Dynamic-Epistemic Logic of Questions and Inquiry

Ştefan A. Minică

Dynamic-Epistemic Logic of Questions and Inquiry

Cover design by Camelia C. Minică, Mircea D. Minică & Ştefan A. Minică.

Printed and bound @ Lulu:
`http://www.lulu.com/content/paperback-book/delq---paperback/12066431`
`http://www.lulu.com/product/hardcover/delq---hardback/18629151`
`http://www.lulu.com/product/ebook/delq---ebook/18634437`

ISBN: 978–90–5776–229–1

The investigations were supported by the Huygens Scholarship Programme (HSP), which was subsidized by the Netherlands Organization for International Cooperation in Higher Education (NUFFIC) and the Institute for Logic, Language and Computation (ILLC).

Memoriei tatălui meu

Contents

Chapter 1

Introduction and Motivation

The logical study of questions is important for various reasons, it has an interesting history, and significant potential for further development and applications.

The interest in questions as an intellectual tool is manifest already in Antiquity, in the emergence of philosophical thinking. The 'Socratic method' makes questions an essential element of argumentation and inquiry. Many subsequent advances in science can be understood as refinements of best methods for raising questions, and finding better instruments for giving answers, leading to progressive theory improvement.

The study of questions in a formal setting coincides with the emergence of modern logic in the beginning of the 20th century. Already Kazimierz Ajdukiewicz suggested in the 1930s that asking questions is a logical act, analogous in importance to drawing conclusions. While this did not become mainstream logic at the time, there has been a constant flow of interest in formal modeling of questions, in (epistemic) logic, epistemology, philosophy of science, artificial intelligence, information theory, game theory, and natural language semantics.

Asking and answering questions has been considered the decisive criterion for defining and recognizing intelligence [87]. Contemporary practical applications of questioning theory include query languages in database theory [61, 36], searching the world wide web, guiding rovers exploring Mars towards the best way of performing inquiry in a new environment [58], or computers able to beat human champions in open-questions contests like Jeopardy, as Watson did recently [29].

There are many traditions in the logical study of questions, which started in a serious manner in the 1960s: cf. the discussion in [13, 48, 44, 117]. It would be tedious to discuss all of these, but here are a few major approaches:

- answer sets and question logic analogous to logics of assertions [13],
- partition representation for question semantics to natural language questioning acts [40],
- questions as epistemic devices exploring truth in the world [48],

– questions related to inferential strategies in erotetic scenarios [117],

– algorithmic modeling of questions inside a process of inquiry [56].

These approaches range from quite concrete to highly abstract. There have even been meta-theoretical results on incompleteness in [43], using Cantor's diagonal argument for sets of answers, and undecidability in [81], which is inherited from the underlying first order logic. But decidability of more constrained reasoning tasks with questions has also been established [15] as well as efficiency of model checking in suitable modal logics.

In this thesis, we want to add something to this existing literature. We will focus on questions as dynamic actions that change some current issue and guide investigation. We will then formalize this view in terms of current 'dynamic-epistemic logics', asking ourselves: what, precisely, is the natural repertoire of questions and question-related actions, and what are the natural valid principles of reasoning about them?

But before getting to this point, we discuss a number of existing approaches to set the scene, identify some major insights that are available, and then state what further things we want to achieve.

1.1 Brief History of Approaches to Questions

We start by a brief overview of major previous approaches to questions that will be most relevant for later developments in this thesis.

– **The Logic of Interrogation** (LoI), close to natural language semantics, [40, 37] starts from considering a set of possibilities. Then a context is defined as an equivalence relation on these. A language containing first order sentences and formulae is used to talk about such structures.

  1. The semantics of both indicative and interrogative sentences is given in terms of 'context change potential'. Indicative sentences change the context by eliminating possibilities. In contrast, interrogative sentences change the context by disconnecting possibilities, so not by providing new data but by 'raising new issues'.[1]

[1]Without delving in all formal details here is how the difference between indicative an interrogative context change is presented in [37]. Indicative sentences provide new data, hence an indicative sentence will change the context by eliminating possible worlds: $C[!\phi] = \{(w, v) \in C \mid w \models \phi \wedge v \models \phi\}$. Interrogative sentences also change the context, not by giving new information but by raising an issue, hence the context change potential of an interrogative sentence is to disconnect certain possible worlds: $C[?\phi] = \{(w, v) \in C \mid \|\phi\|^w = \|\phi\|^v\}$. Where C is the context equivalence relation on W, the set of possibilities, and $||\phi||^w = \{g \in D^{FV(\phi)} \mid w, g \models \phi\}$ the extension of ϕ in a triple (W, D, I) where W is a set of possible worlds, D a set of individuals, and I an interpretation function [37].

2. Next, a notion of semantic entailment is defined also in terms of context change potential. This describes relations between informative and interrogative sentences, both as answerhood (either partial or complete) of an indicative sentence, or as compliance (or licensing) between a context resulting from a sequence of assertions and questions. Using licensing and entailment notions of optimality and informativity for answers can also be defined in this framework.
3. This framework leads to a complete axiomatization in [84] and to a syntactic characterization for the answerhood relation [83] under the assumption of rigid designation. The notion of questions entailment is reduced to the notion of entailment between indicative first order formulae by means of a development of a set of formulae and definability and interpolation for first order logic [84].
4. Indicative sentences as primary information carriers are studied in close connection to questions. Later approaches consider hybrid combination of indicative and inquisitive content of sentences [38, 39] and [18].

– **Interrogative Epistemic Logic** and the resulting *Interrogative Model of Inquiry* (IMI), close to epistemology [49, 48, 78], starts by specifying standard tableau decomposition rules for indicative sentences. Then the resulting tableaux are enriched with an Oracle and rules for questioning moves are added. These rules establish the interaction with the available sources of information: only questions with established presuppositions can be asked and only available information will trigger answers from the Oracle.

1. Knowledge operators can be added to this setting and patterns of dependence and independence between quatifiers and the implicit epistemic quantification are used to specify new decomposition rules. Questions to and answers from the Oracle have also their tableau rules.[2]
2. IMI establishes a connection between a first order questioning model using wh-questions [which object(s) have some property?] and a basic propositional questioning model using the simplest yes-no questions [is p the case?] via the Yes-No Theorem, cf. [49].

[2]Skipping the formal details for the rules describing standard logical operations, here is how an (in)dependence pattern involving the epistemic modality K looks like as tableau rules [49]:

$$\vee^K_L\colon\ \frac{\Pi, S_0[S_1(\vee/K)S_2] \to \Sigma}{\Pi, S_0[(S_1 \wedge f(\overrightarrow{y}) = 0) \vee (S_2 \wedge f(\overrightarrow{y}) \neq 0)] \wedge \overrightarrow{\forall y}(\exists/K)(f(\overrightarrow{y}) = x)) \to \Sigma}$$

$$\exists^K_L\colon \frac{\Pi, S_0[(\exists x/K)S_1[x]] \to \Sigma}{\Pi, S_0[S_1[f(\overrightarrow{y})]], \overrightarrow{\forall y}(\exists/K)(f(\overrightarrow{y}) = x)) \to \Sigma} \quad \exists^K_R\colon \frac{\Pi, (\exists x/K)(t = x) \to \Sigma, S_0[(\exists x/K)S_1[x]]}{\Pi, (\exists x/K)(t = x) \to \Sigma, S_0[(\exists x/K)S_1[x]], S_0[S_1[t]]}$$

for S_1, S_2 occurring inside S_0 within the scope of universal quantifiers $\overrightarrow{\forall y} = (\forall y_0) \dots (\forall y_n)$ and where $\overrightarrow{y} = y_0, \dots, y_n$ and f is a new function symbol.

3. IMI establishes a first link between strategic aspects in questioning and the choice of a decomposition rule in an interrogative tableau via the Strategy Theorem, cf. [49].

A more detailed comparison and a comprehensive merge between IMI and Dynamic Epistemic Logic (henceforth, DEL) can be also found in [41].

– **Dynamic Epistemic Logic** based approaches. Inside the broader DEL paradigm questions also have a more recent but fruitful history which is already very close to our own desiderata. A first account of questions as 'communication acts' was introduced in [8]:

1. Questions are a special case in a more general setting of 'abstract dialogues' or 'communication sequences'. Interrogatives, or queries are in this setting a particular kind of communication acts.[3]
2. Communication acts come with a 'timestamp' determining the legal or illegal '(communication) moves' in a 'dialogue game'.
3. Various kinds of questioning moves are captured by complex preconditions for execution and/or the publicity of the action, the security of the communication channel, responsiveness, sincerity, etc.

Questions as processes. Another recent approach inside the DEL paradigm is [88, 105, 106]. Here questions are understood as processes combining Propositional Dynamic Logic (PDL) tests with sequential composition and choice to model changes in the structure of an epistemic model.

1. An update by a question is a complex program that changes a focus equivalence relation representing the content of the asked formula.[4]
2. Standard notions of answerhood and interrogative entailment can be captured in this setting in a natural way, and a large variety of questioning actions and their presuppositions projections can be modeled such as embedded questions, constituent questions, etc.
3. The questioning moves interact with standard informative actions such as public announcements to describe a complete epistemic dynamics via model restrictions and the focus relation induced by questions.

[3]See also Section 4.1 later in the thesis, where we will further discuss the relevance of this approach for questioning games.

[4]We cannot present all the formal details here, but here is the main idea from [88] in a nutshell: the effect of asking the question whether ϕ is a complex program constructed by composing several simple ones as follows: $\mathbf{F}\phi =_{def} (\texttt{Test}\phi; G; \texttt{Test}\phi) \cup (\texttt{Test}\neg\phi; G; \texttt{Test}\neg\phi)$, where $G = W \times W$ is the total binary relation and $;$, $\cup$, and `Test` have their standard meaning from PDL. The extensional result of executing a question is then given by refining the focus relation: $[\![\mathbf{F}\phi]\!]^M = \{(w, w') \mid M, w \models \phi \text{ iff } M, w' \models \phi\}$.

4. Another relevant contribution in this approach is the fruitful implementation of the DEL theory in a computational framework provided by functional programming in Haskell [106, 109, 25].

There are also related approaches and neighboring fields in which the questioning research agenda is reinforced by similar results and research interests:

– **Inferential Erotetic Logic** [119, 118, 117], starts from an account of inferences with questions as premises and builds an approach to problem solving via 'erotetic search scenarios'. This setting leads to results that link erotetic derivation with a systematic search for an answer to one main wh-question using a 'golden path' of propositional yes/no questions which are the simplest in a logical and epistemological hierarchy.

– **Other Inquiry Calculi** There are other traditions aiming to develop abstract calculi for inquiry using various formal frameworks. The approach in [72] and [26] adds a research agenda to the AGM framework for belief revision and studies resulting logics. The approach from [56, 57] provides an algorithmic model for the role of questions inside a process of inquiry. The approach in [20, 59] derives questioning inference rules from algebraic properties of questions in an analogous way in which Boolean algebra is connected with assertions and Bayesian inference [58, 60]. Finally, the approach in [14] studies questions and answers in an orthoalgebraic framework using a 'decorated' partition theory and a formal theory of 'observables'.

– **Game Theory** Approaches that establish a connection in both directions between logical and game theoretical aspects are not new [50], [80, 77], [90]. But even in standard game theoretic approaches questions and experiments have played an important part as ancillary elements in probabilistic belief dynamics [7]. Approaches to awareness and unawareness in imperfect information games have also many common points with the logic of issue management we are going to develop here. A exhaustive presentation is beyond the scope of this brief history, we also refer to Chapters 4, 5, 6 and 7 where we add a game theoretic twist to DELQ itself (i.e. DEL with Questions), and we also mention [23] which provides a comprehensive link between a DEL approach with binary experiments and agreement theorems. Moreover, there are results inside game theory showing how belief dynamics based on questioning actions gives rise to specific, useful and irreducible properties complementing the dynamics induced by informative actions [46].

A complete presentation of all these fields is beyond the scope of this brief introduction, nevertheless it suggest a coherent research agenda that includes topics that resonate in many other research projects.

1.2 Main Topics in the DELQ Approach

In this broad theoretical background boundaries between alternative approaches aimed at explaining the same underlying phenomena of questioning are blurred and clear distinctions might be only partial and sometimes elusive. One way to distinguish the specific of one approach among many others is by finding pivotal criteria for what it aims to achieve and aspects considered important.

We already listed the achievements of previous approaches, we will focus now on some general desiderata that remain unfulfilled.

Even though the connection between knowledge and questions is of interest in many previous approaches, this is not approached in an setting that makes this connection explicit and with a language that can describe both aspects and their intersection with adequate modalities. The issue-knowledge resolution will be studied at both a static level, expressed by an intersection modality, and a dynamic one, by performing the intersection between the two relations. The intersection will be an essential tool in the formal repertoire needed to understand the interdependence between questions and information.

A next natural desideratum left unfulfilled by previous approaches concerns an interactive framework. We will use a genuine multi-agent setting that allows for higher order questioning and information dynamics, private communication and group notions of knowledge and issues.

Another unfulfilled desideratum concerns an exploration of strategic abilities in questioning scenarios and the connection between a questioning theory and designing efficient querying strategies. We will also provide a bridge between a theory of questions and known search heuristics using backtrack oracles.

Many of the topics already discussed, issues acknowledged and problems raised in previous approaches will constitute the main points to be addressed in our approach. Some of the most relevant open problems received from previous traditions will define the profile of our current approach and will guide the solutions we aim at and the general results that emerge in the DELQ approach.

There are at least three general goals that will characterize our approach inside a more general framework of related approaches, and we state them in retrospect from the very beginning. First, we will be interested in developing a setting in which questions are analyzed and understood in their intricate conceptual, logical, and practical interdependence and essential connection with knowledge and information dynamics. Second, a genuine and fruitful guiding interest for a setting in which questions arise in an interactive multi-agent environment doubled by a study of the strategic aspects that emerge in such a setting. Finally, a third defining criterion for our approach will be a constant interest to find the right balance between expressive power and computational complexity. This will be substantiated in a parallel connection with implementing theoretical aspects and in developing logics that will provide at least decidability of reasoning task and practical efficiency for model checking of realistic questioning scenarios.

The discussion so far and the list of further desiderata already provides the main points on the research agenda to be pursued throughout this thesis. We list them here once more in a systematic way:

- make actions explicit, and determine the repertoire systematically,
- multi-agent character of questions,
- role of strategic interaction,
- role of protocols and global temporal perspective,
- need for a 'reality check' in implementation.

All these constitute relevant components in the general theoretical background on which the approach presented in the following chapters will emerge. There are also many topics for further research and connections with similar or alternative approaches that will only be partially resolved in this thesis. We only mention here one of them and reserve a more exhaustive discussion to Section 1.3 after a more detailed overview of the DELQ approach.

Developing an adequate theory of questioning actions for first order logic and the corresponding wh-questions using our framework will remain an important desideratum for further research. In comparison to most of previous approaches this is an important missing bit in our current approach. However, the current questioning theory can already handle much of the relevant and interesting questioning scenarios involving first order definable properties by using oracles. The main idea here is that once FO entailment between FO properties is independently proved, a DELq theory can be applied to oracles that use such a FOL implication. And designing efficient questioning strategies for such oracles does not require higher order logics. We illustrate this with a concrete example in Chapter 6 using an oracle of first order definable properties.

We end this section with a brief explanation of the rationale behind the implementations included in the thesis. They can be understood in connection with previous implementations for PAL and DEL from [108, 107, 109, 106]. We will provide a significant extension of previous implementations which focus on informative actions with analogous functionality for questioning actions. Moreover, we will also extend the standard DEL functionality with strategic aspects that emerge in games with questions. In addition to previous Haskell implementations for DEL functionality from DEMO, we will also use Alloy Analyzer [52] to capture entailment between oracles of local properties in characterizing solution concepts, and the Haskell module for probabilistic functional programming from [27]. The implementations are used to provide concrete illustrations for many theoretical and algorithmic aspects developed in the text. However, their role goes beyond merely providing illustrations and often they will give rise to interesting conceptualizations and provide additional useful results.

Overview of the Thesis Before going into the details of each chapter, we give a bird's view perspective on the succession of the main topics that will converge into the guiding storyline of the entire thesis.

There are two entangled strains contributing to the main story of this thesis. The first strain will be theoretical and will provide a formal analysis of questioning actions in epistemic and interactive contexts. This storyline will be intertwined and doubled by the storyline of implementing the formal aspects and applying them for analyzing realistic questioning scenarios.

The two story-lines are closely related but the succession of chapters allows them to be followed independently. The even chapters will only contain theoretical aspects while the odd chapters complement these with implementation details and further concrete illustrations of relevant example applications.

The first chapter is an introductory chapter in which we also review some of the previous approaches to questions that are most relevant for this thesis.

We start in Chapter 2 with setting the ground for an analysis of questioning actions by introducing a basic logical framework. We extend the standard epistemic models with equivalence relations for questions and introduce a static language to describe such structures by means of corresponding modalities, most important being the resolution modality which uses the intersection of the two equivalence relations. We then go on to define actions changing both epistemic and questioning structure in such models. Most important are the actions of asking and resolution which operate by refining partitions. Then we add dynamic modalities to this language and provide a completeness result by means of reduction axioms. We conclude the chapter with two appendices containing some theoretical background and definitions respectively the proofs of the main results.

We continue in Chapter 3 by presenting and discussing the implementation behind our logic of questions. This will be a literate *Haskell* program that extends the previous implementation for epistemic model checking from DEMO [107, 108] with questioning specific functionality.

The main new additions are a richer, more expressive language that includes formulas with intersection modalities, model checking for resolution, questioning and epistemic formulae and a general and extensible implementation for complex questioning and resolution dynamic actions that emerge in this framework. We will also prove the implementation useful by analyzing some paradigmatic examples of questioning scenarios in epistemic settings.

We return to a theoretical approach in Chapter 4 by defining and investigating games with questioning moves. We first look at strategic games with two players and question-answer moves. We then extend this basic approach to settings with more players, sequential moves, and oracles encoding interactions between imperfect agents or limitations in external information sources or experimental procedures. We define solution concepts for such games and consider some illustrative examples. We continue by basic results about questioning games and compute outcomes in some illustrative interactions. We then investigate strategic abilities in epistemic games with extensive questioning moves and show how and why the difraction property is an important property of questioning games that is relevant for a general framework for long term questioning actions in inquiry. Two final appendices contain background definitions and proofs of main results.

In Chapter 5 we return to implementation by presenting and discussing the Haskell scripts behind the questioning games introduced in the previous chapter. These also extend basic epistemic functionality from [109] to include strategic aspects specific for a game theoretic approach of questioning actions. Some of the most noticeable new functionalities include the use of nominals to define the execution value of questioning strategies by linking the semantic level based on partitions of the domain with a corresponding syntactic level in expressive harmony. This gives us a notion of strategy equivalence based on the execution value for a question in an issue-epistemic model.

Using this notion of strategy equivalence we build the space of strategy profiles for players in the issue-epistemic model at a state. Then using this local perspective we construct the induced game that represents the questioning interaction at a global level of the model. Finally, we build the game matrix in the induced global game by performing model checking for the goal formulae of the agents and then by aggregating the results over the entire domain. We prove the implementation useful by analyzing in detail the process of counting strategies and goals and computing the outcomes in a representative example of a interactive and competitive questioning scenario and further possible variations.

We also give an algorithm for minimizing issue-epistemic structures while preserving the truth value of formulae using the intersection modalities. This is based on a refinement algorithm that solves the birelational coarsest partition problem. To match this fixedpoint refinement process we also define a notion of behavioral equivalence between issue-epistemic models and we show that this is the adequate invariance notion for our questioning language.

Chapter 6 approaches the topic of designing questioning strategies in problem solving from a theoretical perspective. We take again solving games as our point of departure and a rich test case representative for a more general theory. In this context we investigate the problem of solving the location game (LG) played on a line by finding all Nash equilibria (NE) with pure strategies.

We give a characterization of NE by means of local properties in the game. Then we use an approach based on querying an oracle of local properties to design questioning strategies that solve the game in an efficient way. We end by a discussion of the general relevance of this result for designing querying strategies in problem solving by using oracles of operational properties to solve a principal problem using efficiently available sources of information. As in the previous chapters, two appendix sections conclude the study by presenting some useful background definitions and the proofs for the main results.

Section 6.4 approaches the dynamics of questioning in a setting enriched with probabilities. We start by considering previous approaches to DEL in a probabilistic setting in general, and, in particular approaches that add questions into the mix. Then we give an algorithm for minimizing probabilistic issue-epistemic models while preserving the truth value of formulae using the intersection modalities and probabilistic inequalities. This is based on a refinement algorithm that solves the birelational coarsest partition problem. Based on this we also give a notion of behavioral equivalence between probabilistic issue-epistemic models and we show that this is the adequate invariance notion for our language.

Next, as we did until now, in Chapter 7, we present and discuss the implementation behind the results in the previous chapter. In this case we will use two software tools. The first is, as in the previous chapters Haskell. We will show how queries of local properties in the game can be used to search for equilibrium strategy profiles in an implementation using list comprehension. This approach assumes the existence of oracles of local properties and uses this to search NE.

Because both NE and the local properties are expressible by formulae of first order logic we will use a second implementation, in Alloy Analyzer [52], to build a model for the location game and check assertions about logical entailment within a predetermined scope between formulae expressing local properties and NE. Once more, we will illustrate the use of both implementation tools by considering in detail representative examples of concrete strategic interactions in LG.

Chapter 8 is dedicated to draw the general conclusions and to discuss how our approach gives rise to a coherent research agenda and many directions for further study and broader connections with other relevant topics both inside DEL.

The Sources of Some Material The content in Chapter 2 is based on material which has been previously presented in preliminary versions at the LIRa seminar in Amsterdam and subsequently published in the LIRa Yearbook 2009. It was also presented at the Second International Workshop on Logic, Rationality and Interaction (LORI-II) in Chongqing, China, October 6-11, 2009, and subsequently published in the conference proceedings as [100].

Section 4.4 has been presented in preliminary version at the 9th Conference on Logic and the Foundations of Game and Decision Theory (LOFT 9), in Toulouse, France, 5-7 July, 2010, as [1]. Section 4.4 has been presented and subsequently published in the proceedings of the second ILCLI International Workshop on Logic and Philosophy of Knowledge, Communication and Action, (LogKCA-10), Donostia - San Sebastian, Spain, 3-5 November 2010 as [70].

A preliminary version of implementing questioning dynamics was presented and subsequently published in the proceedings of the Second International Conference on Computer Supported Education (CSEDU 2), in Valencia, Spain, 7-10 April, 2010 as [71]. Section 5.1.1 is the implementation corresponding to the theoretical framework from [1] and was also used in [70]. The content in Chapter 6 originated from a homework exercise while the author was grading assignments for the Strategic Games [4] course at ILLC in 2010 and it has been presented before in the ComSoc seminar in Amsterdam and in the Student Session of the Sino-European Winter School in Logic, Language and Computation (SELLC) Guangzhou, China, December 3-18, 2010, and subsequently included in the electronic proceedings as [69].

1.3 Comparisons with Alternative Approaches

There are many classical traditions [42, 5, 65, 13] that provide useful insights for an approach to questions and many of these are reinforced when considered in the light of our current dynamic logics of questioning actions. It is beyond the scope of this brief introduction to make a comprehensive comparison to all of them, we will only focus on those of them that offer the closest connections.

One which is directly connected to our approach is the active program of Interrogative Model of Inquiry (IMI) [48, 49] and [50, 78, 32]. As already mentioned, questions are treated here as requests for new information, which function intertwined with deductive indicative moves in 'interrogative tableaux'. There is an extensive theory of answerhood, as well as an analysis of various types of question in a predicate-logical setting, beyond what we have done here. The framework has a number of nice theoretical results, including meta-theorems about the scope of questioning in inquiry and discovery. A systematic comparative study of the relations between the two approaches is still needed in order to have a complete picture of the connections. There are previous studies that already started such an investigation and proposed meaningful and fruitful merges between the frameworks. A bridge in this direction has been given by [41], and comparison points can be also found in [76]. Much of what we did in this thesis can also be seen as a further development of some of the themes put forward in IMI.

Such themes also emerge in Inferential Erotetic Logic (IEL) [119, 118, 117] which provides an account of inferences with questions as premises and, starting from this notion, an approach to problem solving via 'erotetic search scenarios' linking erotetic derivation with a systematic search for an answer to one main wh-question using a 'golden path' of propositional yes/no questions which are the simplest in a logical and epistemological hierarchy.

Our approach in Chapter 6 establishes a fruitful link between a questioning theory and known search heuristics using backtrack oracles. This shows that a questioning theory is relevant and can be useful in designing efficient querying strategies by using oracles of first order properties. It also shows that designing such query strategies can be done with yes/no questions alone as long as the higher first order entailment between properties encoded in the available oracles has been established. This comparison is in no way complete but it provides a first step towards a more general theory for efficient problem solving via questioning mechanism design and issue management.

Another close comparison to our approach stemming from the LoI tradition is the recent inquisitive semantics ([39, 38, 18]). Inquisitive semantics gives propositions an "interrogative meaning" defined in a universe of information states over propositional valuations, with sets of valuations expressing issues. This supports a compositional semantics for the language of propositional logic, where, for instance, a conditional is true in an informational sense if every subset (stronger information state) supporting the antecedent also supports the consequent. Interrogative meanings are then defined in terms of generalized partitions of the set of worlds, where partition cells may now also overlap. This is a significant extension of the traditional issue picture. Based on this semantics, a propositional logic arises that describes valid consequence and other important relations between questions, and for questions and answers. The program has found a variety of applications to natural language semantics and pragmatics. Of course, we cannot do full justice to this framework here: for recent updates see [86]. At some level of abstraction, the ideas in this system sound very close to ours: there is information dynamics, questions also change current possibilities, and so on.

However, a systematic study of the formal relationship between the two does not yet exist in the literature. Neither does the scope and purpose of this thesis provide the proper setting for such a comparison. The best we can do at this stage is to give a starting point for a methodological comparison, while acknowledging the fact that further investigation is needed in order to have a complete picture of the existing connections bridging the two approaches in both directions.

We start from a basic observation that seems the key point of difference between the two approaches. Inquisitive semantics puts the dynamic information about questions in a new account of the meaning of interrogative sentences in a propositional language. This is not classical declarative meaning, and hence some deviant propositional logic emerges.

By contrast, the DELQ approach wants to give an explicit account of questions and other actions of issue management, but it does so by means of dynamic modalities on top of a classical logical language. In particular, there is no meaning shift: but rather an expansion of the domain of study of classical logic.

The distinction is similar to one in logic itself (cf. [89, 92]). Intuitionistic logic studies knowledge and information *implicitly* by changing the meaning of the classical logical constants, and then picking a fight with classical logic in the set of 'validities'. By contrast, epistemic logic analyzes knowledge *explicitly* as an additional operator on top of classical propositional logic: there is no meaning shift, but agenda expansion. In our view, DELQ stands in exactly the same relationship to inquisitive logic: it makes the dynamics explicit, and steers away from foundational issues of meaning and validity. Comparisons between the two approaches can be quite delicate (cf. [92]), and the same may also be true here.

Much of what we did in chapters 3 and 5 is an investigation related to themes put forward in the inquisitive approach. We have considered many examples of natural language conversational and epistemic scenarios in which various gradients of mixing information flow with raising issues emerge. We have also given reduction axioms for some of these combinations, and, perhaps most importantly, we have given an unifying method, a way to treat all such gradients in an uniform way by using a unique product update rule. We have also shown how adequate models for this 'art of pragmatic modeling' can be generated using implementation tools. We have also shown how the partition model can be extended to deal with questions inducing a cover representation using the product update rule.

A complete comparison should proceed by giving characterization results by explicit reduction axioms that will bridge the two approaches, this is a very interesting enterprise for future research. Our approach that deals with birelational structures might also bare some relevance to more recent inquisitive approaches [17] that also consider the interaction between issues and attention or awareness.

Finally, our multi-agent approach to question-answer games revealed the fact that the interactive desideratum is not just a collateral extension. We have shown that multi-agent interaction gives rise to concepts about questioning that generalize traditional notions. The questioning games analyzed in Chapters 4 and 5 showed that received standard notions of relevance and informativity for questions turn out to be one particular case in a more general conceptualization that also takes the interaction with epistemic aspects into account.

But the closest comparison is inside DEL itself. There are previous DEL approaches that inspired and guided what we have done. The first DEL approach to questions as communicative actions [8] was already using product update and gave reduction axioms for legal 'timestamped' moves in an abstract dialogue with questions and answers. Our approach shows how this methodology can be lifted from epistemic events to sets of epistemic events representing the answers to questions and shows that this works essentially in the same way as it did for logics focusing on informative actions. The same issues of privacy and publicity that were treated for informative actions emerge for questioning actions.

Another close comparison inside DEL is the approach using a PDL tests and regular operations to refine the focus relation [106, 105, 88]. This is already very close to our approach both in desiderata and in formal details. The main further contribution provided by our approach is to add intersection and show how to handle it in a formal framework that captures the interaction between questions and knowledge. Another relevant comparison point that emerges here consists in the fruitful connection with implementations. The preexisting functionality for dynamic epistemic modeling from DEMO [107, 109] was the starting point for our extension to questions. We add functionality for questioning actions to this framework, we also give algorithms for minimizing birelational models and we link this to an notion of structural equivalence adequate for questions. We also show how the Haskell implementation can be used for extensions modeling strategic aspects in question-answer games and also in basic probabilistic scenarios.

Last but not least we mention the close connection with game theory. This connection is a major theme of the thesis and bridges its topics in both directions.

First we will show how the standard DELQ approach can lead to interesting results when enriched with game theoretic concepts and we will add a more general game theoretic twist to questioning actions in chapters 4 and 5 by considering and analyzing games with questions.

Second, in chapters 6 and 7, we will start from a game theoretical analysis of a concrete interactive situation in the location game on a line and use queries of local properties together with backtrack oracle heuristics to design query strategies for solving the game.

Chapter 2
Dynamic Epistemic Logic of Questions

2.1 Basic Logical System for $\mathbf{DEL}_{\mathrm{Q}}$

In this chapter we set the stage by introducing our basic logical system for representing questions and for giving an account of the dynamics of question answering.

2.1.1 Issue-Epistemic Models

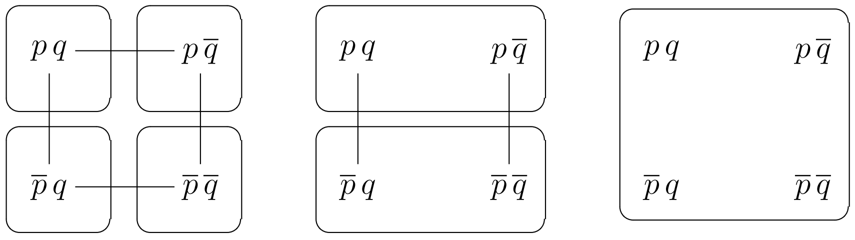
We work over standard epistemic models. In this setting, a simple framework for representing questions uses an equivalence relation over some relevant domain of alternatives, that we will call the 'issue relation'. This idea can be found in many places, from linguistics (cf. [40]) to learning theory (cf. [57]): the current 'issue' is a partition of the set of all possibilities, with partition cells representing the relevant classes of alternatives. We do not need prior assumption about how these equivalence classes are generated. This abstract partition may be induced by a concrete conversation whose current focus are the issues that have been put on the table, or a game where finding out about certain issues has become important to further play, a learning scenario for the regularities implicitly present in our environment, or equally a whole research program with an agenda determining what is currently under investigation. The relevant alternatives or possible worlds are domain dependent: they may range here from simple finite settings like deals in a card game to complex, potentially unbounded, histories representing a total life experience. Formally, all this can be represented in the following way:

2.1.1. Definition. [Issue Epistemic Model] An epistemic issue model is a structure $M{=}\langle W, \approx, \sim, V\rangle$ with the following components: W is a set of possible worlds or states (epistemic alternatives), $\approx$ is an equivalence relation on W (the abstract issue relation), $\sim$ is an equivalence relation on W (epistemic indistinguishability), and $V : \mathtt{P} \rightarrow \wp(W)$ is a valuation for atomic propositions $p \in \mathtt{P}$.

This basic setting can be extended for more general relations which are not assumed to be equivalences to model belief instead of knowledge, or issue structures which lack epistemic introspection or other relevant properties. We will not use such extensions in this thesis, we will further discuss some of them in Chapter 8 and continue to work with equivalence relations.

We intuitively illustrate in Figure 2.1 the formal components from Definition 2.1.1 while describing their workings in the text. In Figure 2.1 and in subsequent diagrams we will use some representation conventions: epistemic indistinguishability will be represented by lines linking possible worlds, and the issue relation will be represented by partition cells, when appropriate, we will also use double lines instead of partition cells to represent issue relations. We use the standard conventions and skip, for brevity, reflexive and transitive relations. Unless otherwise mentioned, we will also assume that the actual world is the top left one and indicate the valuation function by propositional symbols.

Figure 2.1: Examples of Epistemic Issue Models



With this understanding, Figure 2.1 depicts, from left to right, an epistemic issue model in which nothing is known and everything is an issue, a second one in which q is known in the actual world and the issue is to find out about p, and, finally, one in which everything is known in the actual world, and nothing is object of questioning, i.e. we have a universal issue relation on the domain.

2.1.2 Information and Issues: Language and Semantics

To describe facts and properties in and about our static issue structures we will introduce a language with matching modalities. We make a minimal choice of modal and epistemic logic of state spaces plus two new modalities: one describing the issue structure the second talking about both epistemic information and questioning structure. First, $K\varphi$ talks about knowledge or semantic information of an agent, its informal short reading is 'φ is known', and its extended explanation is as usual: 'φ is the case in all epistemically indistinguishable worlds'.

In a similar fashion, we use $Q\varphi$ to say that, locally, in a given possible world, the current structure of the issue-relation makes φ true, in a longer phrasing: 'φ is true in all possible worlds that are issue-equivalent'. This local notion is convenient in itself, but it does not express yet the global assertion that the current issue *is* φ, this will be defined later in terms of the current local notion.

Most importantly, we often need to express a notion that mixes the epistemic and issue relations, expressing (roughly) what would be the case if the issue were resolved and given what is already known in the structure. Technically, we add an intersection modality $R\varphi$ saying that 'φ is the case in all worlds that are both epistemically indistinguishable from and issue equivalent to the current one'.

Finally, in order to describe our structures at a global level, we add a universal modality $U\varphi$ saying intuitively that 'φ is true in all worlds'.

While such modalities offer adequate expressive power for our current purpose and are also frequent in many other settings, they also complicate axiomatization. To deal with such formal complications, we will use the standard device of adding *nominals* naming single worlds to our language (cf. [33, 66] for recent instances of this technique in the *DEL* setting). As a first change, working with nominals requires a modified valuation function in Definition 2.1.1, to a $V : \mathtt{P} \uplus \mathtt{N} \to \wp(W)$ mapping every proposition $p \in \mathtt{P}$ to a set of states $V(p) \subseteq W$, but every nominal $i \in \mathtt{N}$ to a singleton set $V(i)$ of a world $w \in W$.

2.1.2. Definition. [Static Language] The language $\mathcal{L}_{\mathbf{EL_Q}}(\mathtt{P}, \mathtt{N})$ uses two disjoint countable sets $\mathtt{P}$ and $\mathtt{N}$ of propositions and nominals, respectively, with $p \in \mathtt{P}$, $i \in \mathtt{N}$. Its formulas are defined by the following inductive syntax rule:

$$i \mid p \mid \bot \mid \neg\varphi \mid (\varphi \wedge \psi) \mid U\varphi \mid Q\varphi \mid K\varphi \mid R\varphi$$

When needed, dual existential modalities $\widehat{U}$, $\widehat{K}$, $\widehat{Q}$ and $\widehat{R}$ are defined as usual: E or $\widehat{U} := \neg U \neg, \widehat{Q} := \neg Q \neg, \widehat{K} := \neg K \neg, \widehat{R} := \neg R \neg$.

Occasionally we will also use, interchangeably, the following notation which makes the interactions between the relations used in the static modalities more explicit: $i \mid p \mid \bot \mid \neg\varphi \mid (\varphi \wedge \psi) \mid U\varphi \mid \langle\approx\rangle\varphi \mid \langle\sim\rangle\varphi \mid \langle\sim \cap \approx\rangle\varphi$. Customary shortcuts to express disjunction and other boolean connectives are also used in their standard way. Formulas in this static language receive their meaning by:

2.1.3. Definition. [Interpretation] Formulas are interpreted in models M at worlds w by the following recursive clauses: $M \models_w p$ iff $w \in V(p)$, $M \models_w i$ iff $w \in V(i)$, $M \models_w \neg\varphi$ iff not $M \models_w \varphi$, $M \models_w \varphi \wedge \psi$ iff $M \models_w \varphi$ and $M \models_w \psi$, $M \models_w U\varphi$ iff for all $w \in W : M \models_w \varphi$, $M \models_w K\varphi$ iff for all $v \in W : w \sim v$ implies $M \models_v \varphi$, $M \models_w Q\varphi$ iff for all $v \in W : w \approx v$ implies $M \models_v \varphi$, $M \models_w R\varphi$ iff for all $v \in W : w\,(\sim \cap \approx)\,v$ implies $M \models_v \varphi$.

For instance, with this language we can express that the structure of the current issue settles fact φ with the following formula: $U(Q\varphi \vee Q\neg\varphi)$. The term 'settling' is used here in a technical sense, as saying that the issue answers (either explicitly or implicitly) the question whether φ holds. In natural language, there is also the notion of 'settling an issue', an event of finding out which partition cell we are in. This will be a later action of 'issue management', that of *resolution*.

This language can also express that an agent considers it possible that fact φ is not settled by the structure of the current issue: $\widehat{K}(\varphi \wedge \widehat{Q}\neg\varphi)$. The next example says that an agent knows locally that a certain fact φ would be settled by the issue, while it is not settled globally: $KQ\varphi \wedge \neg U(Q\varphi \vee Q\neg\varphi)$.

As for the third modality of 'resolution', it describes intuitively what agents would know if the current issue is resolved. Thus, we can say that in the current epistemic situation the fact expressed by φ is neither known by the agent nor settled by the structure of the issue, but is true upon resolution: $\neg Q\varphi \wedge \neg K\varphi \wedge R\varphi$.

A more complex example is when a fact is neither known nor settled in any world of the model, but it is true in all indistinguishable and issue-equivalent worlds, and it would be settled by a resolution action: $\neg\widehat{U}(K\varphi \vee Q\varphi) \wedge UR\varphi$.

These examples show that our language can express quite complex notions about questions. Many such notions have been already considered in the literature about questions and information flow, but often restricted to factual questions, and without the benefit of a uniform formal language. We include further examples of how the interaction between information and questioning can be expressed by formulas of our language in Section 3.2. As it is the case with every logical language, there is a trade-off between its expressive power and its computational complexity. The main jump in this context is given by the use of the universal modality. We discuss the computational cost that has to be paid for this gain of expresivity in Section 2.7.

2.1.4. Fact. The *intersection modality* $R\varphi$ is not definable in terms of the issue modality $Q\varphi$ and the epistemic modality $K\varphi$.

2.1.1. Proof. A simple bisimulation argument establishes this claim. Consider two issue-epistemic models: $M = \langle W, \approx, \sim, V\rangle$ where $W = \{w, v, u\}, \approx = \{(w, v)\}, \sim = \{(w, u)\}, V = \{p \mapsto \{v, u\}\}$ and $M' = \langle W', \approx', \sim', V'\rangle$ where $W' = \{x, y\}, \approx' = \{(x, y)\}, \sim' = \{(x, y)\}, V' = \{p \mapsto \{y\}\}$. Take the following relation between the models $Z = \{(w, x), (v, y), (u, y)\}$, then worlds w and x are bisimilar and should satisfy the same modal formulas. However, we have both: $M \models_w \neg\widehat{R}p$ and $M' \models_x \widehat{R}p$ □

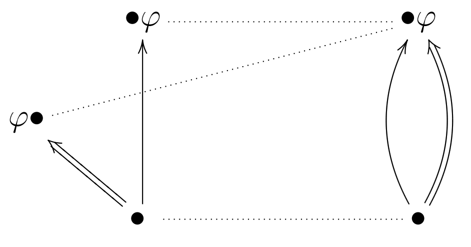


Figure 2.2: The intersection modality is not invariant under bisimulation.

An intuitive illustration of this fact is given in Figure 2.2. Here we can witness that, in particular, $\widehat{R}\varphi$ is not equivalent with $\widehat{K}\varphi \wedge \widehat{Q}\varphi$. However, the use of 'nominals' i from hybrid logic helps us to completeness, by providing the needed valid implication in the converse direction: $\widehat{K}(i \wedge \varphi) \wedge \widehat{Q}(i \wedge \varphi) \to \widehat{R}\varphi$.

2.1.3 Static Logic of Information and Issues

As for reasoning with our language, here are some valid implications: $\widehat{K}\varphi \to \widehat{U}\varphi, \widehat{O}\varphi \to \widehat{U}\varphi, \widehat{R}\varphi \to \widehat{U}\varphi$. However, the following implications are not valid: $R\varphi \to Q\varphi, R\varphi \to \neg Q\varphi, K\varphi \to Q\varphi, Q\varphi \to K\varphi, R\varphi \to K\varphi$.

More generally, we write $\models \varphi$ if the static formula φ is true in every model at every world. The static epistemic logic $\mathbf{EL_Q}$ of information and questions in epistemic issue models is the set of all validities: $\mathbf{EL_Q} = \{\varphi \in \mathcal{L}_{\mathbf{EL_Q}} : \models \varphi\}$.

Next, we consider the set of formulas derivable in the following proof system:

2.1.5. Definition. [Axiomatization] The proof system EL_Q contains the customary (epistemic) S5 axioms for K, Q and R:

1. $Kp \to p$ (Truth), $Kp \to KKp$, $\neg Kp \to K\neg Kp$ (Full Introspection)
2. $p \to Q\widehat{Q}p$, $p \to \widehat{Q}p$, $\widehat{Q}\widehat{Q}p \to \widehat{Q}p$ (equivalence relation for issue),
3. $p \to R\widehat{R}p$, $p \to \widehat{R}p$, $\widehat{R}\widehat{R}p \to \widehat{R}p$ (equivalence relation for resolution),

together with the characteristic axiom for intersection:

4. $\widehat{K}i \wedge \widehat{Q}i \leftrightarrow \widehat{R}i$.

In addition, it contains a standard hybrid logic with universal modality: $\Box(p \to q) \to (\Box p \to \Box q)$, (Distribution), $\neg\Box\neg p \leftrightarrow \Diamond p, \Box \in \{U, K, R, Q\}$ (Duality), $p \to U\widehat{U}p, p \to \widehat{U}p, \widehat{U}\widehat{U}p \to \widehat{U}p, \widehat{U}i, \Diamond p \to \widehat{U}p, \Diamond \in \{\widehat{K}, \widehat{R}, \widehat{Q}\}$, $\Diamond(i \wedge p) \to \Box(i \to p), \Box \in \{U, K, R, Q\}$ (Nominals), From $\vdash_{\mathsf{PC}} \varphi$ infer φ (Prop), From φ and $\varphi \to \psi$ infer ψ (MP), From φ infer $\Box\varphi$, for $\Box \in \{U, K, R, Q\}$ (Necessitation), From φ and $\sigma_{\text{sort}}(\varphi) = \psi$ infer ψ, where σ_{sort} is 'sorted'[1] From $i \to \varphi$ infer φ, for i not occurring in φ, From $\widehat{U}(i \wedge \Diamond j) \to \widehat{U}(j \wedge \varphi)$ infer $\widehat{U}(i \wedge \Box\varphi)$, for $\Diamond \in \{\widehat{K}, \widehat{R}, \widehat{Q}\}$, $i \neq j$, and j not occurring in φ.

We write, as usual, $\vdash_{EL_Q} \varphi$ if φ is provable in the proof system EL_Q. These basic laws of reasoning derive many principles that confirm basic intuitions. For instance, here is the simple proof that agents have introspection about the current public issue: $U(Qp \vee Q\neg p) \vdash_{EL_Q} UU(Qp \vee Q\neg p) \vdash_{EL_Q} KU(Qp \vee Q\neg p)$. Here are some simple derivable principles describing interactions between the epistemic, issue, resolution and universal modalities: $U\varphi \to K\varphi, U\varphi \to Q\varphi, U\varphi \to R\varphi$. These properties are useful for describing epistemic and issue properties from a global perspective. The intersection modality is, in its turn, included in both the issue and the epistemic modalities, a feature that will allow us later on to describe the way they interact in dynamic contexts: $K\varphi \to R\varphi, Q\varphi \to R\varphi$.

All this machinery leads to the following expected general result:

[1]The technical notion 'sorted' and its uses are explained in [85], the main idea being that the substitution is applied distinctly to each syntactic sort of propositions and, respectively, nominals.

2.1.6. THEOREM ($\mathbf{EL_Q}$ COMPLETENESS). *For every formula* $\varphi \in \mathcal{L}_{\mathbf{EL_Q}}(\mathtt{P}, \mathtt{N})$*:*

$$\models \varphi \quad \textit{if and only if} \quad \vdash_{EL_Q} \varphi$$

The proof techniques for results like this one are standard in the literature [85]. We include further relevant details and references in Sections 2.7 and 2.8 and continue towards adding dynamics to our static language.

2.2 Dynamic Logic of Issue Management

In the dynamic epistemic logic methodology, the next step is now to identify basic events of information flow, and expand the so far static logic accordingly.

The situation is completely analogous for the logic of questioning events and actions of 'issue management', and we will proceed in this section in a similar way towards 'dynamifying' the static level by considering structure-changing actions.

2.2.1 Basic Actions of Issue Management

To identify basic actions that change the issue relation in a given model, we first look at some intuitive illustrations. For simplicity, we start with the initial issue as the universal relation, represented as a frame border.

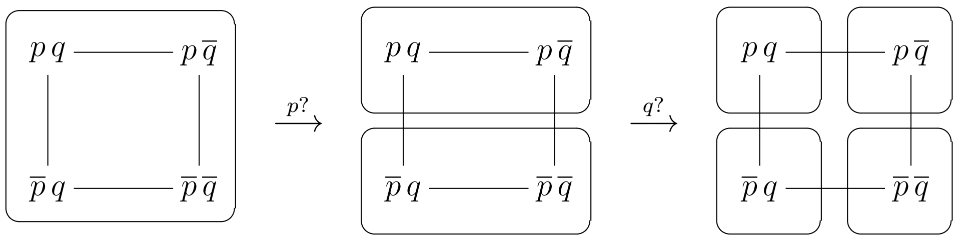


Figure 2.3: Effects of Asking Yes/No Questions.

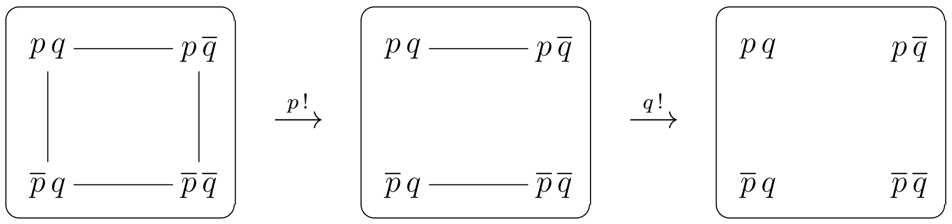


Figure 2.4: Almost Symmetrical Effects of 'Soft' Announcing.

In Figure 2.3, the first transition records the effect of asking a question: the issue relation is split into p and $\neg p$ cells. The second transition illustrates the a second question: the issue partition is further refined.

In Figure 2.4, the first transition is an announcement: the indistinguishability links between p and $\neg p$ worlds are removed. The second transition shows how a second announcement further refines the epistemic partition. Here and henceforth, we use a special sort of event that is congenial to this setting, viz. the

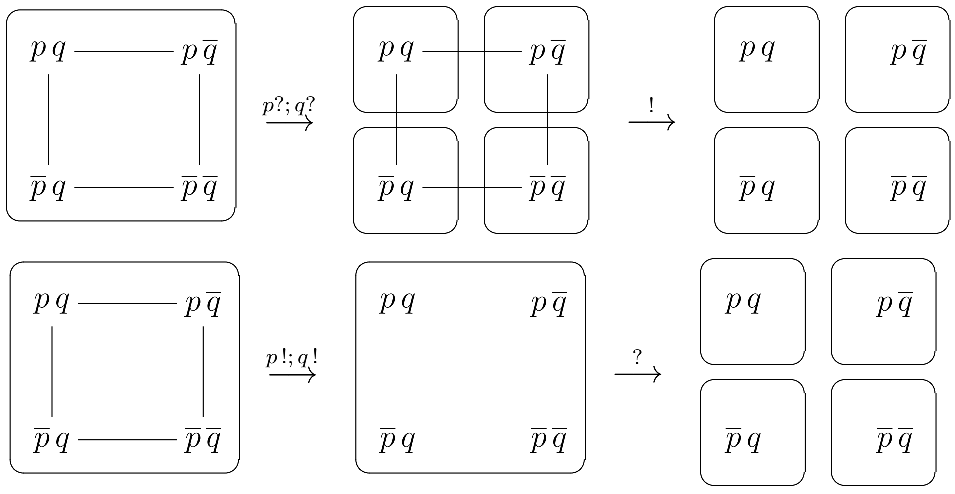


Figure 2.5: Resolving and Refining Actions.

link-cutting announcements $\varphi!$ from [99]. Unlike eliminative public announcements, these do not throw away worlds, but merely cut all links between φ- and $\neg\varphi$-worlds, keeping the whole model available for further reference. In this way, there is a symmetry between a question and a soft announcement. One refines the issue, the other the information partition:

2.2.1. Definition. [Question & Announcement] Let $M = \langle W, \approx, \sim, V\rangle$ be an epistemic issue structure and $\overset{\varphi}{\equiv}_M = \{(w, v) \mid \|\varphi\|^M_w = \|\varphi\|^M_v\}$. Executing action $\varphi?$ in M results in $M_{\varphi?} = \langle W_{\varphi?}, \sim_{\varphi?}, \approx_{\varphi?}, V_{\varphi?}\rangle$, and executing a $\varphi!$ action results in $M_{\varphi!} = \langle W_{\varphi!}, \sim_{\varphi!}, \approx_{\varphi!}, V_{\varphi!}\rangle$, with: $W_{\varphi?} = W_{\varphi!} = W$, $V_{\varphi?} = V_{\varphi!} = V$,

$$\begin{array}{lclclcl} \sim_{\varphi?} & = & \sim & \qquad & \sim_{\varphi!} & = & \sim \cap \overset{\varphi}{\equiv}_M \\ \approx_{\varphi?} & = & \approx \cap \overset{\varphi}{\equiv}_M & & \approx_{\varphi!} & = & \approx \end{array}$$

The symmetry in this mechanism would be lost if we let $p!$ be an executable action only if it is *truthful.* For, the corresponding question $p?$ is executable in every world in a model, even those not satisfying p. The results that will follow can easily be stated for both kinds of announcement: truthful or not.

One attractive feature of this setting is that it suggests further natural operations on information and issues. In particular, Figure 2.5 contains two more management actions. In the first example two Yes/No questions $p?$ and $q?$ are asked, and then a global *resolving* action follows on the epistemic relation. In the second, two announcements $p!$ and $q!$ are made, and a *refinement* action follows on the issue relation, adjusting it to what agents already know. These operations are natural generalizations of asking and announcing:

2.2.2. Definition. [Resolution and Refinement] Let $M = \langle W, \approx, \sim, V\rangle$ be an epistemic issue structure. The execution of the 'resolve' action denoted by !, and of the 'refine' action denoted by ? in model M results in changed models $M_! = \langle W_!, \sim_!, \approx_!, V_!\rangle$, and $M_? = \langle W_?, \sim_?, \approx_?, V_?\rangle$, respectively, with:

$$\sim_? \;=\; \sim \qquad\qquad \sim_! \;=\; \sim \cap \approx$$
$$\approx_? \;=\; \approx \cap \sim \qquad\qquad \approx_! \;=\; \approx$$

Again, the two actions are symmetric. As a way of understanding this, we could introduce a new agent whose role is that of an 'issue manager', dual to the epistemic information agent. It is also useful to have one more issue management action # that *simultaneously* changes both equivalence relations. The effect of executing this in model M is a new model $M_\# = \langle W_\#, \sim_\#, \approx_\#, V_\# \rangle$ with: $W_\# = W$, $\sim_\# = \sim \cap \approx$, $\approx_\# = \sim \cap \approx$, $V_\# = V$.

Here is a final summary of our repertoire of issue management actions:

$[\varphi?]$	Question	$[\varphi!]$	'Soft' announcement
$[!]$	Resolution	$[?]$	Refinement
$[\#]$	Simultaneous resolution	or	'parallel refinement'

Semantic Properties of Issue Management. Our basic actions satisfy some intuitive principles. In particular, our three last ones form an algebra under composition, as illustrated in the following table:

;	!	?	#
!	!	#	#
?	#	?	#
#	#	#	#

With more specific management actions of questions and announcements, the picture is more diverse. In particular, composing these operations is a complex endeavor, and many prima facie laws do not hold, for instance:

2.2.3. Fact. [Composition] The following equations are not valid in DEL_Q:

$$(11)\ \varphi!;! = !;\varphi! \qquad (12)\ \varphi!;? = ?;\varphi! \qquad (13)\ \varphi!;\# = \#;\varphi!$$
$$(14)\ \varphi?;! = !;\varphi! \qquad (15)\ \varphi?;? = ?;\varphi! \qquad (16)\ \varphi?;\# = \#;\varphi?$$
$$(17)\ \varphi?;\psi! = \psi!;\varphi?$$

2.2.1. Proof. The following table lists simple counter-examples, with the following understanding. Each line presents a three-world static starting model i, j, k with information cells between round brackets (,) and issue cells between square brackets [,]. The relevant formulas use nominals for the worlds. □

No.	EIM	φ	ψ
11	$([ij][k])$	$\widehat{K}k \wedge \widehat{Q}j \wedge \neg i$	
12	$[(ij)(k)]$	$\widehat{Q}k \wedge \widehat{K}j \wedge \neg i$	
13	$([ij])([k])$	i	
14	$([ijk])$	i	
15	$[(ijk)]$	j	
16	$([ij])([k])$	i	
17	$([ijk])$	i	$\widehat{R}i$

It is worth noting that some of these examples crucially involve non-factual formulas. E.g., the equalities $\varphi?;\psi! = \psi!;\varphi?$ and $\varphi?;\psi? = \varphi? \cdot \psi?$ are both valid for factual φ, and only fail for formulas φ with non-factual content. This is so because $V_M = V_{\varphi?} = V_{\varphi!}$, that is: the valuation function remains unchanged after all questioning actions. Therefore, by a simple inductive argument, $[\![\varphi]\!]_M = [\![\varphi]\!]_{M'}$ for nonmodal φ, i.e., purely propositional formulas have constant extensions during questioning updates, independently of issue-epistemic changes.

Next, let us see how some known features of information flow in public announcement logic PAL fare with our issue management actions.

Repetition. In PAL, repeating the same assertion $!\varphi$ has no new effects when its content φ is *factual.* But as the Muddy Children puzzle shows, repeating the same epistemic assertion can be informative, and lead to new effects, or in the above short-hand notation: $\varphi! \,; \varphi! \neq \varphi!$. The reason is that when the model has changed, epistemic operators may change truth values.

What about DEL_Q: is asking a question once the same as asking it twice? Again, for factual questions, this is clearly so, given the above semantics: the issue relation no longer changes in the second step. But when the question itself can refer to the issue relation, things are different:

2.2.4. Fact. [Iteration] The equation $\varphi?; \varphi? = \varphi?$ is invalid in DEL_Q.

2.2.2. Proof. Take $\xi := (\widehat{Q}i \to (j \vee k)) \wedge ((\widehat{Q}j \wedge p) \to \widehat{Q}i)$. Both updates for this question, computed as above, change a model with a domain of three worlds i, j, k, a universal issue relation, and a valuation that makes p true at k. □

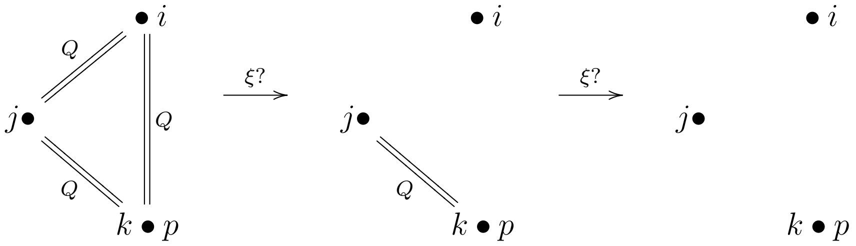


Figure 2.6: Effects of asking the same question twice.

Composition. Next comes a difference with PAL. Public announcement satisfies the valid composition principle that gives the effects of two consecutive announcements with just a single one: $\varphi!; \psi! = (\varphi \wedge [\varphi]\psi)!$ It was observed in [99] and [98], that this does not hold for more complex model changes.[2]

2.2.5. Fact. [Proper Iteration] *There is no question composition principle.*

2.2.3. Proof. If there were one single assertion having just the same effect as a sequence $\varphi?; \psi?$, then, starting with the issue configured as the universal relation on the domain of a model, such a sequence will always induce a two, not four, element partition; this refutation is also depicted in Figure 2.5.[3] □

Related to this are dynamic properties of ordering. While action order makes no difference with purely factual assertions or questions, it does when the content may be of an explicit epistemic or issue-related nature.

We have seen that information update and questions have many subtleties. It is time for a dynamic epistemic logic of issues that can reason about these.

[2]The composition principle also fails in PAL with *protocols*, our topic in Section 6.

[3]This Fact is not a big obstacle. We could easily extend our language with multiple questions, that do not just change partitions on a single-formula basis.

2.2.2 Issue Management: Language and Semantics

In order to talk explicitly about the above changes, dynamic modalities are added to the earlier static language of information and issues:

2.2.6. Definition. [Dynamic Language] Language $\mathcal{L}_{\mathbf{DEL_Q}}(\mathtt{P},\mathtt{N})$ is defined by adding the following clauses to Definition 2.1.2: $[\varphi!]\psi \mid [\varphi?]\psi \mid [?]\varphi \mid [\,!\,]\varphi$.

These are interpreted by adding the following clauses to Definition 2.1.3:

2.2.7. Definition. [Interpretation] Formulas are interpreted in a model M at a world w by: $M \models_w [\varphi?]\psi$ iff $M_{\varphi?} \models_w \psi$, $M \models_w [\varphi!]\psi$ iff $M_{\varphi!} \models_w \psi$, $M \models_w [!]\psi$ iff $M_! \models_w \psi$, $M \models_w [?]\psi$ iff $M_? \models_w \psi$, with $M_{\varphi?}$, $M_{\varphi!}$, $M_?$, $M_!$ as before.

Interesting relations between knowledge, questions and answers can now be expressed in our formal language. We can, for instance, say that a certain question $\psi?$ is entailed in an epistemic-issue model: $U(\widehat{Q}i \to [\psi?]\widehat{Q}i)$, *for all* i[4]. Intuitively this just says that the question does not change the issue structure. We can also express the fact that a sequence of questions entails $\psi?$ with:

$$U([\varphi_0?]\cdots[\varphi_n?]\widehat{Q}i \to [\varphi_0?]\cdots[\varphi_n?][\psi?]\widehat{Q}i)^5 \qquad \textit{for all } i$$

We can also express new notions of entailment, like, for instance, the notion of epistemic global entailment of an arbitrary announcement $\psi!$: $U(\widehat{R}i \to [\psi!]\widehat{R}i)$, *for all* i. A small modification of this in which we relax the previous requirement of abstract global entailment can capture local compliance of answers:

$$[\varphi_0?]\cdots[\varphi_n?](\psi \wedge \widehat{R}i) \to [\varphi_0?]\cdots[\varphi_n?][\psi!]\widehat{R}i^6 \qquad \textit{for all } i$$

Moreover, our language can express basic laws of interrogative reasoning. For instance, we can say that an agent knows in advance that the effect of a question followed by its resolution leads to knowledge of the relevant issue:

$$K[\varphi?][\,!\,]U(K\varphi \vee K\neg\varphi)$$

We include more examples and illustrations of relevant notions capturing epistemic and inquiry interaction in Section 3.2 and proceed to giving a dynamic logic.

[4]Here we assume that each world has a unique nominal i naming it.

[5]This generalizes standard definitions of entailment restricted to factual questions.

[6]Again, this generalizes notions of compliance restricted to propositional formulas.

2.2.3 Dynamic Logic of Informational Issues

We have seen that effects of asking questions are not always easy to keep straight, but also, that there is an interesting structure to management operations on models. Both aspects call for a complete dynamic epistemic logic of questions. Satisfaction and validity are defined as before. The dynamic epistemic logic of questioning is the set of all semantic validities: $\mathbf{DEL_Q} = \{\varphi \in \mathcal{L}_{\mathrm{DEL_Q}}(\mathtt{P},\mathtt{N}) : \models \varphi\}$. We introduce a proof system by adding the reduction axioms below to the earlier proof system EL_Q for the static fragment of the logic.

What follows is a long list of mostly operator commutation principles, interspersed with clauses where 'something happens'. This difference reflects the workings of our semantics of information and issue management:

2.2.8. Definition. [Reduction Axioms] The proof system DEL_Q extends the earlier static logic EL_Q by the following reduction axioms and inference rule:

1. $[\varphi?]a \leftrightarrow a$ (*Asking & Atoms*)
2. $[\varphi?]\neg\psi \leftrightarrow \neg[\varphi?]\psi$ (*Asking & Negation*)
3. $[\varphi?](\psi \wedge \chi) \leftrightarrow [\varphi?]\psi \wedge [\varphi?]\chi$ (*Asking & Conjunction*)
4. $[\varphi?]U\psi \leftrightarrow U[\varphi?]\psi$ (*Asking & Universal Modality*)
5. $[\varphi?]K\psi \leftrightarrow K[\varphi?]\psi$ (*Asking & Knowledge*)
6. *Asking & Resolution*:
$$[\varphi?]R\psi \leftrightarrow (\varphi \wedge R(\varphi \rightarrow [\varphi?]\psi)) \vee (\neg\varphi \wedge R(\neg\varphi \rightarrow [\varphi?]\psi))$$
7. *Asking & Partition*:
$$[\varphi?]Q\psi \leftrightarrow (\varphi \wedge Q(\varphi \rightarrow [\varphi?]\psi)) \vee (\neg\varphi \wedge Q(\neg\varphi \rightarrow [\varphi?]\psi))$$

8-11. The same as items 1 to 4 with $[!]$ instead of $[\varphi?]$.

12. *Resolving & Knowledge*: $[\,!\,]K\varphi \leftrightarrow R[\,!\,]\varphi$
13. *Resolving & Resolution*: $[\,!\,]R\varphi \leftrightarrow R[\,!\,]\varphi$
14. *Resolving & Partition*: $[\,!\,]Q\varphi \leftrightarrow Q[\,!\,]\varphi$

15-18. The same as items 1 to 4 with $[\varphi!]$ instead of $[\varphi?]$.

19. *Answer & Knowledge*[7]:
$$[\varphi!]K\psi \leftrightarrow (\varphi \wedge K(\varphi \rightarrow [\varphi!]\psi)) \vee (\neg\varphi \wedge K(\neg\varphi \rightarrow [\varphi!]\psi))$$

[7] If we assume truthfulness as a precondition of executing an announcement action this axiom (and other ones with a similar structure) does not need the right conjunct and will correspond to the standard DEL axioms for announcement.

20. *Answer & Resolution*:

$$[\varphi!]R\psi \leftrightarrow (\varphi \wedge R(\varphi \rightarrow [\varphi!]\psi)) \vee (\neg\varphi \wedge R(\neg\varphi \rightarrow [\varphi!]\psi))$$

21. *Announcement & Partition*: $[\varphi!]Q\psi \leftrightarrow Q[\varphi!]\psi$

22-25. The same as items 1 to 4 with [?] instead of $[\varphi?]$.

26. *Refining & Knowledge*: $[\,?\,]K\varphi \leftrightarrow K[\,?\,]\varphi$

27. *Refining & Resolution*: $[\,?\,]R\varphi \leftrightarrow R[\,?\,]\varphi$

28. *Refining & Partition*: $[\,?\,]Q\varphi \leftrightarrow R[\,?\,]\varphi$

29. From φ infer $\Box\varphi$, for $\Box \in \{[\cdot?], [\cdot!], [!], [?]\}$ (Necessitation)

We write $\vdash_{DEL_Q} \varphi$ if φ is provable in the proof system DEL_Q.

2.2.9. Theorem (Soundness). *The reduction axioms in DEL_Q are sound.*

2.2.10. Theorem (**DEL$_\mathbf{Q}$** Completeness). *For every formula $\varphi \in \mathcal{L}_{\mathbf{DEL_Q}}$:*

$$\models \varphi \quad \text{if and only if} \quad \vdash_{DEL_Q} \varphi.$$

2.2.4. Proof. This is a standard DEL-style translation argument. Working inside out, the reduction axioms translate dynamic formulas into corresponding static ones. At the end, completeness for the static base logic is invoked. □

Discussion So far we have given a logic of information and questions in standard DEL style. This calculus can derive many further principles, for instance:

The following formula is provable for all factual φ : $\varphi \rightarrow [\varphi?][!]K\varphi$.

2.2.5. Proof.

1	$\varphi \rightarrow (\varphi \wedge R(\varphi \rightarrow \varphi)) \vee (\neg\varphi \wedge R(\neg\varphi \rightarrow \varphi))$	*PC*
2	$\varphi \rightarrow (\varphi \wedge R(\varphi \rightarrow [\varphi?]\varphi)) \vee (\neg\varphi \wedge R(\neg\varphi \rightarrow [\varphi?]\varphi))$	*Factual* φ
3	$\varphi \rightarrow [\varphi?]R\varphi$	*Ak & R*
4	$\varphi \rightarrow [\varphi?]R[!]\varphi$	*Factual* φ
5	$\varphi \rightarrow [\varphi?][!]K\varphi$	*Rs & K*

□

Here and elsewhere, we need the following auxiliary observation.

2.2.11. Fact. For factual formulas φ, with q ranging over management actions, the following equivalence is valid: $[q]\varphi \leftrightarrow \varphi$.

2.2.6. Proof. *The proof proceeds by induction on the structure of formulas, using the Action & Atoms axioms for the base case and the Action & Negation or Action & Conjunction axioms in the inductive steps.* □

Several steps in the previous proof crucially depend on φ being factual, and they would fail otherwise as illustrated in Figure 2.7. The complex formula $Q\neg Kp$ is true initially in every world of the model, but this is not the case anymore after a $\varphi?$ question plus a resolution action [!].

Such changes are not always easy to keep straight, but our logic keeps track of the present, and even more complex cases. But our analysis really shows its power (compared with other approaches to questions) when we consider *multi-agent scenarios* and *protocols for investigation*. These extensions will be taken up in the next two sections.

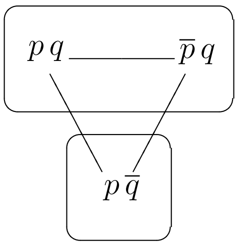


Figure 2.7: The implication $\varphi \to [\varphi?][!]K\varphi$ fails for $\varphi = Q\neg Kp$.

However, we have a remaining issue at this level:

'Hidden validities'. Like with *PAL*, the current axiomatization leave unfinished business. While reduction axioms work on a formula-by-formula basis, they need not describe the general *schematic laws* of the system, such as the earlier composition law for consecutive assertions, that hold under arbitrary substitutions of formulas for proposition letters.[8] This deficit becomes even more urgent here. We saw that our model-changing operations of issue management had a nice algebraic structure. For instance, it is easy to see that resolving is idempotent and commutes with refinement: $!\,;\,!\ =!$ and $!\,;\,?\ =?;\,!$. But our axiomatization for $\mathbf{DEL_Q}$ does not state such laws explicitly, since, by working only from innermost occurrences of dynamic modalities, the completeness argument needed no recursion axioms with stacked modalities like $[!]\,[!]$. Yet this sort of sequential information can be of interest for a logic of issue management.

2.3 Multi-Agent Extensions for $\mathbf{DEL}_{\mathrm{Q}}$

Questions typically involve more than one agent – and hence our restriction to single agents so far misses much of the action. In the following sections we will study multi-agent questioning scenarios.

This will be done in two stages: public questions first, and after that, questions with different informational powers for different agents.

2.3.1 Static Multi-Agent Logic of Information and Issues

The first step is routine. A static language and semantics with many agents follow entirely standard lines, by merely adding a set A of agent labels.

First we modify Definition 2.1.1 to get richer epistemic issue models:

2.3.1. Definition. [Epistemic Issue Model]*An* epistemic issue model *is a structure* $M = \langle W, \overset{a}{\sim}, \overset{a}{\approx}, V\rangle$ *where* W *and* V *are as in Definition 2.1.1 and* $a \in A$*:* $\overset{a}{\sim}$ *and* $\overset{a}{\approx}$ *are binary equivalence relations on* W*.*

[8]Note that the above reduction axioms for atoms typically do not have this substitution property – though many of our more complex reductions axioms do.

2.3.2. Definition. [Static Language] The language $\mathcal{L}_{\mathbf{ELQ_m}}(\mathtt{P}, \mathtt{N}, \mathtt{A})$ is defined by the following inductive syntax rule: $i \mid p \mid \bot \mid \neg\varphi \mid (\varphi \wedge \psi) \mid U\varphi \mid K_a\varphi \mid Q_a\varphi \mid R_a\varphi$. Existential modalities $\widehat{K}_a$, $\widehat{Q}_a$ and $\widehat{R}_a$ are again defined as usual.

2.3.3. Definition. [Interpretation] Formulas are interpreted in models M at worlds w using the same clauses as Definition 2.1.3 for atoms, Boolean and universal combinations, plus the following indexed modal ones: $M \models_w K_a\varphi$ iff for all $v \in W : w \overset{a}{\sim} v$ implies $M \models_v \varphi$, $M \models_w Q_a\varphi$ iff for all $v \in W : w \overset{a}{\approx} v$ implies $M \models_v \varphi$, $M \models_w R_a\varphi$ iff for all $v \in W : w\,(\overset{a}{\sim} \cap \overset{a}{\approx})v$ implies $M \models_v \varphi$.

This language can make agent-dependent distinctions. E.g., it can say that one agent's information is linked with issues for another: $\widehat{K}_a\varphi \to R_b\psi$ or $K_b\psi \to (\varphi \wedge \widehat{Q}_a\neg\varphi)$. It can also say that a fact φ is an issue for some agents and not for others: $U(Q_a\varphi \vee Q_a\neg\varphi) \wedge \neg U(Q_b\varphi \vee Q_b\neg\varphi)$. It can also describe much more deeply intertwined knowledge and issues, we give more examples in Section 3.2.

Next, we need to bring out the dynamics. We first consider public actions as before. These lead to an immediate generalization of earlier results:

2.3.4. Theorem (Public Multi-Agent $\mathbf{DEL_Q}$ Completeness). *The logic of multi-agent epistemic questions is completely axiomatizable.*

Remark on groups. Our static language only contains issue modalities for single agents. But like in epistemic logic, groups of agents have inquiry-related behavior of their own. Thus, a full treatment would require group notions of information (such as *common* and *distributed* knowledge), and operators for 'collective issues' owned by groups rather than individuals.

2.3.2 Agent-specific Questioning with Preconditions

So far, we only had impersonal public questions among many agents. Now we must consider real questions, as asked by one agent to another. As everywhere in this section, we restrict attention to propositional questions:[9] b asks a: 'Is φ the case?' Thus, real 'agency' enters the picture when we consider the structure of questioning acts. But does this also call for new management actions?

Consider an analogy with public announcement logic PAL. Real speech acts of announcement, too, are agent-relative: 'b tells a that φ'. But one usually performs a reduction here: saying that φ is treated as an impersonal public announcement of the conversational *precondition* of this event.

What that precondition is may depend on one's pragmatic theory. But it will usually involve agent-oriented facts such as: 'b knows φ', and perhaps even: 'b believes that a does not know that φ'.

[9]Extension to other types of questions, such as *Wh-questions* are possible but they would need a predicate-logical version of the static epistemic logics.

Thus, agents come in through impersonal public announcement of agent-dependent preconditions.[10] Some common multi-agent preconditions: (1) 'b asks φ' presupposes $\neg K_b\varphi \wedge \neg K_b\neg\varphi$: that is, the questioner must not know the answer to the question she asks, (2) 'b asks φ to a' presupposes $\widehat{K}_b(K_a\varphi \vee K_a\neg\varphi)$: that is, the questioner must consider it possible that the questionee knows the answer.

These make sense for truly informative Gricean questions, and there may be other preconditions. For instance, asking a question usually suggests that one wants to know the answer. But of course, questions come in many varieties. Rhetorical questions by a teacher to the students do not convey that the teacher does not know the answer, and they definitely do not suggest that the teacher is under any illusions whether the students know the answer.

Such typology of questions: 'plain', 'rhetorical', 'Socratic', is not a task for logic, however. Indeed, a language of the sort we have developed here can formulate lots of different preconditions, and logic is neutral on any choice between them. Our further issue management actions like resolution, refinement, or their parallel execution may also involve multi-agent preconditions.

The only thing our logic needs to do at this stage is keep track of possible preconditions for agent-related questioning actions $\varphi?{}^b_a$: 'b asks φ to a'. We assume that some precondition $\mathtt{pre}(\varphi?{}^b_a)$ is given for this, and likewise, a $\mathtt{pre}(\varphi!{}^b_a)$ for an act of b's saying that φ to a. Then we can formulate management actions and their dynamic logic along lines that are by now standard:

2.3.5. Definition. [Questioning Actions] Executing the $\varphi?{}^b_a$ action in model M gives a new model $M_{\varphi?^b_a} = \langle W_{\varphi?^b_a}, \overset{c}{\sim}_{\varphi?^b_a}, \overset{c}{\approx}_{\varphi?^b_a}, V_{\varphi?^b_a}\rangle$. Likewise, executing the $\varphi!{}^b_a$ action produces a new epistemic-issue model $M_{\varphi!^b_a} = \langle W_{\varphi!^b_a}, \overset{c}{\sim}_{\varphi!^b_a}, \overset{c}{\approx}_{\varphi!^b_a}, V_{\varphi!^b_a}\rangle$, where: $W_{\varphi?^b_a} = W_{\varphi!^b_a} = W$, $V_{\varphi?^b_a} = V_{\varphi!^b_a} = V$ and

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\overset{c}{\sim}_{\varphi?^b_a} & = & \overset{c}{\sim} \cap \overset{\mathtt{pre}(\varphi?^b_a)}{\equiv}_M & \overset{c}{\sim}_{\varphi!^b_a} & = & \overset{c}{\sim} \cap \overset{\mathtt{pre}(\varphi!^b_a)}{\equiv}_M \cap \overset{\varphi}{\equiv}_M \\
\overset{c}{\approx}_{\varphi?^b_a} & = & \overset{c}{\approx} \cap \overset{\varphi}{\equiv}_M & \overset{c}{\approx}_{\varphi!^b_a} & = & \overset{c}{\approx}
\end{array}$$

The 'resolution' and 'refinement' actions remain as before.

In this definition, assertions refine the epistemic relation by means of their preconditions and also by their content. In our link-cutting version, this refines the given relation into four equivalence classes.

To make an analogy with world-eliminating announcements, this behaves as if we are restricting the domain to the worlds satisfying only one formula: the conjunction between the content and the precondition.

[10]The same is true if we add further agent-dependent aspects of taking what is said, such as the *reliability* that a assigns to the source b. This may be reflected in degrees of 'softness' of the signal φ, as in dynamic logics of belief revision (cf. [96]). We will not pursue this here, since reliability seems to play less of a role in the process of raising issues. But similar phenomena would come up if one gave speakers 'authority' in raising issues.

Questions act differently: they refine the issue relation with their content, but they also refine the epistemic relation through the information content of their precondition. Our clauses capture these ideas formally.

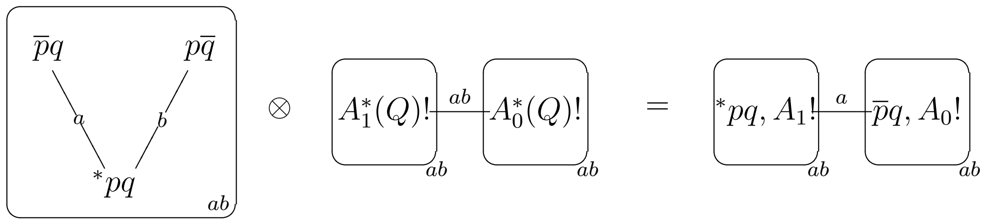


Figure 2.8: Modeling agency by the effects of epistemic preconditions when asking agent-dependent question $(p \wedge q)$? in a multi-agent epistemic-issue environment.

2.3.6. Example. [Solving a Question by Raising It] This multi-agent semantics can deal with interesting phenomena. For instance, the combination of preconditions and issue change can have surprising effects which go beyond mere refinement of the issue relation. In particular, agents can *solve a question by raising it.* Consider the example in Figure 2.8 where by asking a question a simultaneously gives b the answer to the very issue that he raises.

Here agent a asks the question $Q = (p \wedge q)?$ to agent b, with the standard yes/no answers $\{A_0, A_1\}$ and the following combined epistemic preconditions:

$$\texttt{pre}(Q) = \neg(K_a(p \wedge q) \vee \neg K_a(p \wedge q)) \wedge \widehat{K}_a(K_b(p \wedge q) \vee K_b\neg(p \wedge q)),$$

this formula is a precondition for the questioning action, its intuitive meaning captures pragmatic conditions as discussed before: the first conjunct says that the questioner (agent a) does dot know the answer and, the second conjunct, she (agent a) also considers it possible that the questionee (agent b) knows it.

$$\texttt{pre}(A_0(Q)) = \neg(p \wedge q) \wedge \texttt{pre}(Q), \text{ and } \texttt{pre}(A_1(Q)) = p \wedge q \wedge \texttt{pre}(Q).$$

Next we have the two (yes/no) possible answers (0 or false, and 1 or true), they carry their own preconditions, respectively, and the previously explained formula.

Much more complex scenarios can be dealt with, but this will give the flavour for now, see also Section 3.2 for further details.

2.3.3 Dynamic Language, Logic, and Some Design Issues

Now we can add a dynamic language as usual, and interpret it over multi-agent epistemic issue models. This proceeds just as before, in Section 2.2.3

2.3.7. Theorem. *There is a complete dynamic logic with reduction axioms for public multi-agent questions extending the single-agent one of Section 2.2.3.*

The overall completeness argument runs exactly like before, we give the relevant reduction axioms in Section 2.8. A similar completeness theorem for correspondingly modified axioms holds for the usual world-eliminating announcements.

But there are also modifications of our question dynamics that could be studied. For instance, our treatment of questions really amounts to a simultaneous *parallel operation* on epistemic issue models: we change epistemic accessibility and the issue relation at the same time, using the precondition of the question and its content, respectively. An alternative closer to standard systems of DEL might be to do this *sequentially* – but then, for complex questions, updating with the precondition might change the model in a way that affects the subsequent change in the issue relation. The two approaches are not the same. Our feeling is that the parallel version is closer to the intuitive effect of a question.

Moreover, we also think that our system has an independent technical interest, as an example of a new *program operation* in dynamic-epistemic logic: parallel execution of actions on different components of a model.

2.4 Product Update for Multi-Agent DEL$_{\mathrm{Q}}$

Now we move to the real domain of dynamic-epistemic logic: informational actions that involve different observational powers for agents, ranging from differences in public abilities to privacy and hiding. These same phenomena make sense with questions and issue management. There may be differences in agents' powers, and questions can be private just as well as observations. And mixtures can occur, too: a question may be public, but the answer partly private, or an answer may be public, and the question private. Indeed, there are many communicative settings in which subtle distinctions of this sort make sense, and even more so in the setting of epistemic games and competitive scientific research.

The dynamic-epistemic mechanism appropriate to this setting is computing a product update. We include a brief exposition of the method of computing the *product update* between static epistemic models M and event or action models E, in Section 2.7 and refer to the classic paper [9] for further relevant details.

The main idea of performing a product update is to look at epistemic actions as arbitrary events and represent epistemic relations between events in an action model. Such events will have preconditions, telling where (i.e. in which epistemic alternatives) they can occur, or be executed, in our case these are going to be both truthfulness, as with a public announcement, but also questions specific pragmatic considerations, encoded by epistemic formulas.

Then the result of a product update will be a new epistemic structure, in which some actions took place in some worlds, according to their preconditions for execution. This will also define a new space of epistemic possibilities, taking into account both previous uncertainty about the world, encoded in the initial epistemic model, and indistinguishability between the possible epistemic actions, such as questions or possible answers to questions, encoded in the action model.

In what follows, we will merely show that, and how, this method can be extended to include questioning actions and issue management.

The main idea is simply this: we add issue relations to static epistemic models, but also to epistemic event models. It then turns out that the usual product update rule makes sense for both kinds of relation – though there are interesting new points of interpretation and application.

2.4.1 A simple Motivating Example

Before defining a formal counterpart to product update in our new setting, let us discuss what needs to be done in an intuitive way, starting from the basic type of question that we have already studied before.

Consider the simple scenario depicted in Figure 2.9: a public Yes/No question is asked in a single-agent structure. As the relevant events, we have chosen two abstract epistemic signals that model the two possible answers to the question. The epistemic uncertainty relation connects these two events, since the agent does not yet know which one will actually occur. By contrast, the issue relation chosen in this event model does distinguish the answers:

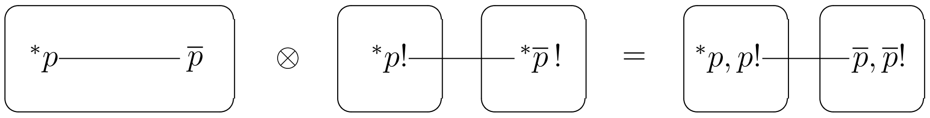


Figure 2.9: Question $p?$ in a single-agent epistemic-issue structure.

As indicated in Figure 2.9, it is easy to see, even without formal definitions yet, that the product update rule gives rise to exactly the new model that we defined earlier in Section 2.1.1. The same is true for the examples we discussed earlier and further ones included and discussed in detail in Section 3.2.

Simple as this proposal looks, it raises some divergence points with DEL.

Observation versus prediction uncertainty In PAL and DEL, epistemic indistinguishability between events represents observational uncertainty of agents. In the present setting, there is yet no observation of the possible future answers, and therefore the epistemic relation rather models *predictive uncertainty.* This is analogous to the situation in games, where 'future ignorance' about a next move is not lack of observational power about present or past moves, but uncertainty about what will happen. Likewise, the issue relation changes its intuitive meaning. The point is not that answer events are issues by themselves, but rather, the issue relation in an event model makes agents aware of possibilities to resolve questions in a future answering move. The issue of awareness also has relevant syntactic aspects. We could call this the *highlighting function* of the issue relation.

Distinguished sets of worlds While the initial epistemic model usually has one actual world, there is no such distinguished world in our questioning-event model. None of the possible answers is yet distinguished at this stage as the real answer. Therefore, we will use a version later where models can have sets of distinguished worlds or sets of distinguished events – a generalization that has also been considered for DEL itself. It is also possible and useful to have both a distinguished set of actual events and an real answer inside that set.

Two kinds of preconditions Answer events have the precondition that their content is true. But in addition, we also saw that questions themselves may have preconditions, such as ignorance of the answer. How should these be represented in the event model? One option is to have the precondition as a separate announcement before the product update, another option (less intuitive, but sometimes more convenient) would be to copy the precondition of the whole question into the preconditions for each answer.

Once we have this simplest setting in place, we want to model more complex cases, that go far beyond what most logics of questions can handle. In particular, some of the phenomena we want to model are the following:

– *Mixtures of prediction uncertainty and observation uncertainty.* An agent knows that some question is asked, but does not know exactly what that question is: it could be either p? or q?.

– *Genuine privacy.* An agent hears the question p?, but another agent thinks that nothing happened.

We now formulate a mechanism that can deal with such scenarios, that often occur in daily life and our natural communicative practice.

2.4.2 Product Update for Questions

It is time now to introduce a more precise formal model that can account for the above described phenomena. In order to do this, we use the following event models enriched with issue structure:

2.4.1. Definition. An *action structure* $\gamma = \langle E, (\overset{a}{\sim})_{a\in A}, (\overset{a}{\approx})_{a\in A}, \mathtt{pre}\rangle$ is a tuple with the following components, where A is a set of agent-labels: E is a set of events (possible future answer events), $\overset{a}{\sim}$ are equivalence relations on E (prediction uncertainty), for $a \in A$, $\overset{a}{\approx}$ are equivalence relations on E (issue highlight relation), for $a \in A$, $\mathtt{pre} : E \to \mathcal{L}$ is a precondition function mapping events $e \in E$ to formulas of the epistemic issue language $\mathcal{L}$. We use pairs (γ, e) for γ an action model and $e \in \wp E$. If e is a singleton we call such pairs *pointed* action structures.

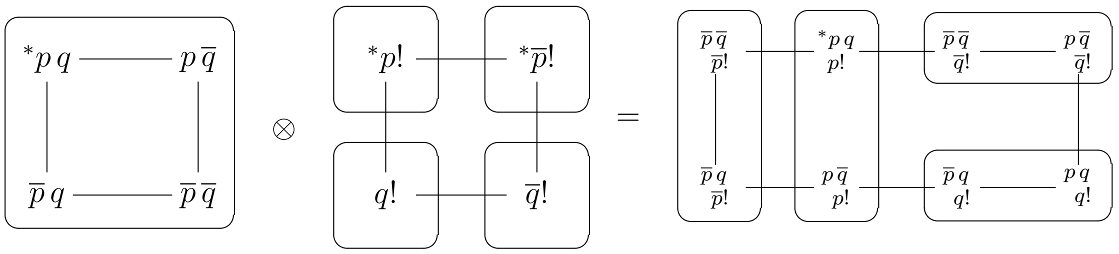
There is a 'parametrization' here to some appropriate language for specifying the preconditions. This language can be a propositional base logic of factual assertions, the richer static epistemic issue language of Section 2.2, or even the full dynamic language $\mathcal{L}_{\mathbf{ELQ_m}}$ below. The latter choice involves a slightly delicate issue of mutual recursion, which is similar to that for DEL in general.

We continue by specifying the way such structures transform given issue epistemic models which provide the underlying changing entities for specific questioning actions. From now on, when convenient and unlikely to cause confusion, we will skip indexing the binary relations and use only $\overset{a}{\sim}$ instead of $(\overset{a}{\sim})_{a \in A}$:

2.4.2. Definition. [Product Update] Given any arbitrary epistemic-issue structure $M = (\langle W, \overset{a}{\sim}, \overset{a}{\approx}, V\rangle, w)$, and action-issue structure $\gamma = (\langle E, \overset{a}{\sim}, \overset{a}{\approx}, \mathtt{pre}\rangle, e)$ the *update product* $M \otimes \gamma = M_{\otimes}^{\gamma} = (\langle W_{\otimes}, \overset{a}{\sim}_{\otimes}, \overset{a}{\approx}_{\otimes}, V_{\otimes}\rangle, (w, e))$ is defined by:

- $W_{\otimes} = \{(w, e) \mid w \in W, e \in E, M \models_w \mathtt{pre}(e)\}$, $V_{\otimes}(w, e) = V(w)$,
- $\overset{a}{\sim}_{\otimes} = \{((w, e), (w', e')) \mid w, w' \in W, e, e' \in E, w \overset{a}{\sim} w', e \overset{a}{\sim} e'\}$,
- $\overset{a}{\approx}_{\otimes} = \{((w, e), (w', e')) \mid w, w' \in W, e, e' \in E, w \overset{a}{\approx} w', e \overset{a}{\approx} e'\}$,

If we need to work with sets of distinguished events, as indicated earlier, we take all pairs of an earlier distinguished world and the (unique) answer event supported by it. The fact that there is a unique event capturing a possible answer depends on how the action model has been set up in the first place. We assume that we work with an adequate action structure, we also show how such structures can be automatically generated in a standard way in Chapter 3. A structure in which two different events can be executed at the real world is allowed by the formalism but will correspond to a questioning action represented by a cover instead of a partition. We discuss these issues at length later in Chapter 3.



$^*p\,q$
$p\,\overline{q}$
$\overline{p}\,q$
$\overline{p}\,\overline{q}$
$\otimes$
$^*p!$
$^*\overline{p}!$
$q!$
$\overline{q}!$
$=$
$\overline{p}\,\overline{q}$ $\overline{p}!$
$^*p\,q$ $p!$
$\overline{p}\,\overline{q}$ $\overline{q}!$
$p\,\overline{q}$ $\overline{q}!$
$\overline{p}\,q$ $\overline{p}!$
$p\,\overline{q}$ $p!$
$\overline{p}\,q$ $q!$
$p\,q$ $q!$

Figure 2.10: Intuitive dynamics for asking more indistinguishable questions.

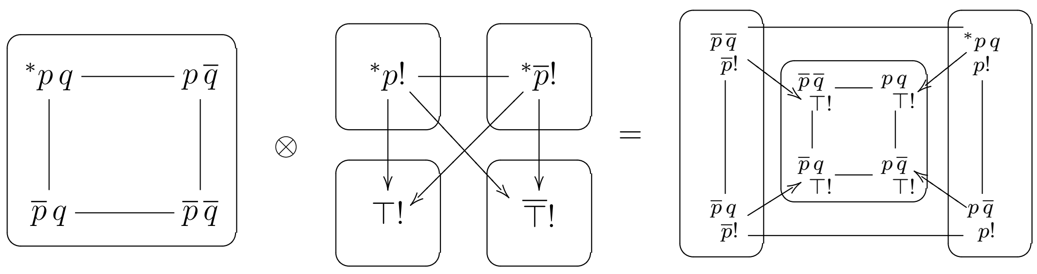
We will now consider some examples demonstrating how this mechanism works. We start by considering our two earlier desiderata:

2.4.3. Example. *Uncertainty about the question.* Figure 2.10 depicts a scenario where a question is asked but, as far as the observing agent is concerned, the content of the question could be either p or q.

This is not yet a multi-agent situation nor does it involve privacy. But it serves its purpose as a prime illustration for the kind of structures we are going to use to represent questions. The first model is a standard issue-epistemic model. The second model is an action model of the type we need to model questioning actions. It contains the possible answers as events of the model.

The third model is the result obtained using the product update rule just described for the first two models. It represents all the possibilities to partition the space of epistemic possibilities given the questioning structure.

2.4.4. Example. *Radical privacy.* Figure 2.11 depicts a question that is not observed by an agent, who thinks that nothing happened. This involves mistaken beliefs of agents, just as in the case of private observations. Accordingly, the accessibility relations might not be reflexive anymore – but as with DEL in general, this changes nothing essential in the product update mechanism.



$^*p\,q$
$p\,\overline{q}$
$\overline{p}\,q$
$\overline{p}\,\overline{q}$
$\otimes$
$^*p!$
$^*\overline{p}!$
$\top!$
$\overline{\top}!$
$=$
$\overline{p}\,\overline{q}$ $\overline{p}!$
$^*p\,q$ $p!$
$\overline{p}\,\overline{q}$ $\top!$
$p\,q$ $\top!$
$\overline{p}\,q$ $\top!$
$p\,\overline{q}$ $\top!$
$\overline{p}\,q$ $\overline{p}!$
$p\,\overline{q}$ $p!$

Figure 2.11: Intuitive dynamics for radical privacy about questions.

This is a good place to discuss how preconditions of answers work during product update. The first action (answer $p!$) is successfully executed in two of the worlds, the remaining ones provide a successful execution for the second event (the complementary answer $\overline{p}!$). Both these events belong to the real question. However, there are two more events in the action structure. One of them (the trivial answer $\top!$) gets executed everywhere, and the remaining one nowhere.

Note that in a certain way the last event is redundant, a model with just the first three events, and the relations between them, could have produced the same effect more efficiently. However, the presence of the $\bot!$ event helps us later on in building relations between questions from relations between answers.

We end with one more example, now involving two agents essentially:

2.4.5. Example. *Privacy in multi-agent scenarios.* Figure 2.12 illustrates how the product update mechanism produces the right result in a scenario where different agents have different information about the content of a public question.

One advantage of the product update mechanism will be that it works for a multi-agent setting of questioning actions just as well as it works in a single-agent setting. In this example we have the same events as before in a single agent

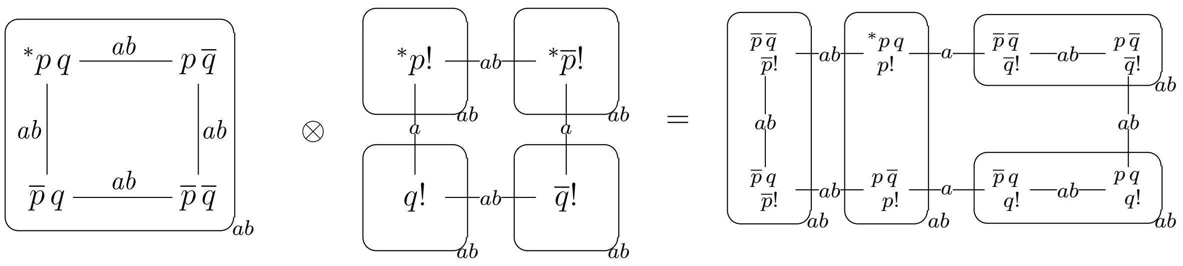


Figure 2.12: Product update for a scenario in which b knows that the content of the question is p? but a considers that the content might have been q?.

setting, but now the epistemic relations between the events are the ones that determine the fact that, after the product update, the epistemic effects for some agents are going to be different, as deriving from the question-action structure.

These examples raise some further issues that we state briefly here.

Special event models for questions. Questions are just a particular type of event, and a more complete treatment might try to capture them as a subclass of the above event models, using the earlier examples as a guide. For instance, it seems plausible that answers to observation-distinguishable questions should be prediction-distinguishable, and further constraints are easily found. We leave a fuller treatment of these issues to future investigation.

Modeling answers as separate events. In Section 3, answers to questions φ? were not modeled separately – though one might think of announcements φ! or $\neg\varphi$! as answer events. In the present setting, an answer event might be modeled explicitly as an event model with its own presuppositions: for instance, the questioner might indicate that he thinks the addressee knows the answer, whereas the answerer communicates that she does know. But also, an answer involves a selection of one possible answer as the actual one, removing the earlier prediction uncertainty, though perhaps still subject to observation uncertainty. It is easy to model such scenarios with the machinery that we have proposed here.

Further issue management actions. Counterparts to our earlier actions of Resolution and Refinement that were introduces in Section 2.2.1 and went beyond simple questions and answers can also be modeled in this style. These require slight adaptations of the present product update rule that we do not pursue here.

The role of syntax: formulation and protocols. Our treatment here has been semantic, identifying propositions with sets of worlds in the usual manner. Thus, the two questions $(p \vee p)$? and p? will have exactly the same effects, and can be considered the same. More generally, our study is not sensitive to differences in output models up to suitably defined *epistemic-issue bisimulation.* This ignores the role of syntactic *formulation* in inquiry, and more modestly, in the appropriateness of answers to a stated question. But actually, the product update mechanism also suggests a more syntactic viewpoint, where the answer events are actual linguistic formulations of answers. This syntactic perspective will return when discussing *protocols* that constrain avenues of investigation.

2.4.3 Dynamic Logics for Product Update

The preceding dynamics can be described in a language extending our systems from previous sections. We proceed in the standard *DEL*-style, we first add dynamic modalities over event models:

2.4.6. Definition. [Dynamic language] Formulas of the *dynamic epistemic issue language* $\mathcal{L}_{\mathbf{ELQ_m}}$ are constructed by the following inductive-syntax rule:

$$i \mid p \mid \bot \mid \neg\varphi \mid (\varphi \wedge \psi) \mid Q_a\varphi \mid R_a\varphi \mid K_a\varphi \mid U\varphi \mid [\gamma?]\varphi \mid [!]\varphi$$

The interpretation of this language is following usual lines. The semantic clauses for boolean and modal formulas are the same as before, the newly added dynamic modalities are defined also as expected:

2.4.7. Definition. [Semantics] Formulas are interpreted with this key clause:

$$M \models_w [\gamma?]\varphi \quad \text{iff} \quad M \otimes \gamma \models_w \varphi, \qquad M \models_w [!]\varphi \quad \text{iff} \quad M_! \models_w \varphi.$$

The resulting dynamic epistemic logic can be axiomatized by standard techniques, in terms of reduction axioms for event modalities:

2.4.8. Theorem. *Any formula in multi-agent logic of questions can be reduced to an equivalent static formula. The key reduction axioms are the following:*

- *Asking & Knowledge:* $[\gamma?]K_a\varphi \leftrightarrow \bigvee_{e\in Q}(\mathtt{pre}(e) \wedge \bigwedge_{\gamma? \overset{a}{\sim} \gamma'?} K_a[\gamma'?]\varphi)$
- *Asking & Partition:* $[\gamma?]Q_a\varphi \leftrightarrow \bigvee_{e\in Q}(\mathtt{pre}(e) \wedge \bigwedge_{\gamma? \overset{a}{\sim} \gamma'?} Q_a(\mathtt{pre}(e) \to [\gamma'?]\varphi))$
- *We also get the axioms involving the resolution modality [!] for free by recalling their corresponding axioms from our previous systems.*

We include the proofs and further discussion of possible extensions of the current framework in Sections 2.8 and 3.2, respectively.

We can now safely say that questions involve the same sort of social issues of privacy and security as informative statements and observations, and our general approach shows how to deal with that in a systematic way.

This allows for a drastic extension of modeling power with regard to what is done in much of the existing literature about questions. Our treatment is by no means complete, but it may show how current logics of questions can undergo the same broadening that epistemic logic has experienced in *DEL*.

2.5 Temporal Protocols in DEL$_{\mathrm{Q}}$

Our final topic in this chapter is another recent theme from dynamic-epistemic logic: longer-term temporal perspective and 'procedural information' (cf. [51]). To make this procedural information explicit, [97] introduces *protocols* in dynamic-epistemic logic. This results in modified versions of *PAL* and *DEL*, which now encode procedural as well as factual and epistemic information. As a technical side-effect, the original reduction axioms no longer do all the work, as procedural information may be irreducible. Completeness proofs become more complex.

But the same considerations apply to questions, perhaps even more so. Single questions usually only make sense in a longer-term temporal perspective of some ongoing inquiry process: an experimental procedure, an ongoing conversation, an information exchange scenario, and so on.

Not everything can be asked, for a variety of reasons such as limitations of measuring instruments, availability of information sources, social conventions or financial resources, and so on. Thus, it makes sense to adapt our dynamic logics to a protocol setting, and we will show how this can be done, while also providing a more realistic theory of inquiry.

We will first present several settings where procedural restrictions on available questions make sense. We do this in some detail, to show a crucial dimension of questions and inquiry that is often ignored in the literature.

Restrictions on types of questions. Clearly, there are strong restrictions on the kinds of question that can be asked in realistic inquiry. We measure 'little things' in science, such as readings of instrument panels, and try to get insights about larger issues. Thus, in a logic setting, we may only be able to ask about atomic questions p?, and not about very complex formulas φ?. The same is true in conversation: we are usually restricted to simple questions that are easy to understand. More generally, we can think of this as a logical hierarchy, from factual questions to epistemic ones ('Did you know that ...?') to complex procedural ones ('What would she say if I asked her what I just asked you?').

These aspects can be illustrated already in the simple setting of propositional logic. Atomic questions may take more time to reach a goal than complex ones.

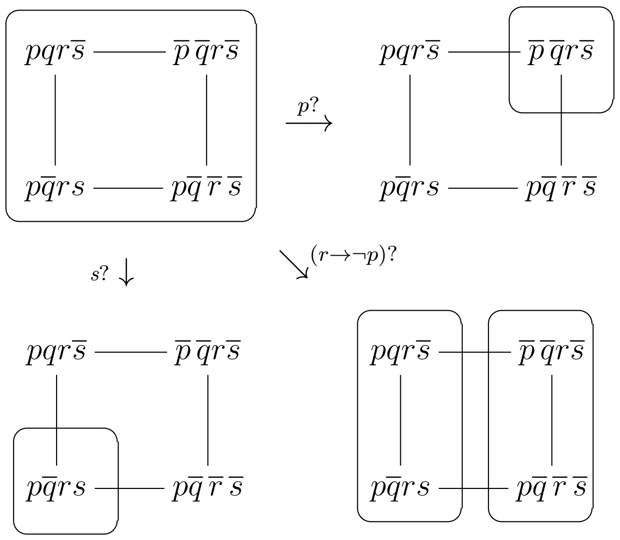

$pqr\overline{s}$
$\overline{p}\,\overline{q}r\overline{s}$
$p\overline{q}rs$
$p\overline{q}\,\overline{r}\,\overline{s}$
$\xrightarrow{p?}$
$s?\downarrow$
$\searrow^{(r\to\neg p)?}$

Figure 2.13: Complex Experiments versus Atomic Questioning

Figure 2.13 gives a very simple illustration. We have here a scenario in which using questions only about propositional atoms has, due to the particular structure of the starting model, different procedural consequences than using unrestricted propositional binary questions. This scenario can lead to an interesting comparison with the usual approch in information theory which measures information bit content in terms of the number of factual Yes/No questions needed to determine the real situation – but it has no restriction to just atomic or other kinds of questions. Thus, one may sometimes need more atomic questions than the usual bit measure would indicate.

Procedural Constraints on Questioning. Even if the available types of question are fixed, there can be further procedural restrictions, having to do, for instance, with access to experimental devices.

Consider a classical mathematical 'Weighing Problem' like the following:

"You have 9 pearls, 1 lighter than the others that are all of equal weight. You can weigh only 2 times with a standard balance. Find the lighter pearl."

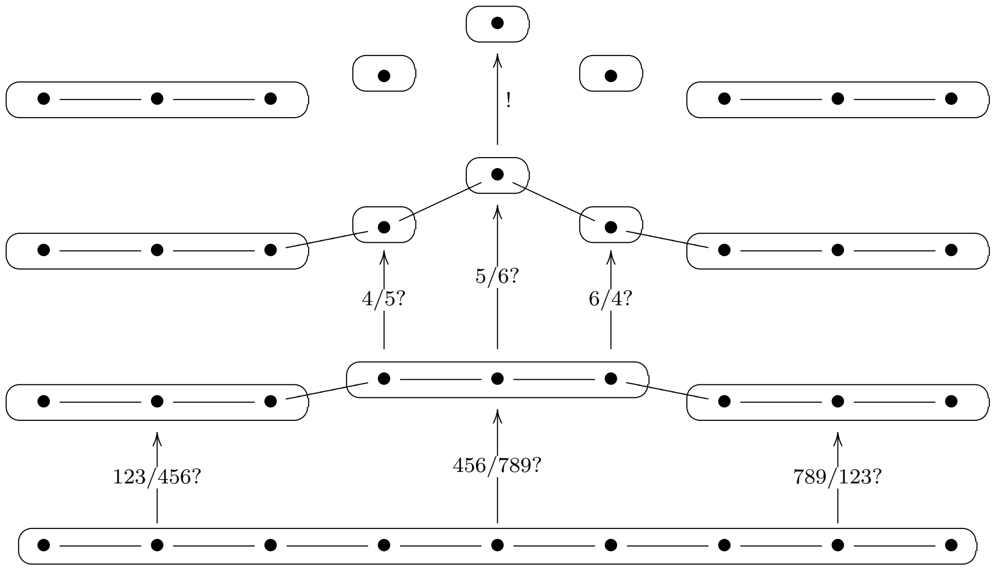


Figure 2.14: Procedural restrictions in the number of available questions

The allowed questions are fixed in this scenario: we can only ask Nature for some atomic balance facts. The resulting process can be pictured in an 'event tree' of possible histories of successive weighings. Figure 2.14 illustrate a few successful evolutions, but there could also be 'bad' ones. For instance, asking (7/8)? after (789/123)? is a history that fails to solve the problem in two steps, and hence it would be a bad experimental procedure. Protocols encode conditional information like 'If measuring 123/789 turns out 789 then measure 7/8, else, if it turns out 123, then measure 2/3, else measure 4/6'. The solution to this weighing problem can be formulated as an interaction between questioning actions and knowledge after resolution: "Weigh 3 pearls against 3, leaving 3 pearls aside. If the result is equal, then weigh 2 of the remaining 3 pearls against each other. If these are equal, then the remaining pearl is the lighter one – if not, then you know which one was lightest. If the initial result was unequal, apply the previous procedure to the lighter group of 3: weigh 2 of these, and you will know."

Restrictions from epistemic properties of inquiry. Even if types of questions and procedural restrictions are fixed, there may still be further relevant restrictions in multi-agent scenarios. In *epistemic games*, for instance, it might be of crucial importance in what temporal order new knowledge is obtained. If the competition finds out the actual world ahead of time, there is no incentive for sharing resources in a social experimental procedure. A further analysis and concrete illustrations of such scenarios are included in Chapter 4.

2.6 Long Term Protocols in Inquiry

Some formalities. The preceding examples can be formalized in an epistemic question protocol logic that follows the general methodology from [97] and [51]:

2.6.1. Definition. [DEL$_\mathrm{Q}$ Protocol] Let Σ be an arbitrary set of epistemic events (management actions). Let Σ^* be the set of finite strings over Σ (finite histories of questioning events). A questioning protocol is a set $\mathcal{H} \subseteq \Sigma^*$ such that: $\mathsf{FinPre}_{-\lambda}(\mathcal{H}) = \{h \mid h \neq \lambda, \exists h' \in \mathcal{H} : h \preceq h'\} \subseteq \mathcal{H}$

Next, we can set up a formal language over these structures. In the truth definition, dynamic modalities now state that there exists a next informational or issue action *available at this stage by the current protocol* whose execution makes the postcondition true.

The construction in the following definition considers only sequences $w\sigma$ such that w is a world in the domain of the initial model M, and σ, σ' are sequences in the state dependent protocol $Q(w)$; σ_n denotes the sequence σ up to its n-th position and $\sigma_{(n)}$ denotes the n-th element in the sequence.

2.6.2. Definition. [Q-Generated Model] Let $M = \langle W, \sim, \approx, V\rangle$ be an arbitrary model and let Q be an arbitrary DEL$_\mathrm{Q}$-protocol over M (a prefix-closed set of finite sequences of questioning events). The Q-Generated Model at level n, $M^n_Q = \langle W^n_Q, \sim^n_Q, \approx^n_Q, V^n_Q\rangle$ is defined by induction on n as follows:[11]

1 $W^0_Q = W$, $\sim^0_Q = \sim$, $\approx^0_Q = \approx$, $V^0_Q = V$;

2 $w\sigma \in W^{n+1}_Q$ iff $w \in \mathrm{dom}(M)$, $\sigma \in Q(w)$, $\mathtt{len}(\sigma) = n+1$, and $w\sigma_n \in W^n_Q$;

3 If $\sigma_{(n+1)} = \varphi?$ then: (a) $(w\sigma, v\sigma') \in \sim^{n+1}_Q$ iff $\sigma_{(n+1)} = \sigma'_{(n+1)}$, and $(w\sigma_n, v\sigma'_n) \in \sim^n_Q$; (b) $(w\sigma, v\sigma') \in \approx^{n+1}_Q$ iff $\sigma_{(n+1)} = \sigma'_{(n+1)}$, $(w\sigma_n, v\sigma'_n) \in \approx^n_Q$, and $(w\sigma_n, v\sigma'_n) \in \overset{\varphi}{\equiv}_{M^n_Q}$;

[11]We include here only questions and resolution actions to save some space. It should be emphasized that other types of management actions can be considered. For instance here is how clauses for announcement and refinement look like:

If $\sigma_{(n+1)} = \varphi!$ then: (a) $(w\sigma, v\sigma') \in \approx^{n+1}_Q$ iff $\sigma_{(n+1)} = \sigma'_{(n+1)}$ and $(w\sigma_n, v\sigma'_n) \in \approx^n_Q$; (b) $(w\sigma, v\sigma') \in \sim^{n+1}_Q$ iff $\sigma_{(n+1)} = \sigma'_{(n+1)}$, $(w\sigma_n, v\sigma'_n) \in \sim^n_Q$, and $(w\sigma_n, w\sigma'_n) \in \overset{\varphi}{\equiv}_{M^n_Q}$;

If $\sigma_{(n+1)} = ?$ then: (a) $(w\sigma, v\sigma') \in \sim^{n+1}_Q$ iff $\sigma_{(n+1)} = \sigma'_{(n+1)}$, and $(w\sigma_n, v\sigma'_n) \in \sim^n_Q$; (b) $(w\sigma, v\sigma') \in \approx^{n+1}_Q$ iff $\sigma_{(n+1)} = \sigma'_{(n+1)}$, $(w\sigma_n, v\sigma'_n) \in \approx^n_Q$, and $(w\sigma_n, v\sigma'_n) \in \sim^n_Q$;

4 If $\sigma_{(n+1)} = !$ then: (a) $(w\sigma, v\sigma') \in \approx_Q^{n+1}$ iff $\sigma_{(n+1)} = \sigma'_{(n+1)}$, and $(w\sigma_n, v\sigma'_n) \in \approx_Q^n$; (b) $(w\sigma, v\sigma') \in \sim_Q^{n+1}$ iff $\sigma_{(n+1)} = \sigma'_{(n+1)}$, $(w\sigma_n, v\sigma'_n) \in \sim_Q^n$, and $(w\sigma_n, v\sigma'_n) \in \approx_Q^n$;

5 For each $a \in \mathtt{P} \uplus \mathtt{N}$, $V_Q^{n+1}(a) = \{w\sigma \mid w\sigma \in W_Q^{n+1}, w \in V(a)\}$.

The class of structures $\mathsf{Forest}(\mathrm{TDEL_Q})$ consists of all models $\mathsf{Forest}(M, Q)$ for some arbitrary model M and some arbitrary $\mathrm{TDEL_Q}$ protocol Q.

We can now define the corresponding temporal structure Using the construction from the previous Definition 2.6.2.

2.6.3. Definition. [Generated ETL Model] Let $M = \langle W, \sim, \approx, V\rangle$ be an arbitrary epistemic-issue model and let Q be an arbitrary $\mathrm{DEL_Q}$-protocol over M (a set of finite sequences of questioning events closed under initial segments). The Generated ETL Model $\mathsf{Forest}(M, Q) = \langle \mathsf{H}, \sim, \approx, V'\rangle$ is defined as follows:

1 $\mathsf{H} = \{h \mid \text{there is a } w \in W, \sigma \in Q \text{ with } h = w\sigma \in W_Q^{\mathtt{len}(\sigma)}\}$;

2 For all histories $h, h' \in \mathsf{H}$ with $h = w\sigma$ and $h' = v\sigma'$, $h \sim h'$ iff $\mathtt{len}(\sigma) = \mathtt{len}(\sigma')$, and $(w\sigma, v\sigma') \in \sim_Q^{\mathtt{len}(\sigma)}$;

3 For all histories $h, h' \in \mathsf{H}$ with $h = w\sigma$ and $h' = v\sigma'$, $h \approx h'$ iff $\mathtt{len}(\sigma) = \mathtt{len}(\sigma')$, and $(w\sigma, v\sigma') \in \approx_Q^{\mathtt{len}(\sigma)}$;

4 For each $a \in \mathtt{P} \uplus \mathtt{N}$, and $h = w\sigma \in \mathsf{H}, h \in V'_Q(a)$ iff $h \in V_Q^{\mathtt{len}(\sigma)}(a)$.

Next we give a truth definition for a matching dynamic language, with dynamic actions involving formulas from the static base language only.

2.6.4. Definition. [Interpretation] The truth definition of formulas at state h in model $\mathsf{Forest}(M, \mathcal{Q}) := \mathsf{Fr}(M, \mathcal{Q}) = \langle \mathcal{H}, \sim, \approx, V\rangle$ uses the following key inductive clauses for the modal operators:

- $\mathsf{Fr}(M, \mathcal{Q}) \models_h K\varphi$ iff $\forall h' \in \mathcal{H} : h \sim h'$ implies $\mathsf{Fr}(M, \mathcal{Q}) \models_{h'} \varphi$;
- $\mathsf{Fr}(M, \mathcal{Q}) \models_h Q\varphi$ iff $\forall h' \in \mathcal{H} : h \approx h'$ implies $\mathsf{Fr}(M, \mathcal{Q}) \models_{h'} \varphi$;
- $\mathsf{Fr}(M, \mathcal{Q}) \models_h R\varphi$ iff $\forall h' \in \mathcal{H} : h\,(\sim \cap \approx)\,h'$ implies $\mathsf{Fr}(M, \mathcal{Q}) \models_{h'} \varphi$;
- $\mathsf{Fr}(M, \mathcal{Q}) \models_h \langle q\rangle\varphi$ iff $hq \in \mathcal{H}$ and $\mathsf{Fr}(M, \mathcal{Q}) \models_{hq} \varphi$;
- $\mathsf{Fr}(M, \mathcal{Q}) \models_h U\varphi$ iff $\forall h' \in \mathcal{H} : h = w\sigma,\ h' = w'\sigma',\ w, w' \in \mathrm{Dom}(M)$, $\sigma, \sigma' \in \mathcal{Q}$, and $\sigma = \sigma'$ implies $\mathsf{Fr}(M, \mathcal{Q}) \models_{h'} \varphi$.

Where q is used as a variable over all issue management actions.

In order to obtain the completeness result we will proceed by the standard Henkin construction of a canonical model. We start constructing the canonical ETL model M^c by first defining the base level $M_0^c = \langle H_0, \overset{a}{\approx}_0, \overset{a}{\sim}_0, V_0\rangle$ using the usual definitions as follows:

- $H_0 = \{w \mid w \text{ is a maximally consistent set of formulas}\}$,
- For each $w, v \in H_0$, let $w \overset{a}{\approx}_0 v$ if and only if $\{\varphi \mid Q_a\varphi \in w\} \subseteq v$,
- For each $w, v \in H_0$, let $w \overset{a}{\sim}_0 v$ if and only if $\{\varphi \mid K_a\varphi \in w\} \subseteq v$,
- For each $p \in \mathtt{P}$ and $w \in H_0$, $w \in V_0(p)$ iff $p \in w$.

2.6.5. Definition. The canonical model $M^c = \langle H^c, \overset{a}{\sim}{}^c, \overset{a}{\approx}{}^c, V^c\rangle$ is defined by:

- $H^c = \bigcup_{i=0}^{\infty} H_i$;
- For each $h, h' \in H^c$ with $h = w\sigma$ and $h' = v\sigma'$, let $h \overset{a}{\approx}{}^c h'$ if and only if $\sigma = \sigma'$ and $w \overset{a}{\approx}_0 v$;
- For each $h, h' \in H^c$ with $h = w\sigma$ and $h' = v\sigma'$, let $h \overset{a}{\sim}{}^c h'$ if and only if $\sigma = \sigma'$ and $w \overset{a}{\sim}_0 v$;
- For every $a \in \mathtt{P} \uplus \mathtt{N}$ and $h = w\sigma \in H^c$, $w\sigma \in V^c(a)$ iff $w \in V_0(a)$.

2.6.6. Definition. [Legal Histories] Let W_0 be the set of all $TDEL_Q$ maximal consistent sets. We define λ_n and $H_n (0 \le n \le d(\Sigma))$ as follows:

- Set $H_0 = W_0$, and for each $w \in H_0$, set $\lambda_0(w) = w$.
- Let $H_{n+1} = \{hq \mid h \in H_n \text{ and } \langle q\rangle\top \in \lambda_n(h)\}$ and
  let $\lambda_{n+1}(h) = \{\varphi \mid \langle q\rangle\varphi \in \lambda_n(h')\}$, for each $h = h'q \in H_{n+1}$.

We need to check that each map λ_n is well defined:

2.6.7. Lemma. *For each $n \ge 0$, for each $w \in H_n$, $\lambda_n(\sigma)$ is maximally consistent.*

2.6.8. Lemma (Truth Lemma). *For every $\varphi \in \mathcal{L}_{\mathbf{EL_Q}}(\mathtt{P}, \mathtt{N}, A)$ and any $h \in H^c$:*

$$\varphi \in \lambda(h) \quad \textit{if and only if} \quad M^c \models_h \varphi.$$

2.6.9. Theorem (**TDEL$_\mathbf{Q}$** Completeness). *The dynamic epistemic question logic $TDEL_Q$ of protocol models is completely axiomatizable.*

The proof of this result follows the standard Henkin construction of a canonical model from maximally consistent sets of formulae. This method has been used before for PAL and DEL in [97], [51] and [22].

Some crucial axioms for $TDEL_Q$ are in this setting *Questions & Partition*:

$$\langle\varphi?\rangle Q\psi \leftrightarrow \langle\varphi?\rangle\top \wedge ((\varphi \wedge Q(\varphi \to \langle\varphi?\rangle\psi)) \vee (\neg\varphi \wedge Q(\neg\varphi \to \langle\varphi?\rangle\psi)))$$

Resolution & Knowledge: $\langle !\rangle K\varphi \leftrightarrow \langle !\rangle\top \wedge R\langle !\rangle\varphi$

The main difference with the earlier reduction axioms is given by the presence of conjuncts like $\langle\varphi?\rangle\top$. These now express the fact that the questioning action described is available according to the protocol. This procedural information may be *sui generis*, and not equivalent to any simple precondition formula in the underlying static language. This information encodes now epistemic aspects, as for *PAL* and *DEL*, but also dynamics of issue structure.

In our current setting the proof needs to be more elaborated because:

– Unlike in the setting of PAL and DEL, we work not with one but with four distinct dynamic modalities.
– The possible sequences of dynamic actions (or legal histories) have a more complex structure as they are constructed by sequential compositions of all these four dynamic modalities.

– Unlike in the setting of PAL and DEL, these new modalities change not one binary relation but two distinct ones plus their intersection.
– The static language has formulas containing modalities describing both the issue structure and the epistemic structure on one hand, and also their mutual interdependence via the resolution modality, on the other hand.

Despite this variety, an adapted version of the proof can be obtained in an analogous way using the following simplifications:

– We will only consider two relevant modalities $[\varphi?]$ and $[!]$, and note that the remaining ones have a completely symmetrical behavior.
– We will only consider the non-commuting modalities. Because relevant modalities have a segregated effect on the components of the static structures, some modalities are unchanged by some dynamic actions. These have a merely commutating behavior as the underlying relation is not changed by the given dynamic action and can be safely ignored.
– Given the equalities presented earlier in 2.2.1, some sequences obtained by sequential composition can be shown to have a redundant effect. In order to simplify the proof we will always collapse such redundant sequences into one single dynamic action. For instance, the sequence !!! is equivalent to !! and ! as far as their their dynamic effects are concerned.
– We will build the proof componentwise, first for the main dynamic modality of questioning $[\varphi?]$, which will only require the choice of the appropriate disjunct in a reduction formula as an additional step in the proof, and the resolution modality $[!]$, for which the intersection modality will be used.

All these aspects show how the dynamic logics of questions suddenly get a much richer field of study than single linguistic speech acts, or information-theoretic question scenarios where anything can be asked and answered. We will further pursue the idea of designing good questioning strategies in inquiry and problem solving considering a concrete example in Chapter 6 of this thesis.

We have shown that dynamic logic of questions can easily accommodate global temporal protocols, thus getting closer to a true logic of inquiry. Of course, the main interest is not in our general completeness theorem, but in analyzing concrete protocols, and more detailed logical effects of various restricted question repertoires. The following chapters will further develop these ideas.

Conclusions and Further Topics In this chapter, we have shown how dynamic logics of questions can analyze various aspects of private and public inquiry. The main contributions we have made so far are the following:

- A rich system of dynamic issue-management actions
- Complete dynamic logics for questioning in DEL style,
- Extension to privacy and product update for questions,
- Extension to temporal protocols for inquiry.

These systems fit entirely within the methodology of dynamic-epistemic logic, and they seem to form a natural complement to what already exists in this area, making the the actions of questioning and resolution explicit in information flow.

2.7 Appendix A: Background Definitions

The theoretical background on which the static epistemic-issue logic is based is that of hybrid logics. In particular here are two general results which are applied in the proof of completeness of the static logic:

2.7.1. Theorem (5.2.10 in [85]). *Every pure $\mathcal{H}$-formula is di-persistent. Conversely, every di-persistent $\mathcal{H}$-formula defines the same class of discrete general frames as a pure $\mathcal{H}$ formula. The same holds for $\mathcal{H}(@)$ and $\mathcal{H}(\mathsf{E})$.*

Pure formulas are formulas that only contain nominals and no proposition letters.

2.7.2. Corollary (5.4.1 in [85]). *Let Σ be any set of pure formulas, very simple modal Sahlqvist formulas and/or shallow modal formulas of $\mathcal{H}(@)$. Then $\mathbf{K}^{+}_{\mathcal{H}(@)}\boldsymbol{\Sigma}$ is strongly complete for the class of frames defined by Σ. Similar for $\mathbf{K}^{+}_{\mathcal{H}}\boldsymbol{\Sigma}$ and $\mathbf{K}^{+}_{\mathcal{H}(\mathsf{E})}\boldsymbol{\Sigma}$.*

Using these general results completeness of our static logic of questions and knowledge follows as a particular application:

2.7.1. Proof (Theorem 2.1.6). We only have to show that the intersection axiom: $\langle\sim\rangle i \wedge \langle\approx\rangle i \leftrightarrow \langle\sim\cap\approx\rangle i$ is a pure formula. The rest of the proof is an application of the more general result (2.7.2 in [85]) to the particular case of an extension of the basic hybrid logic with pure formulas. □

Complexity of Static Logic The main novelty of our static epistemic logic is the use of the intersection modality between issues and knowledge. The intersection modality has been studied before in the DEL literature but only for one relations in the context of group epistemic notions like general knowledge within a group. Already in a setting with only one kind of relation the intersection modality raises technical difficulties as it is not invariant under bisimulation. Using intersection between two different kinds of binary relations gave us the advantage of being able to express interesting connections between questioning actions and knowledge evolution, but also brought all the technical difficulties with it.

To solve these technical aspects we have to use an extended modal logic. Using a hybrid logic with nominals is enough to define the intersection modality as we show in Section 2.1.2. One natural question to ask in this context is: what is the computational price of this gain in expressive power? The answer is available in the standard hybrid logic literature: the K-satisfaction problem is PSPACE-complete, just like the standard modal logic (up to a polynomial).

Using a universal modality provides even more expressive power but raises the complexity from PSPACE to EXPTIME. However the logic still remains decidable as long as our modalities do not need to satisfy the commutation axiom:

$[\sim][\approx]\varphi \leftrightarrow [\approx][\sim]\varphi$. Even so, in many applications which are relevant for describing inquiry in general many interesting facts can be expressed even without the universal modality (see the extensive questioning games studied in Chapter 4).

The step towards undecidability is only made if the logic is enriched with the down-arrow binder. However, in none of the applications that we will consider in the following chapters this will be necessary.

Overall, the complexity profile of hybrid logic is very appealing for a large variety of inquiry and questioning related applications. The task which is most of the time enough for describing information flow via questioning is model checking of formulas involving the intersection modality.

The DEL Methodology The background method used for completeness of the various dynamic logics proposed in this chapter is standard in the DEL literature [9, 24, 93]. We will illustrate below for the reduction axioms in Definition 4.4.7 the major steps of the DEL methodology used several times in this chapter.

2.7.3. Definition. [Translation] The translation of $\mathrm{DEL_Q}$ formulas is given by: $t(p) = p, t(\neg\varphi) = \neg t(\varphi), t(\varphi \wedge \psi) = t(\varphi) \wedge t(\psi), t(\Box_a\varphi) = \Box_a t(\varphi)$, for $\Box_a \in \{K_a, Q_a, R_a\}, t(lhs) = t(rhs)$ for axioms in 4.4.7.

2.7.4. Definition. [Complexity] The complexity of $\mathrm{DEL_Q}$ formulas is given by: $c(p) = 1, c(!) = 1, c(\neg\varphi) = 1 + c(\varphi), c(\varphi \wedge \psi) = 1 + \max(c(\varphi), c(\psi)), c(\Box_a\varphi) = 1 + c(\varphi)$, for $\Box_a \in \{K_a, Q_a, R_a\}, c([q]\psi) = (c(q) + 5) \cdot c(\psi)$, for $q \in \{\varphi?, !\}$.

2.7.5. Lemma (Complexity properties). *For any DEL_Q formulas φ and ψ:*

1. $c(\varphi) \geq c(\psi)$ *if* $\psi \in Sub(\varphi)$

2. $c(lhs) > c(rhs)$ *for axioms in* 4.4.7.

2.7.2. Proof. By induction on ψ. □

2.7.6. Lemma (Translation). *For all DEL_Q formulas we have:* $\vdash \varphi \leftrightarrow t(\varphi)$

2.7.3. Proof. By induction on $c(\varphi)$, using Lemma 2.7.5 and Theorem 2.2.9. □

2.7.7. Theorem (Completeness). *For all DEL_Q formulas it is the case that:*

$$\models \varphi \text{ implies } \vdash \varphi$$

2.7.4. Proof. Suppose $\models \varphi$, therefore, using Theorem 2.2.9 (soundness) together with Lemma 2.7.6 (translation correctness), we know that $\models t(\varphi)$. Hence, by Theorem 2.1.6 (completeness of the static fragment), it follows that $EL_Q \vdash t(\varphi)$. Because $\mathrm{EL_Q}$ is contained in $\mathrm{DEL_Q}$ as a subsystem, we also get that $DEL_Q \vdash t(\varphi)$. And, using Lemma 2.7.6 (translation correctness) $DEL_Q \vdash t(\varphi) \leftrightarrow \varphi$, we obtain that $DEL_Q \vdash \varphi$, as desired. □

This general methodology works in an analogous way for instances of recursive reduction axioms introduced in the present chapter.

Complexity of Dynamic Logic Because reasoning in the the dynamic logic is reduced by the above translation axioms to the static logic, the complexity of the dynamic fragment depends on the properties of the translation. The logic for which complexity results are known is PAL. Therefore we will evaluate the complexity profile of DEL_Q on the background of existing results about PAL [67].

The equivalence-preserving translation used so far cannot avoid an exponential blow-up in formula size even for certain PAL formula instances. However, in the standard literature [67] this is also regarded as a positive quality that allows a language to express the same sentences in a more *succinct* manner. Compared to PAL, formulae in which the main operator is a question modality are translated into a disjunction of two formulae. This makes the blowup in formula size even more dramatic for DEL_Q. In the same time, this can also be seen twice as much as the positive quality of *succinctness* that DEL_Q inherits from results about PAL. The dynamic modalities introduced by DEL_Q allow to express certain properties in a more succinct manner than the static epistemic-issue language.

A different translation [67], which is only preserving satisfiability, does avoid an exponential blowup. This is because in a recursive translation there are exponentially many recursive translation calls but, because some of them have the same content, there are only polynomially many *relevant* translation calls, i.e. clauses using distinct sub-formula instances. This ensures that reasoning in the dynamic fragment stays inside the PSPACE bound of the static epistemic logic. Because both disjuncts of a yes/no question translation formula have a symmetric syntactic structure, the number of relevant translation calls is only increased by a constant. Therefore, reasoning in the dynamic logic of yes/no questions also remains inside the previous bounds for the static issue-epistemic logics.

Product Update The standard sources for product update are [9, 24, 93]. Here we only recall the basics in a nutshell. An event model E consists of a set of relevant events related by epistemic uncertainty links that encode agents' observational powers. Moreover, each event comes with a 'precondition' stating just when it can occur: these drive the information flow when events are observed.

This shows in forming a *product model* $\mathsf{M} \times \mathsf{E}$ consisting of all pairs (s, e) of old worlds $s \in \mathsf{M}$ that satisfy the precondition for event e. The new knowledge of agents is encoded in the new uncertainty relation between pairs, constructed from old uncertainly and event indistinguishability by the 'product rule':

$$(s, e) \sim (t, f) \quad \text{iff} \quad s \sim t \text{ and } e \sim f.$$

Finally, the valuation for proposition letters at (s, e) remains the same as in s: at least in the simple *DEL* systems that we discuss here, there is no factual change.

2.8 Appendix B: Proofs of Main Results

2.8.1. Proof (Theorem 2.2.9). By standard modal arguments. We discuss two cases that go beyond mere commutation of operators.

The first (*Asking & Partition*) explains how questions refine a partition:

$$[\varphi?]Q\psi \leftrightarrow (\varphi \wedge Q(\varphi \rightarrow [\varphi?]\psi)) \vee (\neg\varphi \wedge Q(\neg\varphi \rightarrow [\varphi?]\psi))$$

From left to right. Assume that $M \models_w [\varphi?]Q\psi$, then we also have $M_{\varphi?} \models_w Q\psi$. In case $M \models_w \varphi$, the new issue relation locally refined the old one to φ-worlds, and hence we get the left-hand disjunct on the right. The other case yields the right-hand disjunct. *From right to left.* Properly viewed, the preceding explanation already established an equivalence.

Our second illustration (*Resolving & Knowledge*) shows how a resolution action changes knowledge, making crucial use of the intersection modality:

$$[\,!\,]K\varphi \leftrightarrow R[\,!\,]\varphi$$

$M \models_w [\,!\,]K\varphi$ is equivalent to $M_! \models_w K\varphi$, which is equivalent to $\forall v \in W_! : w \sim_! v$ implies $M_! \models_v \varphi$. As $\sim_! = \sim \cap \approx$,the semantics of our dynamic modality tells us that $\forall v \in W : w\,(\sim \cap \approx)\,v$ implies $M \models_v [\,!\,]\varphi$, which is equivalent to $M \models_w R[\,!\,]\varphi$, as desired.

The rest of the proof proceeds exactly like before along the standard DEL methodology. □

2.8.2. Proof (Theorem 2.3.4). The proof consists in suitable agent labeling of earlier reduction axioms. We illustrate this generalization with the indexed versions of two previous axioms, e.g. *Answer & Knowledge*, *Asking & Partition*:

$$[\varphi!]K_a\psi \leftrightarrow (\varphi \wedge K_a(\varphi \rightarrow [\varphi!]\psi)) \vee (\neg\varphi \wedge K_a(\neg\varphi \rightarrow [\varphi!]\psi)),$$

$$[\varphi?]Q_a\psi \leftrightarrow (\varphi \wedge Q_a(\varphi \rightarrow [\varphi?]\psi)) \vee (\neg\varphi \wedge Q_a(\neg\varphi \rightarrow [\varphi?]\psi)).$$

The rest of the proof proceeds like before along the DEL methodology. □

2.8.3. Proof (Theorem 2.3.7). We display here just three key reduction axioms: – *Asking & Knowledge*, where $\chi = \texttt{pre}(\varphi?\,^b_a)$:

$$[\varphi?^b_a]K_c\psi \leftrightarrow (\chi \wedge K_c(\chi \rightarrow [\varphi?^b_a]\psi)) \vee (\neg\chi \wedge K_c(\neg\chi \rightarrow [\varphi?^b_a]\psi))$$

– *Asking & Intersection*: $[\varphi?^b_a]R_c\psi \leftrightarrow \bigvee_{i\in\{0..3\}}\{\chi_i \wedge R_c(\chi_i \rightarrow [\varphi?^b_a]\psi)\}$ where $\chi_i \in \{\texttt{pre}(\varphi?\,^b_a) \wedge \varphi, \neg\texttt{pre}(\varphi?\,^b_a) \wedge \varphi, \texttt{pre}(\varphi?\,^b_a) \wedge \neg\varphi, \neg\texttt{pre}(\varphi?\,^b_a) \wedge \neg\varphi\}$

– *Announcement & Knowledge*: $[\varphi!^b_a]K_c\psi \leftrightarrow \bigvee_{i\in\{0..3\}}\{\chi_i \wedge K_c(\chi_i \rightarrow [\varphi!^b_a]\psi)\}$, where $\chi_i \in \{\texttt{pre}(\varphi!\,^b_a) \wedge \varphi, \neg\texttt{pre}(\varphi!\,^b_a) \wedge \varphi, \texttt{pre}(\varphi!\,^b_a) \wedge \neg\varphi, \neg\texttt{pre}(\varphi!\,^b_a) \wedge \neg\varphi\}$

The rest of the proof proceeds exactly like before along the standard DEL methodology. □

2.8.4. Proof (Theorem 2.4.8). The proof follows the lines of the standard DEL methodology described in the previous section with the only caveat that now we need to define a complexity measure for formulae containing questioning action modalities by taking the complexity of the preconditions.

The proof also makes use of a definition that will be introduced in Section 3.2, we insert the proof after the needed notions are explained and discussed in detail in Section 3.2. □

2.8.5. Proof (Lemma 2.6.7). Induction on n. For (base case) $n = 0$ it is the case by definition. Suppose (induction hypothesis) that the statement holds for H_n and λ_n. Take (induction step) $\sigma \in H_{n+1}$ with $\sigma = \sigma' q$. By the induction hypothesis, $\lambda_n(\sigma')$ is a maximally consistent set. Moreover, by the construction of H_{n+1}, $\langle q\rangle\top \in \lambda_n(\sigma)$, hence $\lambda_{n+1}(\sigma) \neq \emptyset$. For arbitrary $\varphi \in \mathcal{L}_{TDEL_Q}$, as $\lambda_n(\sigma')$ is a maximally consistent set, either $\langle q\rangle\varphi \in \lambda_n(\sigma')$ or $\neg\langle q\rangle\varphi \in \lambda_n(\sigma')$. If $\langle q\rangle\varphi \in \lambda_n(\sigma')$, then $\varphi \in \lambda_{n+1}(\sigma)$ by construction. If $\neg\langle q\rangle\varphi \in \lambda_n(\sigma')$, then, by the *Neg.* axiom we have $\langle q\rangle\neg\varphi \in \lambda_n(\sigma')$. Thus, $\neg\varphi \in \lambda_{n+1}(\sigma)$ by construction. As φ was arbitrary, we have that for all $\varphi \in \mathcal{L}_{TDEL_Q}$, either $\varphi \in \lambda_{n+1}(\sigma)$ or $\neg\varphi \in \lambda_{n+1}(\sigma)$.

To show that λ_{n+1} is consistent, suppose towards contradiction that there are formulas $\varphi_1, ..., \varphi_m \in \lambda_{n+1}(\sigma)$ such that $\vdash \bigwedge_{i=1}^{m} \varphi_i \to \bot$. Using standard modal reasoning, $\vdash \langle q\rangle\top \to \bigvee_{i=1}^{m}\langle q\rangle\neg\varphi_i$. Since $\langle q\rangle\top \in \lambda_n(\sigma')$, it folllows that $\bigvee_{i=1}^{m}\langle q\rangle\neg\varphi_i \in \lambda_n(\sigma')$. As $\lambda_n(\sigma')$ is a maximally consistent set, there is some j with $1 \leq j \leq m$ and $\langle q\rangle\neg\varphi_j \in \lambda_n(\sigma')$. Using the *Neg.* axiom we have $\neg\langle q\rangle\varphi_j \in \lambda_n(\sigma')$. By construction of $\lambda_{n+1}(\sigma)$ we have $\langle q\rangle\varphi_i \in \lambda_n(\sigma')$ for each $i = 1, \ldots, m$, which contradicts the fact that $\lambda_n(\sigma')$ is consistent. □

2.8.6. Proof (Lemma 2.6.8). The proof needs a definition that will be introduced in Section 3.2, we insert the proof after the needed notions are explained and discussed in detail in Section 3.2. □

2.8.7. Proof (Theorem 2.6.9). The proof follows by a standard argument from Lemma 2.6.8 and the fact that the canonical model is in the class of intended models. □

Chapter 3
Implementing Questioning Dynamics

In this chapter we present and document the Haskell implementation behind $\mathrm{DEL_Q}$. Some outstanding features of the implementation are the following: modeling the dynamics of issue-epistemic updates via questioning actions, intuitive display of update results, intuitive display of both issue-epistemic models and questioning action models, model checking of issue-epistemic formulae in issue-epistemic structures. The chapter presents and explains the code of one main module `DELQ.lhs` as well as its related functionality contained in auxiliary modules. The code is written in Haskell [55], [68], in the style of Knuth's 'literate programming' and it is related to previous epistemic functionality from *DEMO* [107], and specific $\mathrm{DEL_Q}$ functionality from [71].

3.1 A `DEMo`-like Implementation for DEL_Q

In this sections we present the implementation behind the $\mathrm{DEL_Q}$ results of Chapter 2. We present and discuss the main modules, other modules containing auxiliary functionality are added in a final section. Further illustrations of how the code is useful can be found already in the next section based on the theory already introduced in the previous chapter, they are also referred to throughout the text and additional illustrations are added whenever needed.

The working language for this chapter is Haskell. Haskell is a high level, non-strict, purely-functional programming language named after Haskell B. Curry. The overall design of the implementation has a modular structure, with specific modules capturing theoretical aspects and modeling logical components introduced and discussed so far in previous chapters. The `DELQ.lhs` module is behind the general theoretical background for the logic of questions from Chapter 2, and it also contains functionality to model more complex questioning and resolution actions that include questions that are represented by a cover instead of a partition and answering that captures indistinguishability of previous questions.

The following modules are part of the main `DELQ.lhs` module:

`Syntax` The module that defines the data structures for propositional symbols, agent labels and complex formulae of DEL_Q. These definitions are behind syntax of the epistemic language from Section 2.1.2 of Chapter 2, where their intuitive meaning was introduced and discussed extensively.

`Structures` The module that defines the data structures for the issue-epistemic structures and the various action models and event models as well as their inner structure component by component. Specific functionality, like initialization and naming functions are also included in this module. These definitions are behind Section 2.1.1 of Chapter 2, where their intuitive meaning was introduced and discussed in detail.

`BinaryRel` The module containing the type definition and the main functionalities for binary relations. Binary relations are a central component of the issue-epistemic and action structures, therefore operations on and properties of relations are used for many tasks: defining issue and information partitions, computing fixed points, etc. These definitions are behind Section 2.2.1 of Chapter 2, where their intuitive meaning was introduced and discussed.

`Semantics` The module that contains the functionality linking the formal language introduced in the `Syntax.lhs` module to the issue-epistemic models and their functionality from the `Structures.lhs` module. The main pillar of this link is given by the way in which DEL_Q formulae are interpreted in issue-epistemic structures. This is done by a recursive semantic definition for the concept of truth in a structure. All these correspond to Section 2.1.2 of Chapter 2, where they are intuitively illustrated and minutely explained.

`Upgrade` The module containing functionality related to the model upgrade operations discussed so far. These encode model transformations either by means of resolution and refinement based on taking intersections or by using preconditions and taking the product upgrade of issue-epistemic structures and questioning action models. These definitions are behind Section 2.2.1 of Chapter 2, where their intuitive meaning is introduced and discussed.

`Display` Auxiliary module containing functionality used to display epistemic structures themselves and results of various operations involving them such as model-checking of formulae, domain naming, listing of expressible formulae, etc. These have been used in Section 3.2 of Chapter 2.

`Shortcuts` Auxiliary module containing the predefined structures used as examples and illustrations from Section 3.2 of Chapter 2.

3.1.1 The Syntax.lhs Module

```
 1  module Syntax
 2  where
 3  import List
 4
 5  data Prop = P Int | Q Int | R Int | S Int | N Int deriving (Eq,Ord)
 6  data Nomi = Nomi Int deriving (Eq,Ord)
 7
 8  instance Show Prop where
 9    show (P 0) = "p"; show (P i) = "p" ++ show i
10    show (Q 0) = "q"; show (Q i) = "q" ++ show i
11    show (R 0) = "r"; show (R i) = "r" ++ show i
12    show (S 0) = "s"; show (S i) = "s" ++ show i
13    show (N 0) = "n"; show (N i) = "n" ++ show i
14
15  data Agent = A Int | B Int | C Int | D Int | E Int deriving (Eq,Ord)
16
17  instance Show Agent where
18    show (A 0) = "a"; show (A i) = "a" ++ show i
19    show (B 0) = "b"; show (B i) = "b" ++ show i
20    show (C 0) = "c"; show (C i) = "c" ++ show i
21    show (D 0) = "d"; show (D i) = "d" ++ show i
22    show (E 0) = "e"; show (E i) = "e" ++ show i
```

Basic operations on lists are predefined in `Prelude.hs` which is the main Haskell module containing the standard language definitions and functions. The list functionality is imported from the `List` module in line 3.

Next, the data-structure for propositional atoms is defined and its basic functionality is added in lines 5-13. Propositions are going to be symbols possible indexed by an integer, we introduce for this data constructors. We want to be able to test propositional symbols for equality and to compare and order propositional symbols, we make the proposition datatype inherit this properties as they are predefined in the standard Haskell type-system by using the `deriving (Eq,Ord)` command. A special type of propositional symbols are the nominals, line 6. For conceptual reasons we can introduce a distinct data structure for them. For technical reasons propositions and nominals have to belong to disjoint sets as discussed before. In practice, however, this can also be achieved by simply reserving a special symbol for nominals, in our case `N`, line 5. We give here both alternatives and we will use the most convenient one in subsequent functions.

Finally, we want to be able to display propositional symbols, we achieve this by making `Prop` an instance of the predefined `Show` class and by defining the `show` function for propositions. In case the index is 0 this is not going to be displayed, otherwise we concatenate the character for the index and display the resulting list of characters. We continue by defining the data-structure for agent labels and by adding its basic functionality in lines 15-22. These is completely analogous to what we did for propositional symbols.

The final code block, lines 23-54, defines the data-structure for complex formulae and adds its basic functionality. These follow the basic definitions introduced in Section 2.1.2 of Chapter 2: we have constructors for nominals and propositional atoms, then we have the basic boolean connectives, taking list of formulas for conjunction and disjunction, and we have constructors for the static modalities, which take bot a formula and an agent label as parameters. In addition, we also introduce standard group notions for issues, knowledge and their intersection: conditional common knowledge, common knowledge and distributed knowledge, taking a list of agents as an additional parameter, and general knowledge.

```
23 data Formq = Top | Prop Prop | Nomi Nomi
24   | Neg Formq | Conj [Formq] | Disj [Formq]
25   | O Agent Formq | X Agent Formq | K Agent Formq | U Formq
26   | CO [Agent] Formq | DO [Agent] Formq | EO Formq
27   | CX [Agent] Formq | DX [Agent] Formq | EX Formq
28   | CK [Agent] Formq | DK [Agent] Formq | EK Formq
29   | CCO [Agent] Formq Formq | CCK [Agent] Formq Formq
30   | CCX [Agent] Formq Formq deriving (Eq,Ord)
31
32 instance Show Formq where
33   show Top                 = "T"
34   show (Prop p)            = show p
35   show (Nomi i)            = show i
36   show (Neg f)             = '~': show f
37   show (Conj fs)           = '&': show fs
38   show (Disj fs)           = 'v': show fs
39   show (O agent f)         = 'Q': show agent ++ show f
40   show (X agent f)         = 'R': show agent ++ show f
41   show (K agent f)         = 'K': show agent ++ show f
42   show (U  f)              = 'U': show f
43   show (CO group f)        = "CQ" ++ show group ++ show f
44   show (DO group f)        = "DQ" ++ show group ++ show f
45   show (EO f)              = "EQ" ++ show f
46   show (CX group f)        = "CX" ++ show group ++ show f
47   show (DX group f)        = "DX" ++ show group ++ show f
48   show (EX f)              = "EX" ++ show f
49   show (CK group f)        = "CK" ++ show group ++ show f
50   show (DK group f)        = "DK" ++ show group ++ show f
51   show (EK f)              = "EK" ++ show f
52   show (CCO group f1 f2) = "CCQ" ++ show f1 ++ show group ++ show f2
53   show (CCX group f1 f2) = "CCX" ++ show f1 ++ show group ++ show f2
54   show (CCK group f1 f2) = "CCK" ++ show f1 ++ show group ++ show f2
```

Analogous to propositions, we define the `show` function used to display formulae in our language, using prefix notation for boolean connectives which are applied to lists of formulae. Further illustrations of how this code works for some frequently used formula instances are included in Section 3.2.

3.1.2 The Structures.lhs Module

The list functionality is imported as explained before. The syntax-related functionality is imported from the previously described `Syntax.lhs` module in line 4. The content of the `Probability` module will be explained at a later stage.

```
 1  module Structures
 2  where
 3  import List
 4  import Syntax
 5  import Probability
 6
 7  data EIM state = Eim
 8    [state]
 9    [Agent]
10    [(Agent,state,state)]
11    [(Agent,state,state)]
12    [(state,[Prop])]
13    [state]
14    deriving (Eq,Show)
```

Next, the data-structure for issue-epistemic models is defined in lines 7-14 following the basic definitions introduced and discussed in Section 2.1.1 of Chapter 2. Without going again in minute details we remind the main components and explain how they are implemented. The data constructor for issue-epistemic structures takes the following arguments: a list of states, representing the domain of alternatives or possible worlds; a list of labels representing the agents, as introduced before; two lists of triples containing an agent and two states, encoding the issue relation respectively the uncertainty relations, indexed by the corresponding agent label; a list of pairs containing a state and a list of propositional symbols, representing the valuation function assigning propositions to each possible world; the final component is a list of states encoding the actual situation. We want to be able to test for equality and display EIMs, so we make the EIM datatype inherit this properties as they are predefined in the standard Haskell type-system by using the `deriving (Eq,Show)` command. Further illustrations of how this code works for some concrete instances of EIMs are included in Section 3.2.

Next we add the functions needed to work with the structures just defined.

The function `initMq`, lines 16-24, is used to generate issue-epistemic models. It takes a list of agents and a list of propositional atoms as arguments, respectively, and returns a "blisfull ignorance" and "pristine questioning" issue-epistemic model, i.e., each agent has both a universal issue relation and a universal uncertainty relation over all possible combinations of the given propositional atoms. The actual situation remains unspecified, using a list of the entire domain.

```
16  initMq :: (Num state, Enum state) => [Agent] -> [Prop] -> (EIM state)
17  initMq ags props = (Eim worlds ags accs eqq val points)
18    where
```

```
19      worlds = [0..(2^k-1)]
20      k      = length props
21      val    = zip worlds (sortL (powerList props))
22      accs   = [(ag,st1,st2) | ag <- ags, st1 <- worlds, st2 <- worlds]
23      eqq    = [(ag,st1,st2) | ag <- ags, st1 <- worlds, st2 <- worlds]
24      points = worlds
25
26  powerList  :: [a] -> [[a]]
27  powerList  [] = [[]]
28  powerList  (x:xs) = (powerList xs) ++ (map (x:) (powerList xs))
29
30  sortL :: Ord a => [[a]] -> [[a]]
31  sortL  = sortBy (\ xs ys -> if length xs < length ys then LT else
32    if length xs > length ys then GT else compare xs ys)
33
34  sortR :: Ord a => [[a]] -> [[a]]
35  sortR  = sortBy (\ xs ys -> if length xs < length ys then GT else
36    if length xs > length ys then LT else compare ys xs)
37
38  dom ::  EIM a -> [a]
39  dom (Eim worlds _ _ _ _ _ ) = worlds
```

The next step in the generation of an issue-epistemic model is to name all its states using the nominal `n` indexed by integer values for every possible world in the domain. This is done by the `named` function which takes an arbitrary structures and returns it named, see lines 40-45.

```
40  named :: (Eq a) => EIM a -> EIM a
41  named m@(Eim worlds ags accs eqq val points) =
42   (Eim worlds ags accs eqq namedval points)
43     where
44      namedval = zip worlds (map(\x ->((snd (val!!x) ++ [(N((elemIndices
45        (fst (val!!x)) (map fst val))!!0))]))) [0..((length val)-1)])
```

The remaining functions have an auxiliary role as follows: starting from line 26, the `powerList` function defines recursively a list-analogue of the powerset construction, it is used to generate all propositional combinations. The functions `sortL` and `sortR`, line 30, respectively, line 34 give two alternative ways of ordering lists, by comparing first their length and second their content. Both require an ordered type as the predefined list ordering functions are used. Finally, the `dom` function, line 38, takes an EIM and returns its domain.

The data structure for EIMs and its functionality are very flexible, as they can be extended with minor modeling adaptations to capture action structures.

Questioning Action Models The action models representing both questioning and resolution actions are a straightforward adaptation of EIMs. Their data-structures is defined in lines 47-54 following the basic definitions introduced and discussed in Section 2.1.1 of Chapter 2. Again, without going in minute details we remind the main components and explain briefly the implementation design. The data constructor for questioning action structures (QAMs) takes the following arguments: a list of states, representing the arbitrary epistemic events, which in this context stand for possible answers; a list of labels representing the agents, as before; two lists of triples containing an agent and two states, encoding the issue respectively uncertainty relations, as before; a list of pairs containing an event and a issue-epistemic formula, representing the precondition function assigning formulae encoding conditions for execution to questioning events, this replaces the previous valuation function; the final ingredient is a list of events encoding the real or actual question. We want to be able to test for equality and display QAMs, so we make the EIM datatype inherit this properties as they are predefined in the standard Haskell type-system by using the `deriving (Eq,Show)` command. Further illustrations of how this code works for some concrete instances of QAMs are included in Section 3.2 of the current chapter.

```
47 data QM state = Qm
48   [state]
49   [Agent]
50   [(Agent,state,state)]
51   [(Agent,state,state)]
52   [(state,Formq)]
53   [state]
54   deriving (Eq,Show)
```

The data structure for a resolution action, lines 56-63, is defined essentially as the one for questioning actions, with the exception that now the list of answers is replaced by a unique event. However, this structural isomorphism hides the fact that in practice the formulae encoding preconditions for execution have a completely different structure. This is why we have both actions represented by distinct data constructors.

The main arguments in describing and generating questioning action structures are given by a list of ignorant agents a list of knowing, or aware, agents and a list of formulae representing content of binary questions together with a designated formula which stands for the content of the actual questioning action:

```
56 data RM state = Rm
57   [state]
58   [Agent]
59   [(Agent,state,state)]
60   [(Agent,state,state)]
61   [(state,Formq)]
62   state
63   deriving (Eq,Show)
```

The `initAq` function, lines 65-80, generates a model for indistinguishable yes/no questions as follows: two events are generated for each formula, line 70, corresponding to the affirmative and the negative answers; the corresponding positive and negative preconditions are assigned to the generated events, line 71;

the uncertainty relation is constructed starting from line 72, taking into account the distinction between the two lists of ignorant and aware agents, answers to the same question remain undetermined for everyone while answers to different questions are indistinguishable only for the ignorant agents; the issue relation is set to the identity for all agents, line 77; and the distinguished set of events is the binary answer to the real question, line 79. The `precond` function, lines 82-85, is an auxiliary function used to retrieve the precondition for an event.

```
65 initAq :: (Num state, Enum state) =>
66   [Agent] -> [Agent] -> [Formq] -> Formq -> (QM state)
67 initAq iags kags propfs quest =
68   (Qm events (iags ++ kags) accs eqqs prec answers)
69   where
70    events  = [0..(2* (fromIntegral (length propfs)))-1]
71    prec    = zip events (propfs ++ (map (\x-> (Neg x)) propfs))
72    accs  = [(ag,st1,st2) | ag <- iags, st1 <- events, st2 <- events
73              ++[(ag,st1,st2) | ag <-kags, st1 <-events, st2 <- events,
74                  or [(precond prec st1) == (precond prec st2),
75                      (precond prec st1) == (Neg (precond prec st2)),
76                      (precond prec st2) == (Neg (precond prec st1))] ]
77    eqqs    = [(ag,st1,st2) | ag <- (iags ++ kags), st1 <- events,
78                                  st2 <- events, st1==st2]
79    answers = let q = quest in
80       map (\x-> (fst x)) (filter (\x->(or [(snd x)==q,(snd x)==(Neg q)])) prec
81 
82 precond :: (Num state, Enum state) =>
83   [(state,Formq)] -> state -> Formq
84 precond prec e =
85   (map (\x -> (snd x)) (filter (\x -> ((fst x)==e)) prec))!!0
```

The `initAr` function, lines 87-98, generates a model for a resolution action as follows: a list of events matching the formula-list argument is created, line 92, and execution preconditions are assigned to each event, line 93; as previously for questioning actions, the pattern based on the distinction between the list arguments containing ignorant and aware agents is used to generate a universal and an identity uncertainty relation, respectively, line 94 and a universal issue-relation, line 96; finally, the last argument determines the actual event, line 97.

```
87 initAr :: (Num state, Enum state) =>
88   [Agent] -> [Agent] -> [Formq] -> Formq -> (RM state)
89 initAr iags kags propfs res =
90   (Rm events (iags ++ kags) accs eqqs prec answer)
91     where
92       events = [0..(1* (fromIntegral (length propfs)))-1]
93       prec = zip events propfs
94       accs = [(ag,st1,st2) | ag <- iags, st1 <- events, st2 <- events]
95         ++ [(ag,st1,st2) | ag <- kags, st1 <- events, st2 <- events, st1==st2]
96       eqqs = [(ag,st1,st2) | ag <- (iags++kags), st1 <- events, st2 <- events]
97       answer = let q = res in
98             (map (\x-> (fst x)) (filter (\x-> ((snd x)==q)) prec))!!0
```

Such resolution structures work fine with a product update mechanism for public questions, however, in order to capture more complex resolution actions succeeding private questioning actions more elaborate constructions are needed.

The `pinitAr` function, lines 100-114, generates a resolution action structure that allows for some answers to remain indistinguishable as follows:

```
100 pinitAr :: (Num state, Enum state) =>
101   [Agent] -> [Agent] -> [Formq] -> Formq -> [Formq] -> (RM state)
102 pinitAr iags kags propfs res reset =
103     (Rm events (iags ++ kags) accs eqqs prec answer)
104   where
105     events = [0..(1* (fromIntegral (length propfs)))-1]
106     prec = zip events propfs
107     accs = [(ag,st1,st2) | ag <- iags, st1 <- events, st2 <- events,
108       (st1 ‘elem‘ (map (\x -> (fst x)) (filter (\x ->
109       (elem (snd x) reset)) prec))) == (st2 ‘elem‘ (map (\x -> (fst x))
110         (filter (\x -> (elem (snd x) reset)) prec) ) )  ]
111         ++ [(ag,st1,st2) | ag <- kags, st1 <- events, st2 <- events, st1==st2]
112     eqqs = [(ag,st1,st2) | ag <- (iags++kags), st1 <- events, st2 <- events]
113     answer = let q = res in
114               (map (\x-> (fst x)) (filter (\x-> ((snd x)==q)) prec))!!0
```

The first two arguments are, as before, list containing ignorant respectively aware agents used to generate uncertainty relations that allow for epistemic gradients, line 107, instead of just complete uncertainty as before, now partial uncertainty can be modeled; the issue-relation is, as before, universal, line 112; a list of events matching the formula-list argument is created, line 105, and execution preconditions are assigned to each event, line 106 as explained previously; finally, the last function argument is used to determine the actual event, in line 113.

Complex Questioning As with resolution actions, the raw binary questioning structures are just a first approximation, their basic utilities can be extended to deal with more complex questioning functionality in an analogous way. We give here the code for both complex propositional questions with mutually disjoint answers and complex cover-based questions with overlapping answer-sets.

```
115 initPropq :: (Num state, Enum state) =>
116   [Agent] -> [Agent] -> [[Formq]] -> [Formq] -> (QM state)
117 initPropq iags kags frmlist quest =
118   (Qm events (iags ++ kags) accs eqqs prec answers)
119     where
120       events = [0..((fromIntegral (length (foldr (++) [] frmlist)))-1)]
121       prec = zip events ((foldr (++) [] frmlist))
122       accs = [(ag,st1,st2) | ag <- iags, st1 <- events, st2 <- events]
123            ++[(ag,st1,st2) | ag <- kags, st1 <- events, st2 <- events,
124                 (filter (elem (precond prec st1)) frmlist) ==
125                 (filter (elem (precond prec st2)) frmlist)]
126       eqqs = [(ag,st1,st2) | ag <- (iags ++ kags), st1 <- events,
127                                 st2 <- events, st1==st2]
128       answers = let q = quest in
```

```
129              map (\x-> (fst x)) (filter (\x-> ( elem (snd x) q)) prec)
130
131 fst33 (x,_,_) = x
132 snd33 (_,x,_) = x
133 trd33 (_,_,x) = x
```

The `initPropq` function, lines 115-129, generalizes the previous construction from binary yes/no questions to arbitrary partition-based propositional questions. To capture this additional complexity the function arguments are now a list of ignorant agents, a list of aware agents, just like before, but the remaining parameters are lifted one level of abstraction, we have now a list of lists of formulae, standing for the indistinguishable questioning actions and a list of formulae designed to capture the real or actual question.

The function arguments are assumed for now to be adequate for a partition model of questions, i.e. the formulae are assumed to be mutually incompatible, however they do not have to also be jointly exhaustive, the product update mechanism handles such aspects in a standard way. The process of automatic generation proceeds as follows: events are generated for each formula, after collapsing in line 120 by `forldr` one level of abstraction and concatenating all formulae in one list; the corresponding exclusive preconditions are assigned to the generated events, in line 121 by `zip`-ing the two lists; the uncertainty relation is constructed starting from line 122, taking into account the distinction between ignorant and aware agents, answers to the same question remain undetermined for everyone while answers to different questions are indistinguishable only for the ignorant agents, because in the current setting it is possible that answers appear in multiple questions, the answer-list is also used as an additional identification criterion; the issue relation is set, as before, to the identity for all agents, line 126 capturing the fact that questioning actions refine the issue relation; and the actual set of events is taken to be the distinguished formula list given as the last argument, line 128. Further illustrations of how this code works for some concrete examples are included in Section 3.2 later in this chapter.

```
135 initPropCov :: (Num state, Enum state) =>
136   [Agent] -> [Agent] -> [[Formq]] -> ([Formq],Formq) -> (QM state)
137 initPropCov iags kags frmlist quest answ =
138   (Qm events (iags ++ kags) accs eqqs prec answers)
139     where
140       events = [0..((fromIntegral (length (foldr (++) [] frmlist)))-1)]
141       prec = zip events ((foldr (++) [] frmlist))
142       accs = [(ag,st1,st2) | ag <- iags, st1 <- events, st2 <- events]
143            ++[(ag,st1,st2) | ag <- kags, st1 <- events, st2 <- events,
144               (map trd33 (filter (\x -> (fst33 x == st1) )
145               (zip3 events ((foldr (++) [] frmlist)) (foldr (++) []
146               (map (\x -> take (length x) (repeat x)) frmlist)))))
147               == (map trd33 (filter (\x -> (fst33 x == st2) )
148               (zip3 events ((foldr (++) [] frmlist)) (foldr (++) []
149               (map (\x -> take (length x) (repeat x)) frmlist))))) ]
```

```
150       eqqs = [(ag,st1,st2) | ag <- (iags ++ kags), st1 <- events,
151                              st2 <- events, st1==st2]
152       answers = [fst33 ((filter (\x -> (and [(snd33 x) ==
153                 answ, (trd33 x)==quest]) ) (zip3 events ((foldr (++)
154                 [] frmlist)) (foldr (++) [] (map (\x -> take (length x)
155                 (repeat x)) frmlist))))!!0)]
```

The `initPropCov` function, lines 135-155, generalizes the previous construction from partition-based propositional questions to questions based on a cover by eliminating the requirement that answers to questions have to be disjoint and allowing for answers with overlapping extensions.

In order to deal with this additional complexity the `initPropCov` function, lines 135-155, takes as arguments the lists of ignorant and aware agents, just like before, a list of formulae lists, but the last parameter is now a list-formula pair, representing the real question action and the designed actual answer, respectively. Auxiliary projection functions for triples are defined in lines 131-133.

The process of automatic generation proceeds as follows: events are generated for each formula, again by folding all the answers in a single list, in line 140; the arbitrary preconditions are `zip`-ed with the generated events, in line 141; the uncertainty relation is constructed using triples containing events, questions and answers starting from line 142, the distinction between ignorant and aware agents has the same role as before, because in the current setting answers may have nonempty intersections and appear in multiple questions, the answer-list is also used as an additional identification criterion; the issue relation is set, as before, to the identity for all agents, in line 150, capturing the fact that questioning actions refine the issue relation by distinct events, even thought some might have overlapping or identical preconditions; and the actual event is constraint now to a singleton list taking into account both the precondition and the question list to which it belongs, line 152. Further illustrations of how this code works for some concrete examples are included in Section 3.2.

Complex Resolution It is now time to extend our models for resolution actions in a similar way to allow a formal structure matching the gradient level of abstraction in the actions with indistinguishable questions discussed so far.

A first step in this direction will be to parametrize resolution by the history of previous questioning actions. This can be done by the use of preconditions. The `initAres` function, lines 157-177, has the same structure as previous ones but assigns preconditions to event in a more complex way.

The componentwise generation process is the following: events are generated for each formula, by taking binary answers, in line 162; corresponding binary issue preconditions are `zip`-ed with the generated events, in line 163; the uncertainty relation is constructed by pairing events with affirmative respectively negative issue-precondition, (see Section 3.2 for the intuitive example), starting from line 167, the distinction between ignorant and aware agents has the same role as explained before; the issue relation is the universal relation for all agents, line 175, capturing the fact that resolution actions do not raise further issues; and the actual event is determined by the last function argument, at line 176.

Resolution actions can in some cases override the structure of the issue relation and give either more or less information than a mere answer to the questions raised so far would allow. There is no reason why such information flow scenarios should be excluded from the formalism. And indeed such situations can be captures by using the constructor defined by the `initAres2` function, lines 179-192.

The components that are unchanged from the previous constructor are the event set, from line 184, the binary issue-precondition formulae, from line 185, the universal issue relation, from line 190, and the designated actual event, in line 191; the changed component in this new resolution model consists in assigning an universal indistinguishability relation between events to the oblivious agents, starting from line 188, while the aware agents have a transparent access to the content of the resolution action. Further intuitive illustrations of how this code works and some concrete examples of resulting resolution actions and their epistemic effect are included in Section 3.2 later in this chapter.

```
157 initAres :: (Num state, Enum state) =>
158   [Agent] -> [Agent] -> [Formq] -> Formq -> (QM state)
159 initAres iags kags propfs actf =
160   (Qm events (iags ++ kags) accs eqqs prec resol)
161     where
162      events = [0..(2* (fromIntegral (length propfs)))-1]
163      prec = zip events ((map (\x-> (Conj [ x, Disj [ O (iags!!0) x,
164             O (iags!!0) (Neg x)]])) propfs) ++ (map (\x-> (Conj [
165             Neg x, Disj [ O (iags!!0) x, O (iags!!0)
166             (Neg x)]])) (propfs)))
167      accs = [(ag,st1,st2) | ag <- iags, st1 <- events, st2 <- events,
168               ((elemIndices st1 events)!!0) <= (length propfs)-1,
169               ((elemIndices st2 events)!!0) <= (length propfs)-1]
170               ++ [(ag,st1,st2) | ag <- iags, st1 <- events,
171               st2 <- events, ((elemIndices st1 events)!!0) >=
172               (length propfs), ((elemIndices st2 events)!!0) >=
173               (length propfs)] ++ [(ag,st1,st2) | ag <- kags,
174               st1 <- events, st2 <- events, st1==st2 ]
175      eqqs = [(ag,st1,st2) | ag <- (iags ++ kags),st1 <- events,st2 <- events]
176      resol = let q = actf in
177       map (\x->(fst x)) (filter (\x->((snd x)==q)) (zip events propfs))
178
179 initAres2 :: (Num state, Enum state) =>
180   [Agent] -> [Agent] -> [Formq] -> Formq -> (QM state)
181 initAres2 iags kags propfs actf =
182   (Qm events (iags ++ kags) accs eqqs prec resol)
183     where
184       events = [0..(2* (fromIntegral (length propfs)))-1]
185       prec = zip events ((map (\x-> (Conj [ x, Disj [ O (iags!!0) x,
186         O (iags!!0) (Neg x)]])) propfs) ++ (map (\x->(Conj [Neg x,Disj
187        [ O (iags!!0) x, O (iags!!0) (Neg x)]])) (propfs)))
```

```
188       accs = [(ag,st1,st2) | ag<-iags, st1<-events, st2<-events] ++
189         [(ag,st1,st2) | ag<-kags, st1<-events, st2<-events, st1==st2]
190       eqqs = [(ag,st1,st2) | ag <- (iags ++ kags), st1 <-events, st2 <- events]
191       resol = let q = actf in map (\x-> (fst x))
192              (filter (\x-> ( (snd x)==q)) (zip events propfs))
```

The `initAres3` function, lines 194-209, takes resolution actions to the other extreme in this large spectrum of modeling possibilities by allowing resolutions in which all the questions raised so far are publicly resolved for all agents.

The components that are unchanged from the previous constructor are the event set, from line 199, the binary issue-precondition formulae, from line 200, the universal issue relation, from line 207, and the designated actual event, in line 208; the changed component in this new resolution model consists in assigning an identity indistinguishability relation between events to all agents in line 204.

```
194 initAres3 :: (Num state, Enum state) =>
195   [Agent] -> [Agent] -> [Formq] -> Formq -> (QM state)
196 initAres3 iags kags propfs actf =
197   (Qm events (iags ++ kags) accs eqqs prec resol)
198     where
199       events = [0..(2* (fromIntegral (length propfs)))-1]
200       prec = zip events ((map (\x-> (Conj [ x, Disj [ O (iags!!0) x,
201              O (iags!!0) (Neg x)]])) propfs) ++ (map (\x-> (Conj
202              [ Neg x, Disj [ O (iags!!0) x, O (iags!!0)
203              (Neg x)]])) (propfs)))
204       accs = [(ag,st1,st2) | ag <- iags, st1 <- events, st2 <- events,
205              st1 ==st2 ] ++ [(ag,st1,st2) | ag <- kags, st1 <- events,
206              st2 <- events,  st1==st2 ]
207       eqqs = [(ag,st1,st2) | ag <- (iags ++ kags),st1 <- events,st2 <- events]
208       resol = let q = actf in map (\x-> (fst x))
209              (filter (\x-> ( (snd x)==q)) (zip events propfs))
```

Finally, the `initAres4` function, lines 211-229, models a resolution action with an arbitrary gradient of publicity for the answer events. The components that remain unchanged are the event set, from line 216, the binary issue-precondition formulae, from line 217, the universal issue relation, from line 226, and the designated actual event, in line 208; the changed component in this new resolution model consists in assigning an indistinguishability relation that groups the events by division modulo the length of the list of answer events, starting from line 221.

Further intuitive illustrations of how this code works for some concrete resolution actions are discussed in Section 3.2 later in this chapter.

Although all the constructions presented and explained in this section seem to lead to an industrious process, they follow the usual requirements for an exercise in the 'art of modeling' for various scenarios occurring in luxuriant practical applications, moreover, the generation of adequate models is automated, indicating the fact that the choice of the right issue-epistemic gradient for a particular context to be modeled does is not given by logic but is merely a contingent choice determined by an adequate model of the situation at hand.

```
initAres4 :: (Num state, Enum state) =>
  [Agent] -> [Agent] -> [Formq] -> Formq -> (QM state)
initAres4 iags kags propfs actf =
  (Qm events (iags ++ kags) accs eqqs prec resol)
   where
    events  = [0..(2* (fromIntegral (length propfs)))-1]
    prec = zip events ((map (\x-> (Conj [ x, Disj [ O (iags!!0) x,
           O (iags!!0) (Neg x)]])) propfs) ++ (map (\x-> (Conj
           [ Neg x, Disj [ O (iags!!0) x, O (iags!!0)
           (Neg x)]])) (propfs)))
     accs = [(ag,st1,st2) | ag <- iags, st1 <- events, st2 <- events,
            rem ((elemIndices st1 events)!!0)  (length propfs) ==
            rem ((elemIndices st2 events)!!0)  (length propfs)] ++
            [(ag,st1,st2) | ag <- kags, st1 <- events, st2 <- events,
            st1==st2 ]
    eqqs = [(ag,st1,st2) | ag <- (iags ++ kags), st1 <- events,
                           st2 <- events]
    resol = let q = actf in
     map (\x-> (fst x)) (filter(\x->((snd x)==q)) (zip events propfs))
```

The real advantage of having all these models is that they can be handled in a unitary manner by the product update mechanism, as we will see shortly.

3.1.3 The `BinaryRel.lhs` Module

Before we go on to discuss the product upgrade mechanism we will spend some time presenting the basic definitions and functionality used in the implementation for binary relations as already introduced in 2.2.1 of Chapter 2.

The module starts by importing the standard list functionality and defining, in line 5 the datatype for binary relations as a list of pairs. Next a containment function for lists is defined to be used for comparing relations represented as pair-lists, at line 7, also a function that decides identity for relations, by ignoring the order in which the pairs are present in the list, is defined at line 9.

```
 1  module BinaryRel
 2  where
 3  import List
 4
 5  type Rel a = [(a,a)]
 6
 7  containedIn :: Eq a => [a] -> [a] -> Bool
 8  containedIn xs ys = all (\ x -> elem x ys) xs
 9  sameR :: Ord a => Rel a -> Rel a -> Bool
10  sameR r s = sort (nub r) == sort (nub s)
11
12  cnv :: Rel a -> Rel a
13  cnv r = [ (y,x) | (x,y) <- r ]
14
15  infixr 5 @@
16  (@@) :: Eq a => Rel a -> Rel a -> Rel a
```

```
17 r @@ s = nub [ (x,z) | (x,y) <- r, (w,z) <- s, y == w ]
18
19 euclR :: Eq a => Rel a -> Bool
20 euclR r = (cnv r @@ r) ‘containedIn‘ r
21
22 serialR :: Eq a => Rel a -> Bool
23 serialR r = all (not.null) (map (\ (x,y) -> [ v | (u,v) <- r, y == u]) r)
```

Basic relational operations are introduced via the next two functions, the first is the relational converse, in line 12, and relational composition, in line 15, as an infix operator with priority 5. Two useful relational properties are defined afterwards in line 19 euclideanity respectively seriality in line 22.

```
25 rightS :: Ord a => Rel a -> a -> [a]
26 rightS r x = (sort.nub) [ z | (y,z) <- r, x == y]
27
28 lfp :: Eq a => (a -> a) -> a -> a
29 lfp f x | x == f x  = x
30         | otherwise = lfp f (f x)
31
32 rtc :: Ord a => [a] -> Rel a -> Rel a
33 rtc xs r = lfp (\ s -> (sort.nub) (s ++ (r @@ s))) i
34   where i = [(x,x) | x <- xs ]
35
36 tc :: Ord a => Rel a -> Rel a
37 tc r = lfp (\ s -> (sort.nub) (s ++ (r @@ s))) r
```

The image of an element under a relation is defined in line 25 and the reflexive transitive closure and reflexive closure of a relation are introduced in lines 32 respectively line 36 using the least fix-point definition from line 28. These definitions are going to be useful later on, for instance to define the semantics of the common knowledge modality or for partition refinement.

3.1.4 The `Semantics.lhs` Module

The `Sematics.lhs` module uses the functionality presented so far, starting from line 3, to give a recursive definition for truth of DEL$_Q$ formulae in issue-epistemic structures as introduced previously in Section 2.1.2 of Chapter 2.

```
1  module Semantics
2  where
3  import List
4  import Syntax
5  import Structures
6  import BinaryRel
7
8  domain :: Ord state => EIM state -> [state]
9  domain m@(Eim dom _ _ _ _ _) = dom
10
11 smodels :: Ord state => EIM state -> Formq -> Bool
12 smodels m@(Eim _ _ _ _ _ act) fr = (and (map (\x-> (models m x fr)) act))
```

```
13
14 models :: Ord state => EIM state -> state -> Formq -> Bool
15 models m w Top = True
16 models m@(Eim _ _ _ _ val _)     w (Prop p) =
17                      elem p (concat [props|(w',props) <- val, w'==w ])
18 models m w (Neg f)   = not (models m w f)
19 models m w (Conj fs) = and (map (models m w) fs)
20 models m w (Disj fs) = or  (map (models m w) fs)
21 models m@(Eim _ _ _ eqq _ _)     w (O agt f) =
22                   and (map (flip (models m) f) (rightS (relq agt m) w))
23 models m@(Eim _ _ _ _ _ _)   w (X agt f) =
24                   and (map (flip (models m) f) (rightS (relx agt m) w))
25 models m@(Eim _ _ acc _ _ _)     w (K agt f) =
26                   and (map (flip (models m) f) (rightS (relk agt m) w))
27 models m@(Eim dom ags _ eqq _ _) w (U f)     =
28                          and (map (flip (models m) f ) dom)
29
30 relk :: Agent -> EIM a -> Rel a
31 relk a m@(Eim _ _ acc _ _ _) = [ (x,y) | (agent,x,y) <-acc, a == agent]
32
33 relq :: Agent -> EIM a -> Rel a
34 relq a m@(Eim _ _ _ eqq _ _) = [ (x,y) | (agent,x,y) <-eqq, a == agent]
35
36 relx :: (Eq a) => Agent -> EIM a -> Rel a
37 relx a m = intersect (relk a m) (relq a m)
```

In line 8, the auxiliary function takes an issue-epistemic structure and returns its domain. Before defining the usual local definition of truth at a state or from a pointed perspective, we introduce the same notion at the level of a set of worlds from line 11. This notion turns out to be often useful, especially in game contexts where players are uncertain about the real situation.

The most important function in this module is the `models` function introduced in lines 14-28. It takes as input an issue-epistemic structure, a state in its domain and a formula and returns a boolean value. The definitions proceed along the expected lines starting with the boolean cases, for which the valuation of the model provides the needed information, and continuing to formulae containing more complex modalities, for which the functionality related to the issue and epistemic relations becomes crucial. The relevant information about the issues, epistemic uncertainty and their interrelation for an agent or even groups of agents is contained inside the model received as a function parameter. In order to access this information and use it in recursive truth-value computations, three additional auxiliary functions are defined starting from lines 33, 30 and 36, respectively.

3.1.5 The `Upgrade.lhs` Module

We have now all the requested ingredients to proceed towards implementing the functionality needed for the central dynamic actions for $\mathrm{DEL}_{\mathrm{Q}}$ as discussed in Section 2.2.1 of Chapter 2. In this section we will show that the product update can be used as a unifying formal mechanism encompassing the luxuriant variety of questioning actions that can be modeled by DEL$_Q$. We will also show how using the issue relation on top of the traditional epistemic relation can extend the product update mechanism in an interesting and useful way.

The module starts again in the standard way by incrementally importing, at lines 3-7, some needed functionality from modules previously explained.

```
1  module Upgrade
2  where
3  import List
4  import Syntax
5  import Structures
6  import Semantics
7  import Shortcuts
```

Next, the `upgradeq` function, lines 9-27, implements the product update mechanism between issue structures and models of questioning actions using, as before, the componentwise construction of the models' constituents described before.

The new domain consists of pairs of worlds and events with matching preconditions, from line 15; the two models that are the function's arguments are assumed to take the same list of agents, which is also used in the output model, line 17; the new uncertainty and issue relations are constructed by combining the old ones in both the issue-epistemic model and the action model, see lines 18 respectively 21; the new valuation conserves the previous one for the world in the pair, line 24; the new actual world is uniquely determined by the event with matching preconditions in the real question list, line 26.

```
9  upgradeq:: (Eq state, Ord state) =>
10      EIM state -> QM state -> EIM (state, state)
11 upgradeq m@(Eim dom agts accs eqqs val act)
12  q@(Qm evs ags acs eqs prec answ) =
13  (Eim udom uagts uaccs ueqqs uval uact)
14   where
15    udom  = [(w,e) | w <-dom, e <-evs,
16             (models m w (precondition q e))]
17    uagts = [x | x <- agts]
18    uaccs = [(ag,(w,e),(v,f))| ag <- ags, (w,e) <- udom,
19             (v,f) <- udom, (ag,w,v) ‘elem‘ accs,
20             (ag,e,f) ‘elem‘ acs]
21    ueqqs = [(ag,(w,e),(v,f))| ag <- ags, (w,e) <- udom,
22             (v,f) <- udom, (ag,w,v) ‘elem‘ eqqs,
23             (ag,e,f) ‘elem‘ eqs]
24    uval = [((w,e),l) | (w,e) <- udom, l <- (map (\x -> (snd x))
25            val), ((valuation m w) == l)]
26    uact = [(w,e) | w <-act, e <- answ,
27            (models m w (precondition q e))]
```

The functions `valuation`, line 29, and `precondition`, line 33, have an auxiliary role, the first takes an EIM and a state and returns its valuation, the second takes a question model and an event and returns its precondition for execution.

```
29 valuation :: Eq state => EIM state -> state -> [Prop]
30 valuation m@(Eim dom agts accs eqqs val act) w =
31   (snd ((filter (\x -> ((fst x) == w)) val)!!0) )
32
33 precondition :: Eq state => QM state -> state -> Formq
34 precondition q@(Qm evs ags acs eqs prec answ) e =
35   (snd ((filter (\x -> ((fst x) == e)) prec)!!0) )
```

The product update mechanism for resolution will be applied on two sample models: an issue-epistemic model, line 37, and a resolution model, line 40.

```
37 eimsample = (upgradeq (initMq [a,b] [Syntax.P 0,Q 0])
38   (initAq [a] [b] [p,q] p))
39
40 resample = (initAres [a] [b] [p,q] p)
```

Now we have all the required ingredients to introduce an implementation of a product rule that uses both equivalence relations to model resolution actions. The first and most obvious way in which to make good us of the expressive power introduced by the fact that out models use two relations is by considering their intersection in defining the action's effects. The `exclam` function given in lines 42-46 does exactly this. It implements the resolution by the intersection dynamic action defined before. This is a very simple and already useful mechanism that makes the interdependence between questions and information explicit.

```
42 exclam :: (Eq state, Ord state) => EIM state -> EIM state
43 exclam m@(Eim dom agts accs eqqs val act) =
44   (Eim dom agts accsr eqqs val act)
45     where
46     accsr = accs `intersect` eqqs
```

A more complex resolution action, more suitable for realistic scenarios involving the various gradients of publicity in a resolution discussed before will extend the product update mechanism by allowing a combination between the two relations to be the driving force in the product upgrade mechanism.

```
48 upgraderes:: (Eq state, Ord state) =>
49   EIM state -> QM state -> EIM (state, state)
50 upgraderes m@(Eim dom agts accs eqqs val act)
51   mq@(Qm evs ags acs eqs prec answ) = (Eim udom uagts uaccs ueqqs uval uact)
52    where
53     udom  = [(w,e) | w <- dom, e <- evs,  (models m w (precondition mq e))]
54     uagts = [x | x <- agts]
55     uaccs = [(ag,(w,e),(v,f))| ag <- uagts, (w,e) <- udom,
56              (v,f) <- udom, (ag,w,v) `elem` accs, or
57             [(ag,w,v) `elem` eqqs,(ag,e,f) `elem` acs]]
58     ueqqs = [(ag,(w,e),(v,f))| ag <- uagts, (w,e) <- udom,
59              (v,f) <- udom, (ag,w,v) `elem` eqqs,
60              (ag,e,f) `elem` eqs]
```

```
61     uval = [((w,e),l) | (w,e) <- udom, l <- nub (map
62             (\x -> (snd x)) val), ((valuation m w) == l)]
63     uact = [(w,e) | w <- act, e <- answ, (models m w (precondition mq e))]
64
65 recondition :: Eq state => RM state -> state -> Formq
66 recondition r@(Rm evs ags acs eqs prec answ) e =
67   (snd ((filter (\x -> ((fst x) == e)) prec)!!0) )
```

The `upgraderes` function implements this aspects in lines 48-63 by the following componentwise construction: the new domain consists of pairs of wolds and events with matching preconditions, line 53; the function's arguments are an issue-epistemic model and a resolution model assumed to take the same list of agents, which is also used to construct the output model, line 54; the new uncertainty relation is constructed by a disjunctive combination of old uncertainty between states and previous issue equivalence between events, from line 55; the new issue relations are constructed in the same way as before from their corespondents in both the given resolution model and the issue-epistemic model, line 58; the new valuation conserves the valuation of the world in the pair, line 61; the new actual world is determined by the event with matching preconditions in the real question list, line 63. The last function, from line 65, has an auxiliary role, it takes a resolution model and an event and returns its precondition for execution.

3.1.6 The `DELQ.lhs` Module

The `DELQ.lhs` module puts together all the functionality presented so for in an unitary framework. It starts by importing the functionality in the corresponding modules as discussed previously. Then it includes some additional functions.

```
1  module DELQ
2  where
3  import List
4  import Syntax
5  import Structures
6  import Semantics
7  import BinaryRel
8  import Upgrade
9  import Display
10 import Shortcuts
11 import Probability
```

The `extension` function takes a formula and an issue-epistemic model, line 13, and returns the set of states satisfying the formula. The next two functions, line 16 and line 21, return the image respectively the converse image under an epistemic relation for a set of states and an agent given as arguments.

```
13 extension :: (Ord a) => Formq -> EIM a -> [a]
14 extension f m = filter (\x -> (models m x f)) (dom m)
15
```

```
16 boxk :: (Eq b) => [b] -> Agent -> EIM b -> [b]
17 boxk s a m =
18   nub (map snd (filter (\x -> (fst x) ‘elem‘ s) (relk a m)))
19
20 boxk_minus :: (Eq b) => [b] -> Agent -> EIM b -> [b]
21 boxk_minus s a m =
22   nub (map fst (filter (\x -> (snd x) ‘elem‘ s) (relk a m)))
```

The last function in the block, line 24, returns the set of states that satisfy a formula via the intersection modality, making crucial use of nominals.

```
24 intersOp f m = filter (\x -> models m x (k_ a
25   (Conj [f, (o_ a ((noml m1 [x])!!0))]))) (dom m1)
26
27 nominals :: (Eq a) => EIM a -> [Formq]
28 nominals m@(Eim _ _ _ _ val _) = map (\x -> (Prop x))
29   (map (\x -> (x!!((length x)-1))) (map snd val))
30
31 noml :: (Eq a) => EIM a -> [a] -> [Formq]
32 noml m l = map (\x -> (Prop x))
33   ((map (\x -> (x!!((length x)-1))))
34   (map snd (filter (\x -> (elem (fst x) l)) (valuationL m))))
```

Next, there are two functions dealing with generating nominals for an issue-epistemic structure given as input, line 27. This is useful in order to deal with intersection modality and to allow for full expressibility when describing a named model. The second function generates the nominal list for a given subset of the domain in an issue-epistemic structure, which is useful to generate all the expressible formulae given a model and a hybrid language, line 31.

The valuation of a model can be retrieved by the function given in line 36, and can be used to generate all the expressible formulae as disjunctions of nominals in the powerset of the domain, as described starting from line 39.

```
36 valuationL :: EIM a -> [(a, [Prop])]
37 valuationL m@(Eim _ _ _ _ val _) = val
38
39 forms :: (Ord state) => EIM state -> [ Formq ]
40 forms m = map (\y -> Disj y)
41   (map (\x -> noml m x) (powerList (dom m)))
42
43 relk_a :: Agent -> QM state -> Rel state
44 relk_a a m@(Qm _ _ acc _ _ _) =
45   [ (x,y) | (agent,x,y) <- acc, a == agent ]
46
47 relq_a :: Agent -> QM state -> Rel state
48 relq_a a m@(Qm _ _ _ eqq _ _) =
49   [ (x,y) | (agent,x,y) <- eqq, a == agent]
```

Finally, two more auxiliary functions are given at line 43 and line 47, they take a questioning action model and an agent and return the relation pairs indexed by that agent, for the epistemic and the issue relations, respectively.

Further intuitive illustrations of how this code works for some concrete resolution actions are discussed in Section 3.2 of the current chapter.

This concludes the exposition of the core DEL_{Q} functionality contained in the most important modules implementing theoretical aspects. The remaining ones have supporting role with only practical and applicative importance.

3.2 Illustrations using the Implementation

In this chapter we will discuss some general consequences of the DEL_{Q} approach explained so far. We will support and give concreteness to our discussion by using illustrative intuitive examples, for this purpose we will make use of code output for both ease of exposition and computational precision.

In order to do this we will guide the reader through the bare minimum needed to grasp the examples discussed. We do this by linking some of the concrete examples already discussed and illustrated in the text with their representation as both standard tuple-notation and implementation-generated output.

Epistemic Issue Structures (EIMs) can be automatically generated by `DELQ.hs` in a standard way. For facilitating the current presentation we refer the reader to our previous intuitive diagrammatic representation in Figure 2.12. The starting EIM in our Example 2.4.5 is generated and represented as follows:

```
*QPR> dpq (initMq [a,b] [Syntax.P 0,Q 0])
[0,1,2,3]
[(0,[]),(1,[p]),(2,[q]),(3,[p,q])]
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])
[3]
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])
```

Intuitively, this says that we have two agents a, b ignorant about two facts p, q and with a universal issue relation on the domain, the world in which both facts are true represents the actual situation. This also corresponds to the following structure in standard tuple-notation $M = \langle W, \overset{a}{\approx}, \overset{b}{\approx}, \overset{a}{\sim}, \overset{b}{\sim}, V\rangle$: $W = \{0, 1, 2, 3\}$, $V = \{p \mapsto \{1, 3\}, q \mapsto \{2, 3\}\}$, $\overset{a}{\approx} = \overset{b}{\approx} = \{0, 1, 2, 3\} \times \{0, 1, 2, 3\} = \overset{a}{\sim} = \overset{b}{\sim}$.

In a similar fashion, the Action Issue Model (AIM) from Example 2.4.5 can be automatically generated by `DELQ.hs` in a standard way. The second model in Figure 2.12 is generated and represented in the implementation as follows:

```
*QPR> dpa (initAq [a] [b] [p,q] p)
[0,1,2,3]
[(0,p),(1,q),(2,~p),(3,~q)]
(a,[[0,1,2,3]])
(b,[[0,2],[1,3]])
[0,2]
(a,[[0],[1],[2],[3]])
(b,[[0],[1],[2],[3]])
```

Intuitively, we have here two agents a, b and four events corresponding to the possible ways to answers the two questions $p?$ and $q?$. Each of the events has the corresponding formula as its precondition. Here these are the obvious propositional atoms and their negation. Later on, in order to capture more interesting situation such preconditions can become more complex formulae in our epistemic issue language designed to capture questioning content and its relation with agents' information. This also corresponds to the following structure represented in standard tuple-notation $Q = \langle E, \overset{a}{\approx}, \overset{b}{\approx}, \overset{a}{\sim}, \overset{b}{\sim}, P\rangle$: $E = \{0,1,2,3\}$, $P = \{0 \mapsto p, 1 \mapsto q, 2 \mapsto \neg p, 3 \mapsto \neg q\}$, $\overset{a}{\approx} = \{0,1,2,3\}\times\{0,1,2,3\}$, $\overset{b}{\approx} = \{(0,0),(1,1),(2,2),(3,3)\} = Id(E)$, $\overset{a}{\sim} = \overset{b}{\sim} = \{0,2\} \times \{0,2\} \cup \{1,3\} \times \{1,3\}$.

Here is a good place to introduce the basics about how issue-epistemic formulae are generated, used, and represented in Haskell. Formulas can be visualized in `DELQ.hs` in a compact way, we give here some illustrative examples covering most frequently used formula instances:

`DELQ.hs`	Formula	Display
`Neg (Conj [O a p, Neg p])`	$\neg(Q_a p \wedge \neg p)$	`~&[Qap,~p]`
`Disj [Neg (X a (Prop(P 0))), p]`	$R_a p \to p$	`v[~Rap,p]`
`Disj [K a p, k_ a (Neg p)]`	$K_a p \vee \widehat{K}_a \neg p$	`v[Kap,~Ka~~p]`
`U (Disj [O b q, O b (Neg q)])`	$U(Q_b q \vee Q_b \neg q)$	`Uv[Qbp,Qb p]`
`U (Conj [Prop(N 1), X a (K b p)])`	$U(n_1 \wedge R_a K_b p)$	`U&[n1,RaKbp]`

Continuing with our correspondence between the diagramatic representation in Figure 2.12 and the code output, we can see that a is ignorant with regard to what question was asked, we model this by a universal indistinguishability relation on the set of events (answers). The second agent b, on the other hand, is aware of what the content of the questioning action is. The identity issue relation on the domain of events captures the fact that both agents are aware abut possible answers to the questioning actions. Finally, the actual question is $p?$, we represent this by a list of distinguished events in the domain. The set of distinguished events is not a singleton, as in the standard DEL action structure, but the set of possible answers to the real question $\{p, \neg p\}$.

Discussion (Determinism) One important assumption in the standard DEL modeling using product update was the fact that both epistemic structures and action models are *deterministic*. This background assumption was present in the formalism in the uniqueness of the real world and of the actual event. Some agents are ignorant about the real world and some agents cannot determine what epistemic action has taken place. But this kind of nondeterminism is merely a reflection of epistemic uncertainty not genuine lack of determinism in the real situation modeled. The fact that the perfectly informed *modeler* of the epistemic scenario can chose beforehand a designated world and a unique action that really took place reflects precisely the essential deterministic character of the approach.

One natural question at this point is: Don't we have to give up determinism when we want to build an adequate model of questioning actions? A quick answer at this stage is: Not yet. Note that EIMs still have a unique world, and we only introduced multiple events in AIMs. However, we still require our models to be deterministic, even though at a higher level. We only allow for a unique questioning action, which can have multiple, not yet determined, answers corresponding to a partition cell. Besides being conceptually interesting, this aspect will be important from a technical point of view during the completeness proof. For this reason, we give it a distinct statement already at this stage:

3.2.1. Definition. [Deterministic AIMs] A questioning structure is a Kripke frame over a set of agents and events together with a set of events corresponding to multiple (but mutually exclusive) answers to a *unique* designated question.

Using Definition 3.2.1 one can show that the class of EIMs structures is closed under product update with deterministic AIMs and partition questions.

We now return to our exposition of Example 2.4.5. The resulting updated model can now be computed in our implementation of $\mathrm{DEL_Q}$ using standard product update; the result is visualized by `DELQ.hs` in the following way:

```
*QPR> dpq (reornameq (upgradeq
  (initMq [a,b] [Syntax.P 0,Q 0]) (initAq [a] [b] [p,q] p)))

[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,1,2,3,4,5,6,7]])
(b,[[0,2,5,6],[1,3,4,7]])
[6]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
```

The resulting model has a unique actual world and agents have the expected issue relation capturing the natural intuitions, namely, both agents have an issue relation that captures the possible answers but only b can distinguish between the two possibilities, while a is uncertain which of the alternative is the case.

Starting from this first simple illustration, we will proceed towards a more general logical framework in which examples like this can be captured formally. The next step in the standard DEL methodology is to define simultaneously a language that uses modalities over such AIM structures and the structures in which the event-preconditions are formulated in a language describing them.

Let L be a set of atomic sentences closed under boolean operators, epistemic, issue and intersection modalities for a set of agent labels and $[\gamma?]\varphi$ where γ is a deterministic AIM and $\varphi \in L$. The following step in the DEL methodology is to lift the indistinguishability relation from events to action structures. For DEL this was a simple translation from uncertainty between events in a structure with the same domain to an epistemic relation between structures themselves.

Because we have now both epistemic and questioning equivalence relations the task at hand requires more work. Another conceptual difficulty that prevents applying the same strategy comes from the fact that we are already using a reduction. Our AIM structures are meant to capture questioning actions while the domain of an AIM contains not questions but answers to questions. However, we will show that a natural lifting that captures the basic intuitions and keeps the product method unchanged is still possible. For this purpose we will use a standard ordering of questions as sets of answers. This design option has many natural formal properties and also captures the intended intuitions:

3.2.2. Definition. [Lifting] Let $\gamma = \langle E, \overset{a}{\approx}, \overset{a}{\sim}, P, Q\rangle$ and $\gamma' = \langle E', \overset{a}{\approx}', \overset{a}{\sim}', P', Q'\rangle$ be two AIMs, then:

$$\gamma' \overset{a}{\approx} \gamma \quad \text{iff} \quad E = E', P = P' \text{ and } \forall e \in Q, e' \in Q' : e \overset{a}{\approx} e'$$

$$\gamma' \overset{a}{\sim} \gamma \quad \text{iff} \quad E = E', P = P' \text{ and } \forall e \in Q, e' \in Q' : e \overset{a}{\sim} e'$$

We have now all the ingredients for proving the soundness of the axioms given in Section 2.4.3 using the semantics already introduced there. The crucial step used Definition 3.2.1 to restrict the class of structures described.

3.2.1. Proof (Theorem 2.4.8). Let $\gamma = \langle E, \overset{a}{\approx}, \overset{a}{\sim}, P, Q\rangle$ be an AIM. Fix a pair $\langle W, w\rangle$. As γ is a yes/no question, it follows that $W \models_w P(e)$ for some $e \in Q$. Using Definition 3.2.1 we also know that there is a unique such (w, e). Pick the e disjunct in the rhs of the A&P axiom. Assume that $\langle W, w\rangle \otimes \gamma \models Q_a\varphi$. Take some γ' such that $\gamma \overset{a}{\sim} \gamma'$. By Definition 3.2.2, this γ' is of the form $\langle E, \overset{a}{\approx}', \overset{a}{\sim}', P, Q'\rangle$ for some Q' such that $\forall e \in Q, e' \in Q' : e \overset{a}{\sim} e'$. Let $w \overset{a}{\approx} w'$. We have two cases: $Q = Q'$, and $Q \neq Q'$. In both cases we have $e \overset{a}{\approx} e'$ iff $e = e'$. Then (w', e') is a world of $\langle W, w\rangle \otimes \gamma$, and indeed $(w, e) \overset{a}{\approx} (w', e')$. Since Q is a unique set of yes/no answers we have to consider two possibilities for $e \in Q$: if $\langle W, w'\rangle \models \neg\texttt{pre}(e)$ then $\langle W, w'\rangle \models \texttt{pre}(e) \to [\gamma'?]\varphi$ trivially and we are done, otherwise, $\langle W, w'\rangle \models \texttt{pre}(e)$ and we can use the previous assumption. From $\langle W, w\rangle \otimes \gamma \models Q_a\varphi$ we get that $\langle W \otimes \gamma, (w', e')\rangle \models \varphi$. This means that $\langle W, w'\rangle \otimes \gamma' \models \varphi$. Hence $\langle W, w'\rangle \models \texttt{pre}(e) \to [\gamma'?]\varphi$. Since γ' and w' were arbitrary, $\langle W, w\rangle \models \bigwedge_{\gamma? \overset{a}{\sim} \gamma'?} Q_a(\texttt{pre}(e) \to [\gamma'?]\varphi)$. The other direction is similar.

Pick the e disjunct in the rhs of the A&K axiom. Assume that $\langle W, w\rangle \otimes \gamma \models K_a\varphi$. Take some γ' such that $\gamma \overset{a}{\sim} \gamma'$. By Definition 3.2.2, this γ' is of the form $\langle E, \overset{a}{\approx}', \overset{a}{\sim}', P, Q'\rangle$ for some Q' such that $\forall e \in Q, e' \in Q' : e \overset{a}{\sim} e'$. Let $w \overset{a}{\sim} w'$. We have two cases: $Q = Q'$, and $Q \neq Q'$. In both cases we have $e \overset{a}{\sim} e'$. Then (w', e') is a world of $\langle W, w\rangle \otimes \gamma$, and indeed $(w, e) \overset{a}{\sim} (w', e')$. Now our assumption that $\langle W, w\rangle \otimes \gamma \models K_a\varphi$ implies that $\langle W \otimes \gamma, (w', e')\rangle \models \varphi$. This means that $\langle W, w'\rangle \otimes \gamma' \models \varphi$. Hence $\langle W, w'\rangle \models [\gamma']\varphi$. Since γ' and w' were arbitrary, $\langle W, w\rangle \models \bigwedge_{\gamma? \overset{a}{\sim} \gamma'?} K_a[\gamma'?]\varphi$. The other direction is similar.

The rest of the proof continues along the lines of the standard DEL methodology described in the previous section with the only caveat that now we need to define a complexity measure for formulae containing questioning action modalities by taking the complexity of the preconditions. □

We also get the axioms for resolution action for free since we used the most convenient formal model that gives the semantic meaning by taking the previously discussed intersection. Even so, formal convenience does not always match conceptual clarity, we still have an unfulfilled modeling desideratum on our list.

```
*QPR> dpq (exclam (reornameq (upgradeq
  (initMq [a,b] [Syntax.P 0,Q 0]) (initAq [a] [b] [p,q] p))))
[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
[6]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
```

Consider the consequences of taking the undifferentiated epistemic effects of the intersection resolution action in our previous questioning product update model which produce the result represented in the previous code output.

A brief inspection of the resulting model and a basic analysis of its intuitive meaning reveals that it is fair to say that it would be expected that a resolution action can also be modulated epistemicaly. There can be levels of underlying dynamics: some agents can be aware of the fact that a resolution action happens but still be opaque about its content, while for others the content is transparent.

In our example, since *a* was unaware which of the questions was asked, hearing an affirmative answer should not have the same epistemic effect as a transparent answer, or public announcement. Nor should hearing the negative resolution to one of two undistinguished questions have the same effect as receiving a negative answer to any of them. Our rough intersection semantics misses such subtleties.

Once again, we can use the framework of product update to capture the desired epistemic effects. In order to do this we will make crucial use of the gain in expressive power coming from having a language capable of talking about both facts, epistemic realities, questioning conditions and their mutual interdependence. Again, we will proceed first by means of an illustrative example:

```
*QPR> dpq (reornameq (upgradeq
  (initMq [a,b] [Syntax.P 0,Q 0]) (initAq [a] [b] [p,q] p)))
[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,1,2,3,4,5,6,7]])
(b,[[0,2,5,6],[1,3,4,7]])
[6]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
```

The main task is to come up with a formal structure representing a resolution action which captures the natural intuitions about the epistemic evolution. Such a model can be again automatically generated by the implementation in a standard way. The structure we need here is represented by `DELQ.hs` as follows:

```
*QPR> dpa (initAres [a] [b] [p,q] p)
[0,1,2,3]
[(0,&[p,v[Qap,Qa~p]]),(1,&[q,v[Qaq,Qa~q]]),
(2,&[~p,v[Qap,Qa~p]]),(3,&[~q,v[Qaq,Qa~q]])]
(a,[[0,1],[2,3]])
(b,[[0],[1],[2],[3]])
[0]
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])
```

A new feature in comparison to AIMs for questioning actions is the more complex preconditions for resolution, they have now, in addition to the factual component, a questioning aspect designed to capture the conditions on the structure of the issue relation. Next, the resolution does not change the issue structure by raising new questions, therefore both agents have a universal relation. Finally, agent a can only distinguish between affirmative and negative resolution, like when hearing an opaque ‘Yes’ answer in a conversation that could refer to any of two previous questions. We have used agent-dependent epistemic preconditions before, the issue-related content of the preconditions for resolution goes beyond only modeling epistemic effects and factual aspects. In order to capture the intended meaning we add a further generalization to the product update rule. This can now go beyond using only indistinguishability and it is natural to take full advantage of both equivalence relations. The new uncertainty can be now derived from two alternative and complementary sources: both previous uncertainty and issue-indistinguishability. Both aspects and their mutual dependence are needed to capture the dynamics of joint observation and prediction uncertainty.

The rule that makes sense in this context is the following:

$$(w,e)\sim(v,f) \quad \text{iff} \quad w\sim v \text{ and } (w\approx v \text{ or } e\sim f)$$

This rule allows a finer grained modeling of the dependence between two sources of uncertainty in an epistemic scenario, and captures expected intuitions:

```
*QPR> dpq (reornameq (upgraderes (reornameq (upgradeq
  (initMq [a,b] [Syntax.P 0,Q 0]) (initAq [a] [b] [p,q] p)))
  (initAres [a] [b] [p,q] p)))
[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,1,3,5],[2,4,6,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
[6]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
```

The new kind of resolution only partially resolves agent *a*'s prediction uncertainty due to his previous observational limitation about the question asked. He can distinguish the affirmative answers from the negative ones but acquires no further knowledge about the situation. The flow of information is modulated by two distinct sources: previous uncertainty and current observational limitations.

Further gradations in the level of assertive or questioning content implicitly contained in a resolution action can now be spelled out using the same underlying formalism. However, capturing this variation is not a logical task. The underlying logic is given in all cases by the same extended product rule, while the remaining enterprise consists merely in designing the adequate action structures for the concrete resolution scenario to be captured. Witness the following situations:

```
*QPR> dpa (initAres2 [a] [b] [p,q] p)
[0,1,2,3]
[(0,&[p,v[Qap,Qa~p]]),(1,&[q,v[Qaq,Qa~q]]),
(2,&[~p,v[Qap,Qa~p]]),(3,&[~q,v[Qaq,Qa~q]])]
(a,[[0,1,2,3]])
(b,[[0],[1],[2],[3]])
[0]
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])

*QPR> dpa (initAres4 [a] [b] [p,q] p)
[0,1,2,3]
[(0,&[p,v[Qap,Qa~p]]),(1,&[q,v[Qaq,Qa~q]]),
(2,&[~p,v[Qap,Qa~p]]),(3,&[~q,v[Qaq,Qa~q]])]
(a,[[0,2],[1,3]])
(b,[[0],[1],[2],[3]])
[0]
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])

*QPR> dpa (initAres3 [a] [b] [p,q] p)
[0,1,2,3]
[(0,&[p,v[Qap,Qa~p]]),(1,&[q,v[Qaq,Qa~q]]),
(2,&[~p,v[Qap,Qa~p]]),(3,&[~q,v[Qaq,Qa~q]])]
(a,[[0],[1],[2],[3]])
(b,[[0],[1],[2],[3]])
[0]
(a,[[0,1,2,3]])
(b,[[0,1,2,3]])
```

The following code outputs are resulting models capturing the intuitively adequate gradients in information flow predicted by our extended product rule:

```
*QPR> dpq (reornameq (upgraderes (reornameq (upgradeq
  (initMq [a,b] [Syntax.P 0,Q 0]) (initAq [a] [b] [p,q] p)))
  (initAres2 [a] [b] [p,q] p)))
[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,1,2,3,4,5,6,7]])
```

```
(b,[[0,5],[1,3],[2,6],[4,7]])
[6]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])

*QPR> dpq (reornameq (upgraderes (reornameq (upgradeq
  (initMq [a,b] [Syntax.P 0,Q 0]) (initAq [a] [b] [p,q] p)))
  (initAres4 [a] [b] [p,q] p)))
[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,2,5,6],[1,3,4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
[6]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])

*QPR> dpq (reornameq (upgraderes (reornameq (upgradeq
  (initMq [a,b] [Syntax.P 0,Q 0]) (initAq [a] [b] [p,q] p)))
  (initAres3 [a] [b] [p,q] p)))
[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
[6]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
```

The same gradation can be applied to the issue relation, our initial questioning action model was one extreme in a large spectrum of intermediate possibilities of mixing informative and questioning content in various levels. This aspect also opens the possibility of further comparing the DEL approach to alternative approaches of questioning phenomena and to unravel some implicit modeling assumptions in an explicit formal representation.

Another interesting aspect is that the new resolution rule gives a formal way of capturing the widespread phenomenon that much information, even if both truthful and known, might still not be available for announcement. This gives a formal justification for using procedural restrictions in both games with epistemic moves and protocols for inquiry and communication.

So far we have used only Yes/No questions. Doesn't this choice drastically limit our modeling options? The quick answer at this stage is: Not yet. The framework can be extended without any alteration to more complex propositional questions by merely lifting the construction from formulae to lists/sets of formulae. The following code outputs illustrate exactly such situations.

Once again, the extension from binary questions to arbitrary propositional questions falls under the same logical framework using product update. The only extra needed work is an empirical exercise in designing the right AIM for the given scenario to be modeled. This is a very interesting exercise in the 'art of modeling' without significant conceptual or computational consequences.

```
*QPR> dpa (initPropq [a] [b] [[p,Neg p],[Conj [p,q],Conj[Neg p, Neg q],
Disj [Conj [p, Neg q], Conj [q, Neg p]], Conj[Neg p, Neg q]]]
[Conj [p,q],Disj[Conj [p, Neg q], Conj [q, Neg p]]]])

[0,1,2,3,4]
[(0,p),(1,~p),(2,&[p,q]),(3,v[&[p,~q],&[q,~p]]),(4,&[~p,~q])]
(a,[[0,1,2,3,4]])
(b,[[0,1],[2,3,4]])
[2,3,4]
(a,[[0],[1],[2],[3],[4]])
(b,[[0],[1],[2],[3],[4]])

*QPR> dpq (reornameq (upgradeq (initMq [a,b] [Syntax.P 0,Q 0])
(initPropq [a] [b] [[p,Neg p],[Conj [p,q],
Disj[Conj [p, Neg q], Conj [q, Neg p]],Conj[Neg p, Neg q]]]
[Conj [p,q],Disj[Conj [p, Neg q], Conj [q, Neg p]],Conj[Neg p, Neg q]]])))

[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,1,2,3,4,5,6,7]])
(b,[[0,2,4,6],[1,3,5,7]])
[7]
(a,[[0,4],[1],[2,6],[3,5],[7]])
(b,[[0,4],[1],[2,6],[3,5],[7]])
```

The crucial fact here is the requirement that the designed AIM is an adequate one. If it respects the postulate of determinacy and uses appropriate sets of mutually inconsistent answers, then all the formal details can be adapted without ado, and all results transfer in corresponding reformulations.

Discussion: Nondeterminism What about modeling questions that do not have mutually inconsistent answers? In other words, does the DEL product upgrade mechanism still work when we switch from a classical model of questions modeled as partitions to one which has a cover as the underlying formal structure? We will discuss the conceptual implications of this change and we will show that the formal product mechanism still works under minor adaptations.

So we are still working with a unique actual world in our initial EIMs, but now we take a question to be an arbitrary set of formulae seen as subsets of the domain of possibilities, including ones with a nonempty intersection.

Therefore, our previous AIM structures in which we take as the designated set of actual events all the answers to the actual question can generate multiple actual worlds in the model obtained after performing product update.

It can be argued that this is meaningful as a representation of nondeterminism at a subjective level in many scenarios of social interaction, or even in an objective sense in contexts of dependence between measurements of entangled states in quantum physics. However, we identify it as an open problem weather the formal details needed in the proofs can be adapted for a non-deterministic setting, and reserve a study of valid principles for nondeterminism for a future occasion.

Our next step will be to illustrate how our previous mechanism works if we change the questioning structures to ones having both a set of events representing the real question and a designated event for the actual answer. As in the standard DEL approach, and as we did before for issue relations, we use preconditions for action execution to reduce multiple consistent answers to questioning actions with unique effects. This is a good place to illustrate how model checking of formulae involving epistemic and issue effects is performed in our Haskell implementation.

```
*DELQ> dpq m1
[0,1,2,3]
[(0,[n]),(1,[p,n1]),(2,[p,n2]),(3,[q,n3])]
[0,1,2,3]
(a,[[0,1,3],[2]])
(a,[[0,2,3],[1]])
```

The truth-value of formulas can be computed at a state in an epistemic-issue model. The following is an illustration of how `DELQ.hs` performs model checking in the issue-epistemic model `m1` represented in the code output above:

```
*DELQ> models m1 0 (impl (x_ a p) (Conj[o_ a p,k_ a p]))
True
*DELQ> models m1 0 (dimpl (x_ a p) (Conj[o_ a p,k_ a p]))
False
*DELQ> models m1 0 (impl (x_ a q) (Conj[o_ a q,k_ a q]))
True
*DELQ> models m1 0 (dimpl (x_ a (Conj[n3,q]))
                     (Conj[o_ a (Conj[n3,q]),k_ a (Conj[n3,q])]))
True
*DELQ> models m1 0 (Conj
  [o_ a (Conj[n2,p]), k_ a (Conj[n1,p])])
True
*DELQ> models m1 0 (Conj
  [o_ a (Conj[n2,p]), k_ a (Conj[n1,p]), x_ a p])
False
```

Now, we can attach preconditions for mutually consistent answers to distinct events in our action models and we obtain the expected results. The only necessary adjustment is to use a singleton set of real answers in the questioning model during the product computation. In this way the class of deterministic EIMs remains closed under product update. Here is how this can be done for a disjunctive question that can receive three overlapping answers:

```
*QPR> dpa (initPropCov [a] [b]
  [[Disj[p,q],p,q],[Disj[p,Neg q],p,Neg q]] [Disj[p,q],p,q] p)
[0,1,2,3,4,5]
[(0,v[p,q]),(1,p),(2,q),(3,v[p,~q]),(4,p),(5,~q)]
(a,[[0,1,2,3,4,5]])
(b,[[0,1,2],[3,4,5]])
```

```
[1]
(a,[[0],[1],[2],[3],[4],[5]])
(b,[[0],[1],[2],[3],[4],[5]])

*QPR> dpq (reornameq (upgradeq (initMq [a,b] [Syntax.P 0,Q 0])
(initPropCov [a] [b] [[Disj[p,q],p,q],[Disj[p,Neg q],p,Neg q]]
[Disj[p,q],p,q] p)))
[0,1,2,3,4,5,6,7,8,9,10,11,12,13]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[p]),(5,[p]),(6,[p]),(7,[q]),(8,[q]),
 (9,[p,q]),(10,[p,q]),(11,[p,q]),(12,[p,q]),(13,[p,q])]
(a,[[0,1,2,3,4,5,6,7,8,9,10,11,12,13]])
(b,[[0,1,4,5,6,12,13],[2,3,7,8,9,10,11]])
[10]
(a,[[0,4,12],[1,6],[2,7,9],[3,10],[5,13],[8,11]])
(b,[[0,4,12],[1,6],[2,7,9],[3,10],[5,13],[8,11]])
```

This setting can also be used to obtain completeness results about long term inquiry procedures. The discussion so far about adapting the proof techniques from informative actions to questioning actions is equally relevant in a long term protocols setting. We give here the proof for the basic case with binary questions.

3.2.2. Proof (Lemma 2.6.8). By structural induction on $\varphi \in \mathcal{L}_{DELQ}$. Base case: straightforward. Boolean cases: straightforward. We start from the right to left direction: For each $h \in \mathsf{H}^{\text{can}}$:

$$\text{if} \quad \mathcal{H}^{\text{can}}, h \models \varphi \quad \text{then} \quad \varphi \in \lambda(h)$$

We proceed by considering case by case the most relevant situations:

1. Question/Issue Modality Case:

   (a) Questioning Action Case:

   We start by recalling some relevant valid principles in this context, that we will use later on:

   Axiom 1 (Protocol Asking and Negation):

   $$\langle\gamma?\rangle\neg\varphi \leftrightarrow \langle\gamma?\rangle\top \wedge \neg\langle\gamma?\rangle\varphi$$

   Axiom 2 (Protocol Asking and Partition):

   $$\langle\gamma?\rangle Q_i\varphi \leftrightarrow \langle\gamma?\rangle\top \wedge ((\gamma \wedge Q_i(\gamma \to \langle\gamma?\rangle\varphi)) \vee (\neg\gamma \wedge Q_i(\neg\gamma \to \langle\gamma?\rangle\varphi)))$$

   Take an arbitrary history $h \in \mathsf{H}^{\text{can}}$. Assume that $Q_i\psi \notin \lambda(h)$ (1). For simplicity let $h = w\gamma?$ with $w \in W_0$ and $\gamma \in \mathcal{L}_{\text{EL}}$. The argument can easily be generalized to deal with the general case along the lines of the argument below.

By λ being a MCS: (1) $\Rightarrow \neg Q_i\psi \in \lambda(h)$ (2). By the construction of legal histories in Definition 2.6.6 we have: (2) $\Rightarrow \langle\gamma?\rangle\neg Q_i\psi \in \lambda(w)$(3). By Axiom 1 above: (3) $\Rightarrow \neg\langle\gamma?\rangle Q_i\psi \in \lambda(w)$(4) By Axiom 2 above and using de Morgan equations we have: (4) $\Rightarrow \neg\langle\gamma?\rangle\top \vee \neg((\gamma \wedge Q_i(\gamma \to \langle\gamma?\rangle\psi)) \vee (\neg\gamma \wedge Q_i(\neg\gamma \to \langle\gamma?\rangle\psi))) \in \lambda(w)$ (5)
By Construction: $\langle\gamma?\rangle\top \in \lambda(w)$ (6) By Disjunctive Syllogism: (5)&(6) $\Rightarrow \neg((\gamma \wedge Q_i(\gamma \to \langle\gamma?\rangle\psi)) \vee (\neg\gamma \wedge Q_i(\neg\gamma \to \langle\gamma?\rangle\psi))) \in \lambda(w)$ (7) By de Morgan equivalence: (7) $\Rightarrow \neg(\gamma \wedge Q_i(\gamma \to \langle\gamma?\rangle\psi)) \wedge \neg(\neg\gamma \wedge Q_i(\neg\gamma \to \langle\gamma?\rangle\psi)) \in \lambda(w)$(8) By de Morgan again we have: (8) $\Rightarrow (\neg\gamma \vee \neg Q_i(\gamma \to \langle\gamma?\rangle\psi)) \wedge (\gamma \vee \neg Q_i(\neg\gamma \to \langle\gamma?\rangle\psi)) \in \lambda(w)$ And we can rewrite this as:

$$(\neg\gamma \wedge \neg Q_i(\neg\gamma \to \langle\gamma?\rangle\psi)) \vee$$
$$(\neg\gamma \wedge \gamma) \vee (\neg Q_i(\gamma \to \langle\gamma?\rangle\psi) \wedge \gamma) \vee$$
$$(\neg Q_i(\gamma \to \langle\gamma?\rangle\psi) \wedge \neg Q_i(\neg\gamma \to \langle\gamma?\rangle\psi)) \in \lambda(w)$$

We proceed by considering the disjuncts case by case:

Case 1: The $\gamma \wedge \neg\gamma \in \lambda(w)$ case is ruled out by Lemma 2.6.7.

Case 2: If we have $\neg\gamma \wedge \neg Q_i(\neg\gamma \to \langle\gamma?\rangle\psi) \in \lambda(w)$ then we can make the sub-claim:

The following is a consistent set:

$$v_0 = \{\theta \mid Q_i\theta \in \lambda(w)\} \cup \{\neg(\neg\gamma \to \langle\gamma?\rangle\psi)\}$$

Suppose not. Then there are formulae: $\theta_1, \dots, \theta_m$ such that:

$$\vdash \bigwedge_{j=1}^{m} \theta_j \to (\neg\gamma \to \langle\gamma?\rangle\psi)(12)$$

and for $j = 1, \dots, m, Q_i\theta_j \in \lambda(w)$. Then, by standard modal reasoning (Necessitation and Distribution of Q) we have:

$$(12) \Rightarrow \vdash \bigwedge_{j=1}^{m} Q_i\theta_j \to Q_i(\neg\gamma \to \langle\gamma?\rangle\psi)(13)$$

Hence, we also have: (13) $\Rightarrow Q_i(\neg\gamma \to \langle\gamma?\rangle\psi) \in \lambda(w)$(14)
Because λ is a MCS: (14) contradicts (Case 2).

By Lindenbaum's Lemma any consistent set can be extended to a MCS. Hence there exist a MCS v such that $v_0 \subseteq v$ (9). Therefore, by construction we have: (9) $\Rightarrow w \sim_i^0 v$ and $w\gamma? \sim_i^{\text{can}} v\gamma?$ (10) Hence $\Rightarrow \langle\gamma?\rangle\top \in \lambda(v)$ and $\neg\langle\gamma?\rangle\psi \in \lambda(v)$ (11) and by the Axiom 1: (11) $\Rightarrow \langle\gamma?\rangle\neg\psi \in \lambda(v)$. By LH construction: (11) $\Rightarrow \neg\psi \in \lambda(v\gamma?)$(12), hence (12) $\Rightarrow \psi \notin \lambda(v\psi?)$(13). By induction hypothesis: (13) $\Rightarrow \mathcal{H}^{\text{can}}, v\psi? \not\models \psi$ (14) and by standard modal semantics: (14) $\Rightarrow \mathcal{H}^{\text{can}}, w\gamma? \not\models Q_i\psi$, as desired.

Case 3: If we have $(\neg Q_i(\gamma \rightarrow \langle\gamma?\rangle\psi) \wedge \gamma) \in \lambda(w)$ then we can reason completely analogously to case 2.

Case 4: If we have $(\neg Q_i(\gamma \rightarrow \langle\gamma?\rangle\psi) \wedge \neg Q_i(\neg\gamma \rightarrow \langle\gamma?\rangle\psi)) \in \lambda(w)$ we can use case 2 and case 3 to reach the desired conclusion.

(b) Resolution Action Case:
We start by recalling some relevant valid principles in this context:
Axiom 3 (Protocol Resolution and Negation):

$$\langle\,!\,\rangle\neg\varphi \leftrightarrow \langle\,!\,\rangle\top \wedge \neg\langle\,!\,\rangle\varphi$$

Axiom 4 (Protocol Resolution and Partition):

$$\langle\,!\,\rangle Q_i\varphi \leftrightarrow \langle\,!\,\rangle\top \wedge Q_i\langle\,!\,\rangle\varphi$$

This case is a situation of a commutating behavior so we are done.

2. Intersection Modality Case: Behaves completely similar to the already explained Question/Issue modality for both cases (a) and (b) below:
   (a) Questioning Action Case and
   (b) Resolution Action Case.

3. Epistemic/Knowledge Modality Case:

   (a) Questioning Action Case:
   We start by recalling some relevant valid principles in this context:
   Axiom 1 (Protocol Asking and Negation):

$$\langle\gamma?\rangle\neg\varphi \leftrightarrow \langle\gamma?\rangle\top \wedge \neg\langle\gamma?\rangle\varphi$$

   Axiom 6 (Protocol Asking and Knowledge):

$$\langle\gamma?\rangle K_i\varphi \leftrightarrow \langle\gamma?\rangle\top \wedge K_i\langle\gamma?\rangle\varphi$$

   This case is a situation of a commutating behavior so we are done.

   (b) Resolution Action Case:
   We start by recalling some relevant valid principles in this context, that we will use later on:
   Axiom 3 (Protocol Resolution and Negation):

$$\langle\,!\,\rangle\neg\varphi \leftrightarrow \langle\,!\,\rangle\top \wedge \neg\langle\,!\,\rangle\varphi$$

   Axiom 7 (Protocol Resolution and Knowledge):

$$\langle\,!\,\rangle K_i\varphi \leftrightarrow \langle\,!\,\rangle\top \wedge R_i\langle\,!\,\rangle\varphi$$

This is an interesting combination as it relies essentially on the intersection modality so we will give it closer attention.

Take an arbitrary history $h \in \mathsf{H}^{\text{can}}$. Assume that $K_i\psi \notin \lambda(h)$ (1). For simplicity let $h = w!$ with $w \in W_0$ and ! a resolution symbol. The argument can easily be generalized to deal with the general case along the lines of the argument below.

By λ being a MCS: (1) $\Rightarrow \neg K_i\psi \in \lambda(h)$ (2). By the construction of legal histories in Definition 2.6.6 we have: (2) $\Rightarrow \langle\,!\,\rangle\neg K_i\psi \in \lambda(w)(3)$. By Axiom 3 above: (3) $\Rightarrow \neg\langle\,!\,\rangle K_i\psi \in \lambda(w)(4)$ By Axiom 7 above we have: (4) $\Rightarrow \neg(\langle\,!\,\rangle\top \wedge R_i\langle\,!\,\rangle\psi) \in \lambda(w)$ (5). Using de Morgan: (5) $\Rightarrow \neg\langle\,!\,\rangle\top \vee \neg R_i\langle\,!\,\rangle\psi \in \lambda(w)$ (5).

By Construction: $\langle\,!\,\rangle\top \in \lambda(w)$ (6) By Disjunctive Syllogism: (5) and (6) $\Rightarrow \neg R_i\langle\,!\,\rangle\psi \in \lambda(w)$ (7). We make the next sub-claim:

The following set is consistent:

$$v_0 = \{\theta \mid R_i\theta \in \lambda(w)\} \cup \{\neg\langle\,!\,\rangle\psi\}$$

Suppose not. Then there are formulae: $\theta_1, \ldots, \theta_m$ such that:

$$\vdash \bigwedge_{j=1}^{m} \theta_j \rightarrow \neg\langle\,!\,\rangle\psi(12)$$

and for $j = 1, \ldots, m, R_i\theta_j \in \lambda(w)$. Then, by standard modal reasoning (Necessitation and Distribution of Q) we have:

$$(12) \Rightarrow \vdash \bigwedge_{j=1}^{m} R_i\theta_j \rightarrow R_i\neg\langle\,!\,\rangle\psi(13)$$

Hence, we also have: (13) $\Rightarrow R_i\neg\langle\,!\,\rangle\psi \in \lambda(w)(14)$

Because λ is a MCS: (14) contradicts the initial assumption (1).

By Lindenbaum's Lemma any consistent set can be extended to a MCS. Hence there exist a MCS v such that $v_0 \subseteq v$ (9). Therefore, by construction we have: (9) $\Rightarrow w \sim_i^0 \cap \approx_i^0 v$ and from this it follows by construction that $w! \sim_i^{\text{can}} \cap \approx_i^{\text{can}} v!\,(10)$. Hence $\Rightarrow \langle\,!\,\rangle\top \in \lambda(v)$ and $\neg\langle\,!\,\rangle\psi \in \lambda(v)$ (11) and by Axiom 3: (11) $\Rightarrow \langle\,!\,\rangle\neg\psi \in \lambda(v)$. By LH construction: (11) $\Rightarrow \neg\psi \in \lambda(v!)(12)$, hence (12) $\Rightarrow \psi \notin \lambda(v!)(13)$. By induction hypothesis: (13) $\Rightarrow \mathcal{H}^{\text{can}}, v! \not\models \psi$ (14) and by standard modal semantics: (14) $\Rightarrow \mathcal{H}^{\text{can}}, w! \not\models R_i\psi$ and $K_i\psi$, as desired.

We continue now with the remaining converse direction. Base case: straightforward. Boolean cases: straightforward. We will consider the interesting cases involving modalities in more detail next.

1. Epistemic/Knowledge Modality Case:

   (a) Resolution Action Case: Let us start by recalling some relevant valid principles in this context, that will be useful later in the proof:
   Axiom 7 (Protocol Resolution and Knowledge):

   $$\langle\,!\,\rangle K_i\varphi \leftrightarrow \langle\,!\,\rangle\top \wedge R_i\langle\,!\,\rangle\varphi$$

   Take an arbitrary history $h \in \mathsf{H}^{\text{can}}$ such that $h = w!$. The argument can easily be generalized to deal with the general case $h = w!_1 \cdots !_n$. along the lines of the argument below.

   Assume that: $K_i\psi \in \lambda(h)$ (1). Suppose: $h' \in \mathsf{H}^{\text{can}}$ and $h \sim_i h'$ (2). By CM construction: (2) $\Rightarrow h' = v!$, for some $v \in H_0$ with $w \sim_i^0 v$. By LH construction: (1) $\Rightarrow \langle\,!\,\rangle K_i\psi \in \lambda(w)$ (4). By the previously introduced Axiom 7: (4) $\Rightarrow (\langle\,!\,\rangle\top \wedge R_i\langle\,!\,\rangle\psi) \in \lambda(w) = w$ (5). By CM construction: (2) $\Rightarrow w \sim_i^0 v$ (6). Using the modal semantics of R_i: (5) and (6) $\Rightarrow \langle\,!\,\rangle\psi \in v = \lambda(v)$ (7). Using the LH construction we have: (7) $\Rightarrow \psi \in \lambda(v!) = \lambda(h')$ (8). From the Induction Hypothesis it follows that: $\mathcal{H}^{\text{can}}, h' \models \psi$. Therefore, by Modal Semantics we get $\mathcal{H}, h \models K_i\psi$ as desired.

   (b) Questioning Action Case: Let us start by recalling some relevant valid principles in this context: Axiom 6 (Protocol Asking and Knowledge):

   $$\langle\varphi?\rangle K_i\psi \leftrightarrow \langle\varphi?\rangle\top \wedge K_i\langle\varphi?\rangle\psi$$

   This case is a situation of a commutating behavior so we are done.

2. Intersection Modality Case:

   (a) Questioning Action Case:
   Let us start by recalling some relevant valid principles in this context, that will be useful later on during the proof:
   Axiom 8 (Protocol Asking and Intersection):

   $$\langle\gamma?\rangle R_i\varphi \leftrightarrow \langle\gamma?\rangle\top \wedge ((\gamma \wedge R_i(\gamma \rightarrow \langle\gamma?\rangle\varphi)) \vee (\neg\gamma \wedge R_i(\neg\gamma \rightarrow \langle\gamma?\rangle\varphi)))$$

   Take an arbitrary history $h \in \mathsf{H}^{\text{can}}$ such that $h = w\gamma?$. The argument can be generalized to deal with the general case $h = w\gamma_1? \cdots \gamma_n?$. along the lines of the argument below. This generalization essentially relies on having a *unique* choice of the relevant disjunct in every component in the long term history consisting of a finite sequence of questioning actions. The availability of such a unique choice is ensured by Definition 3.2.1.

Assume that: $R_i\psi \in \lambda(h)$ (1). Suppose that we have: $h' \in \mathsf{H}^{\text{can}}$ and $h \sim_i^0 \cap \approx_i^0 h'$ (2). By CM construction: (2) $\Rightarrow h' = v\gamma?$, for some $v \in H_0$ such that $w \sim_i^0 \cap \approx_i^0 v$. From the LH construction it follows that: (1) $\Rightarrow \langle\gamma?\rangle R_i\psi \in \lambda(w)$ (4). By the previously introduced Axiom 8: (4) $\Rightarrow (\langle\gamma?\rangle\top \wedge ((\gamma \wedge R_i(\gamma \rightarrow \langle\gamma?\rangle\psi)) \vee (\neg\gamma \wedge R_i(\neg\gamma \rightarrow \langle\gamma?\rangle\psi)))) \in \lambda(w) = w$. Hence we have either $(\gamma \wedge R_i(\gamma \rightarrow \langle\gamma?\rangle\psi) \in \lambda(w) = w$ or $(\neg\gamma \wedge R_i(\neg\gamma \rightarrow \langle\gamma?\rangle\psi)) \in \lambda(w) = w$. All we have to do at this stage is to pick the right disjunct, because of Definition 3.2.1 we know there is such a disjunct and the result of the choice function is well defined. Because λ is a MCS either one or the other is the case. Here γ is a yes/no question but the argument can also be extended along the same lines to arbitrary partition-based propositional questions.

Assume wlog: $(\neg\gamma \wedge R_i(\neg\gamma \rightarrow \langle\gamma?\rangle\psi)) \in \lambda(w) = w$ (5). By CM construction: (2) $\Rightarrow w \sim_i^0 \cap \approx_i^0 v$ (6). Using the modal semantics of R_i: (5) and (6) $\Rightarrow \langle\gamma?\rangle\psi \in v = \lambda(v)$ (7). Using the LH construction we have: (7) $\Rightarrow \psi \in \lambda(v\gamma?) = \lambda(h')$ (8). From the Induction Hypothesis it follows that: $\mathcal{H}^{\text{can}}, h' \models \psi$. Therefore, by Modal Semantics we get $\mathcal{H}, h \models R_i\psi$ as desired.

(b) Resolution Action Case:

Let us start by recalling some relevant valid principles in this context: Axiom 6 (Protocol Resolving and Intersection):

$$\langle\,!\,\rangle R_i\varphi \leftrightarrow \langle\,!\,\rangle\top \wedge R_i\langle\,!\,\rangle\varphi$$

This case is a situation of a commutating behavior so we are done.

3. Issue/Question Modality Case: Behaves completely similar to the already explained Intersection modality for both cases of (a) Questioning Action and (b) Resolution Action.

This concludes the proof. □

Chapter 4

Games with Questioning Moves

We have introduced in the previous chapter a general logical framework for questions, information flow, and their mutual interdependence. In the present chapter we enrich our study with strategic considerations. We will see that adding a strategic aspect to questioning activities is a natural extension of the logical framework which fits with many intuitive scenarios and captures concrete aspects of rational interaction in many practical applications. We will also show that the game theoretic twist enriches with new theoretical content traditional notions used to describe questions and results about epistemic and interactive scenarios.

This enrichment also fulfills several desiderata already encompassing our approach. A dynamic logic of questions is designed to model interactive scenarios in which agents perform questioning moves and aim to achieve epistemic goals. The main modeling desiderata that we will address are: to develop a formal model of questioning actions using an issue relation and to study the epistemic and strategic effects of such questioning actions in a game theoretical setting.

Approaches that combine logical and game theoretic aspects for informative epistemic actions are not new, we start our chapter with a brief account of previous approaches inside the DEL paradigm. Next we will focus on showing how these classical approaches designed to model informative actions can be extended to model questioning actions in interactive scenarios.

We proceed by first introducing strategic games in which moves are question-answer pairs. We give the basic definitions of such games and then proceed to consider results about existence of solution concepts and analysis of concrete examples. Next we introduce extensive games in which moves are questioning actions whose effects are represented by an issue relation. We continue with studying solution concepts and strategic abilities in epistemic games with questioning moves and general game properties useful to describe them.

4.1 Brief History of Epistemic Games

Several approaches that combine a logic and epistemic approach with a game-theoretic approach for modeling scenarios of rational interaction and communication involving directly or indirectly questioning actions have been previously considered in the literature [50, 77, 80]. We start by briefly surveying the main approaches inside the dynamic epistemic logic paradigm [102], [8], and [2].

4.1.1 Knowledge Games with Epistemic Moves

A first relevant contribution inside the broader DEL paradigm approaches informative epistemic moves in a general game-theoretic setting [102], and defines and studies games with epistemic moves. In brief, this approach starts by considering a set of cards $\mathbf{C}$ and a set of agent-labels $\mathbf{A}$. Let $\mathsf{d} \in \mathbf{A}^{\mathbf{C}}$ be a deal of cards, then the *initial state of a game* for the actual deal of cards d is:

$$(\langle D_{\sharp\mathsf{d}}, (\sim_a)_{a\in\mathbf{A}}, V\rangle, \mathsf{d}), \text{ where } \forall a \in \mathbf{A} : \forall d_1, d_2 \in D_{\sharp\mathsf{d}} : d_1 \sim_a d_2 \Leftrightarrow \\ d_1^{-1}(a) = d_2^{-1}(a) \text{ and } \forall e \in D_{\sharp\mathsf{d}} : \forall c_a \in \mathbf{P} : V_e(c_a) = 1 \Leftrightarrow e(c) = a$$

A *knowledge game state* for cards-deal d is an $S5$ model $(\langle W, (\sim_a)_{a\in\mathbf{A}}, V\rangle, v)$, such that $v \in W$, V_{d}, and: $\forall w \in W : \exists d' \in D_{\sharp\mathsf{d}} : V_w = V_{d'}$, and for all $a \in \mathbf{A}$: $\forall w_1, w_2 \in W$: $\forall d_1, d_2 \in D_{\sharp\mathsf{d}}$: $(w_1 \sim_a w_2, V_{w_1} = V_{d_1}, V_{w_2} = V_{d_2}) \Rightarrow d_1^{-1}(a) = d_2^{-1}(a)$.

Let $(\langle W, (\sim_a)_{a\in\mathbf{A}}, V\rangle, v)$ be a knowledge game state for $d \in \mathbf{A}^{\mathbf{C}}$, then $D_s = \{d' \in D_{\mathsf{d}} \mid \exists w \in W : w \sim_{\mathbf{A}} v, V_w = V_{d'}\}$ are the epistemically relevant card-deals.

Let $(\langle W, (\sim_a)_{a\in\mathbf{A}}, V\rangle, \mathsf{d})$ be a knowledge game state. A *game action* μ for state s is a quintuple $\mu = \langle q, Q, r, R, \mathsf{pub}\rangle$, where $q, r \in \mathbf{A}, Q$ is a covering of W that is coarser than $\sim_r$, $R \in Q$ and pub_r is the identity '=' on Q.

Let $s = (\langle W, (\sim_a)_{a\in\mathbf{A}}, V\rangle, w)$ and $\mu = \langle q, Q, r, R, \mathsf{pub}\rangle$. The game action μ is *executable* in game state s if the answer R contains actual world w, i.e., if r's information state $[w]_{\sim_r}$ is contained in the answer-set.

This framework also offers the possibility to describe epistemic actions as they appear to groups of agents in interaction. For instance, when an action is accessible to all agents in a subgroup $B \in \mathbf{A}$ this can be represented as $\mathsf{pub}_B = (\bigcup_{a\in B} \mathsf{pub}_a)^*$. The equivalence class of $\sim_B$ that contains the answer Q stands for what subgroup B learns in that game action.

4.1.2 Time-stamped Questions in Communication

The second account that we will consider here was introduced in [8]. Here a general setting of 'communication acts' is considered: $[\sigma\vec{\varphi}]\psi$ saying intuitively that "after the communication act $\sigma\vec{\varphi}$ is performed at the current state, ψ will be true at the output sate". 'Abstract dialogues' or 'communication sequences' are defined as words over the alphabet Act^*_Σ of all communications acts considered.

Interrogatives, or queries are in this setting a particular kind of communication acts. For instance, the 'public questioning of agent b by agent a' is of type $PubQ_i(a,b)$ having the following structure: $\Sigma_1 = \{PubQ_a\}, PubQ_i(a,b)_c = \{PubQ_i(a,b)\}$ for all agents $c \in Ag$, $PRE_{PubQ_i(a,b)} = \emptyset$, and $CON_{PubQ_i(a,b)} = ?_i(a,b,Ag)$. The resulting communication act is the act of 'agent a publicly asking b weather φ is true or not'. The notable feature here is the fact that such communication acts come with a 'timestamp i' which will become relevant later on when the communication acts are going to be cast as legal or illegal '(communication) moves' in a 'dialogue game'. Only moves that answer questions that have been previously time-stamped are going to be legal while moves that also consider the later dynamics are not allowed. Many other kinds of questioning moves can be captured in this setting using more and more complex preconditions for execution and/or the publicity of the action, the security of the communication channel, etc. For instance, questions are qualified as 'normal' and 'abnormal', 'private' and 'public', 'secure' or 'intercepted', Socratic, etc.

In this setting, a 'dialogue game' is a pair $G = (S_G, Act_G)$ containing a set of initial epistemic states and available moves, both finite. A 'legal dialogue' is a string $s_0, \alpha_0, s_1, \alpha_1, \ldots, s_n\alpha_n$ of time-stamped epistemic states and communication moves that have to satisfy various reasonable conditions such as, in the case of questioning moves, responsiveness, sincerity, etc.

4.1.3 Public Announcement Games

A more recent approach that models games in which moves are public announcements is [2]. In this framework, the modeling starts from an epistemic structure over two agents. Moves in the game are public announcements. The result of an announcement is the epistemic structure updated with the conjunction of announcement formulae for the two agents. The goals of the agents are represented as epistemic formulae. The payoff for the agents is derived by model checking goal formulae in the output epistemic structures. Both pointed games, which are played locally at a world in the model, and induced games, which are necessary because the agents themselves are uncertain about the actual situation, are defined and analyzed in this setting. The main contribution of this chapter will be to add questioning moves to this formal framework and to study both logical and game theoretic notions and formal properties in settings with questioning moves.

The connection between this approach and question-answer games is very close. While many of the definitions are similar, there are interesting differences already emerging at this level, both conceptually and in terms of basic results. A complete and detailed comparison is beyond the scope of this brief introduction. We include a more detailed comparison starting from and using implementation tools in Chapter 5 and proceed now to introduce definitions for question-answer games in all details as well as illustrations and basic results about them.

4.2 Question-Answer Games

Introduction In general, there is a very close relation between questions and answers. This intimate connection is manifest in many of the historical approaches already discussed and will also be the starting point in the approach we will follow in this chapter. We start with very basic strategic games that have question-answer pairs as constitutive moves. We then explore more complex versions that allow for more realistic features, including more complex questioning actions and sources of information. We then continue our study with a setting of extensive games which have questioning moves as their basic components.

The connection between strategic aspects and information content in questioning activities has been the topic in various studies about games that involve questions and answers such as, for instance: [30, 62]. Moreover, relations between game theory and pragmatic phenomena like the information content of questions and their relevance for solving decision problems have also received attention before, for instance in [110]. Such approaches have revealed close connection between an analysis of questions, information theory and related concepts such as entropy and relevance. However, such approaches usually only consider questioning actions from a single-agent perspective. Following a long tradition, they do not usually involve strategic answers to questions asked to or by other agents. We aim at using an epistemic analysis in game theoretic style in order to integrate such aspects in a broader multi-agent perspective on questioning scenarios.

Questioning Let us give more content to this general perspective by starting from concrete examples. Consider two agents a, and b, and a relevant propositional symbol p. The other alternative, is therefore represented by $\neg p$.

There are several pragmatic preconditions included in the question 'p?' being asked by an agent. She should not know the truth about p, i.e., she does not know p, and she does not know $\neg p$. We are assuming a multi-agent epistemic logic to model questions, where the expression $K_a p$ stands for 'a knows p,' and where the epistemic modality K_a is interpreted with an equivalence relation $\sim_a$. This pragmatic constraint therefore amounts to the precondition $\neg K_a p \wedge \neg K_a \neg p$. The agent asking the question also has expectations about the agent that will answer the question, and that constitutes another pragmatic precondition. She considers it possible that he knows the answer, i.e., $\neg K_a \neg(K_b p \vee K_b \neg p)$. (She also considers it *likely* that he knows the answer, i.e., she believes that tentatively, $B_a(K_b p \vee K_b \neg p)$, where belief and knowledge are combined as in [64]; we may also see that as a combination of preferences and knowledge [99].)

A question φ? splits the domain in the set of states $[\![\varphi]\!]$ where φ is true and the set of states $[\![\neg\varphi]\!]$ where φ is false, i.e., the question induces a dichotomy on the domain of the model. In the approach from the previous chapter, as well as in [100], a question φ? is represented by the *issue* relation $\approx_\varphi$. Such issue relations are natural and intuitive representations for semantic effects of questioning actions in a universe of possible worlds, their use goes back to [40, 56]. We will start this chapter by introducing a model that does not use such issue relations and we will return to them later on. To a certain extent a dichotomy can be also represented as two model restrictions corresponding to the possible answers to the questions modeled as announcements, in standard public announcement logic. Some interesting features that are specific for questions and deserve a minute analysis emerge already at this stage, even without considering an issue relation.

Answers to Questions If a question is addressed to another agent, which might have his own epistemic limitations, there are three possible answers: 'Yes', 'No', and 'I don't know'. A fourth possibility, which is, however, less relevant as an epistemic action, would be: 'I decline to answer the question'. Such a response does not convey an epistemic content. Therefore we will focus only on the three different answers that do have an epistemic content. Neither of these is less or more informative than required in a multi-agent context.

When a question is addressed to an agent b an answer corresponds to the (unique) largest union of b equivalence classes representing the knowledge that *is contained in* the φ-states of the model. This is of course exactly the denotation of the formula $K_b\varphi$. If we make the reasonable assumption that questions are answered truthfully, answers can be encoded as a *public announcement* $K_b\varphi!$ [79].

Similarly, if the answer is "No, I don't" this can be thought as announcing the formula $K_b\neg\varphi$ and its denotation is the complement of the (unique) smallest union of equivalence classes that *contains* the φ-states. The answer "I don't know" results in the remainder, i.e. the union of all $\sim_a$ classes that properly intersect with $\approx_\varphi$. The 'don't know' answer can be conceived as announcement of the formula $\neg K_b\varphi \wedge \neg K_b\neg\varphi$ and this might also fit into the form of an announcement, namely $K_b(\neg K_b\varphi \wedge \neg K_b\neg\varphi)$, so indeed this must also be a union of b-equivalence classes.

This in fact shows that answers to questions can be seen as *rough sets* [75]. Given the set $[\![\varphi]\!]$ (i.e., the subset of the domain consisting of the φ-states), in rough set terms known as the *target*, take the lower and upper $\sim_b$ approximation of the target, i.e., $\underline{\sim_b}([\![\varphi]\!])$ and $\overline{\sim_b}([\![\varphi]\!])$. If the answer to the question is 'yes', the actual state is in the lower approximation. If the answer is 'no', the actual state is in the complement of the upper approximation. If the answer is 'don't know', the actual state is in the upper approximation minus the lower approximation. See also Figure 4.1 later in the section for an intuitive illustration.

In dynamic epistemic logic, a public announcement is interpreted as a model restriction. Therefore, answering the question can be seen as executing one of three possible such restrictions, a non-deterministic program so to speak.

Games with Questions and Answers Extending all this into a game theoretical scenario is as natural as it can be. The only aspect that needs clarification is the goal of the game. What are the goals of the players? In the case of asking p, it is most natural to think of the goal as achieving knowledge about p: $K_a p \vee K_a \neg p$. But in this case it is not so clear if there is an opponent. Clearly if the agent to whom the question is addressed has no interest in the issue he will answer the question, but he will not be playing a game just yet.

But scenarios featuring genuine strategic aspects are frequent in many competitive situations involving questions in a multi-agent setting. Take as a quick illustration a scenario in which you and I are both spies looking to find out a particular secret. The secret is the truth about p. Your goal is to get to know it before me and my goal is to get to know it before you, i.e., each agent has a goal:

$$\gamma_a = (K_b p \vee K_b \neg p) \to (K_a p \vee K_a \neg p),$$

$$\gamma_b = (K_a p \vee K_a \neg p) \to (K_b p \vee K_b \neg p),$$

In other words, it is not a problem if you know it, as long as I already know it too, and vice versa. The epistemic results of asking and answering are new information states wherein we can check whether or not each player's goal is fulfilled. This determines a payoff function and thus the outcome of the epistemic game.

So in order to play a game with questions and answers the players need a goal, and that goal can be an epistemic formula. Why should a player answer a question if that means giving away information that may make him lose the game? He has no reason whatsoever. However, just like in real life, if you wish the other person to loosen his information strings, you may only expect to obtain that by giving away some information yourself. The procedural version of this expectation is a game where each player may *choose* between different questions to ask but where the other player addressed by that question is *obliged* to answer.

Pointed and induced question-answer games Given two agents a and b, an epistemic model $M = (S, \sim_a, \sim_b, V)$ encodes their uncertainty about facts and about each other; Two formulas γ_a and γ_b in the logical language express what they wish to achieve by their questions. In order to achieve their goals, agent a asks a question $\varphi?$ to agent b, to which b is obliged to respond with 'yes' (I know that φ), 'no' (I know that $\neg\varphi$), or 'don't know' (I don't know whether φ). And similarly for b asking a question to a. We don't want to keep saying that all the time, so from now on we may refer to the two agents as i and j, where $i \neq j$, and i may be either a or b. We assume that both agents ask their question simultaneously, and that subsequently both agents answer the question simultaneously. Of course, more realistic communicative settings would allow for agents to ask questions and respond to them in any order, such generalizations will be modeled as extensive games in a later section of this chapter. The question formulas can be thought of as defining the strategies for the agents.

Executing the strategy φ for agent i can be thought of as follows. Agent i asks $\varphi?$ to j. If $M, s \models K_j\varphi$, then j answers (announces) "Yes, I know that φ". If $M, s \models K_j\neg\varphi$, then j answers "No, I know that $\neg\varphi$". Otherwise, j answers "I don't know whether φ". The resulting model restriction depends on both answers, e.g., if a asks φ? to which b responds $K_b\varphi!$ and b asks ψ? to which a responds $K_a\neg\psi!$, the result is the restricted model $M|(K_b\varphi \wedge K_a\neg\psi)$. We can capture these alternatives with a construct $\overline{K}_i\varphi$, for 'agent i answers the question φ?', defined as follows. Given an epistemic model M and a state $s \in M$, if $M, s \models K_i\varphi$, then $\overline{K}_i\varphi \equiv K_i\varphi$; if $M, s \models K_i\neg\varphi$, then $\overline{K}_i\varphi \equiv K_i\neg\varphi$; and else $\overline{K}_i\varphi \equiv \neg(K_i\varphi \vee K_i\neg\varphi)$.

Alternatively, we can represent the question by an issue relation $\approx_\varphi$ and the public announcement of answering the question in the link-cutting way from the previous chapters in this thesis going back to [99, 100]. A strategy will be in the current setting a tripartite question, and its execution value consists of the answer to it. As the three epistemically relevant possible answers are mutually exclusive, each world of the model uniquely determines the execution value for each strategy. The values will then be aggregated at the global level of the model.

We associate two different strategic games with these questions, their answers and agents' goals: pointed question-answer games and not-pointed or global question-answer games. Both are needed: a player may not know what the actual state is, and therefore not know which game he is playing. These definitions are adaptations of similar concepts in [2] with the only notable exception that when modeling questions the pointed game has to use a more complex payoff function.

4.2.1. DEFINITION. [Pointed question-answer game] The *state game* or *pointed question-answer game* $G((M, s), \gamma_a, \gamma_b)$ associated with state $s \in M$ of goals γ_a and γ_b for agents a and b respectively, is the strategic game defined by:

- $N = \{a, b\}$;
- for $i = a, b$, $A_i = \{\varphi? \mid \varphi \in \mathcal{L}\}$;
- for $i = a, b$, $u_i^s(\varphi, \psi) = \begin{cases} 1 & \text{if } M, s \models \langle(\overline{K}_b\varphi \wedge \overline{K}_a\psi)!\rangle\gamma_i \\ 0 & \text{otherwise} \end{cases}$

Note that the set of strategies A_i is the same in all states of the model. The next definition gives a state-independent perspective on question-answer games.

4.2.2. DEFINITION. [Question-answer game] Given state games $G((M, s), \gamma_a, \gamma_b)$ for each $s \in M$, the *induced game* or *question-answer game* $G(M, \gamma_a, \gamma_b) = \langle N, \{A_i' : i \in N\}, \{u_i : i \in N\}\rangle$ is the strategic game defined by

- $N = \{a, b\}$;
- for $i = a, b$, A_i' is the set of uniform functions from S to A_i;
- for $i = a, b$, $a_a', a_b' \in A_i'$, $u_i(a_a', a_b') = \frac{\sum_{s\in S} u_i^s(a_a'(s), a_b'(s))}{|S|}$.

In Definition 4.2.2, a strategy a_i for player i is *uniform* iff for all $s, t \in S$: $s \sim_i t$ implies $a_i(s) = a_i(t)$. It can be easily shown that this corresponds to a Bayesian game [45], in the sense that it has the same Nash equilibria; we will return to this aspect in more detail later in the section.

As there are countably infinitely many formulas in the basic propositional language, there will be infinitely many strategies in a pointed question-answer game, and therefore also in an induced question-answer game. However, in order to avoid such an explosion and an unnecessary overkill by strategy proliferation we can propose the following major simplification for the notion of strategy.

4.2.3. Definition. [Strategy equivalence] Let an epistemic model M be given. Two strategies $\varphi?$ and $\psi?$ for a pointed question-answer game played in the model M are the same (equivalent) for agent i if:

$$\begin{aligned}\{[\![K_j\varphi]\!]_M, [\![K_j\neg\varphi]\!]_M, [\![\neg(K_j\varphi \vee K_j\neg\varphi)]\!]_M\} &= \\ \{[\![K_j\psi]\!]_M, [\![K_j\neg\psi]\!]_M, [\![\neg(K_j\psi \vee K_j\neg\psi)]\!]_M\}\end{aligned}$$

Note that it is common knowledge to a and b if two strategies are the same. This is so because we are comparing the denotations of formulas involving φ and ψ in the model, independent of the actual state.

We include a more detailed discussion of this notion of strategy equivalence and additional examples in Section 5.2.

Given the requirement of uniform strategies in the global game, instead of seeing a strategy in the induced game as a function from *states* to formulas, we can also see such a strategy for agent i as a function from *i-equivalence classes* to formulas, and therefore as a function from *formulas* characterizing i-equivalence classes to formulas. In other words, we can see then as conditional strategies.

With these simplifications there are fewer strategies: given that our starting models are always finite, there are only a finite number of non-equivalent strategies. The next subsection contains an example where both players only have two 'real' (i.e., non-equivalent) strategies. Subject to the identification of strategies with the same tri-partition, the number of strategies for i is a function of the number of the equivalence classes for agent j in the model M. We discuss further the counting of strategies based on this notion of strategy equivalence in Section 5.2. We will consider here the simple special case where the 'don't know' alternative is always empty as a first approximation of how strategies can be counted.

Also we will assume that the worlds in the domain of the model can be named either by using nominals or by a characteristic epistemic formula true only at one world. This can always be constructed if the starting epistemic model has been minimized under epistemic bisimulation before the play of the question-answer game begins and the set of formulae representing strategies is determined.

If the strategies for i have the form $K_j\varphi?$, there are only two and not three answers, namely only 'yes' and 'no', where the second answer contains now both previously negative and unknown answer alternatives. If strategies are required to have this form for both agents, we call this a *dichotomous question-answer game*. The number of strategies is now even lower. It can be counted as follows, assuming that we work with $\{a, b\}$-connected models. If we assume that player i has m_i equivalence classes and player j has m_j equivalence classes. Then the number of pure strategies for player i will be $2^{m_j m_i - m_i}$.

There are $2^{m_j - 1}$ different dichotomies for player j i.e. coarsenings of player j's partition, and for each of m_i different equivalence classes for the requesting player i, she may choose one of those questions, therefore the total number of pure strategies in the game will be $(2^{m_j-1})^{m_i} = 2^{m_j m_i - m_i}$.

We include further examples and illustrations in the next chapter.

It is time now to take stock of where we stand with our formal model so far. First, our model is based on some implicit but drastic simplifications. These have been useful so far because they provide concreteness to the formal model as a initial approximation. We discuss some of these simplifying assumptions and some further desiderata for games with more realistic features.

Simplifications In the discussion throughout this section, we disregard pragmatic constraints of questions. One may ask a question to which she already knows the answer. In other words, we did not consider that the question φ? is also an informative update / public announcement of the formula:

$$(\neg K_a \varphi \wedge \neg K_a \neg \varphi) \wedge \neg K_a \neg (K_b \varphi \vee K_b \neg \varphi).$$

However, incorporating such additional constraints does not raise any significant conceptual or technical difficulties. It merely restricts the players' strategies.

Avoiding to answer the question is not modeled as a move in the version of question-answer game considered so far. However, it is not problematic, as that response can be modeled as the trivial announcement.

There are two players only, that ask each other questions. If there are more than two players, one has to specify who is addressed by the question. Again, this is doable, and an answer to the question would still be a public announcement.

So far, we only assume that the two players ask each other a single question, and that they ask the question at the same time; say, by writing down the question on a piece of paper, putting it in an envelope, and then exchanging envelopes. For the answer, they again exchange envelopes.

The most natural interaction between players involving questions and answers is where they ask each other questions in turn not simultaneously, such that a question is answered before the next question is asked. Such scenarios can be modeled but they will require an approach that uses games in extensive form.

However, in order to make sure that our formal model is relevant for realistic applications we have to find the right balance between a naive technical convenience and an approach that can capture a reasonable level of conceptual complexity. For this we list a number of further modeling desiderata.

Modeling Desiderata, Further Extensions We have shown how epistemic models come with natural games that model interesting phenomena, and suggest interesting logical questions. Our games are very simple, but this starting point itself is an advantage, since well-chosen simple games are a first start for more complex scenarios. Nevertheless, we wish to go beyond that. Most natural from the viewpoint of our general aims are the following extensions:

A first desideratum is to have a richer account of questions that is adequate at least for analyzing questions as possible moves in inquiry. As mentioned in the introductory Section 4.2, a pragmatic constraint on questions is that the interrogator does not know the answer. This amounts to the precondition

$$\neg K_a\varphi \wedge \neg K_a\neg\varphi \quad (i)$$

for a question φ?. The agent asking the question also has expectations about the agent that will answer the question, and that constitutes another pragmatic precondition. She considers it possible that he knows the answer, i.e.,

$$\neg K_a\neg(K_b\varphi \vee K_b\neg\varphi) \quad (ii)$$

For a given model, such pragmatic constraints result in a (further) reduction of strategies than the reduction already achieved by strategy equivalence. For instance, given (i), in the example of Section 4.2 the agents would not have had any strategic choice! The trivial question would not be allowed. Also, it makes sense to consider games in which the trivial answer 'I decline to answer the question' / announcement of $\top$ is always allowed; or games where this is out-ruled. All such variations can be accommodated in the current model.

Extensive games with longer sequences of moves are an obvious desideratum, and so are logics for them and finding sequential equilibria in extensive games.

An approach that generalizes questioning games by considering more than two agents; multiple or partially ordered goals per agent; a notion of goal equivalence; non-uniform probability distributions over the worlds in the domain, etc.

A final desideratum would be to have multiple goals and also goals that are structured in a richer way, for instance forming a linear order with some of them having a higher priority or relevance over the remaining ones.

4.3 Questioning Games with Oracles

Most of the applications that are relevant for modeling inquiry and scientific research consider alongside strategic questions of rational agents objective answers from Nature or other generic entity like an Oracle or other abstract information sources. The strategic aspects involved in playing a questioning game depend also on what are the sources of information available during the game.

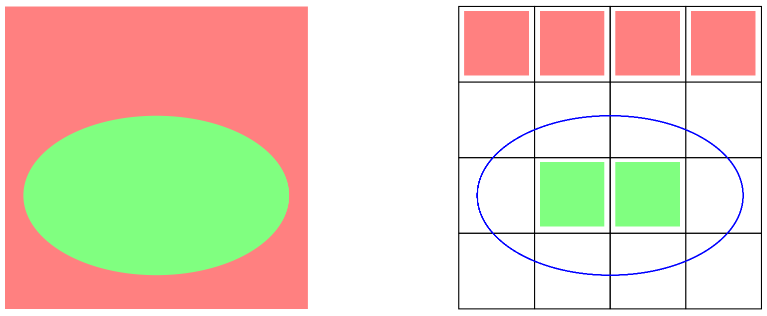
In this section we will address the following modeling desiderata specified earlier: we will consider a setting with more than two players, we will consider a setting in which we can study various sources of information available and a more complex model of questioning: both classical or *dichotomous* questions and epistemically modulated, or *trichotomous* questions. We will show by means of some illustrative examples how these features can be easily integrated in our setting of epistemic games introduced in the previous section. Even more, we will show that the standard approach to questioning actions in an epistemic neutral environment can be captured in an epistemic setting as a particular case, when a perfectly informed source of information is available in the model.

Moreover, we will also show how standard results about games in general transfer in the expected way to question-answer epistemic games (QAGs). One property investigated will be the (in)existence of Nash equilibria[1]. For games with pure strategies in general there might be the case that no Nash equilibrium exist. We show that this is also the case for QAGs using a setting in which this result can be presented in a compact (even though, admittedly, graphically demanding) way.

However, we will not have questions represented by an issue partition yet, or games with sequential moves, these desiderata will be addressed later on in Section 4.4.2. Let us start by considering a paradigmatic example:

4.3.1. Example. The answer to a question φ? can be seen as a rough set. In the figure below, the $(4 \times 4 = 16)$ cells represent the equivalence classes of the agent answering the question. The blue ellipse corresponds to $\{\varphi\}$, the green region to the answer 'yes' and the red region to the answer 'no' (and this could have been the other way around), and the white region (necessarily) to 'don't know'.

Figure 4.1: Bipartite versus Tripartite Questioning: adding an epistemic partition (right) enriches the standard approach of formal structure in answering (left).



We can generalize the setting from this example in the following way. Let W be a set of possible worlds, $R \in W \times W$ an equivalence relation on W, and K the

[1]The notion of Nash equilibrium we will use is the standard one, see Definition 6.6.1 in the Theoretical Background appendix 6.6 and/or Definition 4.4.15 later on in this chapter.

modality for R. Let $P \subseteq W$ be the extension of φ. We define $S3_P \in W^3$ by:

$$S3_P = \{S0_P, S1_P, S2_P\}$$

where we have: $S0_P = \{w \mid M \models_w K\neg\varphi\}, S1_P = \{w \mid M \models_w K\varphi\}$,

$$S2_P = \{w \mid M \models_w \neg K\neg\varphi \wedge \neg K\varphi\}.$$

Discussion Intuitively, K represents the query-available information. It can be given by the limitations in the subjective knowledge of the agent or group of agents, that will answer the query in a multi-agent query setting. It can equally well represent the objective limits in generic answering capabilities like limitations of measurement instruments, or contingencies of experimental design, restrictions in computing power, or lack of specific resources needed for problem solving. Such limitations can also be encoded by an oracle function.

We can now define an order on questions in the following way:

Let S be the set of tripartite questions expressible over a given domain W and $S3, S3' \in S$ such that $S3 = \{S0, S1, S2\}$, $S3' = \{S0', S1', S2'\}$, we define an order $\leq \in S^2$ in the following way:

$$S3 \leq S3' \Leftrightarrow \forall X \in S3\ \exists Y \in S3' : X \subseteq Y$$

This ordering is related to the order over the subsets of the domain, representing assertions, in a natural way. Let $P, P' \subseteq W$ be the extensions of φ, φ', respectively. We define an order $\leq \in \wp(W)^2$ in the following way:

$$P? \leq P'? \Leftrightarrow \{P\} \leq \{P'\} \Leftrightarrow S3_P \leq S3_{P'}$$

It follows from this definition that $\varphi? \leq \varphi'? \Leftrightarrow P? \leq P'?$.

The two orders can be further lifted in a natural way to questions understood as sets of answers. Let $Q = \{P_1, \ldots, P_n\}, Q' = \{P'_1, \ldots, P'_m\}$, $P_i, P'_j \subseteq W$, we define an order $\leq \in \wp(\wp(W))^2$ in the following way:

$$Q? \leq Q'? \Leftrightarrow \{P_1, \ldots, P_n\} \leq \{P'_1, \ldots, P'_m\} \Leftrightarrow \forall P_i \in Q\ \exists P'_j \in Q' : S3_{P_i} \leq S3_{P'_j}$$

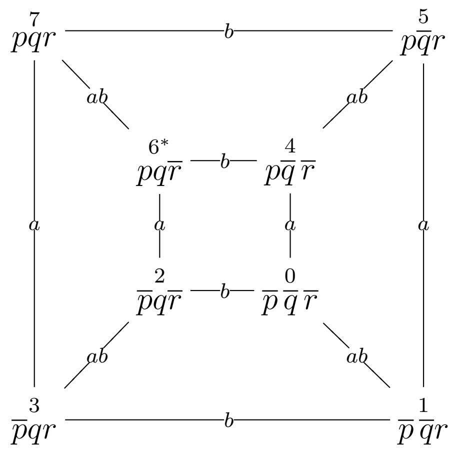
This is a natural ordering of questions which depends on the information available in the epistemic structure considered. However, it turns out that the standard ordering of questions is just a particular case that can be obtained as follows: Let P_R be the partition induced by R on W, $|P_R| = c$ be number of equivalence classes in P_R and $|W| = s$ be the size of W, i.e. the number of possible worlds. When $R = \mathsf{Id}(W) = \Delta(W^2)$ we have $|P| = c = s = |W|$, and we obtain as a particular case the standard ordering between questions:

$$Q \leq Q' \Leftrightarrow \forall P \in Q, \exists P' \in Q' : P \subseteq P'$$

In the following application we will investigate a scenario in which both kinds of sources of information and their corresponding questions coexist. In this setting it becomes important from a strategic point of view to chose the right source to address the question, or the most efficient oracle, which is also a crucial aspect in designing efficient questioning strategies in inquiry.

Epistemic Games with Subjective Agents and Objective Environment

We start with an epistemic situation where three agents a, b, c have various levels of knowledge about three facts p, q, r, as follows: a knows q, b knows p and c is fully informed. This epistemic situation can be represented in a relational structure like the one depicted in Figure 4.2.



$\overset{7}{pqr}$
$\overset{5}{p\overline{q}r}$
$\overset{6^*}{pq\overline{r}}$
$\overset{4}{p\overline{q}\,\overline{r}}$
$\overset{2}{\overline{p}q\overline{r}}$
$\overset{0}{\overline{p}\,\overline{q}\,\overline{r}}$
$\overset{3}{\overline{p}qr}$
$\overset{1}{\overline{p}\,\overline{q}r}$
b
ab
a

Figure 4.2: The M_7 model

We specify an epistemic game to be played inside this structure by the following components: the moves that the agents are allowed to make in the game and the goals that each player tries to achieve by making choices during the game. As described in previous sections both of these components are formulas from the specified epistemic language discussed before.

Each questioning move is paired by a truthful and informative answer to it. In the current setting the answer can come from any agent that has the requested information and was addressed by the question. Each player's payoff is computed in the usual way at a given world inside the epistemic model that results after the initial model is updated with the complete information contained in all the answers for all the asked questions and from every information source.

An intuitive illustration of the game just described is given in Figure 4.3. Although this is not going to be our concern in this section we mention the fact that the given strategies could have been obtained in a uniform way as a result of a meaningful set of pragmatic pre- and/or post-conditions for epistemic actions. For our present purpose it suffices to consider all the game components as primitive. The formal definition is as follows (we use here $p?(!_b)$ as a shortcut notation for a sequence in which a asks $p?$ and b answers, and, similarly, $r?(!_c)$ for a sequence in which b asks $r?$ and c answers accordingly):

- The set of players is: $N = \{a, b, c\}$,
- The strategy-sets are: $S_a = \{p?(!_b), r?(!_c)\}, S_b = \{p?(!_a), r?(!_c)\}, S_c = \{\}$,
- The goal-formulas for each of the players are: $\gamma_a = K_a p, \gamma_b = K_b q, \gamma_c = K_c \bot$

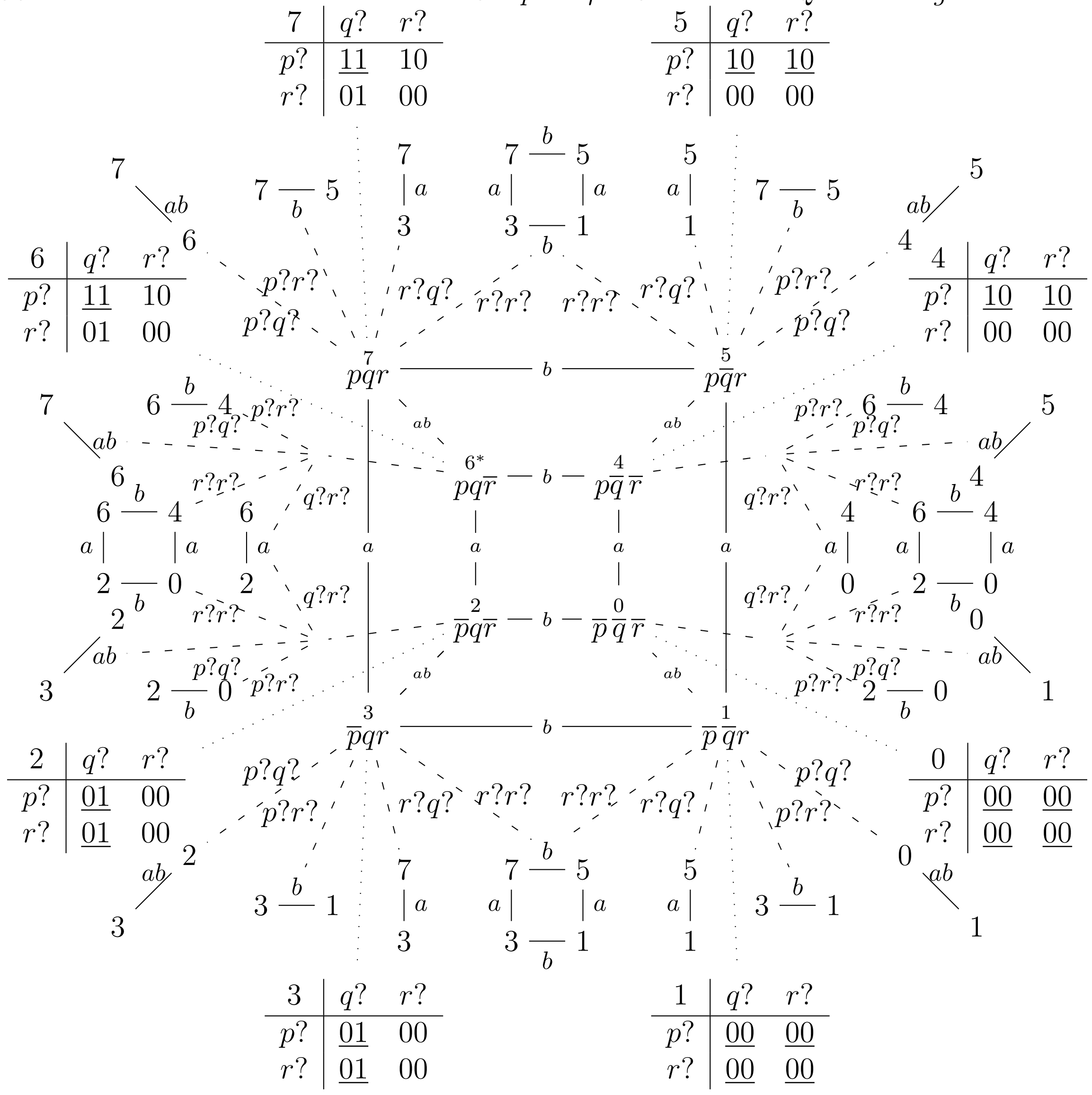


Figure 4.3: Question-answer updates and local games in the epistemic model M_7.

– The payoff for each player is computed by the following function:

$$p_i^w(s_i, s_{-i}) = \begin{cases} 1 & \text{if } M_{?!} \models_w \gamma_i \\ 0 & \text{otherwise} \end{cases}$$

where $M_{?!}$ is $M_7 \mid_! \varphi$ for $\varphi := \bigwedge(s_i, s_{-i})$. We denote by $\bigwedge(s_i, s_{-i})$ a strategy profile, in which s_{-i} is the tuple of strategies for players other than i. The notation $\bigwedge(s_i, s_{-i})$ is a shortcut for $s_a \wedge \cdots \wedge s_c$.

The local state-games that result from this definition in each world of the M_7 model are depicted in Figure 4.3. To economize on space we use an abbreviated notation that assumes but does not represent explicitly the strategies and the payoff values for player c, which will be 0 in all situations given the goal γ_c.

The local state-games played in each of the worlds of the epistemic model M_7 are represented in Figure 4.4. The Nash equilibria in these games are underlined. We can notice that the distinction between *de re* and *de dicto* concepts remains pertinent also for games in which strategies are questions and answers in the same way they were for games in which strategies are public announcements. Note also that even if the strategies in the normal form for the state-games are questions in fact the effect of the corresponding answer is the one that changes the epistemic structure. The semantic effects of questions are not yet modeled at this stage.

0	$q?$	$r?$
$p?$	$\underline{00}$	$\underline{00}$
$r?$	$\underline{00}$	$\underline{00}$

1	$q?$	$r?$
$p?$	$\underline{00}$	$\underline{00}$
$r?$	$\underline{00}$	$\underline{00}$

2	$q?$	$r?$
$p?$	$\underline{01}$	00
$r?$	$\underline{01}$	00

3	$q?$	$r?$
$p?$	$\underline{01}$	00
$r?$	$\underline{01}$	00

4	$q?$	$r?$
$p?$	$\underline{10}$	$\underline{10}$
$r?$	00	00

5	$q?$	$r?$
$p?$	$\underline{10}$	$\underline{10}$
$r?$	00	00

6	$q?$	$r?$
$p?$	$\underline{11}$	10
$r?$	01	00

7	$q?$	$r?$
$p?$	$\underline{11}$	10
$r?$	01	00

Figure 4.4: Local state-games in the epistemic structure M_7

Note also that in this setting the same strategy can have different semantic effects in various worlds of the model. For instance, asking the question $r?$ in world 7 will lead to a truthful and informative announcement of $r!$ and the corresponding update to a model with a domain containing only worlds with odd numbers: $\{7531\}$. In contrast, asking the same question $r?$ in world 0 will lead to a truthful and informative announcement of $\overline{r}!$ and the corresponding update to a model with a domain containing only worlds with even numbers: $\{6420\}$. The more general inequality $dom(M\ |_?\ \varphi_0?, \ldots, \varphi_n?) \neq dom(M\ |_?\ \varphi_0?, \ldots, \varphi_n?)$ is the reason why the same strategy profile can lead to different payoff values.

We continue with an analysis of the induced Q-A game. Conceptually, the induced game is the game that is played globally in the whole epistemic model not just locally in each world of the model. Its formal definition is as follows:

– The set of players is: $N = \{a, b, c\}$,

– The strategy-sets contain all strategies uniform on information cells:

$S_a = \{S_0^a, S_1^a, S_2^a, S_3^a\}$	$S_b = \{S_0^b, S_1^b, S_2^b, S_3^b\}$
$S_0^a = \{(7632, p?), (5410, p?)\}$	$S_0^b = \{(7654, q?), (3210, q?)\}$
$S_1^a = \{(7632, p?), (5410, r?)\}$	$S_1^b = \{(7654, q?), (3210, r?)\}$
$S_2^a = \{(7632, r?), (5410, p?)\}$	$S_2^b = \{(7654, r?), (3210, q?)\}$
$S_3^a = \{(7632, r?), (5410, r?)\}$	$S_3^b = \{(7654, r?), (3210, r?)\}$

– The goal-formulas for each of the players are: $\gamma_a = K_a p, \gamma_b = K_b q, \gamma_c = K_c \bot$

– The payoff for each player is computed by the following function:

$$p_i^M(s_i, s_{-i}) = \frac{\sum_{w \in W} p_i^w(s_i(w), s_{-i}(w))}{|W|}.$$

If we compute the players' payoffs in the local state games for each world and strategy profile according to this definition we obtain the following results:

(S_b, S_a)	0	1	2	3	4	5	6	7	(p_a^M, p_b^M)
(S_0^b, S_0^a)	0,0	0,0	0,1	0,1	1,0	1,0	1,1	1,1	0.50, 0.50
(S_0^b, S_1^a)	0,0	0,0	0,1	0,1	0,0	0,0	1,1	1,1	0.25, 0.50
(S_0^b, S_2^a)	0,0	0,0	0,1	0,1	1,0	1,0	0,1	0,1	0.25, 0.50
(S_0^b, S_3^a)	0,0	0,0	0,1	0,1	0,0	0,0	0,1	0,1	0.00, 0.50

(S_b, S_a)	0	1	2	3	4	5	6	7	(p_a^M, p_b^M)
(S_1^b, S_0^a)	0,0	0,0	0,0	0,0	1,0	1,0	1,1	1,1	0.50, 0.25
(S_1^b, S_1^a)	0,0	0,0	0,0	0,0	0,0	0,0	1,1	1,1	0.25, 0.25
(S_1^b, S_2^a)	0,0	0,0	0,0	0,0	1,0	1,0	0,1	0,1	0.25, 0.25
(S_1^b, S_3^a)	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,1	0.00, 0.25

(S_b, S_a)	0	1	2	3	4	5	6	7	(p_a^M, p_b^M)
(S_2^b, S_0^a)	0,0	0,0	0,1	0,1	1,0	1,0	1,0	1,0	0.50, 0.25
(S_2^b, S_1^a)	0,0	0,0	0,1	0,1	0,0	0,0	1,0	1,0	0.25, 0.25
(S_2^b, S_2^a)	0,0	0,0	0,1	0,1	1,0	1,0	0,0	0,0	0.25, 0.25
(S_2^b, S_3^a)	0,0	0,0	0,1	0,1	0,0	0,0	0,0	0,0	0.00, 0.25

(S_b, S_a)	0	1	2	3	4	5	6	7	(p_a^M, p_b^M)
(S_3^b, S_0^a)	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0.50, 0.00
(S_3^b, S_1^a)	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0.25, 0.00
(S_3^b, S_2^a)	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,0	0.25, 0.00
(S_3^b, S_3^a)	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0.00, 0.00

By putting together all the obtained values we can construct the following payoff matrix for the induced Q-A game (again player c's payoffs are omitted):

	S_0^b	S_1^b	S_2^b	S_3^b
S_0^a	$\underline{0.50}, \underline{0.50}$	0.50, 0.25	0.50, 0.25	0.50, 0.00
S_1^a	0.25, 0.50	0.25, 0.25	0.25, 0.25	0.25, 0.00
S_2^a	0.25, 0.50	0.25, 0.25	0.25, 0.25	0.25, 0.00
S_3^a	0.00, 0.50	0.00, 0.25	0.00, 0.25	0.00, 0.00

As expected, the only Nash equilibrium in this game is given by the strategy profile in which every player asks the question about its goal formula.

We conclude the analysis of the induced game and close this section about Nash equilibria in Question-Answer epistemic games with the following result:[2]

[2]The notion of Nash equilibrium we will use here is the standard one, see Definition 6.6.1 in the Theoretical Background appendix 6.6 and/or the previous footnote in this chapter.

4.3.2. Proposition (No Nash Equilibrium). *There exist Question-Answer Epistemic Games with Oracles in which no Nash equilibrium can be found.*

4.3.1. Proof. This fact can be witnessed by changing the following goal formulas in our running example throughout this section:

$$\gamma_a = (p \leftrightarrow \overline{q}) \wedge W_a r \wedge (W_a p \vee W_b q)$$

and symmetrically γ_b for the second player b, and where $W_i\varphi = K_i\varphi \vee K_i\overline{\varphi}$.

The rest of the proof consists in building the corresponding game matrix. We include all the details in Section 4.7. □

A natural question at this point is whether or not a similar result can be obtained for games with question-answers without oracles that were discussed in Section 4.2. Conceptually there seems to be no difference between these, however, in practice it is more difficult to obtain and display a similar result in the initial setting of question answer games. We have to leave this as an open question, and we conclude this topic with merely the following tentative result:[3]

4.3.3. Conjecture. *An example of a Question-Answer epistemic game with no Nash Equilibrium exists, and it will have at least 256 strategy profiles.*

Nash Equilibria under Syntactic Restrictions The inexistence result for Nash equilibrium in the previous section can be interpreted as an undesirable situation if a design of an inquiry strategy is meant to offer incentive for cooperation in research. Therefore it would be of interest to have a way of identifying inquiry patterns in which Nash equilibria can be always found under special restrictions.

Positive goals and most informative answers Consider the fragment of the *positive formulae* of $\mathcal{L}$: $\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid K_i\varphi \mid [\varphi]\varphi$ where $p \in \Theta$. This notion of positive formulae is found in [103]. It is an extension of [91] who observed that purely epistemic, i.e. without announcement operators, positive formulae are *preserved* under sub-models. This leads immediately to the result that, if both players have goal formulae that are positive, *every pointed question-answer game has a Nash equilibrium.* We can easily grasp why this is true. Let s be the actual state of the Kripke model, for which we play the game. If player i asks a question to player j that makes j reveal all he knows, either i's goal is now

[3]The number of strategy profiles in the next conjecture can be justified as follows: a model constructed starting from only two propositional atoms cannot produce the dynamics described in Proposition 4.3.2. So at least three atoms are needed. A model in which any of the players is fully informed cannot produce the dynamics described in Proposition 4.3.2. So each agent has to have an epistemic partition with at least two equivalence classes. In a model with eight worlds and two equivalence classes an agent has at least $2^3 \times 2$ strategies in the iduced game.

realized, or there is no way she can realize that goal. This is because if there were a weaker announcement by j also realizing that goal, a further model restriction would preserve the goal, as it is positive. Therefore, asking the question that elicits the most informative answer is a weakly dominant strategy for i, for the pointed question-answer game for state s. And as this holds for both players, that must be a Nash equilibrium of the pointed question-answer game for state s. As s was arbitrary, this holds for all pointed games.

4.3.4. Proposition. *Whenever both players' goals are positive formulae, each pointed question-answer game has a Nash equilibrium.*

However, what question elicits the most informative answer might not be known to either of the players, as they may be unable to distinguish the actual state from another state in which the same question does not elicit the most informative answer. This makes for another difference between games with questioning moves and games with announcement moves. This is also of more general interest than just for positive goal formulae.

Most informative answers We can call a strategy for a given agent weakly dominant *de re*, if the agent knows that it is weakly dominant (i.e., if in all states indistinguishable for that agent, including the current state, that strategy is weakly dominant). The agent has a weakly dominant strategy *de dicto* if he has a weakly dominant strategy in all indistinguishable states – but now it may be a different one in each, so he cannot choose which one to execute. The *de dicto*/*de re* distinction is well known in the knowledge and action literature [54]. Similarly, we can distinguish a *de dicto* Nash equilibrium from a *de re* Nash equilibrium, wherein all players know what their equilibrium profiles are. We can see how these notions are useful in describing questioning games: the induced question-answer game has an equilibrium if (not necessarily only if) all pointed games have a *de re* equilibrium: each player has a *uniform* strategy that is weakly dominant.

This aspect reveals a real difference between public announcement games and question-answer games. From a player's perspective, there is such a thing as a most informative announcement (tell them all you know). If all goals are positive, then the most informative announcement is a weakly dominant strategy in all points of the public announcement game. And because players know what their most informative announcement is, this is therefore an Nash equilibrium strategy of the induced public announcement game.

But the question that elicits the most informative answer from another player cannot be called the most informative question from the questioning player's point of view. In a different state in the same equivalence class for that player, the question to elicit the most informative answer may be a different question, as the responding player may be in a different equivalence class there. So even when all goals are positive, induced question-answer games may not have an equilibrium.

However, finding an example remains an open problem. We will return to the topic of describing strategic abilities for questioning actions in Section 4.5.

4.3.1 Bayesian Games

The induced question-answer game corresponds to a Bayesian game. This can be shown in an analogous way as the corresponding result for public announcement games in [2]. We merely give an overview of the argument here.

Note that the difference between a game with announcements and one with questions and answers consists only in how the strategies are defined. Once a set of strategies is fixed the remaining overall structure of the argument follows along similar directions. Given an arbitrary question-answer game we can define an associated Bayesian game as follows:

– $N = \{1, 2\}$, – $\Omega = W$, – $\tau_a(w) = [w]_{\overset{a}{\sim}}$,

– $T_a = \{[w]_{\overset{a}{\sim}} : w \in W\}$, where $[w]_{\overset{a}{\sim}} = \{v : w \overset{a}{\sim} v\}$,

– $S_a = \{\varphi : \{[\![K_{-a}\varphi]\!]_M, [\![K_{-a}\neg\varphi]\!]_M, [\![\neg(K_{-a}\varphi \vee K_{-a}\neg\varphi)]\!]_M\} =$
$\{[\![K_{-a}\psi]\!]_M, [\![K_{-a}\neg\psi]\!]_M, [\![\neg(K_{-a}\psi \vee K_{-a}\neg\psi)]\!]_M\}\}$,

– $\mu(v, [w]_{\overset{a}{\sim}}) = \begin{cases} \frac{1}{|[w]_{\overset{a}{\sim}}|} & \text{if } v \in [w]_{\overset{a}{\sim}} \\ 0 & \text{otherwise,} \end{cases}$ – $u_a(s_a, s_{-a}, w) = u_a^w(s_a, s_{-a})$,

where u_a^w is the utility function of the w-local question-answer game, and $a, -a$ are variables ranging over N such that $a \neq -a$.

A signal t_a for player a corresponds to an a-equivalence class. In a Bayesian game, the combination of a player a and a signal t_a defines a virtual player (a, t_a), who has the same strategies $s_a \in S_a$ at his disposition. But this amounts to the same as our definition involving the same players a employing strategies s'_a that are uniform across equivalence classes and that are conditional from states to strategies $s_a \in S_a$, and therefore can be also seen as conditional at a higher level from equivalence classes to strategies s_a.

In our simplified modeling, all states w in a given Kripke model get equal a priori probability of $\mu(w) = \frac{1}{|W|}$, i.e., uniform over the entire domain. (And this is their probability for all agents.) A more general approach would have a given probability distribution as a parameter in the modeling of the game.

However, a uniform distribution is a reasonable assumption from the perspective of the observer or modeler of such a multi-agent system: given common knowledge of the structure of the model, as usual in multi-$S5$ conditions, there is no reason to prefer one state over another one. For example, if the Kripke model represents uncertainty about card deals, and the cards are shuffled and drawn blindly from the pack by the players, there is no reason to consider any given card deal (possible world / state) more likely than any other card deal.

After receiving their signal, each agent conditionalizes the probability mass over its equivalence class, that is, $\mu(w|t_a) = \frac{1}{|[w]_{\sim_a}|}$ (in fact, we can assume the received signal t_a to be the corresponding equivalence class $[w]_{\sim_a}$). This means that for each state inside the class the probability is non-zero and uniform in that class, and for each state t outside that class, the probability is 0. (Of course, we are now talking about the probability *for a given agent.*)

For example, continuing our parallel with card games, after the cards have been dealt and a player has picked up his cards, the agent only considers card deals possible wherein she holds that hand of cards, but no longer any of the remaining card deals. Further to this, in the absence of information to the contrary (i.e., assuming 'fair play') each of the possible deals of cards wherein she holds that hand of cards will be also considered to be equally likely.

We will consider scenarios in which this is not the case anymore in the setting with probabilistic extensions to DELQ from Chapter 7.

This provides the key to view an induced question-answer game as a Bayesian game. In induced games the payoff for agent a is computed as

$$u_a(s'_1, s'_2) = \frac{\sum_{w \in W} u_a^w(s'_1(w), s'_2(w))}{|W|}$$

whereas for a Bayesian game, one would get a sum (see [2])

$$\sum_{w \in W} \mu(w|t_a) u_a^w(s'_1(w), s'_2(w)) = \sum_{w \in W} \frac{u_a^w(s'_1(w), s'_2(w))}{|[w]_{\sim_a}|}$$

Although these sums may be different, they induce the same order on payoffs and thus they induce the same Nash equilibria.

4.3.5. Proposition. *Any induced question-answer game has a corresponding Bayesian game, and for a given model and fixed goal formulae for the players they have the same Nash equilibria.*

So far in this chapter we have introduced a formal model for question-answer games and we have studied both logical and game theoretical properties that emerge in such a setting. Such results are very useful as a starting point.

However, we still have two crucial unfulfilled modeling desiderata on our list: a model that uses genuine questioning actions, represented by the previously discussed issue relation, and a model of questioning actions as moves in an inquiry process, which consists of long term sequences of questioning moves.

We will address both these desiderata in further detail during the following section. These will provide a more general setting adequate for modeling more realistic questioning scenarios in both inquiry and communication.

4.4 Extensive Questioning Games

Introduction Questions are ubiquitous in communication and social interaction and they are essential for inquiry and scientific research. One possibility to approach questions is by their intricate connection with other informative actions in their natural linguistic or scientific environment. This was the perspective followed so far in the current chapter and it was closely related to a more general approach which was initiated by [1, 2] and previous ones going back to [102].

Besides the fact that questions are intertwined with various kinds of informative actions which have been extensively studied inside DEL, there is still place for an approach that takes questioning actions as primitive entities. For all relevant purposes questions cannot be studied completely independently of answers, announcements or other information providing activities, however, it is possible and desirable to identify and separate the information seeking essence of questioning actions, and acknowledge their intrinsic important role in strategic reasoning and rational interaction. This motivates recent approaches that study questions and actions of issue-management inside the DEL paradigm like [100, 105], and previous ones going back to [8].

There are two main desiderata previously discussed that are going to be addressed during this section. The first is to capture genuine questioning using a model based on issue relations, the second one consists in defining a formal model for sequential questioning games.

Questioning actions are knowledge-seeking actions, they do not give information straight away but raise and manage issues or highlight possible alternatives for future informative actions. This will also be the perspective followed in the remaining of this chapter which will study questioning actions in a logical and a game-theoretical framework as long-term rational and interactive activities. The aim of this section is to proceed in this direction by defining and studying formally extensive games in which moves available to players are questioning actions and players seek to achieve epistemic goals.

4.4.1 Some Preliminary Notions

We will use epistemic issue structures as the basis for our investigations:

4.4.1. Definition. [Epistemic-Issue Model] An Epistemic-Issue Model (EIM) is a tuple $\langle W, N, (\overset{i}{\approx}, \overset{i}{\sim})_{i\in N}, P, C, V\rangle$ with the following components: W a set of possible worlds or epistemic alternatives, $N = \{1, 2, \dots, n\}$ a set of labels representing agents, $\overset{i}{\approx} \in W \times W$ a binary issue relation on the domain W, for $i \in N$, $\overset{i}{\sim} \in W \times W$ a binary indistinguishability relation on W, for $i \in N$, P and C a sets of symbols representing propositions and nominals, respectively, such that $P \cap C = \emptyset$ and $|V(c)| = 1$, for any $c \in C$, $V : P \cup C \to \wp(W)$ a standard propositional valuation function. A pointed EIM (M, w^*) is an EIM together with a designated world $w \in W$. A set-pointed EIM (M, Q^*) is an EIM with a designated set of worlds $Q \subseteq W$.

Intuitively, such structures model both the uncertainties agents have about the world and also their agenda for inquiry, or the issues they would like to have resolved by future answers to their questions. Another important feature in our EIMs is the use of nominals. These are propositional symbols which are true in only one world, thus naming it. We will use the symbol $\mathbf{K}_{\mathsf{eim}}$ to refer to the class of all epistemic issue models (EIMs). In this section we assume that $(\overset{i}{\sim}, \overset{i}{\approx})_{i \in N}$ are equivalence relations on the domain W.

Such basic structures will serve the purpose of the present approach but they can be enriched in various ways, for instance, by adding more components to the structure in order to represent beliefs alongside knowledge and issues. This can be done either by using a plausibility relation between states or by introducing a probability distribution over the domain of possibilities.

In order to talk about EIMs we will introduce a logical language that can describe the epistemic-issue structure in a static way:

4.4.2. Definition. [Static Language] The static language of Epistemic Logic of Questions (EL_Q), denoted by $\mathcal{L}_{\mathsf{EL_Q}}$, is defined by the following BNF:

$$\varphi ::= i \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid Q_a\varphi \mid R_a\varphi \mid K_a\varphi$$

with P a set of propositional symbols, N a set of nominal symbols, $P \cap N = \emptyset$, A a set of agent-labels, $p \in P$, $i \in N$ and $a \in A$.

Various fragments of this language will be referred to in various places below using the following notation: $\mathcal{L}_{\mathsf{EL}}$ will denote the language of epistemic logic, which is the fragment without nominals and the issue Q and intersection R modalities. $\mathcal{L}_{\mathsf{HL}}$ will denote the language of hybrid logic, which is the fragment without the knowledge, issue and intersection modalities.

The semantics for our language is the standard modal semantics, using the usual Boolean clauses and the standard relational clauses using $\approx$ for Q and $\sim$ for K. The intersection modality R is be defined using $\approx \cap \sim$ as:

$$M \models_w R\varphi \quad \text{iff} \quad \text{for all } v \in W : w\,(\sim \cap \approx)\,v \text{ implies } M \models_v \varphi$$

This basic language will serve the purpose of this section, it can also be extended if needed. Richer versions usually include the universal modality $U\varphi$ and group notions for knowledge $C_G\varphi$ or even issue or intersection.

Validities in EIMs are captured by the axioms given in Definition 4.4.3. This are standard axioms for hybrid logic with nominals and intersection. The intersection axiom makes crucial use of nominals, for the left to right direction. The substitution rule has to keep track of both propositions and nominals.

4.4.3. Definition. [Axiomatization] The EL_Q proof system contains:

– minimal modal logic axioms for epistemic and issue modalities:

$\neg\Box\neg p \leftrightarrow \Diamond p, \Box(p \to q) \to (\Box p \to \Box q), \Box \in \{Q, R, K\}$,

– standard S5 axioms for for epistemic and issue modalities:

$p \to \Box\Diamond p, p \to \Diamond p, \Diamond\Diamond p \to \Diamond p, \Diamond \in \{\widehat{Q}, \widehat{R}, \widehat{K}\}$,

– an intersection axiom for the static resolution modality:

$\widehat{K}i \wedge \widehat{Q}i \leftrightarrow \widehat{R}i$, where $i \in N$ is a nominal,

– an axiom governing the behavior of nominals:

$\Diamond(i \wedge p) \to \Box(i \to p), \Box \in \{Q, R, K\}$

– together with standard derivation rules for hybrid logic:

$\frac{\vdash \varphi, \sigma_{\text{sort}}(\varphi) = \psi}{\vdash \psi}, \frac{\vdash \varphi, \vdash \varphi \to \psi}{\vdash \psi}, \frac{\vdash_{\mathsf{PC}} \varphi}{\vdash \varphi}, \frac{\vdash \varphi}{\vdash \Box\varphi}, \frac{i \to \varphi}{\varphi},$

for $\Box \in \{Q, R, K\}$, i not occurring in φ, and σ a sorted substitution.

Note that for ease of notation and readability alone we have omitted subscripting the Q, R, K modalities inside the axioms.

So far we have a logic that can describe knowledge and issues from a static perspective. But for the purpose of a questioning game we also need to be able to describe the way questioning actions change the knowledge of the players. We need a way to describe knowledge and issues from a dynamic perspective. In order to be able to also describe knowledge-change and issue-dynamics and, moreover, the interaction between the two during a game, we will introduce the following model-changing actions:

4.4.4. Definition. [Questioning] Let $M = \langle W, (\overset{i}{\approx}, \overset{i}{\sim})_{i \in N}, V\rangle$ be an arbitrary EIM. A question action, represented as $\varphi?$, changes the model M into a new model $M \otimes \varphi? = \langle W_{\varphi?}, (\overset{i}{\approx}_{\varphi?}, \overset{i}{\sim}_{\varphi?})_{i \in N}, V_{\varphi?}\rangle$, in the following way:

$$\overset{i}{\approx}_{\varphi?} \quad = \quad \overset{i}{\approx} \cap \overset{\varphi}{\equiv}_M$$

while leaving the other components unchanged, where the set of pairs of worlds in M with equivalent φ value is denoted by $\overset{\varphi}{\equiv}_M = \{(w, v) : \|\varphi\|^M_w = \|\varphi\|^M_v\}$.

Intuitively, this action says that a questions changes a model by refining the issue partition with the content of a formula φ in a specified language. Such a partition-based approach goes back to [40] where it was used for giving semantics of questions in natural language; here we add a dynamic dimension to questioning actions and place them in a more abstract framework of inquiry and scientific discovery. We also add another dynamic action:

4.4.5. Definition. [Resolution] Let $M = \langle W, (\overset{i}{\approx}, \overset{i}{\sim})_{i \in N}, V \rangle$ be an arbitrary EIM. A resolution action, represented as !, changes the model M into a new epistemic-issue model $M \otimes ! = \langle W_!, (\overset{i}{\sim}_!, \overset{i}{\approx}_!)_{i \in N}, V_! \rangle$, in the following way:

$$\overset{i}{\sim}_! \;=\; \overset{i}{\sim} \cap \overset{i}{\approx}$$

while leaving all the other components in the initial model M unchanged.

Intuitively, the resolution action transforms the model to a situation in which all the questions raised so far would have received one answer or another. This is done by changing the underlying uncertainty to reflect the structure of the issue relation in a given situation.

We will enrich our initial static modal language with two more dynamic modalities which will make reasoning about the flow of information, issue management and question answering possible in a formal way.

4.4.6. Definition. [Dynamic Language] By adding two dynamic modalities: $[\varphi?]\varphi$ and $[\,!\,]\varphi$ to the BNF from Definition 4.4.2 we obtain the full language of Dynamic Epistemic Logic of Questions (DEL_Q), denoted by $\mathcal{L}_{\mathsf{DEL_Q}}$.

Intuitively, a formula like $[\varphi?]\psi$ says that "after a questioning action about φ is performed, ψ holds" or "after asking weather φ, ψ is the case". Similarly, $[\,!\,]\varphi$ says that "after performing a resolution action, φ holds".

A note about notation: The $[\varphi?]$ notation is already familiar from Chapter 2 where was used to denote dynamic modalities. Previously in this chapter we used $\varphi?$ to refer to player's strategies in a game. From now on strategies in a game are also going to be questioning actions, nevertheless, we will try to use the square parenthesis in a consistent way to disambiguate between the two whenever the context might require additional precision.

The semantics of these dynamic modalities is also specified in the standard way using the previously introduced model transformations. We will only need here the minimal system given by the following clauses:

$$M \models_w [\varphi?]\psi \quad \text{iff} \quad M \otimes \varphi? \models_w \psi, \qquad M \models_w [\,!\,]\psi \quad \text{iff} \quad M \otimes ! \models_w \psi,$$

The transformed models are computed as in the action definitions previously introduced. This language can be extended if needed.

4.4.7. Definition. [Reduction Axioms] Formulas in $\mathcal{L}_{\mathsf{DEL_Q}}$ can be reduced to equivalent formulas in $\mathcal{L}_{\mathsf{EL_Q}}$ using the following reduction axioms:

– $[\varphi?]a \leftrightarrow a$ (Asking & Atoms), $[\,!\,]a \leftrightarrow a$ (Resolving & Atoms),

– $[\varphi?]\neg\psi \leftrightarrow \neg[\varphi?]\psi$, $[\,!\,]\neg\varphi \leftrightarrow \neg[\,!\,]\varphi$ (Asking/Resolving & Negation),

– $[q](\psi \wedge \chi) \leftrightarrow [q]\psi \wedge [q]\chi$, $q \in \{\varphi?, !\}$ (Asking/Resolving & Conjunction),

– $[\varphi?]Q\psi \leftrightarrow (\varphi \wedge Q(\varphi \rightarrow [\varphi?]\psi)) \vee (\neg\varphi \wedge Q(\neg\varphi \rightarrow [\varphi?]\psi))$ (Asking & Issue),

– $[\varphi?]R\psi \leftrightarrow (\varphi \wedge R(\varphi \rightarrow [\varphi?]\psi)) \vee (\neg\varphi \wedge R(\neg\varphi \rightarrow [\varphi?]\psi))$ (Asking & Intersection),

– $[\,!\,]Q\varphi \leftrightarrow Q[\,!\,]\varphi, [\,!\,]R\varphi \leftrightarrow R[\,!\,]\varphi$ (Resolving & Issue/Intersection),

– $[\varphi?]K\psi \leftrightarrow K[\varphi?]\psi, [\,!\,]K\varphi \leftrightarrow R[\,!\,]\varphi$ (Asking/Resolving & Knowledge).

All these provide a way of reasoning about dynamic actions of asking questions and resolving issues. We will now proceed to defining formally games in which moves are questioning actions as described so far.

4.4.2 Definitions of Main Notions

In this section we give formal definitions for extensive questions games, building on the preliminary notions introduced so far. Next we proceed by giving some intuitive examples and illustrations of EQGs. After that we discuss modeling choices, make some conceptual clarifications, and study solution concepts, strategic abilities and other formal properties of EQGs.

In EQGs, like in any other game or interactive activity, players have objectives or goals and they try to reach these objectives by making appropriate moves during the play of the game. In a setting involving questions such goals will have an epistemic character: players want to acquire some knowledge, but these can also be complex higher-order epistemic situations, say "I want to find out something in a way that prevents you from knowing that I know", and so on. Players can also have competing, convergent or even incompatible goals. We represent such aspect formally by epistemic formulas.

4.4.8. Definition. [Epistemic-Issue Goal Structure] An Epistemic-Issue Goal Structure (EIGS) is a tuple $\langle M, G \rangle$ with the following components:

– $G = \{G_1, G_2, \ldots, G_n\}$ a set of goal-sets, one for each agent $i \in N$, containing formulas $G_i = \{\gamma_1^i, \gamma_2^i, \ldots, \gamma_k^i\}$ from $\mathcal{L}_{\mathsf{EL}}$, and M an EIM.

An EIGS is (set)-pointed if and only if M is a (set)-pointed EIM.

Achieving such epistemic goals depends on moves made during the game, but it also depends on the structure of the epistemic-issue model in which such a game is played. Therefore, any EQGs will be characterized by the local epistemic perspective of each of the players involved.

Such goal structure need not have the simplest structure considered here, it is realistic to assume that agents might also have their research agenda structured in a certain way, for instance by a priority order over the goals or even a more complex graph structure with formulas as vertices.

Questioning and resolution moves made during the play of the game will change the structure of the initial epistemic situation. Such transformations have an interactive character, they are determined by a combination of choices for each of the players. The resulting model can be computed using the model-changing operations introduced in the previous section.

Finally, we can determine if players achieved their objectives by model checking their goal-formulas in the resulting epistemic-issue structure. We use here a language containing only epistemic modalities for simplicity, but it is conceivable that agents might also have objectives with regard to the structure of the issue at the end of a questioning game, such aspects could be also modeled using a richer language for the goal formulas. We capture all these aspects in a formal way by means of the following definition:

4.4.9. Definition. [Extensive Question Game] The Local Extensive Question Game (LEQG) associated with a pointed EIGS $E = \langle (M, w), G\rangle$ is a tuple $T = \langle E, H, J, F, (U_i)_{i \in N}\rangle$ with the following components:

– $H = \{h_0, h_1, \dots, h_z\}$ a set of histories such that:

  - $h_i = \emptyset$, for some $i \in \{0, \dots, z\}$,
  - if $h = \langle q_0, q_1, \dots, q_{l-1}, q_l\rangle \in H$, then $l + 1 \leq |N|$,
  - if $h = \langle q_0, \dots, q_{l-1}, q_l\rangle \in H$ and $h' = \langle q_0, \dots, q_{l-1}\rangle$, then $h' \in H$,
  - if $h = \langle q_0, \dots, q_{l-1}\rangle \in H$ and $h' = \langle q_0, \dots, q_{l-1}, q_l\rangle$, then $h' \in H$, for $q_l = \mathtt{min}([\varphi]_M)$, and $[\varphi]_M = \{\psi \in \mathcal{L}_{\mathsf{HL}} : \|\varphi\|_M = \|\psi\|_M \subseteq W\}$

– $J : H \setminus Z \to N$ a turn-function assigning players to non-terminal histories, s.t. $J(h) = J(h')$ iff $|h| = |h'|$, with $Z = \{h \in H : |h| = |N|\}$,

– $F : H \to \mathbf{K}_{\mathsf{eim}}$ a model-function assigning EIMs to histories as follows:

$$F(h) = \begin{cases} M & \text{if } h = \emptyset \\ F(\langle q_0, q_1, \dots, q_{l-1}\rangle) \otimes q_l? & \text{otherwise,} \end{cases}$$

– $U_i : Z \to \mathbb{R}$ a utility-function assigning utilities for player $i \in N$ to terminal histories in the following way:

$$U_i(h, \varphi) = \begin{cases} 1 & \text{if } M_h \otimes ! \models_w \varphi \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad U_i(h) = \sum_{\gamma^i_k \in G_i} U_i(h, \gamma^i_k).$$

This definition describes the outcome of a questioning game as played in a given possible world. But since the players have epistemic uncertainties, they do not know precisely which is the real situation, hence what is the game that they are playing. In any given state they also consider a number of many other epistemic alternatives. Therefore, we also need a method to aggregate the payoffs obtained in many local games in an expected utility given the entire structure of the initial epistemic-issue model.

4.4.10. Definition. The Set-Local Extensive Question Game (SEQG) associated to a set-pointed EIGS $E = \langle (M, Q), G \rangle$ is defined as the corresponding LEQG (Definition 4.4.9) with the following modification of utility function: $U_i : Z \to \mathbb{R}$ assigns utilities for $i \in N$ to terminal histories as follows:

$$U_i(h) = \frac{\sum_{\gamma_k^i \in G_i}^{w \in Q} U_i(h, \gamma_k^i, w)}{|Q|}$$

where $U_i(h, \gamma_k^i, w) = U_i(h, \gamma_k^i)$ in the LEQG associated with $E = \langle (M, w), G \rangle$. The Global/Induced Extensive Question Game (GEQG) associated with an arbitrary EIGS $E = \langle M, G \rangle$ is the SEQG $E = \langle (M, Q), G \rangle$ in which $Q = W$.

We discuss briefly some of the modeling choices in this formal definitions before we proceed to some concrete examples and illustrations. We already mentioned the importance of the language in which goals are formulated. The same holds for the language in which questioning moves are formulated during the play of a game. The set of available actions at a give history depends on the available questioning actions. It also depends on various pragmatic and epistemic preconditions that one might want to capture in the formal model. We have made here a very general choice by using a very expressive language and by ignoring any pragmatic or epistemic preconditions for question execution. Further, more realistic, options and the consequences that derive from them are discussed in later sections. Also, we assume that an order on the set of formulas is available when choosing the minimal representative from an equivalence class of formulas. We can also think about versions in which resolution actions are available at non-terminal histories, or versions in which players get more than one question during one play, such options are captured by sequential compositions of EQGs as defined here.

Also we make no assumption with regard to the sources of answers that are available in the game. Games in which answers can only come from other players versus versions in which answers can come from nature or an oracle can both be captured by means of pragmatic, epistemic or other arbitrary restrictions on the questioning and resolution actions.

Another important aspect about interactive questioning activities is their cooperative versus competitive character. We can imagine settings in which competing research programs play a questioning game with nature or against each other as well as situations in which inquiry scenarios have a cooperative setting and the only design requirement is the efficiency of acquiring new knowledge in a coalition of convergent research programs. All such aspects can be captured and described formally using our definitions. We discuss such issues in greater detail in later sections.

4.4.3 Examples and Illustrations

We will resort to intuitive examples to illustrate the formal definitions. Consider the very simple concrete example with just two agents and two facts as depicted in Figure 4.5 below. The initial model is represented in the top-left corner. Questioning and resolution moves are indicated with arrows labeled accordingly.

Interesting phenomena can be already described in this simple setting. For instance, we can compare two interrogative scenarios with respect to their fairness as cooperative experimental procedures, depicted in Figure 4.5.

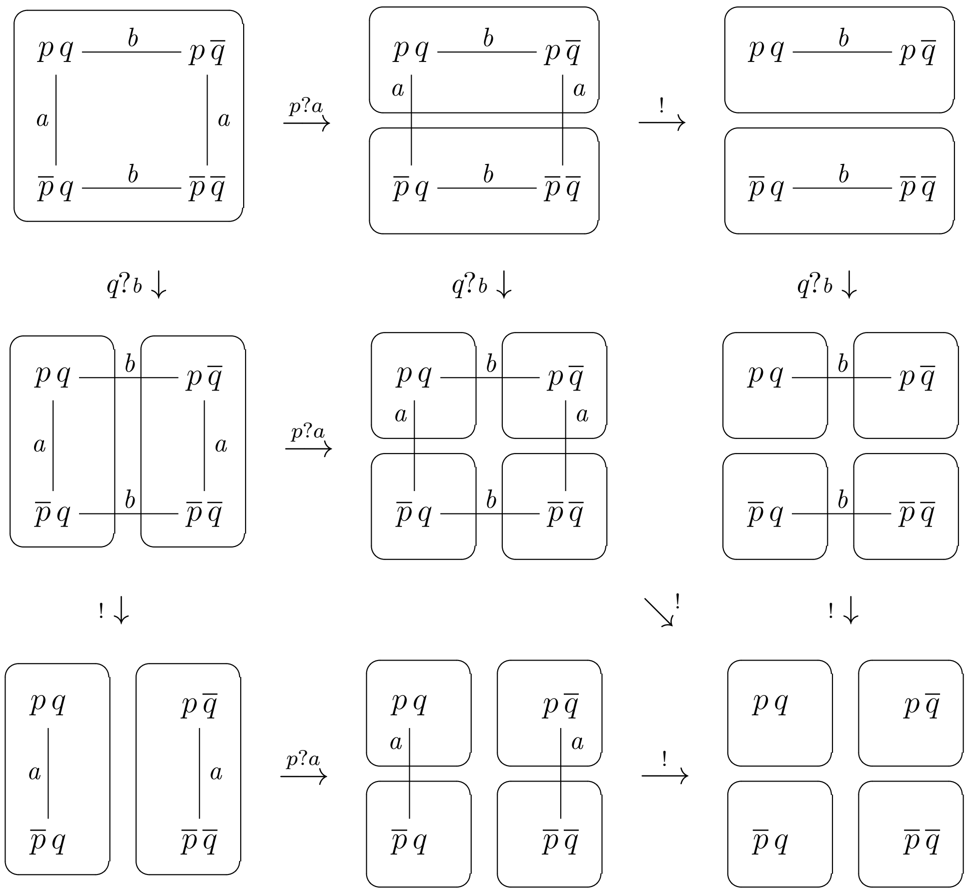


Figure 4.5: EQG example of fairness in cooperative experimental procedures

If we describe an interrogative structure using the set of sequences of available moves, or histories in the game, we can notice that, for instance, the following two experimental protocols are very different in terms of fair-learning during the cooperative inquiry process. In the first scenario, represented by the protocol

$$Q_1 = \{p?, q?, p?\top?, q?\top?, p?q?, q?p?\}$$

one player can become fully informed about the world before the other one does, while in the second one

$$Q_2 = \{\top?, p?, q?, \top?q?, \top?p?, p?q?, q?p?\}$$

this possibility is ruled out, players become fully informed only simultaneously.

For instance, the goal formula

$$\gamma_b = (K_a p \vee K_a \neg p) \rightarrow (K_b q \vee K_b \neg q)$$

and its symmetric counterpart, are both true in the corners on the main diagonal and are false, respectively, in both corners on the secondary diagonal.

The figure also illustrates another interesting feature: extensive games can be composed to form sequences. Each corner of the second diagonal is the endpoint for the previously described game. However, the resulting model can be the starting point for a new extensive game, starting in the resulting model from previous plays as illustrated and proceeding further by new questioning actions.

Even more complex situations and subtleties about research procedures can be captured using the modeling strength of various procedural restrictions in the available experimental protocol, see [97] for such an account in a setting with informative epistemic actions. All such facts about what players can achieve by playing questioning games can be captured in a systematic way using a logic of strategic ability. We will discuss some of these aspects in a later section. Before that we will focus on some properties of EQGs.

Considering a setting in which questions are represented explicitly by means of an issue relation has further advantages. For instance, we can define a measure of relevance of questions in inquiry. Previous notions of relevance exist in the literature, in the present setting, however, we can capture strategic aspects resulting from both the general goal of inquiry and the available sources of information.

Both can be described as partitions of the epistemic domain. Let W be a set of possible worlds, $R, R', R'' \in W \times W$ equivalence relations on W, and K, Q, G the modalities for R, R', R'', respectively.

Let $P_R = \{C_1, \ldots, C_n\}, P_{R'} = \{C_1, \ldots, C_{n'}\}, P_{R''} = \{C_1, \ldots, C_{n''}\}$ be the partitions induced in W by R, R', R'', respectively. Let $\mathsf{Par}(W)$ be the set of W-partitions. For $P, P' \in \mathsf{Par}(W)$ we define $\Cap : \mathsf{Par}(W)^2 \rightarrow \mathsf{Par}(W)$ by:

$$S = P \Cap P' = \{C_1, \ldots, C_n\} \Cap \{C'_1, \ldots, C'_{n'}\} = \bigcup \{C_i \in P \mid \exists C'_j \in P' : C_i \subseteq C'_j\}$$

Let $P_{R'} = \{C_1, \ldots, C_{n'}\}$ be the partition induced in W by a sequence of questioning actions $Q = \{q_1, \ldots, q_k\}$. We define the *source-relevance* or *evidence-relevance* of Q by:

$$R_S(Q) = |P_R \Cap P_{R'}|$$

For $P, P' \in \mathsf{Par}(W)$ we define $\Cap_q : \mathsf{Par}(W)^2 \to \mathsf{Par}(W)$ as follows:

$$S = P \Cap_q P' = \{C_1, \ldots, C_n\} \Cap \{C'_1, \ldots, C'_{n'}\} = \bigcup \{C_i \in P \mid C_i \subseteq C'_1 \in P'\}$$

Let P_S be the bipartition $P_S = \{P \Cap P', W \setminus (P \Cap P')\}$ for $Q = \{q_1, \ldots, q_k\}$. The notion of *goal-relevance* or *inquiry-relevance* of Q is defined as follows:

$$R_Q(Q) = |P_{R''} \Cap_q P_S|$$

These definitions of goal and source relevance of a question give rise to a natural order on question sequences:

$$Q \leq_S Q' \Leftrightarrow R_S(Q) \leq R_S(Q')$$

$$Q \leq_Q Q' \Leftrightarrow R_Q(Q) \leq R_Q(Q')$$

4.4.11. Example. As an illustration, consider the following situation:

$$W = \{w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, , w_9\},$$

$$P_R = \{\{w_1, w_4, w_7, w_2, w_5, w_8\}, \{w_3, w_6, w_9\}\}$$
$$P_{R''} = \{\{w_1, w_4\}, \{w_9, w_6, w_3\}, \{w_7, w_2, w_5, w_8\}\}$$
$$P_Q = \{\{w_1, w_4\}, \{w_7, w_8, w_2, w_5, w_3, w_6, w_9\}\}$$
$$P_{Q'} = \{\{w_1, w_2, w_5\}, \{w_8, w_7, w_4, w_3\}, \{w_9\}, \{w_6\}\}$$

Here we have:

$$R_S(Q) = 2 < 5 = R_S(Q') \text{ and } R_Q(Q) = 2 > 0 = R_Q(Q')$$

This measure of relevance can express that Q' is more relevant for the available sources of information in the inquiry, but less relevant for the overall goal of the inquiry. The situation is reversed for Q, it turns out to be less efficient in using the available sources of information, but more useful for solving the main goal of the inquiry. This is because not all information serving the goal is available.

Other traditional notions that are conceptually related with the notion of relevance like, for example the entropy of a question and the informativity of an answer are suitable for similar epistemic modulations.

4.4.4 Imperfect Information in EQGs

As in games with sequential moves, all actions in an EQG are observable. This means that any player is fully informed about (can distinguish between) any move made previously in the game, either by herself or by other players. This fact is captured in the formalism by the fact that there are no uncertainty lines linking EIMs belonging to different histories.

Even so, it would not be completely accurate to consider that EQGs are games of perfect information due to the fact that agents still have uncertainties during the game, however, not about the moves played but about the worlds in the epistemic model. Because of these particularities a conceptual clarification with regard to the status of imperfect information in EQGs might be of interest.

4.4.12. Fact. Let $P = \{p_1, \dots, p_n\}$ and $A = \{a_1, \dots, a_m\}$ be sets of propositional atoms and agent-labels, respectively. Any epistemic-issue model can be represented as an imperfect information game in the following way:

– $N = \{a_1, \dots, a_m\} \cup \{c\}$, (here c is the chance player, or Nature),

– $S_c = \wp P = 2^{|P|}$, $S_i = \emptyset$ for $i \in N \setminus \{c\}$, (Nature chooses a possible world or epistemic alternative, the other players do nothing),

– $Q_i = \{D_1, \dots, D_k\} \subset \wp(\wp P)$, for $i \in N \backslash \{c\}$ (the players receive an arbitrary issue partition [the same]),

– $I_c = \{\{w\} \mid w \in \wp P\}$, $I_i = \{C_1, \dots, C_k\} \subset \wp(\wp P)$ for $i \in N \setminus \{c\}$ (each agent receives an information partition consistent with the valuation).

In the light of this fact we can say that each game with questioning moves starts from a pre-existing epistemic-issue model M and proceeds according to the previously introduced definitions. It is then of interest to find the properties that characterize precisely EQGs as games of imperfect information. There are results in the literature that characterize models generated by protocols of dynamic epistemic logic in terms of very general properties:

4.4.13. Fact. [Representation [97]] ETL models generated by state-dependent DEL protocols have the following properties: Propositional Stability, Syncronicity, Perfect Recall, Local No Miracles & Local Bisimulation Invariance. For PAL protocols we also have: Reflexive Events & Distinguished Events.

In the context of EQGs questioning actions resemble PAL and DEL actions but are in addition parametrized by a player which performs the action and the notion of bisimulation should also describe both epistemic and issue relations. We will also discuss the *Difraction Property* (DP) which is of interest especially for describing strategic abilities in EQGs in a later section.

4.4.5 Strategies and Solution Concepts

In this section we analyze EQGs using standard logical and game-theoretical notions and techniques. Strategies are defined in a standard way, like in [73]:

4.4.14. Definition. [Strategy] In a EQG a strategy for player $i \in N$ is a function $S_i : \{h \mid J(h) = i\} \setminus Z \to \{q \mid h^\frown q \in H\}$ that assigns an available question to each non-terminal history in which it is player i's turn to move. A strategy profile $S = (S_i)_{i \in N}$ is a tuple of strategies, one for each player. The outcome $O(S)$ of a strategy profile $S = (S_i)_{i \in N}$ is the history $h = \langle q_0, \ldots, q_l \rangle \in Z$ s.t. for $0 \leq k \leq l$ we have $S_{J(q_0,\ldots,q_k)}(q_0, \ldots, q_k) = q_{k+1}$.

Counting strategies. In the very abstract setting used so far the number of strategies at any node is a function of the size of the model $|S_i| = 2^{|W|}$ and the number of strategies for player i moving at history h is $(2^{|W|})^{|h|}$.

Another standard requirement for games with imperfect information is that the players must have uniform strategies, or that the strategies take into account the players' epistemic situation. There are various ways in which pragmatic and epistemic considerations can be used to capture this aspect in a setting of EQGs. For instance, we can add the following requirement:

$$\text{for } i = J(h') \text{ we have } F(h) \models \neg(K_i q_{l+1} \vee K_i \neg q_{l+1})$$

to item 1-4 in Definition 4.4.9, in an EIM M with $|W| = m$ and where agent i has n equivalence classes, then we are interested in the number of informative questions for player i in M. There are 2^{m-1} bi-partitions of W and 2^{n-1} unions of equivalence classes for i not including $\emptyset$ and W. Therefore the following set $R_i = \{\varphi : M \models_s \varphi \wedge \widehat{K}_i \neg\varphi, \varphi \in \mathcal{L}_{\mathsf{HL}}, s \in W\}$ contains all yes/no questions that are informative for i in M and, up to logical equivalence, there are $r_i = |R_i| = 2^{m-1} - 2^{n-1}$ such questions.

Intuitively this says that the criterion for strategy equivalence requires that the player only asks questions about what she does not know at the global level. However we can also require that questions are about what the agent doesn't know locally, in a given state w in an information cell C_i, with: for $i = J(h')$ we have $F(h) \models_w \neg(K_i q_{l+1} \vee K_i \neg q_{l+1})$ in this case the set $R_i = \{\varphi : M \models_s \varphi \wedge \widehat{K}_i \neg\varphi, \varphi \in \mathcal{L}_{\mathsf{HL}}, s \in C_i\}$ of locally informative questions depends on the structure of the information cell C_i of w inside the player's information partition. And since there are $2^{|c_i|-1}$ subsets of C_i, not including $\emptyset$ and W, and $2^{|W|-|C_i|}$ subsets of $W \setminus C_i$, we get $|R_i| = (2^{|C_i|-1}) \times 2^{|W|-|C_i|}$ logically non-equivalent strategies.

Note also that the expressive power of the language we use to express the questions is also a parameter in this counting. If we replace $\mathcal{L}_{\mathsf{HL}}$ by another language, say $\mathcal{L}_{\mathsf{EL}}$, the numbers might be different, depending on the concrete structure of the model, we get the previous numbers as upper bounds on the total number of logically non-equivalent strategies.

There are multiple other modeling options that could be used here to capture reasonable pragmatic preconditions. For instance, in order to capture a sequential game structure it is very plausible to require the following:

$$\text{for all } q_k \in h \text{ and } q_{l+1} \in \mathcal{L}_{\mathsf{HL}},\ [\![q_{l+1}]\!]_{F(h)} \neq [\![q_k]\!]_{F(h)},$$

intuitively, this says that players do not repeat a question which, even if informative, was already asked before. For $|M| = m$, and n information cells for agent i, the set $R_i = \{\varphi : M \models_s \varphi \wedge \widehat{K}_i \neg\varphi, \varphi \in \mathcal{L}_{\mathsf{HL}}, s \in W\}$ contains, up to logical equivalence, $r_i = |R_i| = 2^{m-1} - 2^{n-1}$ yes/no questions that are informative for i in M. But now the history of previously asked questions further restricts available actions, keeping r_i only as an upper bound.

The maximum number of allowed questions for a player i moving at history $h = \{q_0, \ldots, q_l\}$ will be given by $d_i^h = r_i \times r_{J(h^{-1})} \times r_{J(h^{-2})} \times \cdots \times r_{J(h^{-|h|})}$ where $h^{-n} = \{q_0, \ldots, q_{l-n}\}$. The total number of strategies for player i at history h will be at most $\binom{d_i^h}{g_i^h} = \frac{d_i^h!}{g_i^h!(d_i^h - g_i^h)!}$ and at least $\binom{d_i^h}{g_i^h} - |h|$ where $g_i^h = d_{J(h^{-1})}^{h^{-1}} \times d_{J(h^{-2})}^{h^{-2}} \times \cdots \times d_{J(\emptyset)}^{\emptyset}$. Finally, the number of strategy profiles in the EQG for M is $g_i^h \times \cdots \times g_i^h$ for all $i \in N$ and $h \in H$ for which $J(h) = i$.

For versions in which agents ask questions to each other not only to Nature, even more pragmatic preconditions make sense, like, for instance, that the questioner considers it possible that the questionee knows the answer.

Even disregarding pragmatic constraints, another aspect that is crucial to the notion of strategy equivalence is the expressive power of the language for goal formulas. If a certain language cannot express some goals then the number of strategies having equivalent effects upon execution will decrease.

Solution concepts. The general framework introduced so far to model EQGs can be adapted to capture a variety of concrete situations. But once a notion of strategy equivalence is fixed, it is even more important to study solution concepts in EQGs. These have standard game-theoretic definitions, like in [73]:

4.4.15. Definition. [Nash Equilibrium] An EQG Nash equilibrium is a strategy profile S^* s.t. for every player $i \in N$ and every strategy s_i of i we have:

$$U_i(O(S^*_{-i}, S^*_i)) \geq_i U_i(O(S^*_{-i}, S_i)).$$

We have now all the ingredients to obtain an analogous impossibility result analogous to what we had for strategic games:

4.4.16. Fact. There are EQGs in which no pure Nash equilibrium exists.

We only have to construct an appropriate example to establish this fact. We include the details in Section 4.7.

There are other solution concepts which are relevant for games in extensive form. These concepts also provide an adequate analysis for EQGs as particular cases of extensive-form games. One such solution concept, adequate for settings with sequential moves is subgame perfect equilibrium.

We also reserve an extensive discussion regarding the theoretical relevance of such an analysis in the general context of a logical approach to discovery and inquiry for a future occasion and here merely present some basic facts.

Many other interesting problems that are of interest for a general account of inquiry and scientific discovery in which questions play a genuine role emerge at this point. We end this section by listing some of the most relevant ones.

– Explore other solution concepts, considered to be more adequate for games with sequential moves, such as sub-game perfect equilibrium.

– Find systematic connections between existence of solution concepts and syntactic properties of goal formulas and/or strategies.

– Explore a conceptual framework for relevance of questioning and resolution in competitive and/or cooperative interrogative scenarios.

4.5 Strategic Abilities in Questioning

4.5.1 Describing Strategic Abilities

Players' abilities to achieve certain outcomes during the play of an EQG can be also described using a logical language. Such a language will contain "forcing modalities" like, for instance, in [90]:

$M, s \models \langle G, i\rangle\varphi$ *"player i has a* uniform *strategy for playing game G starting from state s which forces a set of outcomes satisfying φ in M"*

or other modalities expressing strategic abilities as in [34, 53]:

$M, q \models \langle\langle A\rangle\rangle\varphi$ *"there is a collective strategy S_A such that φ holds for every path [...] that may result from agents A executing strategy S_A from state q onward"*

Without introducing all the formal details of such frameworks, we mention below some well known facts and discuss their relationship with EQGs.

For perfect information games various fixed-point recursive characterizations of forcing modalities exist in the literature, like the game-logic from [90]:

$$\langle G, i\rangle\varphi \leftrightarrow (end \wedge \varphi) \vee (turn_i \wedge \Diamond\langle G, i\rangle\varphi) \vee (turn_j \wedge \Box\langle G, i\rangle\varphi)$$

or the ATL fixed-point axiom for strategic ability from [34]:

$$\langle\langle A\rangle\rangle\Box\varphi \leftrightarrow \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$$

However, if we consider imperfect information games in general these fixed-point recursive axioms are known to not be valid anymore. Given the special status of imperfect information in EQGs discussed before, it is natural to ask if such recursive axioms are valid in game structures generated by EQGs. In order to approach this question we need the following fact:

4.5.1. FACT. Any EQG induces a Concurrent Epistemic Game Structure. For EQG $E = \langle M, G\rangle$ the corresponding CEGS $S = \langle n, Q, \Pi, \pi, d, \delta, (\overset{a}{\sim})_{a\in\Sigma}\rangle$ is constructed in the following way (cf. Definition 1 in [34], also, [53] p. 440):

– $n = |N|$, – $Q = W \cup (W \times \wp(W)^n)$, – $\Pi = P$,

– $\pi(q) = \begin{cases} \{p \in P \mid q \in V(p)\} & \text{if } q \in W, \\ \{p \in P \mid \mathtt{fst}(q) \in V(p)\} & \text{otherwise} \end{cases}$

– $d_a(q) = |\wp(W)|$, for all $a \in \Sigma$,

– $\delta(q, j_1, \dots, j_n) = \begin{cases} (q, \langle j_1, \dots, j_n\rangle) & \text{if } q \in W, \\ q & \text{otherwise.} \end{cases}$

together with a family of equivalence (indistinguishability) relations $(\overset{a}{\sim})_{a\in\Sigma}$, one for each agent $a \in \Sigma$, with $\overset{a}{\sim} \subseteq Q \times Q$ computed in the following way:

– $\overset{a}{\sim} = \begin{cases} \{(q, q') \mid (q, q') \in \overset{a}{\sim}_M\} & \text{if } q, q' \in W, \\ \{(q, q') \mid \mathtt{snd}(q) = \mathtt{snd}(q'), (q, q') \in \overset{a}{\sim}_{F(\mathtt{snd}(q))\otimes !}\} & \text{otherwise.} \end{cases}$

We only describe here one level of questioning and resolution actions but this can be generalized in a similar fashion to any number of such iterations.

The Difraction Property. We discussed before some general properties of EQGs, one that becomes particularly relevant in this context because of its relevance in questioning scenarios is the one we will call the *difraction property*. The name is intended to capture the basic intuition behind the notion, namely that the value of the formula diverges in alternative histories. However, the formal definition is completely independent of this intuition, it is the following:

4.5.2. DEFINITION. [Difraction Property] We say that an epistemic game structure EGS satisfies the Difraction Property for φ (DP$_\varphi$) if, for some histories $h, h' \in H$, the following conditions:

– $h \sim h'$, $h \models \varphi$ and $h' \models \varphi$,

– there is some action (transition) q, such that $h^\frown q, h'^\frown q \in H$,

imply that the following property obtains:

– $h^\frown q \models \varphi$ and $h'^\frown q \not\models \varphi$ or vice versa.

If an EGS has the DP$_\varphi$ for any formula $\varphi \in \Gamma$ then it also has DP for Γ. If formula φ has factual content (only) we have Factual Difraction (FD), if formula φ has issue/epistemic content, Issue/Epistemic Difraction (I/ED).

Intuitively, the difraction property states that whenever a formula φ has uniform truth value in indistinguishable histories, and the same action q is available in both nodes/histories (q might be an issue/learning action [question, resolution, announcement], but does not have to), it follows that, after executing the same action q in both states/histories, the truth value of φ diverges (it is no longer the same in both resulting histories).

We used in Definition 4.5.2 a setting in which actions and transitions are assumed to coincide, such a setting is not specific to ATL but it is commonplace in game-theoretic contexts. However, Definition 4.5.2 can be straightforwardly reformulated even in a setting where actions and transitions are distinct as long as the transition function is deterministic.

DP is interesting because it seems to be essential for the failure of fixed-point axioms for strategic ability in imperfect information games. It can be shown [35] that both directions of the recursive ATL axiom characterizing strategic ability fail in the class of epistemic game structures that satisfy DP.

We show below that, in particular, the recursive axiom characterizing strategic ability holds in the class of non-DP epistemic game structures.

4.5.3. Fact. The axiom $\langle\langle A\rangle\rangle\Box\varphi \leftrightarrow \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$ is valid in the class of epistemic game structures that do not satisfy the difraction property (DP).

We include the details of the proof in the following appendix (Section 4.7).

What we still have to show is that epistemic game structures without DP indeed capture an interesting class of imperfect information games, in particular, that they contain some epistemic games with sequential moves. We show here, in particular, that EQGs do not have DP for positive formulas. The positive fragment of EL_Q, denoted $\mathcal{L}^+_{\mathsf{EL_Q}}$, is defined by the following BNF:

$$\begin{array}{rl} \varphi ::= & i \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \\ \chi ::= & \chi \wedge \chi \mid \chi \vee \chi \mid Q_a\varphi \mid R_a\varphi \mid K_a\varphi \end{array}$$

4.5.4. Fact. Epistemic game structures generated by extensive question games (EQGs) do not have the difraction property (DP) for positive formulas.

We include the details of the proof in the following appendix (Section 4.7).

We assume in Fact 4.5.4 a setting with questioning and resolution actions but this is not essential for the proof, link-cutting model transformations or non world-eliminating announcements are also captured in such a setting by $[\varphi!] = [\varphi?][!]$.

4.5.2 Conclusions and Further Research

In this section we have defined extensive questioning games in a formal way using the general framework of dynamic epistemic logic with questioning and resolution actions and we have also introduced some illustrative examples.

In this general setting we studied solution concepts in EQGs, in particular, we have shown that there are EQGs with no pure Nash equilibrium. The possibility of describing strategic abilities of players in EQGs in a logical language was explored. In this context we have shown how some strategic ability recursive axioms are preserved in imperfect information games without DP and that EQGs do not have DP for positive formulas.

Many interesting problems and topics for future research emerged, such as: study other solution concepts in the context of EQGs and the relation between existence of solution-concepts and syntactic structure of goal formulas, conceptual clarifications regarding notions of informativity and relevance of questioning moves in scenarios of competitive or cooperative inquiry, find all the remaining valid principles characterizing strategic ability in EQGs.

4.6 Appendix A: Background Definitions

Public Announcement Logic The logic of public announcements was one of the earliest developed in the DEL paradigm. It is also one of the most studied and well known members of the DEL family. Because we also used it in the first sections of this chapter we include a brief summary here.

The language $\mathcal{L}$ of public announcement logic (PAL) [79] defined over a set of agents $N = \{1, \dots, n\}$ and a set of primitive propositions Θ is given as follows, where i is an agent and $p \in \Theta$ is a propositional symbol:

$$\varphi ::= p \mid K_i\varphi \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\varphi_1!]\varphi_2$$

We write $\langle\varphi_1!\rangle\varphi_2$ resp. $\widehat{K}_i\varphi$ for the duals $\neg[\varphi_1!]\neg\varphi_2$ and $\neg K_i\neg\varphi$.

A *Kripke structure* or *epistemic model* over N and Θ is a tuple $M = (S, \sim_1, \dots, \sim_n, V)$ where S is a set of states, $\sim_i \subseteq S \times S$ is an epistemic indistinguishability relation that is assumed to be an equivalence relation for each agent i, and $V : \Theta \to S$ assigns primitive propositions to the states in which they are true. A *pointed Kripke structure* is a pair (M, s) where s is a state in M. In this chapter and beyond we will also assume that Kripke structures are *finite*.

The interpretation of formulae from the public announcement language is defined in a pointed Kripke structure at state as follows:

$M, s \models p$ iff $p \in V(p)$, $M, s \models \neg\varphi$ iff not $M, s \models \varphi$

$M, s \models K_i\varphi$ iff for every t such that $s \sim_i t$, $M, t \models \varphi$; and

$M, s \models \varphi \wedge \psi$ iff $M, s \models \varphi$ and $M, s \models \psi$

$M, s \models [\varphi!]\psi$ iff $M, s \models \varphi$ implies that $M|\varphi, s \models \psi$,

where $M|\varphi = (S', \sim'_1, \ldots, \sim'_n, V')$ such that $S' = \{s' \in S : M, s' \models \varphi\}$, $\sim'_i = \sim_i \cap (S' \times S')$, and $V'(p) = V(p) \cap S'$. For $\{s' \in S : M, s' \models \varphi\}$ we also write $[\![\varphi]\!]_M$.

Strategic Game A *strategic game* is a triple $G = \langle N, \{A_i : i \in N\}, \{u_i : i \in N\}\rangle$ where: $N = \{1, \ldots, n\}$ is the finite set of *players*; for each $i \in N$, A_i is the set of *strategies* (or *actions*) available to i. $A = \times_{j \in N} A_j$ is the set of *strategy profiles*; and for each $i \in N$, $u_i : A \to \mathbb{R}$ is the *payoff function* for i, mapping each strategy profile to a number. Notation $(a_1, \ldots, a_n)[a_i/a'_i]$ stands for the profile wherein strategy a_i is replaced by a'_i. A strategy profile is a (pure strategy) *Nash equilibrium* if every strategy is the best response of that agent to the strategies of the other agents, i.e., if the agent can not do any better by choosing a different strategy given that the strategies of the other agents are fixed. Formally, a profile $(a_1, \ldots, a_n)$ is a Nash equilibrium if and only if for all $i \in N$, for all $a'_i \neq a_i$, $u_i((a_1, \ldots, a_n)[a_i/a'_i]) \leq u_i(a_1, \ldots, a_n)$.

A strategy for an agent is *weakly dominant* if it is at least as good for that agent as any other strategy, no matter which strategies the other agents choose. Formally, a strategy a_i for agent i is weakly dominant if and only if for all strategies $a'_1, \ldots, a'_n$ for players $1, \ldots, n$ respectively, $u_i(a'_1, \ldots, a'_n) \leq u_i((a'_1, \ldots, a'_n)[a'_i/a_i])$.

Bayesian game The most common model of strategic games with imperfect information is the *Bayesian game* [45]. Our presentation of Bayesian games is as in [73]. A *Bayesian strategic game* $BG = \langle N, S, \{A_i : i \in N\}, \{T_i : i \in N\}, \{Pr_i : i \in N\}, \{\tau_i : i \in N\}, \{u_i : i \in N\}\rangle$ has the following components:

N is the set of players; S is the finite set of *states* s modeling the players' uncertainty about each other; and for each $i \in N$: A_i is the set of strategies or choices in the game; T_i is the set of *signals* t_i that may be observed by player i, $\tau_i : S \to T_i$ is the *signal function* of player i; Pr_i is a probability measure on S (the *prior belief* of player i) such that $Pr_i(\tau^{-1}(t_i)) > 0$ for all $t_i \in T_i$, that is, each player's received signal is correct with strictly positive probability; and finally u_i is a payoff function on the set of probability measures over $A \times S$ (instead of a payoff function on the set of action profiles a).

4.7 Appendix B: Proofs of Main Results

4.7.1. Proof (Fact 4.3.2). The following are the answers corresponding to each strategy profile in the game:

(S_b, S_a)	0	1	2	3	4	5	6	7	(a_M, b_M)
(S^b_0, S^a_0)	$\overline{p}, \overline{q}$	$\overline{p}, \overline{q}$	$\overline{p}, q$	$\overline{p}, q$	$p, \overline{q}$	$p, \overline{q}$	p, q	p, q	$(\quad, m)$
(S^b_0, S^a_1)	$\overline{r}, \overline{q}$	$r, \overline{q}$	$\overline{p}, q$	$\overline{p}, q$	$\overline{r}, \overline{q}$	$r, \overline{q}$	p, q	p, q	$(\quad, \quad)$
(S^b_0, S^a_2)	$\overline{p}, \overline{q}$	$\overline{p}, \overline{q}$	$\overline{r}, q$	r, q	$p, \overline{q}$	$p, \overline{q}$	$\overline{r}, q$	r, q	$(\quad, \quad)$
(S^b_0, S^a_3)	$\overline{r}, \overline{q}$	$r, \overline{q}$	$\overline{r}, q$	r, q	$\overline{r}, \overline{q}$	$r, \overline{q}$	$\overline{r}, q$	r, q	$(m, \quad)$

(S_b, S_a)	0	1	2	3	4	5	6	7	(a_M, b_M)
(S^b_1, S^a_0)	$\overline{p}, \overline{r}$	$\overline{p}, r$	$\overline{p}, \overline{r}$	$\overline{p}, r$	$p, \overline{q}$	$p, \overline{q}$	p, q	p, q	$(\quad, \quad)$
(S^b_1, S^a_1)	$\overline{r}, \overline{r}$	r, r	$\overline{p}, \overline{r}$	$\overline{p}, r$	$\overline{r}, \overline{q}$	$r, \overline{q}$	p, q	p, q	$(m, \quad)$
(S^b_1, S^a_2)	$\overline{p}, \overline{r}$	$\overline{p}, r$	$\overline{r}, \overline{r}$	r, r	$p, \overline{q}$	$p, \overline{q}$	$\overline{r}, q$	r, q	$(\quad, m)$
(S^b_1, S^a_3)	$\overline{r}, \overline{r}$	r, r	$\overline{r}, \overline{r}$	r, r	$\overline{r}, \overline{q}$	$r, \overline{q}$	$\overline{r}, q$	r, q	$(\quad, \quad)$

(S_b, S_a)	0	1	2	3	4	5	6	7	(a_M, b_M)
(S^b_2, S^a_0)	$\overline{p}, \overline{q}$	$\overline{p}, \overline{q}$	$\overline{p}, q$	$\overline{p}, q$	$p, \overline{r}$	p, r	$p, \overline{r}$	p, r	$(\quad, \quad)$
(S^b_2, S^a_1)	$\overline{r}, \overline{q}$	$r, \overline{q}$	$\overline{p}, q$	$\overline{p}, q$	$\overline{r}, \overline{r}$	r, r	$p, \overline{r}$	p, r	$(\quad, m)$
(S^b_2, S^a_2)	$\overline{p}, \overline{q}$	$\overline{p}, \overline{q}$	$\overline{r}, q$	r, q	$p, \overline{r}$	p, r	$\overline{r}, \overline{r}$	r, r	$(m, \quad)$
(S^b_2, S^a_3)	$\overline{r}, \overline{q}$	$r, \overline{q}$	$\overline{r}, q$	r, q	$\overline{r}, \overline{r}$	r, r	$\overline{r}, \overline{r}$	r, r	$(\quad, \quad)$

(S_b, S_a)	0	1	2	3	4	5	6	7	(a_M, b_M)
(S^b_3, S^a_0)	$\overline{p}, \overline{r}$	$\overline{p}, r$	$\overline{p}, \overline{r}$	$\overline{p}, r$	$p, \overline{r}$	p, r	$p, \overline{r}$	p, r	$(m, \quad)$
(S^b_3, S^a_1)	$\overline{r}, \overline{r}$	r, r	$\overline{p}, \overline{r}$	$\overline{p}, r$	$\overline{r}, \overline{r}$	r, r	$p, \overline{r}$	p, r	$(\quad, \quad)$
(S^b_3, S^a_2)	$\overline{p}, \overline{r}$	$\overline{p}, r$	$r, \overline{r}$	r, r	$p, \overline{r}$	p, r	$\overline{r}, \overline{r}$	r, r	$(\quad, \quad)$
(S^b_3, S^a_3)	$\overline{r}, \overline{r}$	r, r	$\overline{r}, \overline{r}$	r, r	$\overline{r}, \overline{r}$	r, r	$\overline{r}, \overline{r}$	r, r	$(\quad, m)$

We give next the payoff matrix of the induced game mentioned in Fact 4.3.2.

	S^b_0	S^b_1	S^b_2	S^b_3
S^a_0	0.00, 1.00	0.25, 0.75	0.25, 0.75	0.50, 0.50
S^a_1	0.25, 0.75	0.00, 1.00	0.50, 0.50	0.25, 0.75
S^a_2	0.25, 0.75	0.50, 0.50	0.00, 1.00	0.25, 0.75
S^a_3	0.50, 0.50	0.25, 0.75	0.25, 0.75	0.00, 1.00

□

4.7.2. Proof (4.4.16). This simple fact is witnessed by considering an EQG with two agents a and b informed only about facts q and p, respectively, and only aware of propositional atoms p, q, r, in which a has the following goal formula:

$$\gamma_a = (p \leftrightarrow \overline{q}) \wedge W_a r \wedge (W_a p \vee W_b q)$$

and symmetrically γ_b for the second player b, and where $W_i = K_i\varphi \vee K_i\overline{\varphi}$. The model described informally so far in Fact 4.3.2 can be precisely specified using in our later Haskell implementation by the following model transformations:

```
*QAGM> m7 = upd ( upd ( initM [a,b] [P 0, Q 0, R 0] )
                ( info [a] (P 0) ) )
                ( info [b] (Q 0) )
```

and the resulting EQG played in `m7` has the following outcomes:

	s^a_0	s^a_1	s^a_2	s^a_3
s^b_0	$1.00, 0.00$	$0.75, 0.25$	$0.75, 0.25$	$0.50, 0.50$
s^b_1	$0.50, 0.50$	$0.75, 0.25$	$0.75, 0.25$	$1.00, 0.00$

As in the previous example, we have a cycling pattern of local optima that causes global inexistence of NE. $\Box$

4.7.3. Proof (Fact 4.5.3). Let $S = \langle n, Q, \Pi, \pi, d, \delta, (\overset{a}{\sim})_{a\in\Sigma}\rangle$ be an arbitrary epistemic game structure without DP. Suppose that $\langle\langle A\rangle\rangle\Box\varphi \not\to \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$. Then, for some $q \in Q$, we have $q \models \langle\langle A\rangle\rangle\Box\varphi$ but $q \not\models \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$. In case $q \not\models \varphi$ we are done. If $q \not\models \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle\Box\varphi$ then, by the semantics, for all uniform A-strategies F_A, for some computation $\lambda \in out(q, F_A)$, we have $\lambda[1] \not\models \langle\langle A\rangle\rangle\Box\varphi$. This means, by the semantics, that for all uniform A-strategies F_A, for some computation $\lambda' \in out(\lambda[1], F_A)$, and for some position $i \geq 0$, we have $\lambda'[i] \not\models \varphi$. From $q \models \langle\langle A\rangle\rangle\Box\varphi$ we have, by the semantics, that there exist a uniform A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ and all positions $i \geq 0$, we have $\lambda[i] \models \varphi$. But we also have, for all $q \in Q$ and $a \in \Sigma$, that $q \overset{a}{\sim} q$, because $\overset{a}{\sim}$ is an equivalence relation. Therefore, S must satisfy DP, against the assumption. As q and S are arbitrary this holds for all non-DP EGS. The other direction is similar. $\Box$

4.7.4. Proof (Fact 4.5.4). If for an atomic φ we have $h \sim h'$, $h \models \varphi$ and $h' \models \varphi$ then, for any questioning action q, we also have both $h^\frown q \models \varphi$ and $h'^\frown q \models \varphi$, because for any EIM $M = \langle W, \sim, \approx V\rangle$ and $M' = M \otimes q = \langle W, \sim', \approx', V'\rangle$ we have, by $[\varphi?], [\,!\,]$ definitions, that $V = V'$. If φ is a con(/dis)junction of positive formulas $\psi\wedge(/\vee)\chi$ then we use the induction hypothesis. If φ is a modal formula $Q\psi$ we have to consider two cases. If $h^\frown q \approx h'^\frown q$ we are done. If $h^\frown q \not\approx h'^\frown q$ then, suppose that for some $k \in H$, $h^\frown q \approx k^\frown q$ and $k \models \neg\psi$, and, for all $k' \in H$ such that $h'^\frown q \approx k'$, we have $k' \models \psi$. In the first case φ is a positive formula only if ψ is factual. But then we also have $k \models \neg\psi$ because questioning actions do not change factual content. Then by the definition of $[q?]$, for any epistemic-issue models $M = \langle W, \sim, \approx V\rangle$ and $M' = M \otimes q = \langle W, \sim', \approx', V'\rangle$, we have $\approx' = \approx \cap \overset{q}{\equiv}_M$, hence we must also have $h \approx k$ but then, by the semantics of Q, we have $h \not\models \varphi$ and this contradicts the initial assumption. For resolution $[\,!\,]$ actions, indistinguishability $\sim$ and formulas with epistemic $K\psi$ or mixed $R\psi$ content the argument is analogous. $\Box$

Chapter 5
Implementing Questioning Games

In this chapter we present and document the implementation behind the question answer games discussed in previous chapter. Some outstanding features of the implementation are the following: a representation strategies in game as questions in an epistemic model, computing resulting strategy profiles and intuitive display of formulae representing game-moves, computation of both local and global epistemic actions induced by questioning strategies and intuitive display for updated epistemic models, model checking of goal formulae in resulting epistemic structures, implementation of epistemic games with questioning moves and computation and intuitive display of the corresponding matrix for the induced game. The chapter presents and explains the literate *Haskell* (cf. [55, 68]) code of the `QAGames.lhs` module covering the theoretical aspects surrounding the epistemic games with questioning moves discussed in Chapter 4. Previous epistemic functionality from *DEMo* [107], *DEMo-light* [109], and *DELq* functionality from [71] is also explained and used in the current implementation. Finally, the `PAGs.lhs` module is used to compare questioning games with games with announcements.

In the last section we will define the birelataional coarsest partition problem and we will give an algorithm for minimizing issue-epistemic models using a notion of behavioral equivalence that is adequate for the questioning language.

5.1 Implementing Questioning Games

Much of the theoretical notions introduced so far and even more so much of the functionality contained in the discussed implementation have their utility enriched in applications involving various scenarios of strategic interaction between rational agents via dynamic questioning and informative actions.

This section implements extensions of basic epistemic functionality to model epistemic games with informative and questioning moves in the style of [2] and [1]. We will first model games in which moves of players are epistemic actions which are public announcements triggered by mutual strategic questions, and in which the players seek to achieve goals represented by epistemic formulae.

We will use as the point of departure the standard dynamic epistemic functionality for dynamic epistemic logic from *DEMo-light* [109]. An introductory course presentation of both DEL and its basic Haskell implementation, in a version without vocabulary change, is also available in [104]. The *DEMo-light* modules in [109] provide all the needed functionality to achieve the purpose of this section.

Staring from here we add an extension module that uses the PAL and DEL functionality for informative actions and builds the additional functionality needed to capture the strategic aspects involved in question answer games.

The working of the code is afterwards illustrated in all minutia in Section 5.2 using a paradigmatic example. We will only present hereafter the main features of the code and use it for examples already discussed in Chapter 4.

The main utility provided by the code consists in taking an epistemic-goal structure with two main components a model and a pair of goal formulae and building from these the game matrix for the strategic question-answer game played in the given epistemic model with the given goals.

The following modules are imported by the main `QuestionGames.lhs` module:

`ModelsVocab` The module defines the basic data structures for epistemic models and epistemic formulae using an underlying propositional vocabulary and the underling strong Kleene calculus.

`ActionVocab` The module defines the data structures for action models and uses it to implement action model update. The module also contains the implementation of public announcements.

`ChangeVocab` The module adds functionality for factual change and model checking for epistemic logic alongside with concrete illustrations for how to implement perception and information dynamics.

`ShortctsGms` The module contains various syntactic sugars useful for defining and working with question-specific notions, like, for instance, the tripartite extension used as the basis for strategy equivalence, the domain naming functionality, etc. It also contains additional useful functions that have only an ancillary role in display and related computations.

5.1.1 The `QAGames.lhs` Question-Answer Games Module

The module starts by importing background functionality as described before, lines 2-10. The next seven functions introduce utilities needed to define execution values for players' strategies based on the structure of the epistemic model. First a formula is linked with a subset of states in the epistemic domain, line 12. The second level also takes into account the indistinguishability relation for an agent when computing the relevant extension of a formula line 15. Given any model as input one can determine using nominals the number of formulae with different extensions, line 21, this also gives an upper bound to the number of strategies in the game. A questioning action lifts the extension of a formula to a set of three extensions corresponding to the triple of epistemicaly relevant answers, line 25.

```
1  module QAGames where
2  import Control.Monad
3  import List
4  import Data.Ord (comparing)
5  import CombinatoricsGeneration
6  import qualified Data.Set as Set
7  import ShortctsGms
8  import ModelsVocab hiding (m0)
9  import ActionVocab hiding (upd,public,preconditions,voc)
10 import ChangeVocab
11
12 extension :: (Ord a) => EpistM a -> Form -> Maybe [a]
13 extension m f = filterM (\x -> (isTrueAtMayb m x f)) (dom m)
14
15 knowsExtension3 :: (Ord state) =>
16   EpistM state -> Agent -> [[Maybe [state]]]
17 knowsExtension3 m ag =
18   [[extension m (K ag x), extension m (Neg (Disj[(K ag x),
19   (K ag (Neg x)) ])), extension m (K ag (Neg x))] | x <- (forms m)]
20
21 forms :: (Ord state) => EpistM state -> [ Form ]
22 forms m = map (\y -> Disj y) (map (\x -> noml m x)
23   (powerList (dom m)))
```

After sorting the tripartite extension of the epistemic answers, line 29, the upper bound of questioning strategies having non equivalent execution value in a model can be sensibly lowered by merging all formulae with equivalent execution value into only one representative strategy, line 34.

```
25 formsK3 :: (Ord state) =>
26   EpistM state -> Agent -> [(Form, [Maybe [state]])]
27 formsK3 m ag = zip (forms m) (knowsExtension3 m ag)
28
29 formsK3sort :: (Ord state) =>
30   EpistM state -> Agent -> [(Form, [Maybe [state]])]
31 formsK3sort m ag = zip (forms m) (map sort
32   (knowsExtension3 m ag))
33
34 formsK3nuby :: (Ord state) =>
35   EpistM state -> Agent -> [(Form, [Maybe [state]])]
36 formsK3nuby m ag = nubBy (\x y -> (snd y) ==
37 (snd x)) (zip (forms m) (map sort (knowsExtension3 m ag)))
```

The following four functions use the notions implemented so far to compute global strategy profiles in the game. The first aspect concerns the fact that strategies are uniform inside the same information cell but might differ between elements of the partition, line 39. Next the individual strategies are globally aggregated to form strategy profiles for all the agents, line 45.

```
39 straGlob :: (Eq state, Ord state, Num state) =>
40   EpistM state -> Agent -> [ [ ([state],Form) ] ]
41 straGlob m ag = cartProd (map (\x ->
42   (zip (take (length (formsK3nuby m (aminus m ag))) (repeat x))
43   (map fst (formsK3nuby m (aminus m ag))))) (infopart m ag))
44
45 profGlob :: (Eq state, Ord state, Num state) =>
46   EpistM state -> [[[([state],Form)]]]
47 profGlob m = cartProd [straGlob m ((agents m) !! 0) ,
48   straGlob m (aminus m ((agents m) !! 0))]
49
50 w2infoCells :: (Eq state, Ord state, Num state) =>
51   EpistM state -> state -> [[state]]
52 w2infoCells m w = [infocell m ((agents m)!!0) w,
53   infocell m ((agents m)!!1) w]
54
55 w_prof2forms :: (Eq state, Ord state, Num state) =>
56   EpistM state -> state -> [[([state],Form)]] -> [Form]
57 w_prof2forms m w p = map snd (filter (\x -> (fst x) ==
58   infocell m ((agents m)!!0) w) (p!!0) ++
59   filter (\x -> (fst x) == infocell m ((agents m)!!1) w) (p!!1))
```

Each world in the model corresponds to a list of information cells in the two agents' partitions, as computed in line 50. Furthermore, each global strategy profile determines a list of corresponding formulae, line 55. Next, each of the formulae determined by the global profiles have a corresponding execution value, as calculated in line 61, following the tripartite model discussed before.

```
61 w_prof2exeVal :: (Eq state, Ord state, Num state) =>
62   EpistM state -> state -> [[([state],Form)]] -> [Form]
63 w_prof2exeVal m w p = [ barval3 m w ((agents m)!!0)
64   ((w_prof2forms m w p)!!0), barval3 m w ((agents m)!!1)
65   ((w_prof2forms m w p)!!1) ]
66
67 w_prof2answ :: (Eq state, Ord state, Num state) =>
68   EpistM state -> state -> [[([state],Form)]] -> Form
69 w_prof2answ m w p = Conj (w_prof2exeVal m w p)
70
71 w_prof2upd :: (Eq state, Ord state, Num state) =>
72   EpistM state -> state -> [[([state],Form)]] -> EpistM state
73 w_prof2upd m w p = upd_pa m (w_prof2answ m w p)
74
75 w2updates :: (Eq state, Ord state, Num state) =>
76   EpistM state -> state ->  [EpistM state]
77 w2updates m s = map (\x -> (w_prof2upd m s x)) (profGlob m)
```

The result is a conjunction of execution values, one for each agent, line 67. And each answer computed as a conjunction of execution values leads to a corresponding update of the initial model, as computed starting from line 71.

Taking each state of the model as the point of departure, each global strategy profile can be executed in it, leading to a list of corresponding updates, line 75.

```
79  w2pays :: (Eq state, Ord state, Num state) =>
80    EpistM state -> state -> Form -> [Integer]
81  w2pays m w g = map (\x -> paynumber x w g) (w2updates m w)
82
83  prof2w_pays :: (Eq state, Ord state, Num state) =>
84    EpistM state -> [[([state],Form)]] -> Form -> [Integer]
85  prof2w_pays m p g =
86    map (\x -> paynumber (w_prof2upd m x p) x g) (dom m)
87
88  prof2w_paysum :: (Eq state, Ord state, Num state) =>
89    EpistM state -> [[([state],Form)]] -> Form -> Integer
90  prof2w_paysum m p g = foldr (+) 0 (prof2w_pays m p g)
```

The next and final stage in constructing the game matrix consists in assigning payoffs to game outcomes, this is done by the following four functions. The payoff value is determined at each world by model checking agents' goal formulae in the resulting updated model, line 79. The result of model checking goal formulae can be lifted from local states to a list of worlds in the domain and global strategy profiles, line 83. The payoff values of local strategy execution are aggregated in a global sum corresponding to each strategy profile, computed at line 88.

Finally, the qagn function computes the resulting entries in the matrix induced by the question-answer game played in the given epistemic model, line 92.

```
92  qagn :: (Eq state, Ord state, Num state) =>
93    EpistM state -> (Form,Form) -> [(Integer,Integer)]
94  qagn m g = zip (map (\x -> prof2w_paysum m x (fst g))
95  (profGlob m)) (map (\x -> prof2w_paysum m x (snd g)) (profGlob m))
```

Two additional auxiliary functions, which both assume a named epistemic model as input, are used to compute the epistemic projection of a formula, first as a bipartite epistemic announcement, line 98, and second as a tripartite epistemic answer to a question, starting from line 101. This pair of function also give a concise comparison point between games played with spontaneous announcement moves and games played with questioning moves followed by answers.

```
97  barval2 :: (Ord state) => EpistM state -> state -> Agent -> Form -> Form
98  barval2 m s ag f | isTrueAtMayb m s (K ag f) == Just True = K ag f
99                   | otherwise = Neg (K ag f)
100
101 barval3 :: (Eq state, Ord state) =>
102             EpistM state -> state -> Agent -> Form -> Form
103 barval3 m s ag f | isTrueAtMayb  m s (K (aminus m ag) f) == Just True =
104                     K (aminus m ag) f
```

```
105                     | isTrueAtMayb  m s (K (aminus m ag) (Neg f)) ==
106                       Just True = K (aminus m ag) (Neg f)
107                     | otherwise =
108                       (Neg (Disj[(K (aminus m ag) f),
109                       (K (aminus m ag) (Neg f)) ]))
```

5.1.2 The PAGs.lhs Extension Module

The infrastructure used to implement games with questioning actions uses answers as informative actions in their simplest form as public announcements. In this section we will show how this infrastructure can be extended to accommodate games with informative epistemic games. We present the remaining ancillary functionality as a separate module merely for ease of exposition, it is in fact a constitutive part of the `QAGames.lhs` module, sharing much of the functionality.

The `infopart` function reconstructs the information partition of a given agent in an epistemic model `m`, line 111. Next, using the function at line 114, the information cell determined by an additional state parameter can be retrieved.

```
111 infopart :: (Ord state) => EpistM state -> Agent -> [[state]]
112 infopart m a = rel2partition (dom m) (rel a m)
113
114 infocell :: (Ord state, Num state) =>
115   EpistM state -> Agent -> state -> [state]
116 infocell m a w =
117   (filter (\x -> (elem w x)) (infopart m a))!!0
```

At line 119 and following the information cell is mapped over the list of states in the domain, this will be useful for further processing in the next steps. The first processing step converts the information cells from sets of states to lists of nominals for the states, line 125. This list is further processed by mapping a disjunction over the nominals in the function starting from line 129.

```
119 kKpartState :: (Ord state, Num state) =>
120   EpistM state -> Agent -> state -> [[state]]
121 kKpartState m a s =  map (\x -> foldl union [] x)
122   (filter (\x -> ((infocell (named m) a s) `elem` x))
123   (powerList (infopart (named m) a)))
124
125 cellpartnomK :: (Ord state, Num state) =>
126   EpistM state -> Agent -> state -> [[Form]]
127 cellpartnomK m a s = map (\x -> (noml (named m) x)) (kKpartState (named m) a s)
128
129 cellpartdisK :: (Ord state, Num state) =>
130   EpistM state -> Agent -> state -> [Form]
131 cellpartdisK m a s = map Disj (cellpartnomK (named m) a s)
132
133 cellpartdisKbar :: (Ord state,Num state)=> EpistM state -> Agent -> state -> [Form]
134 cellpartdisKbar m ag s = map (\x->(barval2 m s ag x)) (cellpartdisK (named m) ag s)
```

Finally, the bipartite epistemic value of the disjunctions is computed by mapping model checking operations over the previous list, line 133.

```
136 realcellstratsK :: (Eq state, Ord state, Num state) =>
137   EpistM state -> Agent -> [state] -> [Form]
138 realcellstratsK m ag c = cellpartdisKbar m ag (c!!0)
139
140 strategiesKK :: (Eq state, Ord state, Num state) =>
141   EpistM state -> Agent -> [ [ ([state],Form) ] ]
142 strategiesKK m a = cartProd (map (\x ->
143   (zip (take (length (realcellstratsK m a x)) (repeat x))
144   (realcellstratsK m a x))) (infopart m a))
145
146 profilesKK :: (Eq state, Ord state, Num state) =>
147   EpistM state -> [[[([state],Form)]]]
148 profilesKK m = cartProd [strategiesKK m ((agents m) !! 0),
149   strategiesKK m (aminus m ((agents m) !! 0))]
```

Taking only one representative state instead of an entire partition cell simplifies further computations by preserving only the minimally required information, line 136. Global strategies are computed at line 140 for a model and a player given as inputs by assigning to every cell in the information partition a formula.

This ensures that strategies are information dependent by being uniform over cells in the knowledge partition. Global strategies for agents are lifted in global strategy-profiles by taking the Cartesian product of individual strategies, in the function starting at line 146. The pair of announcements corresponding to each strategy profile is computed at line 151 by filtering relevant formulae for each equivalence class. This gives rise to a joint announcement formula obtained by taking the conjunction of individual announcements, in line 156.

```
151 announcementsKK :: (Eq state, Ord state, Num state) =>
152   EpistM state -> state -> [[Form]]
153 announcementsKK m s = (map (\y->(map snd ((filter (\x->(elem s (fst x))) (y!!0))
154   ++ (filter (\x -> (elem s (fst x))) (y!!1) )))) (profilesKK m))
155
156 jointannouncementKK :: (Eq state, Ord state, Num state) =>
157   EpistM state -> state -> [Form]
158 jointannouncementKK m s = map (\x -> (Conj x)) (announcementsKK m s)
159
160 updatesKK :: (Eq state, Ord state, Num state) =>
161   EpistM state -> state -> [EpistM state]
162 updatesKK m s = map (\x -> (upd_pa m x)) (jointannouncementKK m s)
163
164 paynumber :: (Eq state, Ord state, Num state) =>
165   EpistM state -> state -> Form -> Integer
166 paynumber m s f | isTrueAtMayb m s f == Just True = 1
167                 | otherwise = 0
168
169 lpayKK::(Eq state,Ord state,Num state)=>EpistM state->state->Form->[Integer]
170 lpayKK m s f = map (\x -> (paynumber x s f)) (updatesKK m s)
```

```
171
172 gpayKK :: (Eq state, Ord state, Num state) =>
173   EpistM state -> Form -> [[Integer]]
174 gpayKK m f = map (\x -> (lpayKK m x f)) (dom m)
175
176 sumgpayKK :: (Eq state, Ord state, Num state) =>
177   EpistM state -> Form -> [Integer]
178 sumgpayKK m f = foldr (zipWith (+))
179   (take (length (profilesKK m)) (repeat 0)) (gpayKK m f)
180
181 pagn :: (Eq state, Ord state, Num state) =>
182   EpistM state -> (Form,Form) -> [(Integer,Integer)]
183 pagn m g = (zip (sumgpayKK m (fst g)) (sumgpayKK m (snd g) ))
```

The joint announcement formulae determine the list of resulting updated models, which are computed by the function starting at line 160.

The final stage in computing the normal form of the epistemic game consists in model-checking goal formulae in the updated models and assigning payoffs to the obtained results. The function from line 164 converts the result of model checked goal formula into a corresponding numerical value.

The following function, from line 169, maps the conversion over the list of resulting update models, returning all the numerical values in a list. The next function, line 172, lifts this computation even higher by abstracting away the particular state and projecting over the entire list of states in the domain. A final step consists in taking the sum of all the values in the function from line 176.

Finally, the `pagm` function, line 181, computes the resulting game matrix.

5.2 Illustrations using the Implementation

Now that the implementation details have been discussed and explained in detail, we continue by introducing one extended illustrative analysis of a concrete questioning game and discuss some theoretical aspects that emerge.

The illustration below follow the code description and explanation. The output will stay very close to the succession of processing functions in Haskell implementation for games with questioning moves previously discussed. Whenever needed the names of the functions used in the output can be used refer back to code sections for further details, documentation and further explanations.

5.2.1. Example. [We Are Spies] Consider the following three-state epistemic goal structure where a knows the truth about p and b knows the truth about q:

$$\gamma_a = K_b p \to K_a q \qquad \begin{tikzcd} \overline{q}p \arrow[r, dash, "a"] & qp \arrow[r, dash, "b"] & q\overline{p} \end{tikzcd} \qquad \gamma_b = K_a \neg q$$

We start by considering the epistemic model used in the 'We Are Spies' example (henceforth WaS). The starting epistemic structure is:

```
*QAGames> displayS5 m79
[1,2,3]
[(1,[p]),(2,[q]),(3,[p,q])]
(a,[[1,3],[2]])
(b,[[1],[2,3]])
[1,2,3]
```

DELq formulae can be evaluated in such structures. Formulas are represented and visualized in Haskell in a compact way, see Section 3.2 for some intuitive examples. Another important element in modeling QAGs are the dynamic epistemic actions changing the underlying epistemic structures. Such epistemic actions can be questions or answers and they are represented as model transformations.

For instance, the result of updating the epistemic model considered in Example 1 with the formula $K_a p$ is computed in and visualized by `QAGames.lhs` as follows:

```
*QAGames> displayS5 (upd_pa m79 (K a p))
[1,3]
[(1,[p]),(3,[p,q])]
(a,[[1,3]])
(b,[[1],[3]])
[1,3]
```

Finally, the truth-value of formulas can be computed at a state in a model. The following is an illustration of how `QAGames.lhs` performs model checking:

```
*QAGames> isTrueAtMayb (upd_pa m79 (K a p)) 1 (K b q)
False
```

This says intuitively that in the model of Example 5.2.1 updated by announcing $K_a p$ the formula $K_b q$ does not hold at the world with index 1.

5.2.1 Counting Strategies

There are two ways to consider a strategy in a question-answer game. One is the syntactic way, not very convenient because makes the number of possible moves in the game infinite as φ, $\varphi \wedge \varphi$, etc. are syntactically different strategies. If not for conceptual reasons, at least as far as an efficient implementation is considered desirable, it would be useful to restrict this number.

The second, semantic, perspective can be used to reduce this number: two strategies are equivalent if they have the same extension. This is the notion of logical equivalence between formulas. Even so, for every subset of the domain there are infinitely many formulae that have that extension, but for a finite model we will have only a finite set of subsets. It is enough to take one representative of each equivalence class. For practical reasons we will use the minimal representative.

When using a semantic approach for strategy equivalence and syntactic entities in a language to represent the moves in the game is very useful to establish a meaningful correspondence between the two levels. This can be done by naming a model, that is adding a nominal to each world's valuation list. A nominal is a propositional atom which is true in only one world in the domain:

```
*QAGames> displayS5 (named m79)
[1,2,3]
[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])]
(a,[[1,3],[2]])
(b,[[1],[2,3]])
[1,2,3]
```

This can be then used to construct the set of all extensionally non-equivalent formulae expressible in the language with respect to a given epistemic model.

In the (WaS) example we have as many extensionally non-equivalent formulae:

```
*QAGames> forms (named m79)
[v[],v[n2],v[n1],v[n1,n2],v[n],v[n,n2],v[n,n1],v[n,n1,n2]]
```

Such formulae are disjunction of propositional letters true at only one world, or nominals, which correspond directly to subsets of the domain after the worlds in the initial epistemic model have been named.

As discussed before, using nominals provides extra expressive power at no computational expense. If, however, the model we are working with is minimized under bisimulation, nominals are not needed, because a characteristic formula of epistemic logic could be used instead to identify any world. Even so, we will see that in the context of associating payoffs to worlds the number of worlds satisfying the same formulae matters and minimization under bisimulation erases relevant information for computing the payoffs in a question-answer game.

However, our example only has local games with two, not eight, strategies. This is because some of the formulae have equivalent semantic effects in the game. This proliferation, however, turns out to be redundant in the sense that it can be avoided using a more efficient compact representation without affecting solution concepts. For this reasons, our reduction gives rise to a natural and economic criterion of strategy equivalence in question answer games. This basic fact is captured by the following more general result for equilibria with pure strategies:

5.2.2. Fact. Let M be a named epistemic model, $\Pi = \wp(\mathsf{dom}\ M)$ be the powerset of M's domain. $|X|_M$ be the set of all formulae equiextensional with $X \in \Pi$. $\Phi = \{|X|_M \mid X \in \Pi\}$ be the set of all co-extensional formulas expressible in M, $||X||_M = \{S_0, S_1, S_2\}$ be the execution-value of a formula with extension X, where $S_0 = \{w \in M \mid M \models_w K_a\neg\varphi\}$, $S_1 = \{w \in M \mid M \models_w K_a\varphi\}$, $S_2 = \{w \in M \mid M \models_w \neg(K_a\neg\varphi \vee K_a\neg\varphi)\}$, $\Gamma = \{||X||_M \mid X \in \Pi\}$ be the set of all formulae which are epistemically equiextensional, $[\varphi]_M$ the epistemical equiextensionality equivalence class of φ in M. Then, tfaeq:

– $p = (s_i, s_{-i})$ is a NE in $G(\Phi)$,

– $p^* = (s_i^*, s_{-i})$ is a NE in $G(\Gamma)$, for any $s_i^* \in [s_i]_M$,

where $G(X)$ is the epistemic game in which player's strategies belong to X.

Based on this result, for both reasons of conceptual simplicity and computational convenience we will adopt this further simplification. This will provide a principled definition for a strategy in an epistemic game with question moves which is conceptually clear and computationally efficient.

For instance, in (WaS) asking q? or p? would both have identical effects. And they are both globally different from the effect of asking the trivial question $\top$?:

```
(v[n1,n2],[[2],[1,3],[]])    --the first element represents q's extension
(v[n,n2],[[1,3],[],[2]])     --the second element is p's execution value

(v[n,n1,n2],[[1,2,3],[],[]]) --Top's exec. value differs from both p's & q's
```

We use this basic observation as a notion of strategy equivalence for questions. When computing the strategy set in the global game we only take the minimal representatives in each equivalence class so determined.

Strategy profiles in the global game are the cartesian product of global strategies and choices in global strategy profiles are uniform across agents' information cells like in games with imperfect information. The global strategy profiles in the induced game of (WaS) example are given by:

```
*QAGames> display 1 (profGlob (named m79))
[[([1,3],v[]),    ([2],v[])],      [([1],v[]),    ([2,3],v[])]]
[[([1,3],v[]),    ([2],v[])],      [([1],v[]),    ([2,3],v[n2])]]
[[([1,3],v[]),    ([2],v[])],      [([1],v[n2]),  ([2,3],v[])]]
[[([1,3],v[]),    ([2],v[])],      [([1],v[n2]),  ([2,3],v[n2])]]
[[([1,3],v[]),    ([2],v[n2])],    [([1],v[]),    ([2,3],v[])]]
[[([1,3],v[]),    ([2],v[n2])],    [([1],v[]),    ([2,3],v[n2])]]
[[([1,3],v[]),    ([2],v[n2])],    [([1],v[n2]),  ([2,3],v[])]]
[[([1,3],v[]),    ([2],v[n2])],    [([1],v[n2]),  ([2,3],v[n2])]]
[[([1,3],v[n2]),  ([2],v[])],      [([1],v[]),    ([2,3],v[])]]
[[([1,3],v[n2]),  ([2],v[])],      [([1],v[]),    ([2,3],v[n2])]]
[[([1,3],v[n2]),  ([2],v[])],      [([1],v[n2]),  ([2,3],v[])]]
[[([1,3],v[n2]),  ([2],v[])],      [([1],v[n2]),  ([2,3],v[n2])]]
[[([1,3],v[n2]),  ([2],v[n2])],    [([1],v[]),    ([2,3],v[])]]
[[([1,3],v[n2]),  ([2],v[n2])],    [([1],v[]),    ([2,3],v[n2])]]
[[([1,3],v[n2]),  ([2],v[n2])],    [([1],v[n2]),  ([2,3],v[])]]
[[([1,3],v[n2]),  ([2],v[n2])],    [([1],v[n2]),  ([2,3],v[n2])]]
```

A brief inspection of the set of global strategy profiles confirms the fact that there are indeed only two strategies with distinct execution value in the game model. The strategies correspond to disjunction of nominals that are minimal in their equivalence classes. In our example this are the disjunction of the empty list and a disjunction with only one element `n2`.

Having a clear cut concept of strategy equivalence will serve as the starting point in computing the outcomes and the construction of the game matrix.

5.2.2 Computing the Outcomes

The next lists contain the answers induced by the global profiles at states 1,2,3:

```
*QAGames> display 1 (map (\x->(w_prof2exeVal (named m79) 1 x))
  (profGlob (named m79)))
[[b]-v[],   [a]-v[]]
[[b]-v[],   [a]-v[]]
[[b]-v[],   -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[],   -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[],   [a]-v[]]
[[b]-v[],   [a]-v[]]
[[b]-v[],   -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[],   -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[n2], [a]-v[]]
[[b]-v[n2], [a]-v[]]
[[b]-v[n2], -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[n2], -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[n2], [a]-v[]]
[[b]-v[n2], [a]-v[]]
[[b]-v[n2], -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[n2], -v[[a]v[n2],[a]-v[n2]]]
```

Answers are epistemic formulae about strategies as nominal disjunctions:

```
*QAGames> display 1 (map (\x->(w_prof2exeVal (named m79) 2 x))
  (profGlob (named m79)))
[[b]-v[],                  [a]-v[]]
[[b]-v[],                  [a]-v[n2]]
[[b]-v[],                  [a]-v[]]
[[b]-v[],                  [a]-v[n2]]
[-v[[b]v[n2],[b]-v[n2]],  [a]-v[]]
[-v[[b]v[n2],[b]-v[n2]],  [a]-v[n2]]
[-v[[b]v[n2],[b]-v[n2]],  [a]-v[]]
[-v[[b]v[n2],[b]-v[n2]],  [a]-v[n2]]
[[b]-v[],                  [a]-v[]]
[[b]-v[],                  [a]-v[n2]]
[[b]-v[],                  [a]-v[]]
[[b]-v[],                  [a]-v[n2]]
[-v[[b]v[n2],[b]-v[n2]],  [a]-v[]]
[-v[[b]v[n2],[b]-v[n2]],  [a]-v[n2]]
[-v[[b]v[n2],[b]-v[n2]],  [a]-v[]]
[-v[[b]v[n2],[b]-v[n2]],  [a]-v[n2]]
```

Answers to questions by agent a are formulae describing agent a's epistemic state with regard to the content of the question, in a tripartite execution value.

```
*QAGames> display 1 (map (\x -> (w_prof2exeVal (named m79) 3 x))
  (profGlob (named m79)))
[[b]-v[],                  [a]-v[]]
[[b]-v[],                  -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[],                  [a]-v[]]
[[b]-v[],                  -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[],                  [a]-v[]]
[[b]-v[],                  -v[[a]v[n2],[a]-v[n2]]]
[[b]-v[],                  [a]-v[]]
```

```
[[b]-v[],                -v[[a]v[n2],[a]-v[n2]]]
[-v[[b]v[n2],[b]-v[n2]], [a]-v[]]
[-v[[b]v[n2],[b]-v[n2]], -v[[a]v[n2],[a]-v[n2]]]
[-v[[b]v[n2],[b]-v[n2]], [a]-v[]]
[-v[[b]v[n2],[b]-v[n2]], -v[[a]v[n2],[a]-v[n2]]]
[-v[[b]v[n2],[b]-v[n2]], [a]-v[]]
[-v[[b]v[n2],[b]-v[n2]], -v[[a]v[n2],[a]-v[n2]]]
[-v[[b]v[n2],[b]-v[n2]], [a]-v[]]
[-v[[b]v[n2],[b]-v[n2]], -v[[a]v[n2],[a]-v[n2]]]
```

The next lists contain all the updated models in example WaS at worlds 1,2,3:

```
*QAGames> display 1 (map showS5 (w2updates (named m79) 1))
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[1,3],  [(1,[p,n]),(3,[p,q,n2])],           (a,[[1,3]]),(b,[[1],[3]]),      [1,3]]
[[1,3],  [(1,[p,n]),(3,[p,q,n2])],           (a,[[1,3]]),(b,[[1],[3]]),      [1,3]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[1,3],  [(1,[p,n]),(3,[p,q,n2])],           (a,[[1,3]]),(b,[[1],[3]]),      [1,3]]
[[1,3],  [(1,[p,n]),(3,[p,q,n2])],           (a,[[1,3]]),(b,[[1],[3]]),      [1,3]]
[[1],    [(1,[p,n])],                        (a,[[1]]),(b,[[1]]),            [1]]
[[1],    [(1,[p,n])],                        (a,[[1]]),(b,[[1]]),            [1]]
[[1],    [(1,[p,n])],                        (a,[[1]]),(b,[[1]]),            [1]]
[[1],    [(1,[p,n])],                        (a,[[1]]),(b,[[1]]),            [1]]
[[1],    [(1,[p,n])],                        (a,[[1]]),(b,[[1]]),            [1]]
[[1],    [(1,[p,n])],                        (a,[[1]]),(b,[[1]]),            [1]]
[[1],    [(1,[p,n])],                        (a,[[1]]),(b,[[1]]),            [1]]
[[1],    [(1,[p,n])],                        (a,[[1]]),(b,[[1]]),            [1]]
```

These correspond to all the possible executions of strategy values as joint public announcements of the players' formulae.

These also represent all the game evolution histories and the resulting epistemic structures in which the goal formulae are going to be model checked to determine the payoffs and the overall result of the game.

The difference between the updates at worlds in the domain is determined by the different execution values for the strategies at the worlds.

```
*QAGames> display 1 (map showS5 (w2updates (named m79) 2))
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[2],    [(2,[q,n1])],                       (a,[[2]]),(b,[[2]]),            [2]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[2],    [(2,[q,n1])],                       (a,[[2]]),(b,[[2]]),            [2]]
[[2,3],  [(2,[q,n1]),(3,[p,q,n2])],          (a,[[2],[3]]),(b,[[2,3]]),      [2,3]]
[[2],    [(2,[q,n1])],                       (a,[[2]]),(b,[[2]]),            [2]]
[[2,3],  [(2,[q,n1]),(3,[p,q,n2])],          (a,[[2],[3]]),(b,[[2,3]]),      [2,3]]
[[2],    [(2,[q,n1])],                       (a,[[2]]),(b,[[2]]),[2]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[2],    [(2,[q,n1])],                       (a,[[2]]),(b,[[2]]),            [2]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[2],    [(2,[q,n1])],                       (a,[[2]]),(b,[[2]]),            [2]]
[[2,3],  [(2,[q,n1]),(3,[p,q,n2])],          (a,[[2],[3]]),(b,[[2,3]]),      [2,3]]
```

```
[[2],    [(2,[q,n1])],                      (a,[[2]]),(b,[[2]]),          [2]]
[[2,3],  [(2,[q,n1]),(3,[p,q,n2])],         (a,[[2],[3]]),(b,[[2,3]]),    [2,3]]
[[2],    [(2,[q,n1])],                      (a,[[2]]),(b,[[2]]),          [2]]
```

It is easy to check that the resulting updated models represented here as code output correspond to the resulting models from Example 5.2.1.

```
*QAGames> display 1 (map showS5 (w2updates (named m79) 3))
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[1,3],  [(1,[p,n]),(3,[p,q,n2])],          (a,[[1,3]]),(b,[[1],[3]]),    [1,3]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[1,3],  [(1,[p,n]),(3,[p,q,n2])],          (a,[[1,3]]),(b,[[1],[3]]),    [1,3]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[1,3],  [(1,[p,n]),(3,[p,q,n2])],          (a,[[1,3]]),(b,[[1],[3]]),    [1,3]]
[[1,2,3],[(1,[p,n]),(2,[q,n1]),(3,[p,q,n2])],(a,[[1,3],[2]]),(b,[[1],[2,3]]),[1,2,3]]
[[1,3],  [(1,[p,n]),(3,[p,q,n2])],          (a,[[1,3]]),(b,[[1],[3]]),    [1,3]]
[[2,3],  [(2,[q,n1]),(3,[p,q,n2])],         (a,[[2],[3]]),(b,[[2,3]]),    [2,3]]
[[3],    [(3,[p,q,n2])],                    (a,[[3]]),(b,[[3]]),          [3]]
[[2,3],  [(2,[q,n1]),(3,[p,q,n2])],         (a,[[2],[3]]),(b,[[2,3]]),    [2,3]]
[[3],    [(3,[p,q,n2])],                    (a,[[3]]),(b,[[3]]),          [3]]
[[2,3],  [(2,[q,n1]),(3,[p,q,n2])],         (a,[[2],[3]]),(b,[[2,3]]),    [2,3]]
[[3],    [(3,[p,q,n2])],                    (a,[[3]]),(b,[[3]]),          [3]]
[[2,3],  [(2,[q,n1]),(3,[p,q,n2])],         (a,[[2],[3]]),(b,[[2,3]]),    [2,3]]
[[3],    [(3,[p,q,n2])],                    (a,[[3]]),(b,[[3]]),          [3]]
```

The following are the payoffs obtained by player a in the game of Example WaS:

```
*QAGames> display 1 (zip3 (w2pays (named m79) 1
    (imp (Disj[K b p, K b (Neg p)]) (Disj[K a p, K a (Neg p)]))
  (w2pays (named m79) 2
    (imp (Disj[K b p, K b (Neg p)]) (Disj[K a p, K a (Neg p)]))
  (w2pays (named m79) 3
    (imp (Disj[K b p, K b (Neg p)]) (Disj[K a p, K a (Neg p)]))))
(1,1,1)                (1,1,1)
(1,1,1)                (1,1,1)
(1,1,1)                (1,1,1)
(1,1,1)                (1,1,1)
(1,1,1)                (1,1,1)
(1,1,1)                (1,1,1)
(1,1,1)                (1,1,1)
(1,1,1)                (1,1,1)
```

The following are the payoffs obtained by player b in the game of Example WaS:

```
*QAGames> display 1 (zip3 (w2pays (named m79) 1
     (imp (Disj[K a p, K a (Neg p)]) (Disj[K b p, K b (Neg p)]))
  (w2pays (named m79) 2
     (imp (Disj[K a p, K a (Neg p)]) (Disj[K b p, K b (Neg p)]))
  (w2pays (named m79) 3
     (imp (Disj[K a p, K a (Neg p)]) (Disj[K b p, K b (Neg p)]))))
(1,0,0)                (1,0,0)
(1,1,1)                (1,1,1)
(1,0,0)                (1,0,0)
(1,1,1)                (1,1,1)
```

```
(1,0,0)                (1,0,0)
(1,1,1)                (1,1,1)
(1,0,0)                (1,0,0)
(1,1,1)                (1,1,1)
```

The payoff obtained by the players is constructed by averaging the local result of checking goal formulae over the entire domain. For ease of readability we will only take here the sum without dividing it to the size of the domain.

The next output gives the game matrix for the Q-A game in the WaS example:

```
*QAGames> display 4 (qagn (named m79)
  ((imp (Disj[K b p, K b (Neg p)]) (Disj[K a p, K a (Neg p)])),
  (imp (Disj[K a p, K a (Neg p)]) (Disj[K b p, K b (Neg p)]))))
(3,1)(3,3)(3,1)(3,3)
(3,1)(3,3)(3,1)(3,3)
(3,1)(3,3)(3,1)(3,3)
(3,1)(3,3)(3,1)(3,3)
```

The following outputs give more interesting variations of the WaS example:

```
*QAGames> display 4 (qagn (named m79)
  ((imp (Disj[K b q, K b (Neg q)]) (Disj[K a q, K a (Neg q)])),
  (imp (Disj[K a p, K a (Neg p)]) (Disj[K b p, K b (Neg p)]))))
(1,1)(1,3)(1,1)(1,3)
(1,1)(1,3)(1,1)(1,3)
(3,1)(3,3)(3,1)(3,3)
(3,1)(3,3)(3,1)(3,3)

display 4 (qagn (named m79)
   ((imp (Disj[K b p, K b (Neg p)]) (Disj[K a q, K a (Neg q)])),
   (imp (Disj[K a q, K a (Neg q)]) (Disj[K b p, K b (Neg p)]))))
(2,2)(1,3)(2,2)(1,3)
(2,2)(1,3)(2,2)(1,3)
(3,1)(3,3)(3,1)(3,3)
(3,1)(3,3)(3,1)(3,3)
```

5.2.3 Counting Goals

There are sixteen global strategy profiles in the game from the WaS example:

```
*QAGames> length (profGlob (named m79))
16
```

Each strategy profile can have different execution values, for worlds in the domain, three in our example. Each execution value leads to one update. Hence there are as many possibilities to consider, some of which might be isomorphic:

```
*QAGames> 16 * 3
48
```

The model representing the entire game history can be obtained as the disjoint union of every goal execution as follows:

```
*QAGames> displayS5 hyst
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,
29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,
54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,
79,80,81,82,83,84,85,86,87,88]
[(1,[p]),(2,[q]),(3,[p,q]),(4,[p]),(5,[q]),(6,[p,q]),(7,[p]),(8,[p,q]),
(9,[p]),(10,[p,q]),(11,[p]),(12,[q]),(13,[p,q]),(14,[p]),(15,[q]),(16,[p,q]),
(17,[p]),(18,[p,q]),(19,[p]),(20,[p,q]),(21,[p]),(22,[p]),(23,[p]),(24,[p]),
(25,[p]),(26,[p]),(27,[p]),(28,[p]),(29,[p]),(30,[q]),(31,[p,q]),(32,[q]),
(33,[p]),(34,[q]),(35,[p,q]),(36,[q]),(37,[q]),(38,[p,q]),(39,[q]),(40,[q]),
(41,[p,q]),(42,[q]),(43,[p]),(44,[q]),(45,[p,q]),(46,[q]),(47,[p]),(48,[q]),
(49,[p,q]),(50,[q]),(51,[q]),(52,[p,q]),(53,[q]),(54,[q]),(55,[p,q]),(56,[q]),
(57,[p]),(58,[q]),(59,[p,q]),(60,[p]),(61,[p,q]),(62,[p]),(63,[q]),(64,[p,q]),
(65,[p]),(66,[p,q]),(67,[p]),(68,[q]),(69,[p,q]),(70,[p]),(71,[p,q]),(72,[p]),
(73,[q]),(74,[p,q]),(75,[p]),(76,[p,q]),(77,[q]),(78,[p,q]),(79,[p,q]),
(80,[q]),(81,[p,q]),(82,[p,q]),(83,[q]),(84,[p,q]),(85,[p,q]),(86,[q]),
(87,[p,q]),(88,[p,q])]
(a,[[1,3],[2],[4,6],[5],[7,8],[9,10],[11,13],[12],[14,16],[15],[17,18],[19,20],
[21],[22],[23],[24],[25],[26],[27],[28],[29,31],[30],[32],[33,35],[34],[36],
[37],[38],[39],[40],[41],[42],[43,45],[44],[46],[47,49],[48],[50],[51],[52],
[53],[54],[55],[56],[57,59],[58],[60,61],[62,64],[63],[65,66],[67,69],[68],
[70,71],[72,74],[73],[75,76],[77],[78],[79],[80],[81],[82],[83],[84],
[85],[86],[87],[88]])
(b,[[1],[2,3],[4],[5,6],[7],[8],[9],[10],[11],[12,13],[14],[15,16],[17],[18],
[19],[20],[21],[22],[23],[24],[25],[26],[27],[28],[29],[30,31],[32],[33],
[34,35],[36],[37,38],[39],[40,41],[42],[43],[44,45],[46],[47],[48,49],[50],
[51,52],[53],[54,55],[56],[57],[58,59],[60],[61],[62],[63,64],[65],[66],[67],
[68,69],[70],[71],[72],[73,74],[75],[76],[77,78],[79],[80,81],[82],[83,84],
[85],[86,87],[88]])
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,
29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,
54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,
79,80,81,82,83,84,85,86,87,88]
```

This extended structure captures all the needed information about the game. As far as the truth value of epistemic formulae is concerned many of these worlds are equivalent.

Using the standard epistemic functionality in our code we can minimize this model under bisimulation and name it. This minimization process is desirable and useful especially for very large instances, however, it also has some drawbacks for our current purpose, which is more then just preserving truth of the formulae.

```
*QAGames> displayS5 (named (bisim hyst))
[0,1,2,3,4,5,6,7,8,9]
[(0,[p,n]),(1,[q,n1]),(2,[p,q,n2]),(3,[p,n3]),(4,[p,q,n4]),(5,[p,n5]),
(6,[q,n6]),(7,[q,n7]),(8,[p,q,n8]),(9,[p,q,n9])]
(a,[[0,2],[1],[3,4],[5],[6],[7],[8],[9]])
(b,[[0],[1,2],[3],[4],[5],[6],[7,8],[9]])
[0,1,2,3,4,5,6,7,8,9]
```

```
*QAGames> displayS5 ((bisim hyst))
[0,1,2,3,4,5,6,7,8,9]
[(0,[p]),(1,[q]),(2,[p,q]),(3,[p]),(4,[p,q]),(5,[p]),(6,[q]),
(7,[q]),(8,[p,q]),(9,[p,q])]
(a,[[0,2],[1],[3,4],[5],[6],[7],[8],[9]])
(b,[[0],[1,2],[3],[4],[5],[6],[7,8],[9]])
[0,1,2,3,4,5,6,7,8,9]
```

Next, we show how one can compute the minimal model more efficiently using partitions to capture the tree structure of the game history. Let W be the domain of a model $M = \langle W, R_a, V\rangle$ for $a \in A$. Let L_W be the lattice of partitions of W with $\bigwedge$ refinement and $\bigvee$ coarsening. Let P_a be agent a's information partition. Let $L_W \mid P_a = \downarrow P_a$ be the a-conditional partition lattice of W. Then

$$min(M) = \bigcup_{i\in A}(L_W \mid P_i) \cup \bigvee_{i\in A}\{L_W \mid P_i\}$$

This confirms that there are as many logically non-equivalent goal formulae:

```
*QAGames> length (forms (named (bisim hyst)))
1024
```

This is an efficient and convenient model transformation that preserves the truth value of all modal formulae. However, this also erases relevant information about the strategic aspects of the epistemic game. For instance, there are as many worlds that satisfy the formula: $p \wedge \neg q \wedge \widehat{K}_a\widehat{K}_b K_a\neg p$:

```
*QAGames> sum (map (\x -> (paynumber hyst x f0)) (dom hyst))
12
```

Whereas, after the model is minimized under bisimulation it appears as if the goal is satisfied only once. This will affect the computation of the final payoffs in the game whenever the model is minimized under standard bisimulation.

```
*QAGames> filter (\x -> (isTrueAt (named (bisim hyst)) x
  (Conj [p, Neg q, (k a (k b (K a (Neg p))))])) ) (dom (named (bisim hyst)))
[0]
*QAGames> filter (\x -> (isTrueAt (bisim hyst) x
  (Conj [p, Neg q, (k a (k b (K a (Neg p))))])) ) (dom (bisim hyst))
[0]
```

The same happens for the following goal formulas satisfied twelve times:

```
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [p, q, k a (Neg q), k b (Neg p) ])) ) (dom (hyst))
[3,6,13,16,31,35,45,49,59,64,69,74]
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [q, Neg p, (k b (k a (K b (Neg q))))])) ) (dom (hyst))
[2,5,12,15,30,34,44,48,58,63,68,73]
```

The following formulae are each satisfied eight times in the following worlds:

```
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [p, Neg q, (k a (K b  (Conj [p,q]))]])) ) (dom (hyst))
[7,9,17,19,60,65,70,75]
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [K b  (Conj [p,q]),k a (Neg q)])) ) (dom (hyst))
[8,10,18,20,61,66,71,76]
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [K a (Conj [p,q]),k b (Neg p)])) ) (dom (hyst))
[38,41,52,55,78,81,84,87]
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [q, Neg p, (k b (K a  (Conj [p,q]))]])) ) (dom (hyst))
[37,40,51,54,77,80,83,86]
```

And the following formulae are satisfied each in the listed worlds, respectively:

```
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [(K a (Conj [p,q])),(K b (Conj [p,q]))]))) (dom (hyst))
[79,82,85,88]
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [(K a (Conj [Neg p,q])),(K b (Conj [Neg p,q]))]))) (dom (hyst))
[32,36,39,42,46,50,53,56]
*QAGames> filter (\x -> (isTrueAt hyst x
  (Conj [(K a (Conj [p,Neg q])),(K b (Conj [p,Neg q]))]))) (dom (hyst))
[21,22,23,24,25,26,27,28]
```

To avoid this information loss we can use a hoarding version of bisimulation contraction, to keep track of how many worlds we have in each contracted equivalence class across the model. This can be represented by encoding the size of the bisimilarity equivalence class as a parameter in the minimal model. A refined criterion of goal equivalence should also keep track of this amount. See the notion of probabilistic bisimuation from Chapter 8 for a comparison.

Considering a strictly extensional criterion, and no restriction with regard to the expressive power of the language or the syntactic structure of the formulas expressing goals we have so many logically equivalent goal formulae:

```
*QAGames> length (nubBy (\x y -> (snd y) == (snd x))
  (zip (forms (named (bisim hyst))) (map (extension (named (bisim hyst)))
  (forms (named (bisim hyst))))))
1024
```

As this example already illustrates, even the analysis of the simplest scenarios can lead to structures with a very large number of states. This is even more so for the setting using product update for more complex questioning and resolution actions like the ones we discussed in the previous chapter. This is usually called the ‘state explosion’ problem and to avoid it is desirable to have algorithms for minimizing structures while preserving behavioral equivalence.

5.3 Birelational Coarsest Partition Problem

The main distinctive feature in our approach to questions so far was the use of the intersection between two accessibility relations. This has technical consequences with regard to behavioral invariance for issue-epistemic models that were already explained and discussed in previous chapters. The main idea is that the intersection modality is not invariant under bisimulation.

However, the ability to work with models that capture the same structural behavior with a minimal number of states is very important in practice and also desirable form a theoretical point of view. This is the problem of computing for a given model the minimal bisimilar model.

For epistemic models this problem is solved using the standard notion of bisimulation by a standard partition refinement algorithm. This is a problem also related to minimizing the number of states in a finite automaton, see [74] and [109] for further details.

For issue-epistemic models we cannot just use the standard minimization under bisimulation procedure for the already mentioned reason that this does not adequately capture the behavior of the intersection modality.

In this section we define the Birelational Coarsest Partition Problem in similar way the coarsest partition problem is defined in [74] and we give an algorithm that solves it similar to the algorithm from [109]. The main advantage of having such an algorithm for issue-epistemic structures is that it provides a way to capture invariance of issue-epistemic formulae, including, in special, formulae containing the intersection modality which are going to be also preserved by this algorithm.

5.3.1. Definition. [Bistability] Let W be a finite set and $K \subseteq W \times W$ and $Q \subseteq W \times W$ be two binary relations over W. A set $B \subseteq W$ is bistable with respect to another set $D \subseteq W$ if either $B \subseteq (Q \cap K)(D)$ or $B \cap (Q \cap K)(D) = \emptyset$.

A partition P of W is bistable with respect to a set $D \subseteq W$ if all the blocks belonging to P are bistable as sets with respect to D. P is self-bistable if it is bistable with respect to each of its own blocks as sets.

Two consequences of bistability, defined as in [74] useful during later proofs:

1. Bistability is *inherited* under refinement; that is, if Q is a refinement of P and P is bistable with respect to a set S, then so is Q.
2. Bistability is *inherited* under union; that is, if P is bistable with respect to two sets Q and S, then P is also stable with respect to $Q \cup S$.

The *birelational* coarser partition problem is that of finding for two given relations K and Q and initial partition P over a set W the coarsest stable refinement of W, i.e., the partition such that every other stable partition is a refinement of it, so that it has the fewest blocks. Lemma 5.3.4 will also show that the coarsest stable refinement is unique.

For any partition Q and subset $S \subseteq W$, let $split(S_R, Q)$ be the refinement of Q obtained by replacing each block $B \in Q$ such that $B \cap R(S) \neq \emptyset$ and $B \setminus R(S) \neq \emptyset$

by the two blocks $B' = B \cap R(S)$ and $B'' = B \setminus R(S)$. The set S is a *splitter* of Q if $split(S_R, Q) \neq Q$.

5.3.2. Definition. [Double Split] For any partition Q and subset $S \subseteq W$, let $split(S_2, Q)$ be the refinement of Q obtained by replacing each block $B \in Q$ such that $B \cap (Q \cap K)(S) \neq \emptyset$ and $B \setminus (Q \cap K)(S) \neq \emptyset$ by the two blocks $B' = B \cap (Q \cap K)(S)$ and $B'' = B \setminus (Q \cap K)(S)$.

The set S is a *double splitter* of Q if $split(S_2, Q) \neq Q$.

Note that Q is unstable with respect to S if and only if S is a splitter of Q. The same for bistable and double split. Two properties of the *split* function, defined as in [74], that are going to be used in later proofs are:

1. Function *split* is *monotone* in the second argument; that is, if $U \subseteq W$ and P is a refinement of Q then $split(U, P)$ is a refinement of $split(U, Q)$.
2. Function *split* is *commutative*. The coarsest partition of P bistable with respect to both S and Q is $split(S, split(Q, P)) = split(Q, split(S, P))$.

Algorithm 1 Compute Coarsest Birelational Stable Refinement

Precondition: Q is a partition of W, R_1, R_2 are binary relations on W
Postcondition: Q is the coarsest bistable refinement stable for R_1 and R_2

```
repeat
    Find a set $S$ that is a union of $Q$-blocks and is a double-splitter of $Q$
    Replace $Q$ by $split(S, Q)$
    Find a set $S$ that is a union of $Q$-blocks and is a $R_1$-splitter of $Q$
    Replace $Q$ by $split(S, Q)$
    Find a set $S$ that is a union of $Q$-blocks and is a $R_2$-splitter of $Q$
    Replace $Q$ by $split(S, Q)$
until $Q$ is self-bistable and stable with regard to $R_1$ and $R_2$
```

5.3.3. Lemma (Invariance). *Algorithm 1 maintains the invariant that every coarsest stable refinement of the initial partition P is also a refinement of the current partition Q.*

The proof is very similar to the proof in [74], it proceeds by induction on the number of refinement steps using the four properties discussed so far. The new contribution is the use of the previously described split function for two relations.

5.3.1. Proof (Lemma 5.3.3). By induction on the number of refinement steps. The base case is established by definition. Suppose we have proved that the invariant in maintained before a refinement step of a partition Q by a splitter set S. If R is an arbitrary coarsest stable refinement of P. Because S is a union of Q-blocks and R is a refinement of Q by IH, S is a union of blocks of R. Therefore, R is bistable with respect to S. Because the *split* function is monotone, $R = split(S, R)$ is a refinement of $split(S, R)$. □

5.3.4. Lemma (Termination). *The refinement process in Algorithm 1 is correct and terminates after at most $n - 1$ refinement steps, having computed the unique coarsest bistable partition.*

The proof is very similar to the proof in [74], it is based on the fact that the number of blocks in Q is at most $|W| = n$.

5.3.2. Proof (Lemma 5.3.4). Because the number of partition cells in Q for a finite domain is at least one and at most $|W| = n$ and because each refinement step either increases this number or has already reached the least fixed point, the algorithm terminates after $n - 1$ refinement steps. After the step in which no further refinement steps are possible, Q is bistable and by Lemma 5.3.3 any bistable refinement is a refinement of Q. Hence Q is the unique coarsest bistable refinement. □

We will use Algorithm 1 as the core component inside a model minimization process that preserves modal formulae in the questioning language, including formulae containing intersection modalities. This will be very similar with the iterative refinement process from Algorithm 1 but for which some extra processing has to be performed in order to make it adequate for the propositional and local structure of issue-epistemic models.

There are two main aspects that need additional careful consideration: the first one concerns the starting state of the algorithm, the second one concerns the final steps and the resulting model.

At the beginning of the algorithm, before the refinement process starts there is some preprocessing needed i.e. we have to make sure that the initial partition is not an arbitrary one but one that respects atomic harmony, and is therefore a partition in propositionally equivalent blocks.

After the refinement process reaches a fixed point we still need to apply some postprocessing steps in order to build a new issue epistemic model from the existing structure of the partition blocks and the relational structure in the initial issue-epistemic model.

All we have to do before we can prove correctness and termination for our algorithm is to spell out all the details in the `build` function, that is describe the way in which the final model is constructed.

Intuitively, we take the new domain to be the set of partition blocks, the new issue relation is build from the initial one as follows:

$$[x] \approx [y] \quad \text{iff} \quad \forall z \in [x] \, \exists v \in [y] : z \approx v$$

The new epistemic relation is constructed from the initial one in an analogous manner. The valuation of each minimal representative of a partition block is assigned to the block containing it. And the issue of actuality is solved via block membership projection.

Algorithm 2 Compute The Minimal Questioning Model

Precondition: M is an arbitrary Issue-epistemic Model

Postcondition: M' is the minimal PIM intersimilar to M

```
$P \leftarrow \{C_i \mid C_i \subseteq \mathtt{dom}(M), \forall\, v, w \in C_i : V(w) = V(v)\}$
repeat
    Find a set $S$ that is a K-splitter of $P$ and a union of $P$ blocks
    $P \leftarrow \mathtt{split}(S, P)$
    Find a set $S$ that is a Q-splitter of $P$ and a union of $P$ blocks
    $P \leftarrow \mathtt{split}(S, P)$
    Find a set $S$ that is a 2-splitter of $P$ and a union of $P$ blocks
    $P \leftarrow \mathtt{split}(S, P)$
until $P$ is self-bistable and stable with regard to Q and K
$M' \leftarrow \mathtt{build}(M, P)$
```

We can now show that modal formula in the questioning language are preserved during the process of model minimization:

5.3.5. Lemma (Preservation). *Algorithm 2 ensures that questioning modal formulae true in the initial model are also true in the minimal model, including, in special, formulae using intersection modalities.*

The proof proceeds by induction on the structure of a modal formula using the previous definitions for the split and double-split functions used in the refinement steps and the construction of the new model via the `build` function.

5.3.3. Proof (Lemma 5.3.5). Let M be an arbitrary probabilistic issue epistemic model. Let M^- be constructed from M via Algorithm 3.

We will proceed by constructing a double splitting set from any formula that changes its truth value from M to M^-.

Let φ be a diffraction formula, i.e. let $M \models_w \varphi$ and $M^- \not\models_{[w]} \varphi$.

We continue by induction on the structure of the formula φ.

If φ is an atomic propositional formula then we have from the definition of the `build` function that $\varphi \in V^-([w])$ and we are done.

If φ is a negation or a conjunction we apply IH to the constituent formula(e).

If $\varphi := \langle \approx \cap \sim \rangle \psi$ then, from $M \models_w \langle \approx \cap \sim \rangle \psi$ we obtain by modal semantics that $\exists x \in W : w \approx \cap \sim x$ and $M \models_x \psi$. Using the induction hypothesis we get that $M^- \models_{[x]} \psi$.

On the other hand, from $M^- \not\models_{[w]} \varphi$ we obtain that $\forall\, [x] \in W^- : [w] \approx \cap \sim [x]$ implies $M^- \not\models_{[x]} \psi$. Therefore, it must be the case that $([w], [x]) \notin \approx \cap \sim$. From the definition of the `build` function we have that $\exists v \in [w]$ and $\exists y \in [x]$ such that $(v, y) \notin \approx \cap \sim$.

Assume wlog that $v \not\approx y$ then we can easily check that if we take the set $S = [w]$ we have that $\mathtt{split}(S, P) \neq P$. Hence S is a double splitter set of P, and P could not have been obtained as the fixed point of the refinement process because is not self-bistable.

For the remaining cases of formulae using the epistemic and questioning modalities Q and K the argument proceeds completely analogously using $\approx$ and $\sim$ and constructing a splitter set. □

5.3.6. Lemma (Termination). *The refinement process in Algorithm 2 is correct for behavioral equivalence and terminates after at most* $|W| + 1$ *refinement steps, having computed the unique minimal model intersimilar to the initial model.*

The proof is based on the fact that we always start from an issue-epistemic structure with a finite domain. Therefore he number of blocks in the initial propositional equivalence partition Q is at most $|W| = n$.

5.3.4. Proof (Lemma 5.3.6). Because the starting propositional equivalence partition in a finite issue-epistemic structure has at most as many cells as the size of the domain of the model, the number of partition cells in P is at least one and at most $|W| = n$. Each time the **repeat** loop is executed the number of blocks in the partition is increased, therefore in at most $n + 1$ steps the **repeat** cycle will bistabilize the partition and by Lemma 5.3.3 no further refinement is possible, so no smaller model can exist. By Lemma 5.3.5 the model built is intersimilar with the initial one. Hence M' is the unique minimal model intersimilar to M and therefore behaviorally equivalent. □

To conclude, we have provided an algorithm that can be used to minimize issue-epistemic models while preserving the truth value of formulas using intersection modalities. This will also be very useful for minimizing probabilistic issue-epistemic models later on in the chapter.

The insights provided by the minimization algorithm can be made more precise and can be generalized in a structural notion describing invariance for the issue-epistemic language. For this we have to come up with an adequate notion of behavioral equivalence for issue-epistemic structures. In order to achieve this we will need a notion of invariance between issue-epistemic models. This notion cannot be standard bisimulation because, as we discussed before, the formulae containing intersection modalities are not preserved under the standard epistemic bisimulation. Hence the first step will be to propose an improved version of behavioral equivalence. We will need a notion that is adequate in the following sense: it will preserve the truth value of all formulae needed to reason about questioning scenarios, in special formulae using the interdependence between the two relations issue and epistemic, but also those using only one.

We will call this notion *intersection bisimulation* or *intersimulation* for short, to highlight the main feature that it is designed to address.

5.3.7. Definition. [Intersection Bisimulation] An intersection bisimulation between two IEMs M and M' is a relation $Z \subseteq W \times W'$ defined as in Definition 6.5.4 but in which the first two clauses describing the probabilistic aspects are ignored. So an intersimulation has the following properties:

> Atomic harmony: if sZs' then $V(s) = V'(s')$
>
> Forward relations: for both relations: sZs' and sRt for some $t \in W$ implies that there is some $t' \in W'$ with $s'R't'$ and tZt'
>
> Forward intersection: if sZs' and both sRt and sSt hold for some $t \in W$ then there is some $t' \in W'$ with both $s'R't'$ and $s'S't'$ and tZt'
>
> Backward relations: symmetrical with the forward clause for relations
>
> Backward intersection: symmetrical with the forward clause for intersection

Given any two pointed issue-epistemic models $\langle M, s\rangle$ and $\langle M', s'\rangle$ we use the following shortcut notation $\langle M, s\rangle \underline{\leftrightarrow} \langle M', s'\rangle$ to say that the models are intersimilar, that is there exist a relation Z such that sZs' and Z is an intersimulation.

We will now argue that the notion of intersimulation is the adequate invariance notion for a language with intersection modalities:

5.3.8. Theorem (Invariance). *For any two pointed PIMs M and M' and for any formula φ in a questioning language with intersection modalities we have: if $M, w \underline{\leftrightarrow} M', w'$ then $M \models_w \varphi$ iff $M' \models_{w'} \varphi$.*

The proof proceeds by induction on the structure of the formula φ.

5.3.5. Proof (Theorem 5.3.8). Atomic and Boolean cases follow straightforwardly from the atomic clause and IH. For the modal formulae: In case $\varphi := \widehat{R}\psi$, suppose xZy and $M \models_x \varphi$. From xZx' we get using Definition 6.5.4 that $\mathtt{zig}(QKW_x, Q'K'W'_y)$ which is a shortcut notation for the following condition: $\mathtt{zig}(\{v \in W \mid xQvKx\}, \{v' \in W' \mid yQ'v'K'y\})$. This is equivalent with: $\forall v \in \{v \in W \mid xQvKx\} : \exists v' \in \{v' \in W' \mid yQ'v'K'y\} : vZv'$, which is further translated into $\forall v \in \{v \in W \mid (x, v) \in Q \;\&\; (v, x) \in K^{-1}\} : \exists v' \in \{v' \in W' \mid (y, v') \in Q' \;\&\; (v', y) \in K'^{-1}\} : vZv'$. By IH we obtain that $M \models_v \psi$ iff $M' \models_{v'} \psi$. We also have $\forall v \in \{v \in W \mid (x, v) \in Q \cap K\} : \exists v' \in \{v' \in W' \mid (y, v') \in Q' \cap K'\} : vZv'$. Hence we can conclude, as desired, that: $M' \models_{v'} \varphi$. The other direction is similar. The cases for the remaining modalities are completely analogous using the corresponding relation and the matching clause in Definition 6.5.4. $\square$

We will show in Section 6.5 that this notion can also stand as the first building block for a notion of probabilistic intersection bisimulation.

Chapter 6

Querying Strategies and Probabilities

6.0.1 Introduction and Motivations

There are three interconnected motivations behind the content that will be developed in the current chapter. The first one continues the main themes from the previous chapters and concerns the development of a general and efficient theory of designing good questioning strategies in problem solving.

There are two complementary but also important motivations that stem from the nature of the concrete example considered. The first one is game theoretical and consists in finding solution concepts for a particular example of a strategic interaction. The second one has an algorithmic nature and aims at finding a method for computing such equilibria in an efficient way.

As the first motivation is the one that connects directly with the approach in previous and subsequent chapters it will be thoroughly pursued in this chapter. In addition, the remaining two motivations will also be addressed in various degrees and will receive solutions for special classes of problems.

We start by introducing the concrete theoretical context in which oracles of local properties will be defined and strategies for querying such oracles will be applied. This is a general framework of rational interaction and we will consider a concrete game scenario as the working example. Then we will proceed to defining query oracles and strategies used to resolve problems in this framework.

6.1 Querying Strategies in Solving Games

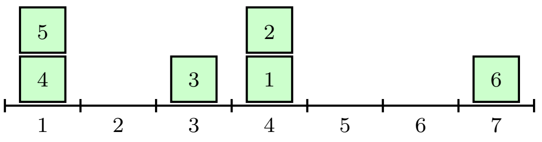
Intelligent interaction between rational agents is a ubiquitous phenomenon in our highly, and increasingly, connected contemporary world. Finding solutions in scenarios of rational interaction becomes even more relevant when agents are competing for, and depleting, scarce or limited resources.

In this general context, the location game is a paradigmatic example of a social interaction in an environment with limited resources for which finding efficient solutions is in the same time theoretically interesting and also potentially practically important. The location game has a very simple formal representation that turns out to be relevant for a wide range of practical applications such as over-fishing the oceans, overloading communication networks, deciding over an optimal positioning on open markets or inside a spectrum of political preferences, and the list can be further extended.

6.1.1 The Location Game and its Applications

In this section we start by considering an intuitive example of the location game played on a line (LG_l) and subsequently introduce the formal definition behind it. Using this illustrative example, we will also show how the formal definition behind LG_l is relevant for modeling practical applications in a large variety of concrete scenarios of rational interaction in social contexts.

An Intuitive Example. Consider a coastline divided in seven distinct fishing regions. Several fishing companies can chose one of these regions as their fishing location. The choice determines their access to resources which are evenly distributed along the seven regions, in this example the resources are fish population. Each location is assigned a unit of payoff and each player gains access to the payoff units in the locations that are closest to his choice. If several players have chosen the same location and if a location is equally distant from several players the payoff is evenly split between those players.



We can represent this in the above figure in which we have $N = \{1, 2, 3, 4, 5, 6\}$ the set of players, with $p = |N| = 6$, and each player can chose a location from the set $S_i = \{1, 2, 3, 4, 5, 6, 7\}$ of available locations, with $l = |S_i| = 7$, for $i \in N$. As an alternative representation, we can think of the locations as apartment buildings on a densely populated street of a crowded city, and of the players as companies deciding where to open a new shop. The model assumes that each inhabitant will become the customer of the shop situated in the closest location to his living place. For instance, inhabitants living in the locations 6 and 7 will become the customers of the closest shop, which in this example is 6. Therefore player 6 will get a payoff of $p_6 = 1 + 1 = 2$ consisting of the sum of unit payoffs in locations 6 and 7. Alternatively we can think of the locations in the game as representing a spectrum of political preferences for voters in an election and the candidates will have to chose a platform that will attract the most voters, assuming that the voters will cast their ballot for the candidate representing the closest position to their preferences. In this case, for instance, players 1 and 2, which have chosen the same position 4 will attract the most sympathetic voters in locations 4 and 5 of the preference spectrum and will split evenly the payoff. In this example, the two candidates will have each the following payoff $p_1 = p_2 = \frac{1}{2} + \frac{1}{2} = 1.00$. As a final illustration, we can think of the locations as hot-spots in a wireless network providing Internet access to all users in their range. In this case the users that are equally closer to the same network hub will have to share its limited resources. In our example, the bandwidth of the hub from position 2 will be evenly allocated to the closest users, which are players 3, 4 and 5, giving them a total payoff of $p_3 = 1 + \frac{1}{3} = 1.33$, and $p_4 = p_5 = \frac{1}{2} + \frac{1}{3} = 0.83$.

To sum up, all this luxuriant variety of concrete practical applications can be described in a unitary way by a very simple formal model. We give the formal definition for the location game on a line below:

6.1.1. Definition. [Location Game] $\mathrm{LG_l}$ is defined in the following way:

- $N = \{1, \ldots, n\}$,
- $S_i = \{1, \ldots, m\}$, for $i \in N$,
- for $i \in N$, $p_i(s_i, s_{-i}) =$

$$\begin{cases} m \;/\; n & \text{if } \forall j \neq i\; s_i = s_j, \\ \dfrac{p_n + \texttt{div}_2(p_r - p_n - 1)}{n_s} + \dfrac{\texttt{mod}_2(p_r - p_n - 1)}{n_s + n_r} & \text{if } \forall j \neq i\; s_i \leq s_j, \\ \dfrac{\texttt{div}_2(p_n - p_l - 1) + m - p_n + 1}{n_s} + \dfrac{\texttt{mod}_2(p_n - p_l - 1)}{n_s + n_l} & \text{if } \forall j \neq i\; s_i \geq s_j, \\ \frac{1 + \texttt{div}_2(p_n - p_l - 1) + \texttt{div}_2(p_r - p_n - 1)}{n_s} + \frac{\texttt{mod}_2(p_r - p_n - 1)}{n_s + n_r} + \frac{\texttt{mod}_2(p_n - p_l - 1)}{n_s + n_l} & \text{otherwise.} \end{cases}$$

where $p_n = s_i, p_r$ and p_l are the strategies that are right-, respectively, left-proximal (as locations) to s_i, n_r and n_l are the numbers of players that have chosen a p_r, respectively, a p_l strategy, n_s is the number of players in the same location as i, and m is the number of positions (locations) in the game.

6.1.2 Solving LG and Computing Solutions Efficiently

The General Problem. The list of possible practical applications for this very simple model is potentially endless, however, the main concern behind all these examples should be a more general problem. This general problem is finding a way to solve such interactive situations. This means to define solution concepts for such games and design algorithms that find such solutions. For instance, in our example, player 5 can increase her payoff by moving to location 5, and other players can increase their payoff by deviating from their initial choices. Then a general question is 'can we find a configuration of choices in which all players are satisfied with the result?'. This solution concept is a Nash equilibrium of the game [73], the formal definition is also included in Section 6.6.

Our main goal will be to use a general theory of queries to an oracle of local properties for strategy profiles to find an algorithm that computes efficiently a solution for the location game on a line formulated as a general problem:

Consider the location game for l locations and p players. Compute all NE_p.

We will not consider NE with mixed strategy, this is known to always exist in strategic games, and computing it is a hard problem [21]. We will only consider pure strategy NEa, these do not always exist in strategic games.

Motivating the Research. As we mentioned already, the main motivation during this chapter consists of establishing a useful link between questioning actions theories, their implementations as software tools [70, 71] and potential further applications to general problem solving.

This was already discussed in a more general framework in previous chapters. In the current context we will rather be concern to a realistic application in which the general abstract theory can be put to work and will produce useful results.

The chapter is also motivated by, and addresses in various degrees, both game-theoretical and algorithmic aspects. We will discuss now these aspects in a little more detail before continuing with pursuing the main goal.

Given a general standard definition for game solutions, the next step is to design an algorithm that finds them. Several standard algorithms for finding Nash Equilibria exist in the literature, like, for instance, iterated elimination of strictly dominated strategies (IESDS) [73, 4]. A general game theoretic motivation is to find such solutions for all game instances, or, if this cannot be dome for all cases, to show that standard algorithms solve some particular classes of cases. Interesting facts are known about solutions of particular instances of LG_l using standard algorithms, like, for instance, the fact that the IESDS procedure solves LG_l for $n = 2k + 1$ locations and 2 players in k rounds [4].

Another general motivation has an algorithmic nature, it aims at solving games in an efficient way. For the general case, computing NE_p is a search problem if the game matrix is given explicitly as an input. In a game with $N = \{1, 2, \dots, n\}$ players the search space is the number of strategy profiles $p = |S_1| \times |S_2| \times \cdots \times |S_n|$ where S_i is the strategy set of player i, for $i \in N$. In order to determine if any given strategy profile is a NE_p a number of $t = (|S_1| - 1) \times (|S_2| - 1) \times \cdots \times (|S_n| - 1)$ tests have to be performed. In the worst-case scenario $p \times t$ tests are needed to compute pure NE in a matrix. This makes the general algorithm exponential in the number of players.

Several proposal of efficient algorithms for solving games exist in the literature: some use heuristics to reduce the search space [82], others use special game properties to limit computations to only few relevant interactions [116].

Overview. The content is structured in the following way: we start in Section 6.2 with an analysis of of the computational complexity of NL$_p$ in LG$_l$ using an oracle model for backtrack searches. We first show that the local properties of the game are essential for an algorithm that computes NL$_p$ in LG$_l$ efficiently. We classify the local properties in negative ones, used to select relevant profiles and construct query strategies, and positive ones, used to define a backtrack oracle, and to characterize NL$_p$ in LG$_l$. Then we show that local properties alone are not sufficient for for an algorithm that computes NL$_p$ in LG$_l$ efficiently.

Next we give a method that computes NL$_p$ in LG$_l$ efficiently using cycles of profile fragments. We continue in Section 6.3 by giving a characterization of NL$_p$ in LG$_l$ by meas of local properties in the game. We conclude, in Section 6.3.3, by a brief presentation of some implementation tools used in the process and by indicating some open problems for future research. The final Sections 6.6-7.2 contain further technical definitions and proofs for the main results.

6.2 Computing NE$_p$ in LG$_l$ by QS$_b$

In this section we will use structural and local properties in LG$_l$ to design an efficient algorithm for computing NE$_p$. A first observation about LG$_l$ is that we have various levels of symmetry in the game. For example, we can use the fact that all the players have the same strategies to simplify the general structure of the search space for NE$_p$. In general, for a LG$_l$ with n players and m locations, the number of tests needed to compute NE$_p$ is:

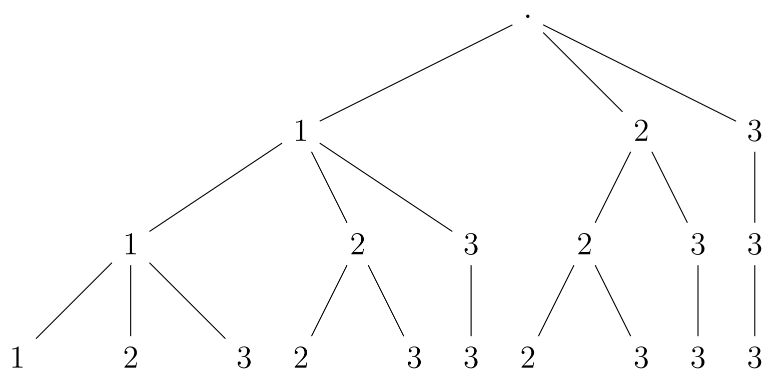
$$f(n,m) = m^n \cdot (m-1) \cdot n \in \Omega(m^n)$$

This will generate a search space that is also exponential in the input, the number of players and locations in the game. However, the fact that payoffs are also computed symmetrically, suggest this can be further improved.

6.2.1 Solving LG by Querying Local Properties

Another important observations is that payoffs are computed symmetrically for all players. This allows us to further reduce the search space by using this symmetry and consider only the so called *canonical* games. These are only the strategy profiles in which locations chosen in increasing order by players are weakly increasing. If some NE$_p$ are found among canonical strategy profiles then all other NE$_p$ in the game are isomorphic with one of these up to permutations of players.

Counting Canonical Games. The figure below illustrates the way the canonical games are constructed in a game with 3 locations and 3 players:



The depth of the tree corresponds to the number of players while the branching factor represents the available choices for each player. For instance, if the first player has chosen 2, location 1 is no longer available as a choice for the players having higher indexes, here the second and the third players.

Local Properties are Necessary. Let $\mathtt{cp}(n, m)$ denote the number of canonical profiles for a LG_l with n players and m locations. We can see from the example in the figure above that $\mathtt{cp}(3, 3) = 3 + 2 + 1 + 2 + 1 + 1 = 10$. We can generalize some basic observations about how this counting is performed:

– for $n = 1$ player and arbitrary m locations, $\mathtt{cp}(n, m) = m$,

– for $m = 1$ location and arbitrary n players, $\mathtt{cp}(n, m) = 1$,

– for arbitrary n players and m locations we will have:

$$\mathtt{cp}(n, m) = \mathtt{cp}(n-1, m) + \mathtt{cp}(n-1, m-1) + \cdots + \mathtt{cp}(n-1, 2) + \mathtt{cp}(n-1, 1)$$

Putting all these observations together, we see that $\mathtt{cp}(n, m)$ can be computed for arbitrary m and n by the following recursive function:

$$\mathtt{cp}(n, m) = \begin{cases} m & \text{if } n = 1, \\ 1 & \text{if } m = 1, \\ \mathtt{cp}(n-1, m) + \mathtt{cp}(n, m-1) & \text{otherwise.} \end{cases}$$

Next we give a Haskell function computing the number of canonical profiles:

```haskell
cp :: Integer -> Integer -> Integer
cp l 1 = 1
cp 1 p = 1
cp l p = cp (l-1) p + cp l (p-1)
```

For example, in a game with $n = 10$ players and $m = 11$ locations the number of strategy profiles in which the chosen locations are weakly increasing is:

```
*CANG> cp 10 11
184756
```

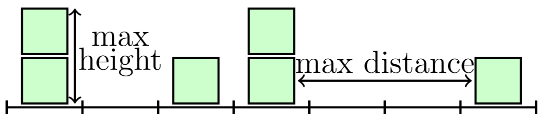
This shows that the number of canonical profiles will also generate a search space exponential in the size of the input, here the number m of locations:[1]

$$\mathtt{cp}(m, n) \in \Omega(1.6^m)$$

and that the symmetry of the game is not enough, and further local properties are essentially needed for an algorithm that computes NE_p in LG_l efficiently.

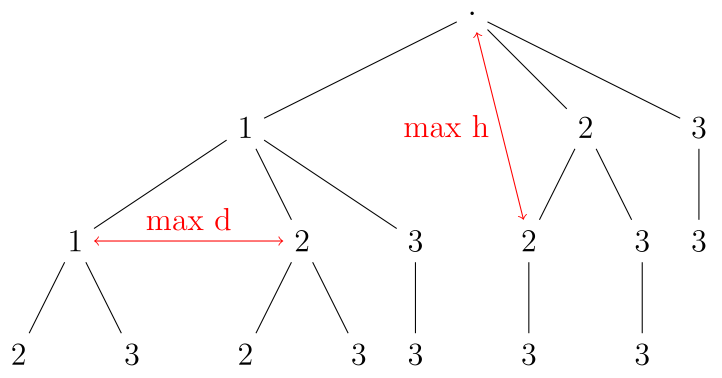
Querying Strategies by Local Properties. We will try to further restrict the search space for NE_p in LG_L by safely ignoring strategy profiles that despite being canonical cannot satisfy further conditions required by to be a NE_p. The background model during this section assumes the existence of an oracle of local properties for NE_p. Intuitively, such an oracle can answer the question about NE_p using local properties of the game, the formal definition is included in Section 6.6. For instance, let h denote the number of players occupying a location and assume that we can show that if $h > 2$ then that strategy profile is not a NE_p. Then many canonical strategy profiles can be safely excluded from the search space. In a similar way, if we can show that in any NE_p we have $d < 2$, where d is the maximal distance between any two players we can further restrict the search space by safely ignoring some location choices for certain players. Using such local properties, we can construct a query strategy to select only relevant profiles for further investigation. Intuitively, a query strategy (QS) is an economic way of selecting only relevant profiles, the formal definition is included in Section 6.6.

Negative Local Properties. We can be sure that a query strategy is correct because all the profiles that are ignored cannot be NE_p. We are using only negative properties to construct the query strategy. The figure below gives an intuitive illustration for the negative properties described above:



[1]We can check that cp increases faster than the Fibonacci sequence: take as base case $fib(1) = 2 \leq cp(1, 2) = 2$ (here the first argument of cp corresponds to the only argument of fib while the second argument is one unit greater). We get from $cp(l, p) = cp((l-1), p)+cp(l, (p-1))$ and the IH that $fib(n) \leq cp(n, n + 1)$. As $fib(n) \in \Omega(1.6^n)$ we can use it as a rough estimate.

Let us consider our previous example with $p = 3$ players and $l = 3$ locations, in which $d = h = 2$ are the maximal distance and height, respectively. The way in which a query strategy restricts the search space by considering only relevant profiles can be represented intuitively by the following tree:



Let $\texttt{rp}(l, p, d, h)$ denote the number of relevant strategy profiles. We can see that for our example $\texttt{rp}(3, 3, 2, 2) = 2 + 2 + 1 + 1 + 1 + 1 = 8$. We can notice that the maximal distance restricts the branching factor in this tree and the maximal height makes further profiles irrelevant due to previous choices.

In order to be correct for finding NE_p a query strategy has to include further queries for local properties, as indicated in the following definition:

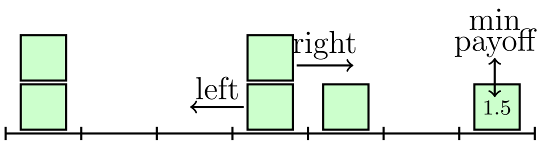
6.2.1. Definition. [Local Query] An **LPQ** of order k is a k-tuple over $\{0, 1, \mathsf{X}\}$:

$$q = \langle \underbrace{\mathsf{X}_1\mathsf{X}_2\cdots\mathsf{X}_n}_{S_0}\underbrace{\mathsf{X}_{n+1}\mathsf{X}_{n+2}\cdots\mathsf{X}_{n+n}}_{S_1}\cdots\underbrace{\mathsf{X}_{mn+1}\mathsf{X}_{mn+2}\cdots\mathsf{X}_{(m+1)n}}_{S_m}\rangle$$

for $k = n(m + 1)$, having the following structure:

- $S_0, S_1, \ldots, S_m$ are locally relevant strategy profiles,
- each q_{in+j}, for $0 \le i \le m$, $0 \le j \le n$, is a S_i local property,
- every $q_{in+j} = 1$ and $q_{in+j} = 0$ represents a Y or a N answer from an oracle of local properties, respectively.

Positive Local Properties. Queries of local properties can include other aspects of the game. For instance, we can inquire weather a relevant profile satisfies a requirement of minimal payoff u, assuming that we can show that in any NE_p profile we have that $u_i \ge 1$ for any player i. Another important positive property for NE_p describes the game dynamics. For instance the proximal move property or unit deviation property will have to be satisfied by any NE_p strategy profile. The following figure illustrates such positive properties:



Local Properties are not Sufficient. Intuitively, the time complexity for a deterministic query strategy is the maximum number of queries required to find all the positive answers from the backtrack oracle, formal definitions and references to results are included in Section 6.6.

Hence it is important to count how many relevant profiles a query strategy will have to consider. We will do this first for the concrete example considered so far and next give a method to extend this counting for the general case. The counting of relevant profiles in the given example proceeds as follows:

$$\begin{aligned}\texttt{rp}(3,3,2,2) &= 2+2+1+1+1+1 = 8\\ \texttt{rp}(3,3,2,2) &= \texttt{rp}(2,2,2,2)+\texttt{rp}(1,2,2,2)+\texttt{rp}(2,1,2,2)+\texttt{rp}(1,1,2,2)\\ \texttt{rp}(3,3,2,2) &= \ \texttt{rp}(1,1,2,2)+\texttt{rp}(1,0,2,2)+\texttt{rp}(0,1,2,2)+\\ &\quad\ \texttt{rp}(1,2,2,2)+\texttt{rp}(2,1,2,2)+\texttt{rp}(1,1,2,2)\end{aligned}$$

We can generalize these basic observations in the following Haskell function:

```haskell
rp :: Integer -> Integer -> Integer -> Integer -> Integer
rp l p h d | l == 1 = 1
           | p == 1 = d
           | l < 1 = 0
           | p < 1 = 0
           |otherwise = (rp (l-1) (p-1) h d)+(rp (l-2) (p-1) h d)
              +(rp (l-1) (p-2) h d)+(rp (l-2) (p-2) h d)
```

This can be used as a recursive method of counting relevant profiles in LG. However, we can also notice that this is not a safe recursion[2] and therefore the number of needed tests is still exponential in the input of the game:

$$\texttt{rp}(l,p,h,d) \in \Omega((d+1)^p)$$

This negative result[3] motivates the approach developed in the next section.

[2]Safe recursion is defined in [12] as the class of constant, projection, successor, predecessor, and conditional functions closed under predicative recursion on notation and safe composition. A result of implicit complexity shows the equivalence between polynomially computable and safely recursive functions [12]. We can also obtain the lower bound by taking $c = 1$ and $n_0 = 3$.

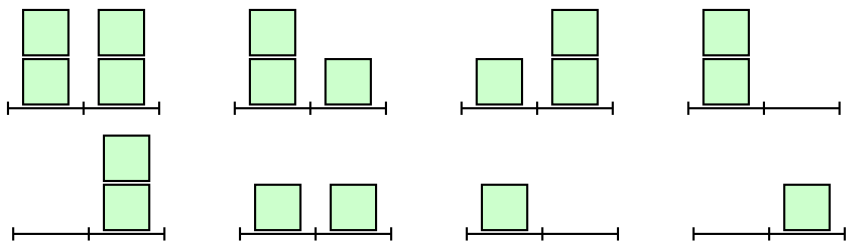
[3]The previous lower bound for computing the number of local properties strategy profiles can be considered a negative result because it shows that in order to find all NE in the game one would need a number of queries of local properties that is exponential in one of the input parameters of the game, namely the number of players.

6.2.2 Efficient Solutions as Cycles of Profile Fragments

In this section we assume a characterization of NE by local properties and we give an efficient algorithm for solving LG_l for arbitrary locations m, and players n, using the maximum height h and distance d. The crucial new feature is the definition of local strategy fragments and of local matching queries used to construct cycles of strategy fragments, requiring a constant amount of queries.

Using Cycles of Local Profile Fragments. Let us assume that the maximum distance in a NE_p profile is d, then any strategy profile will can be decomposed in fragments of length $d+1$. If we can also show that the maximum height in a NE_p profile is h, then we can generate all possible fragments of length $d+1$ in a combinatorial way. We illustrate below the profile fragment construction for the case in which $d = 1$ and $h = 2$ and include the general definition shortly after. An additional aspect that we have to consider is the fact that the border fragments have particular properties and require a special treatment. We will therefore construct three kinds of profile fragments, two for the left and right limits, respectively and one for the middle profile regions.

For $d = 1$, and $h = 2$ let the set of all atomic local profile fragments be $F^h_d := \{22, 21, 12, 20, 11, 02, 10, 01\}$ representing all possible combinations of $h+1$ column heights in fragments of length $d+1$, as illustrated below:



Left and right fragment sets are the cartesian product of atomic fragments:

$$S_0 = S_2 = \{22, 21, 12, 20, 11, 02, 10, 01\} \times \{22, 21, 12, 20, 11, 02, 10, 01\}$$

The middle fragment set is also constructed from the atomic fragment set:

$$S_1 = S_0 \times \{22, 21, 12, 20, 11, 02, 10, 01\}$$

For the general case, for arbitrary large values for the maximum distance and maximum height a profile fragment is defined as follows:

6.2.2. Definition. [Profile Fragment] For arbitrary d, h let F^h_d be defined by:

$$\begin{aligned} F^h_d := &\{(x_0, .., x_d) \mid 0 \le x_i \le h, 0 \le i \le d\} \\ &\cup \{(x_0, .., x_d) \mid 0 < x_i \le h, x_i = x_{i+1}, 0 \le i \le d\} \end{aligned}$$

We have now all the ingredients to define a query strategy using fragments:

6.2.3. Definition. [Local Properties Querying Strategy] An **LPQS** is a finite decision tree T in which nodes are LPQs:

$$q = \langle\underbrace{\mathsf{X}_1\mathsf{X}_2\cdots\mathsf{X}_n}_{S_0^n}\rangle \text{ or } q = \langle\underbrace{\mathsf{X}_1\mathsf{X}_2\cdots\mathsf{X}_n}_{S_1^n}\rangle \text{ or } q = \langle\underbrace{\mathsf{X}_1\mathsf{X}_2\cdots\mathsf{X}_n}_{S_2^n}\rangle$$

or local matching queries (LMQs) with the following structure, $1 \le k \le d$:

- $q = \langle S_0^n[d-k+1]\overset{?}{=}S_1^m[0], S_0^n[d-k+2]\overset{?}{=}S_1^m[1], \ldots, S_0^n[2d+2]\overset{?}{=}S_1^m[k-1]\rangle$
- $q = \langle S_1^m[2d-k+2]\overset{?}{=}S_1^n[0], S_1^m[2d-k+3]\overset{?}{=}S_1^n[1], \ldots, S_1^m[3d+3]\overset{?}{=}S_1^n[k-1]\rangle$
- $q = \langle S_1^n[2d-k+1]\overset{?}{=}S_2^m[0], S_1^n[2d-k+2]\overset{?}{=}S_2^m[1], \ldots, S_1^n[3d+3]\overset{?}{=}S_2^m[k-1]\rangle$

and with the edges linking LPQS nodes constructed in the following way:

- PQ$(S_0^n) \xrightarrow{\mathsf{N}}$ PQ(S_0^{n+1}), PQ$(S_0^n) \xrightarrow{\mathsf{M}}$ MQ(S_0^n, S_1^0), if $n < l_0$;
  PQ$(S_0^n) \xrightarrow{\mathsf{N}}$ noNE$_p$, PQ$(S_0^n) \xrightarrow{\mathsf{M}}$ MQ(S_0^n, S_1^0) for $n = l_0$,
- MQ$(S_0^n, S_1^m) \xrightarrow{\mathsf{N}}$ MQ(S_0^n, S_1^{m+1}), MQ$(S_0^n, S_1^m) \xrightarrow{\mathsf{M}}$ PQ(S_1^m), if $m < l_1$;
  MQ$(S_0^n, S_1^m) \xrightarrow{\mathsf{N}}$ PQ(S_0^{n+1}), MQ$(S_0^n, S_1^m) \xrightarrow{\mathsf{M}}$ PQ(S_1^m), if $m{=}l_1, n{<}l_0$;
  MQ$(S_0^n, S_1^m) \xrightarrow{\mathsf{N}}$ noNE$_p$, MQ$(S_0^n, S_1^m) \xrightarrow{\mathsf{M}}$ PQ(S_1^m), if $m{=}l_1, n{=}l_0$;
- Analogously for PQ(S_1^n), PQ(S_2^n), MQ(S_1^n, S_1^m), MQ(S_1^n, S_2^m)
  with PQ$(S_2^n) \xrightarrow{\mathsf{N}}$ noNE$_p$, PQ$(S_0^n) \xrightarrow{\mathsf{M}} \exists$ NE$_p$ for $n = l_0$.

The figure below illustrates a representative segment of a query strategy with local properties and local matching as a decision tree. This strategy is correct for an NE$_p$ oracle of local properties as each answer from the oracle allows us to prune one outgoing branch at the respective node such that all the queries from the root to the leafs form a cover for the backtrack oracle.

$$\begin{tikzcd}
 & & & & & & & \mathrm{PQ}(S_0^n) \arrow[dllll, "\mathsf{N}"'] \arrow[drrr, "\mathsf{M}"] & & & & & \\
 & & & \mathrm{PQ}(S_0^{n+1}) \arrow[dll, dotted, "\mathsf{N}"'] \arrow[dr, "\mathsf{M}"] & & & & & & & \mathrm{MQ}(S_0^n,S_1^0) \arrow[dll, "\mathsf{N}"'] \arrow[drr, dotted, "\mathsf{M}"] & & \\
 & \mathrm{PQ}(S_0^{l_0}) \arrow[dl, "\mathsf{N}"'] \arrow[dr, "\mathsf{M}"] & & & \mathrm{MQ}(S_0^{n+1},S_1^0) \arrow[d, "\mathsf{N}"'] \arrow[dr, "\mathsf{M}"] & & & & \mathrm{MQ}(S_0^n,S_1^n) \arrow[dl, "\mathsf{N}"'] \arrow[drr, "\mathsf{M}"] & & & & \vdots \arrow[d, dotted, "\mathsf{M}"] \\
\mathrm{noNE} & & \mathrm{MQ}(S_0^{l_0},S_1^0) \arrow[dl, "\mathsf{N}"'] \arrow[dr, "\mathsf{M}"] & & \vdots & \vdots & & \mathrm{MQ}(S_0^n,S_1^{n+1}) \arrow[dl, "\mathsf{N}"'] \arrow[dr, "\mathsf{M}"] & & & \mathrm{PQ}(S_1^n) \arrow[dl, "\mathsf{N}"'] \arrow[dr, "\mathsf{M}"] & & \mathrm{NE} \\
 & \vdots & & \vdots & & & \vdots & & \vdots & \vdots & & \vdots &
\end{tikzcd}$$

Counting Cycles of Local Profile Fragments. Since the previous query strategy is deterministic, its time complexity is the maximum number of queries needed to determine all the instances for which the oracle gives a positive answer. So we have to determine how many such queries there are for arbitrary locations l, players p, maximal height h, and max distance d.

The first relevant counting parameter will be number of possible profile fragments, this number is obtained from the combinatorial content in Definition 6.2.2.

$$|F_d^h| = |P_{d+1}^{h+1}| + h = \frac{(h+1)!}{(d-h)!} + h$$

The second counting parameter is the total number of profile segments containing two units of atomic profile fragments for the right and left boundaries respectively three units for the middle parts:

$$l_0 = |S_0| = |F_d^h \times F_d^h|, \quad l_1 = |S_1| = |F_d^h \times F_d^h \times F_d^h|$$

$$|\text{LPQ}(S_i^x)| = n, \text{ for any } i \in \{0,1,2\}, 0 \leq x \leq |S_i|$$

$$|\text{LMQ}(S_i^x, S_j^z)| = 2d \pm k, \text{ for any } i,j \in \{0,1,2\}, k < d$$

$$|\{\text{LMQ}(S_i^x, y) : y \in S_j\}| = |S_j|, \text{ for any } i,j \in \{0,1,2\}$$

$$\text{any LMQ}(S_1^x, S_1^z) \text{ is repeated at most } k \cdot |S_1|^{|S_1|} \text{ times}$$

Therefore, the execution time (i.e. max required queries) for this LPQS is in

$$O\Bigg((F_d^h)^{l_1^{l_1}} \Bigg)$$

The notable feature here is that neither l nor p is a parameter in this counting. This means that NE_p can be computed in constant time for increasing number of players and locations, as long as the values of d and h are determined.[4]

6.3 Characterizing NE_p in LG_l by BO_c

In this section we give a characterization of NE_p in LG_l using local properties of the game and use this to define a backtrack oracle (BO). The local game properties can describe various game aspects, we mention here a relevant list:

– Maximum column (tower) height (MH)

[4]For all games with $p < l$ the value of h is determined to be at most 2 (cf. Claim 6.3.3) and for $p = l - 1$ the value of d can be also shown to be at most 2 (cf. Claim 6.3.4). In order to be of use for other cases e.g. $p = l - 2$ or $p = l - 3$, etc. this observation should be also paired by a corresponding result analogous to Claim 6.3.4 for $p = l - 2$ or $p = l - 3$, etc.

– Maximal empty distance (MD), Minimal distance (mD),

– Minimal payoff value for a player (MP),

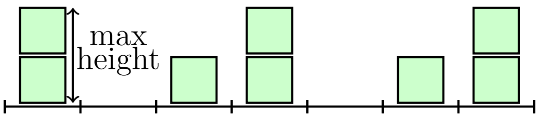
– Incentive for proximal move (unit deviation) (PM).

6.3.1 Local Properties Oracle Characterizing NE$_p$

Both NE on one side, and the local properties we define below are definable by first order logic formulae. So the definition of a backtrack oracle of local properties reduces to checking entailment between the NE definition and the conjunction of formulae for local properties of a given strategy profile. We proceed below by giving both the formal definitions and an intuitive illustration for each of the properties used to characterize NE$_p$ in LG$_l$:

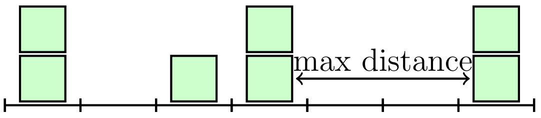
6.3.1. Definition. [Maximum Height, case invariant] A strategy profile $S = (s_i, s_{-i})$ is P2 (MH) iff $\forall i, j, k$:

$$(i \neq j \wedge i \neq k \wedge j \neq k \wedge s_i = s_j) \rightarrow s_k \neq s_j$$

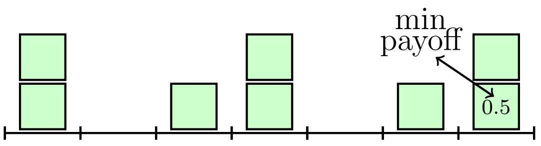


6.3.2. Definition. [Maximum Distance, case dependent] A strategy profile $S = (s_i, s_{-i})$ is P3 iff (MD) $\forall i$:

$$s_i = 1 \vee s_i = 2 \vee \exists j : s_i - s_j = 1 \vee s_i - s_j = 2$$

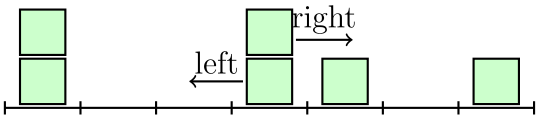


6.3.3. Definition. [Minimum Payoff, case dependent] A strategy profile $S = (s_i, s_{-i})$ is P1 (MP) iff $\quad \forall i : p_i(s_i, s_{-i}) \geq 1$



6.3.4. Definition. [Proximal Move, case invariant] A strategy profile $S = (s_i, s_{-i})$ is P4 (PM) iff $\forall i, s_i$:

$$p_i(s_i - 1, s_{-i}) \leq p_i(s_i, s_{-i}) \geq p_i(s_i + 1, s_{-i})$$



We proceed by a series of entailments between local properties leading to the characterization of NE_p in LG_l.

The corresponding proofs are included in Section 6.7.

6.3.1. Claim (NE $\Rightarrow$ MP). *For $m < n$, S is $NE_p \Rightarrow S$ is P1.*

6.3.2. Claim (NE $\Rightarrow$ MH). *For $m < n$, S is $NE_p \Rightarrow S$ is P2.*

6.3.3. Claim (MP & PM $\Rightarrow$ MH). *For $m < n$, S is P1 and P4 $\Rightarrow S$ is P2.*

6.3.4. Claim (MP & PM $\Rightarrow$ MD). *For $m = n - 1$, S is P1 and P4 $\Rightarrow S$ is P3.*

6.3.5. Proposition (Local NE Characterization). *For $n = m - 1$,*

$$S \text{ is } NE_p \Leftrightarrow S \text{ is } P1 \text{ and } P4$$

The proof of this proposition is completely independent of the implementation we used so far to study LG. The general structure of the proof for this particular combination of local properties also hints at possible ways to extend the result to larger classes of cases. We will discuss some of these in the next section. Another aspect that emerges from the structure of the proof in Section 6.7 is the fact that it generates an very large number of similar cases without giving rise to any essentially new conceptual aspect during the analysis. This makes it a prototypical example of a claim suitable for an analysis using an exhaustive search of a large space of possibilities. This can be performed relatively quick by contemporary computers and is hugely time consuming and tedious for humans.

Therefore it makes sense to use a software tool to automate this process. We will use for this purpose Alloy analyzer [52]. Before presenting some of the results obtained we will briefly explain the theoretical background behind the Alloy architecture. Further implementation details are discussed in Chapter 7.

It is well known that the validity problem is undecidable for first order logic. Alloy uses a relational logic that contains first order logic, therefore the automatic process has to rely on a compromise. The type of compromise behind model checking in general is to check for validity in a limited scope. This is complementary to the compromise behind a theorem prover in which the process is semi-automated and the failure to build a proof might mean either a faulty assertion or a misguided proof strategy by the user.

What Alloy does is to perform an exhaustive search for an assignment to variables that satisfies the initial constraint in a multi-dimensional but limited search space. When checking an assertion Alloy looks for a refutation in a huge space of test cases i.e. possible assignments to variables. This provides a scope complete analysis in the sense that if the analysis finds a counterexample then the assertion is not valid and if the analysis fails to find a counterexample then the assertion is valid within the specified scope. However, it might still be the case that the assertion is not valid in general as it is still possible that it has a counterexample in a larger scope.

Despite this obvious theoretical limitation, what makes this approach useful in practice is the fact that models in a small scope already contain most of the properties that are present in larger models. Testing for such properties in a small scope gives a lot of information about all larger models sharing the property. This has been called in the literature *the small scope hypothesis* and it basically says that: if an assertion is invalid it most probably has a small counterexample.[5]

6.3.6. Corollary (in Alloy, scope 7). *For $n < m$,*

$$S \textit{ is } NE_p \Leftrightarrow S \textit{ is } P2 \textit{ and } P4$$

However, such a result obtained using Alloy Analyzer can only be a partial solution. We will further discuss its relevance in the Section 7.2.2 that presents the Alloy Analyzer implementation in further detail.

For now, we close this section with the following open problem:

6.3.7. Conjecture (NE Characterization). *For $n < m$,*

$$S \textit{ is } NE_p \Leftrightarrow S \textit{ is } P4$$

6.3.2 Generalizing Fragment Cycle Solution to LG_l

In this section we give some illustrations of how the result works for the class of games with $n{=}m{-}1$ and point out how this can be generalized to other game classes, using the same pattern with case dependent local properties.

Note that in Proposition 6.3.5 we have a characterization of NE using local properties that are case dependent. This means that for games in which, for instance, $n = m - 3$ the maximum distance might be different from 1. On the other side, Corollary 6.3.6, gives a characterization by case invariant local properties. This is a more general result as it applies to any number of players and locations and, except for the limited scope, would satisfy our initial game theoretical motivation in the most general way: we have a method to compute equilibria for all location games. However, there is a trade-off here between the game theoretical and the algorithmic motivations behind this analysis. The properties used in Corollary 6.3.6 cannot be used to construct profile fragments

[5]This hypothesis is formulated and defended in [52] in a very general framework. In order to avoid the ad hoc character that this hypothesis might have for LG and to make the hypothesis convincing in the concrete context of the game analyzed one has also to establish what a small enough model would be that already contains all the relevant local properties.

and therefore can not generate an efficient computation. We have to find a right balance between the generality of the solution and the possibility to compute and implement it efficiently.

We will use Haskell and Alloy output to illustrate NE strategy profiles, more details about the implementation tools used are included in Section 7.2. As both NE and the Local Properties can be expressed as FOL formulae, Alloy can be used to test FOL entailment (up to a finite scope). But the construction of local fragments depends on the maximum profile distance which can be different from case to case. In Haskell a QSLPs can be implemented as a lists comprehension. The list of relevant strategy profiles can be further filtered using the other local properties characterizing NE profiles. For example, when we input 5 locations and 4 players we get the following output:

```
*LocGame> nelg 5 4
[[2,2,4,4]]
```

This output is a list of choices that represents abstractly the strategy profile intuitively represented in Figure 6.1 below. We can notice that it is decompos-

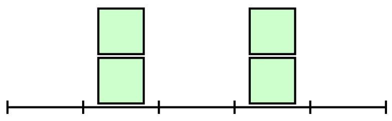


Figure 6.1: A P1-4 strategy profile in LG with 5 locations and 4 players

able by the following pattern $P_1{=}S_0S_1^1S_2$ representing the profile fragment cycle highlighted in Figure 6.2.

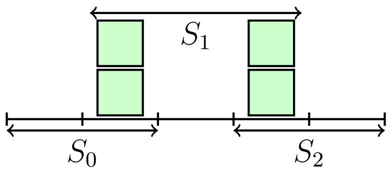


Figure 6.2: Cycles of profile fragments in LG with 5 locations and 4 players

Such observations generalize to larger number of locations, as Haskell output:

```
*LocGame> nelg 7 6
[[2,2,3,5,6,6],[2,2,4,4,6,6]]
*LocGame> nelg 11 10
[[2,2,3,5,5,7,7,9,10,10],[2,2,4,4,6,6,8,8,10,10]]
```

And can be lifted into more general results solving large classes of cases:

6.3.8. PROPOSITION. *For $l = p + 1$ and p even the pattern of profile fragment cycles $P_n = S_0S_1^*S_2$ is one of the (possible many) Nash equilibria, for * denoting matching sequential composition,* $S_0 = \langle 0202\rangle, S_1 = \langle 020202\rangle, S_2 = \langle 2020\rangle$.

Similar observations, with more complex patterns, generalize in the complement class of odd p, $l = p - 1$ cases, witness the examples:

```
*LocGame> nelg 6 5
[[2,2,3,5,5],[2,2,4,5,5]]
*LocGame> nelg 8 7
[[2,2,3,5,5,7,7],[2,2,4,4,6,7,7]]
```

And can again be lifted into general results solving large classes of cases:

6.3.9. Proposition. *For $l = p + 1$ and p odd the pattern of profile fragment cycles $P_n = S_0 S_1^1 (S_1^2)^* S_2$ is one of the (possible many) Nash equilibria, where the concatenation and * represent matching sequential composition, and for $S_0 = \langle 0210 \rangle, S_1^1 = \langle 10202020 \rangle, S_1^2 = \langle 20202020 \rangle, S_2 = \langle 2020 \rangle$.*

The method can be further generalized to other classes of cases to obtain analogous results. The caveat here is that the maximal distance d is a local property that can change from case to case. Even so, our result can be straightforwardly applied to any class of games for which a maximal distance has been determined. This will only require a analogous proof of Proposition 6.3.5 using a corresponding property P3 as in Definition 6.3.2 for increasing d values.

6.3.3 Concluding Remarks and Further Topics

What we did in this chapter was to show how a questioning theory can have interesting applications in problem solving in general and we took the location game as a paradigmatic example. We established a link between a setting with propositional questions and oracles of first order properties. In this setting we showed how questioning strategies can be used to reduce a goal question or the main problem to operational "smaller" questions and how this can lead in particular cases to efficient algorithms that solve the initial question.

A basic theory for efficient strategies of questioning oracles was the main motivation of the chapter. However, we also had game theoretical and algorithmic motivations to guide the concrete application considered. These also received partial solutions and in the same time opened the way for further research.

We end with problems and topics for future research: Does a general, case-independent NE-characterization exist? Can the location game be analyzed by a potential argument? For the location game on the line this is not possible, as there are examples in which no NE exists, but it might be possible for other configurations. Can the method be used for other LGs: on a circle, in a network, etc.? Can the analysis be extended: mixed strategies, probabilistic query-strategies?

6.4 Probabilistic Extensions

6.4.1 General Probabilistic DELs

Approaches adding probabilistic aspect to dynamic epistemic logic have themselves a lively and interesting recent history. The main theoretical framework from which all these start is a probabilistic static epistemic logic. This framework has been established in classical papers such as [28].

The main task of a dynamic logic with probabilities is to add rules governing the dynamics of how such static models change, especially rules governing the evolution of the probabilistic components. For this purpose several proposal have been considered in the literature. Before we proceed with the main topics of this chapter we will make a brief presentation of previous approaches that add probabilities to dynamic epistemic logic.

A first version is the framework from [63], we give below a schematic presentation of its most important concepts.

The probabilistic component in a probabilistic epistemic model is defined as a partial function, some worlds may not be in the domain of the function representing probabilities. For the worlds that are in the domain of the function the usual probabilistic properties have to be the case.

$$P : (A \times W) \to (W \rightharpoonup [0,1]), \text{ such that } \forall\, a \in A \,\forall\, w \in W :$$
$$\sum_{v \in dom(P(a,w))} P(a,w)(v) = 1$$

The use of a partial function is to capture the effect of dynamic actions. Worlds in which an announced formula is not true is excluded from the domain of the function.

The new probabilistic component of the model obtained after the informative action is computed from the old probability component in the following way:

$$P_\varphi(a,u)(v) = \begin{cases} P_\varphi(a,u)(v) & \text{if } P_\varphi(a,u)(\varphi) = 0 \\[2ex] \dfrac{P_\varphi(a,u)(v)}{P_\varphi(a,u)(\varphi)} & \text{otherwise, given that } v \in dom(P_\varphi(a,u)) \end{cases}$$

The computation proceeds by case distinctions taking into consideration exceptions for both zero probability value and the fact that the function does not range over the entire domain.

The approach also offers a definition for the notion of epistemic probabilistic bisimulation and a way to compute minimization of epistemic models under epistemic probabilistic bisimulation.

A further elaborated version is the framework from [94], we give here a brief account of the main conceptual and formal innovations.

The basic static models are again standard epistemic models to which an extra probabilistic component is attached. This time the probability function is a total function, assigning a value, which can be zero, to each world in the domain:

$$P : A \to (S \to (S \to [0,1]))$$

Besides this modification in the static structures, the dynamic actions introduced in this framework are also more general. Now the update is not made by announcing a formula but there are complex action models that have a probabilistic structure of their own. This probabilistic structure is twofold.

1. First there is a probability over a set of preconditions. The function pre assigns to each precondition $\varphi \in \Phi$ a probability distribution over the set of events E, where Φ is set of pairwise inconsistent precondition sentences.

2. Second, there is a probability over events. For each agent i, P_i assigns to each event e a probability distribution over the set of events E.

Such structures model the dynamics of both a subjective observation probability and an objective occurrence probability. This allows for a description of the dynamics of probability, considering three complementary sources: the prior probability of the states in the initial model, the conditional occurrence probability for events given certain properties expressible by formulae in the language, and finally, the subjective probabilistic uncertainty about observed events.

The usual components in the updated model are computed in the standard way. We only mention here the new probabilistic component of the model resulting from an update mechanism that handles both accessibility relations and probabilities, which is computed in the following way:

$$P'_i((s,e)(s',e')) := \begin{cases} 0 & \text{if } \sum_{s''\in S}^{e''\in E} P_i(s)(s'') \cdot \mathsf{pre}(s'',e'') \cdot P_i(e)(e'') = 0 \\ \dfrac{P_i(s)(s') \cdot \mathsf{pre}(s',e') \cdot P_i(e)(e')}{\sum_{s''\in S}^{e''\in E} P_i(s)(s'') \cdot \mathsf{pre}(s'',e'') \cdot P_i(e)(e'')} & \text{otherwise} \end{cases}$$

This still has to use a case distinction to handle the case of zero probability, but the mechanism can handle much more interesting combinations.

Despite conceptual and formal variability of the particular frameworks, both approaches follow the same general DEL methodology of dinamifying a standard static probabilistic epistemic logic by means of adding reduction axioms to the ones describing the static structures. And they both use a probabilistic product update rule that can increase the size of the model after dynamic actions.

6.4.2 Questioning-Related Probabilistic DELs

More recently, probabilistic dynamic logics using with modalities for questioning actions have been also used to model the agreement theorem from [7]. We give here a brief account of the approach from [23] which uses a setting with binary experiments for this purpose.

Again the probabilistic component of a static epistemic structure is a standard probability distribution over the domain of possible worlds.

$$\mu_i(w) : W \to [0,1]$$

In addition, two supplementary conditions are imposed on such probability distributions. These are justified by intuitive properties and have formal advantages.

$$(i) \quad \mu_i(w)(w) > 0 \text{ for all } w \in W$$

$$(ii) \quad \mu_i(w)(v) = 0 \text{ for } (w, v) \notin R_i$$

The first requirement ensures that no division by zero is going to occur in the computation of a new probability, and captures the intuitive principle of truthfulness at the actual world via a probabilistic aspect.

The second requirement corresponds to the fact that it is unintuitive to assign a positive probability to a world which is not considered possible.

The value of the probabilistic component after a binary experiment can be now described without making a case distinction for zero probabilities:

$$\mu_i^e(w)(v) := \frac{\mu_i(w)(\{v\} \cap E_i[w])}{\mu_i(w)(E_i[w])} \quad \text{or} \quad \mu_i^e(w)(x) := \mu_i(w)(x \mid E_i[w])$$

Both experiments and informative epistemic actions such as public announcements can be modeled in this way by restricting the class of models to ones with reasonable factual and epistemic properties:

$$\mu_i^{\varphi}(w)(v) := \frac{\mu_i(w)(\{v\} \cap [\![\varphi]\!]^{\mathbb{M}})}{\mu_i(w)([\![\varphi]\!]^{\mathbb{M}})} \quad \text{or} \quad \mu_i^{\varphi}(w)(x) := \mu_i(w)(x \mid [\![\varphi]\!]^{\mathbb{M}})$$

Another tradition that devotes attention to questions in a probabilistic context is the one originated in [11]. We will give here a brief account of two more recent approaches from [113] and [41] that use this general framework.

The approach to questions in this setting follows the directions of the classical approach devoted to assertions or informative sentences. The first step is to associate a measure of informativity to a formula in the following way:

$$inf(s)(\varphi) = \log_2 \frac{1}{P(s)(\varphi)} = -\log_2 P(s)(\varphi)$$

where the probability of φ in a probabilistic model is given by:

$$P(\varphi) = \sum_{w \in [s] \text{s.t.} M, w \models \varphi} P(s)(w)$$

The corresponding notion of quantitative relevance or informativity for a question is that of entropy. The entropy of a question $Q = (\gamma_1, \ldots, \gamma_k)$, in a pointed probabilistic model such that $M, s \models \mathsf{pre}(Q)$ is defined as follows:

$$E(s)(Q) = \sum_{1 \leq i \leq k} P(s)(\gamma_i) \times inf(s)(\gamma_i)$$

The informational value of an assertion φ in a pointed probabilistic model such that $M, s \models \mathsf{pre}(Q)$ with respect to a question $Q = (\gamma_1, \ldots, \gamma_k)$ is also defined in an analogous way using the difference between the entropies before and after:

$$IV_Q(s)(\varphi) = E(s)(Q) - E_\varphi(s)(Q)$$

where the received information about φ changes the previous entropic value by conditionalization in the following way:

$$E_\varphi(s)(Q) = \sum_{1 \le i \le k} P(s)(\gamma_i \mid \varphi) \times \mathit{inf}(s)(\gamma_i \mid \varphi)$$

Next the value or relevance of assertions are measured against the background of a question. This way the relevance of an assertion φ with respect to a question is greater than the relevance of another using the following informativity criteria:

$$IV_Q(s)(\varphi_1) > IV_Q(s)(\varphi_2), \text{ or}$$

$$IV_Q(s)(\varphi_1) = IV_Q(s)(\varphi_2) \text{ and } \mathit{inf}(s)(\varphi_1) < \mathit{inf}(s)(\varphi_2)$$

Intuitively this corresponds to the amount of change in the entropy of the question brought about by the assertions considered.

An analogous notion for the relevance of a question needs the preliminary notion of expected informational value of a question $Q' = (\chi_1, \ldots, \chi_l)$ with respect to another question $Q = (\gamma_1, \ldots, \gamma_k)$, such that $M, s \models \mathsf{pre}(Q)$ and $M, s \models \mathsf{pre}(Q')$. This is defined by taking the average as follows:

$$EIV_Q(s)(Q') = \sum_{1 \le i \le l} P(s)(\chi_i) \times IV_Q(s)(\chi_i)$$

Intuitively this corresponds to the average amount of change in the value of the main questions' entropy over all possible answers brought about by the question under consideration.

6.5 Minimizing under Probabilistic Bisimulation

Probabilistic issue-epistemic models (PIMs) can be used to describe the interdependence between questioning actions and epistemic and doxastic effects in multi-agent systems. Such approaches have a wide range of application in reasoning about information dynamics in multi-agent systems and in artificial intelligence. Approaches that study both questioning and epistemic dynamics have been considered recently [8, 105, 100, 93, 23, 41]. There exist various logics that describe both static validities in epistemic structures enriched with probabilities [28] and dynamic reduction axioms for probabilistic update [63, 94]. Adding questions to this background allows to study the dynamic interdependence between knowledge and questions using the intersection of two relations, but with the technical complications already discussed namely that it is not invariant under bisimulation. An important prolegomenon to all approaches that will study questioning aspects in a probabilistic setting should therefore be to develop a framework for dealing with this aspect. This will be useful for minimizing models in which the product update rule generates redundant states which are behaviorally equivalent.

In this background, the main motivation of this section is to establish some metatheory and to give an implementation that can handle questioning specific functionality as introduced in [71, 100] in a probabilistic setting. However, the main concern is neither an axiomatization for the static logic nor dynamic reduction axioms but rather metatheory for PIMs, in particular that of defining structural equivalence and computing model minimization for PIMs.

We start by introducing the underlying probabilistic issue-epistemic structures (henceforth PIMs) which will be the changing structures in our investigations:

6.5.1. Definition. [Probabilistic Issue-Epistemic Model] A Probabilistic Issue-Epistemic Model (PIM) is a tuple $\langle W, N, (\overset{i}{\approx}, \overset{i}{\sim}, D_i)_{i \in N}, P, C, V \rangle$ containing:

- W a set of possible worlds or epistemic alternatives,
- $N = \{1, 2, \dots, n\}$ a set of labels representing agents,
- $\overset{i}{\approx} \in W \times W$ a binary issue relation on W, for $i \in N$,
- $\overset{i}{\sim} \in W \times W$ an uncertainty relation on W, for $i \in N$,
- $D_i : W \to (W \to [0, 1])$ is a world-dependent probability distribution over the domain W, for $i \in N$,
- P, C are sets of propositions, respectively, nominals such that $P \cap C = \emptyset$ and $|V(c)| = 1$, for any $c \in C$,
- $V : P \cup C \to \wp(W)$ is a sorted valuation function.

A a set-pointed PIM (M, Q) is a PIM together with a designated set of worlds $Q \subseteq W$, if Q is a singleton set we call the PIM is pointed and denote this (M, w) for $w \in W$.

Intuitively, such structures model both the uncertainties agents have about the world and also their agenda for inquiry, or the issues they would like to have resolved by future answers to their questions. Another important feature in our PIMs is the use of nominals. These are propositional symbols which are true in only one world, thus naming it. In this paper we will assume that $(\overset{i}{\approx}, \overset{i}{\sim})_{i \in N}$ are equivalence relations on the domain W of possibilities.

Finally, the probabilistic information of the agents at a world about the domain of possibilities is also represented by PIMs. We will use the symbol $\mathbf{K}_{\mathsf{pim}}$ to refer to the class of all probabilistic issue-epistemic models (PIMs).

Various languages that describe such structures have been used in the literature. One classic reference for a static language to reason about probabilistic structures is [28]. This has a set of basic propositions describing events closed under Boolean operators. Primitive weight terms an formulae are added to the language to describe the probability of the events by linear inequalities. Validities in such languages are captured by probabilistic axioms and axioms for linear inequalities. Languages that extend the static language with informative dynamic actions have been proposed in [94, 63], these are based on a notion of product update that handles the qualitative components in the way already discussed and adds extra reduction axioms for the probabilistic component as described in this chapter. Approaches that add both probabilities and questions into the mix have also been proposed in the literature [23, 41]. These usually add a second equivalence relation to the static structures and use an intersection modality to describe the epistemic dynamics. As discussed before, this requires a more expressive language, that employs nominals naming worlds, or even higher order devices like binary experiment nominals naming subset pairs.

In this section we will not focus on the dynamics of probabilistic questioning. Such a study is possible inside the DEL methodology with the extensions already discussed and it reveals interesting general properties for questioning phenomena. We reserve such a study for a future occasion, and instead, as a prolegomenon, we focus on a notion of invariance that is adequate for static structures and on giving a minimization algorithm for static structures. The main advantage of having a minimization algorithm that preserves behavioral equivalence is, as discussed before, that of avoiding the 'state explosion' problem which usually arises in practical applications.

In order to describe and reason about PIMs we introduce a basic static probabilistic issue-epistemic logical language:

6.5.2. Definition. [Static Language] The static language of Probabilistic Epistemic Logic of Questions (PEL_Q), denoted by $\mathcal{L}_{\mathsf{EL_Q}}$, is given by the next BNF:

$$\varphi ::= n \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid Q_i\varphi \mid R_i\varphi \mid K_i\varphi \mid D_i\varphi \geq k$$

with $p \in P$ a propositional symbol, $n \in C$ a nominal symbol, $i \in N$ an agent-label, $k \in \mathbb{Q}$ and $P \cap C = \emptyset$.

Various fragments of this language will be referred to in various places below using the following notation: $\mathcal{L}_{\mathsf{EL}}$ will denote the language of epistemic logic, which is the fragment without nominals and the issue Q and intersection R modalities and without the probabilistic $D\varphi \geq k$ inequalities. $\mathcal{L}_{\mathsf{HL}}$ will denote the language of hybrid logic, which is the fragment with no knowledge, issue and intersection modalities and no probabilistic inequalities. Finally, $\mathcal{L}_{\mathsf{PL}}$ will denote the language of probabilistic logic which is the fragment with no knowledge, issue and intersection modalities. We use usual syntactic shortcuts $\widehat{Q} = \neg Q \neg$, etc.

6.5.3. Definition. [Interpretation] The semantics for our language is standard, using the usual Boolean clauses and the expected relational clauses with $\approx$ for Q and $\sim$ for K. The probabilistic inequalities also have the expected meaning:

$$M \models_w D_i\varphi \geq k \quad \text{iff} \quad \sum_{v \in [\![\varphi]\!]_M} D_i(w)(v) \geq k$$

and the intersection modality R is defined using $\approx \cap \sim$ as:

$$M \models_w R_i\varphi \quad \text{iff} \quad \forall v \in W : w\,(\overset{i}{\approx} \cap \overset{i}{\sim})\,v \Rightarrow M \models_v \varphi$$

where $[\![\varphi]\!]_M = \{w \in W \mid M \models_w \varphi\}$ is the extension of φ.

This basic language will serve the purpose of this paper, it can also be extended if needed. Richer versions usually include the universal modality $U\varphi$, group notions for knowledge $D_G\varphi, C_G^\psi\varphi$ or even for issue or intersection, and probabilistic linear inequalities $k_0 D_a\varphi_0 + \cdots + k_n D_a\varphi_n \geq k$.

A note on notation: in a very strict sense $D_i(w)(v)$ is in fact a function with three arguments $D(w, v, i)$ that is represented as $D : W \times W \times A \to \mathbb{R}$ or $D : W \to (W \to (A \to \mathbb{R}))$, however, for ease of notation sometimes the third argument will be given as an index, using $D_i(w)(v)$ interchangeably for $D(w, v, i)$. Also, when some of the arguments are the same, the function can reuse the value of only one input, or receive it also as an extra index, so, for example $X(w, e)$ or $X_e(w, e)$ are used interchangeably for $X(w, e, e)$ or $X(w, e)(e)$ for ease of notation.

Introducing dynamics in this setting follows the main directions of the DEL methodology discussed so far. In the remaining part we will spend some time studying the static language. It is useful to have a notion of bisimulation that extends to also model the probabilistic behavior. We will give a notion of probabilistic issue bisimulation that is very similar to the notion of probabilistic epistemic bisimulation proposed previously in the literature, like, for instance in [63, 94].

Moreover, we will establish a correspondence between this notion of bisimulation and the refinement process in Algorithm 1. This will capture the interaction between issue and epistemic relations in an adequate way and will also add the needed clauses for the probabilistic component.

Discussion The use of nominals was so far motivated, as already discussed, by a need to achieve "expressive harmony" between the static and the dynamic fragments of our language. However convenient and formally elegant the solution of using nominals proved to be so far it also has a drawback that needs to be discussed in this context. This drawback consists in the fact that adding nominals in the valuation of each world makes the notion of bisimulation completely useless as the very basic clause of atomic harmony at propositional level will never be satisfied so no useful partition refinement process can start because the initial partition will already be a partition in which each cell is a singleton.

Algorithm 3 Compute The Minimal Questioning Probabilistic Model

Precondition: M is an arbitrary Probabilistic Issue-epistemic Model

Postcondition: M' is the minimal PIM behaviorally equivalent to M

```
$P \leftarrow \{C_i \mid C_i \subseteq \texttt{dom}(M), \forall\, v, w \in C_i : V(w) = V(v)\}$
repeat
    if $\exists\ S \subseteq \texttt{dom}(M) : S = C_k \cup \cdots \cup C_l$ for some blocks $C_k, \ldots, C_l \in P$
    and $S$ is a K-splitter of $P$ then
        $P \leftarrow \texttt{split}(S, P)$
    else
        skip
    end if
    if $\exists\ S \subseteq \texttt{dom}(M) : S = C_k \cup \cdots \cup C_l$ for some blocks $C_k, \ldots, C_l \in P$
    and $S$ is a Q-splitter of $P$ then
        $P \leftarrow \texttt{split}(S, P)$
    else
        skip
    end if
    if $\exists\ S \subseteq \texttt{dom}(M) : S = C_k \cup \cdots \cup C_l$ for some blocks $C_k, \ldots, C_l \in P$
    and $S$ is a 2-splitter of $P$ then
        $P \leftarrow \texttt{split}(S, P)$
    else
        skip
    end if
until $P$ is self-bistable and stable with regard to Q and K
$M^- \leftarrow \texttt{build}(M, P)$
$D_v(w) \leftarrow \sum_{w \in [w]} D_v(w)$, $\forall\ a \in A$
$D_v([w]) \leftarrow D_v(z)$, for $z = \texttt{min}([w])$, $\forall\ a \in A$
$D_{[v]}([w]) \leftarrow D_z([w])$, for $z = \texttt{min}([v])$, $\forall\ a \in A$
$M' \leftarrow \texttt{buildPro}(M^-, D)$
```

However, in practice, many applications using product update will in fact generate structures that have a more economic model that is bisimilar to the initial one and in which reasoning about questioning actions can be performed more efficiently. For this reason at least the notion of intersection bisimulation is a very useful one, and having a minimization algorithm that captures invariance under this notion is desirable.

6.5.4. Definition. [Probabilistic Bisimulation] A probabilistic bisimulation between two PIMs M and M' is a relation $Z \subseteq W \times W'$ defined as follows: $\forall\, X \subseteq W, \forall\, Y \subseteq W' : \forall\, x \in W, \forall\, y \in W' : xZy \Rightarrow \forall\, a \in A : \forall\, p \in P :$

$$\exists\, X' \subseteq W' : D_a(x, X) \leq D'_a(y, X') \ \&\ \mathtt{zag}(X, X')$$

$$\exists\, Y' \subseteq W : D'_a(y, Y') \leq D_a(x, Y) \ \&\ \mathtt{zig}(Y, Y')$$

$$\mathtt{zig}(QKW_x, Q'K'W'_y), \mathtt{zag}(QKW_x, Q'K'W'_y), x \in V(p) \Leftrightarrow y \in V'(p)$$

$$\mathtt{zag}(QW_x, Q'W'_y), \mathtt{zag}(KW_x, K'W'_y), \mathtt{zig}(QW_x, Q'W'_y), \mathtt{zig}(KW_x, K'W'_y)$$

where we use the following shortcuts: $\mathtt{zig}(X, Y) := \forall\, x \in X : \exists\, y \in Y : xZy$, $\mathtt{zag}(X, Y) := \forall\, y \in Y : \exists\, x \in X : xZy$, $RX_x := \{v \in X \mid xRv\}$, and $RSX_x := \{v \in X \mid xRvSx\}$, for $xRySx := (x, y) \in R \wedge (y, x) \in S^{-1}, xRy := (x, y) \in R$.

We will now prove that this is the adequate invariance notion for our language.

6.5.5. Theorem (Invariance). *For any two pointed PIMs M and M' and for any formula $\varphi \in \mathcal{L}_{\mathsf{ELQ}}$ if $M, w \leftrightarroweq M', w'$ then $M \models_w \varphi$ iff $M' \models_{w'} \varphi$.*

In order to make good use of our notion of behavioral invariance, we also provide in Algorithm 3 a method to perform model minimization.

The core of the algorithm follows the previous two algorithms in this section and adds a final post-processing stage, lines 20-23, to construct the probabilistic component in the minimal model.

The working of the algorithm is also illustrated on a concrete example in the following chapter presenting the implementation for probabilistic DELQ.

The correspondence between our notion of invariance and the setting of the minimization by refinement process in Algorithm 3 is established by the following result showing that the minimization process is adequate:

6.5.6. Theorem (Minimization). *Let M be an arbitrary PIM, and M' the PIM obtained by applying birelational refinement together with probabilistic cell additivity as presented in Algorithm 3.*

Then M' is the minimal PIM that is probabilistically intersimilar to M.

The proof is incremental, building on previous proofs of Lemmas 5.3.5 and 5.3.6. And adding the extra details needed for probabilistic components.

This can be complemented with an implementation for probabilistic aspects. The implementation builds incrementally on the modules from Chapter 3 by enriching them with data structures and functionality specific for a probabilistic setting. We also build on preexisting functionality from [27] and use it in the context of probabilistic issue-epistemic models. We show how the implementation is useful in practice by analyzing several examples and also by illustrating the minimization algorithm for probabilistic models and in the context of dynamic probabilistic questioning and resolution actions.

6.6 Appendix A: Background Definitions

6.6.1. Definition. [Nash Equilibrium] A pure Nash equilibrium is a profile s^* of pure strategies s.t. for any player $i \in N$ and every strategy s_i of i we have:

$$p_i(s^*_{-i}, s^*_i) \geq_i p_i(s^*_{-i}, s_i).$$

6.6.2. Definition. [Query] A query Q of oder n is an n-tuple over the alphabet containing the symbols $\{0, 1, \mathsf{X}\}$. Replacing X with 0 or 1 induces a partial order on queries: for instance, $1\mathsf{XXX} \supseteq 10\mathsf{X}1$; the minimal elements in the order are the atomic queries, which do not contain X.

6.6.3. Definition. [Backtrack Oracle] A backtrack oracle (BO) of order n is a function:

$$A : \{0, 1, \mathsf{X}\}^n \rightarrow \{\mathsf{Y}, \mathsf{N}, \mathsf{M}\}$$

satisfying the following properties:

– no M is given as a response to an atomic query, and

– if $A(Q) = \mathsf{Y}$ or $A(Q) = \mathsf{N}$, and $Q' \subseteq Q$,

then $A(Q') = \mathsf{Y}$ or $A(Q') = \mathsf{N}$, respectively.

6.6.4. Definition. [Oracle Cover] A cover of order k for a BO is a set of queries

$$S = \{Q_0, Q_1, \ldots, Q_k\}$$

such that every atomic query is contained in some of the queries from S to which the BO answers Y or N.

6.6.5. Definition. [Oracle Search] A BO search is a method of finding every BO query Q_i for which $A(Q_i) = \mathsf{Y}$, or conclude that there are none.

6.6.6. Definition. [Search Strategy] An order n search strategy is a finite decision tree T in which nodes are queries of order n with two outgoing branches N and M.

6.6.7. Definition. [Correct Search Strategy] A search strategy T is correct for oracle A if for every leaf z of the answers-pruned tree (the answers from the oracle A allow to prune one outgoing branch from each query node in T):

- if S is the set of queries lying on the path from the root to z,

  then S is a cover for A.

6.6.8. Theorem (Theorem 1 from [16]). Let $\mathcal{C}$ be a collection of oracles of order N which have K-covers. Let T be a probabilistic search strategy which is correct for $\mathcal{C}$: $Etime(T, \mathcal{C}) \geq \frac{1}{4}\left(\frac{N - \log_2 K}{\log_2 N \log_2 K}\right)^{\lfloor \log_2 K \rfloor}$.

For a deterministic search strategy the time complexity is given by the maximum number of queries required.

6.7 Appendix B: Proofs of Main Results

6.7.1. Proof (Claim 6.3.1). Suppose not, then $\exists i : p_i(s_i, s_{-i}) < 1$ for some NE$_\mathrm{p}$ (s_i, s_{-i}). Because $m < n$, $\exists s_i^* : p_i(s_i^*, s_{-i}) \geq 1$, so (s_i, s_{-i}) cannot be NE$_p$. □

6.7.2. Proof (Claim 6.3.2). Suppose not, then $\exists i, j, k : i \neq j \wedge i \neq k \wedge j \neq k \wedge s_i = s_j = s_k$ and $S = (s_i, s_j, s_k, s_{-ijk})$ is NE$_p$. Hence, $p_i(s) + p_j(s) + p_k(s) \geq 3$, using Claim 6.3.1. Hence, wlog $\forall s^* \in S : s^* \neq s_i - 1 \wedge\ s^* \neq s_i - 2$.

Thus, $p_i(s_i - 1, s_{-i}) > p_i(s_i, s_{-i})$ and (s_i, s_{-i}) cannot be NE$_p$. □

6.7.3. Proof (Claim 6.3.3). Suppose not, then $\exists i, j, k : i \neq j \wedge i \neq k \wedge j \neq k \wedge s_i = s_j = s_k$ and $S = (s_i, s_j, s_k, s_{-ijk})$ is P4. We have $p_i(s) + p_j(s) + p_k(s) \geq 3$, because S is P1. Hence, wlog $\forall s^* \in S : s^* \neq s_i - 1 \wedge s^* \neq s_i - 2$. Thus, $p_i(s_i - 1, s_{-i}) > p_i(s_i, s_{-i})$ and (s_i, s_{-i}) cannot be P4. □

6.7.4. Proof (Claim 6.3.4). Suppose not, then, if S is P1 and P4, by Claim 6.3.3, S is P2, so we have to consider the following two cases:
(1) $\exists\ i : s_i$ and $\forall\ k : (s_k \neq s_i \wedge s_k \neq s_i - 1 \wedge s_k \neq s_i - 2) \vee (s_k \neq s_i \wedge s_k \neq s_i + 1 \wedge s_k \neq s_i + 2)$. Assume wlog that $s_k \neq s_i \wedge s_k \neq s_i - 1 \wedge s_k \neq s_i - 2$ then, in the best case, $\dfrac{\sum_{s_k \geq s_i+3} p_k(S)}{|\{k \mid s_k \geq s_i + 3\}|} = \dfrac{n-4}{m-1} = \dfrac{n-4}{n-2} < 1$, so S is not P1.
Otherwise, (2) $\exists\ i, j : s_i = s_j$ and $\forall\ k : (s_k \neq s_i \wedge s_k \neq s_i - 1 \wedge s_k \neq s_i - 2) \vee (s_k \neq s_i \wedge s_k \neq s_i + 1 \wedge s_k \neq s_i + 2)$. Assume wlog that $s_k \neq s_i \wedge s_k \neq s_i - 1 \wedge s_k \neq s_i - 2$ then, because S is P1 and P4, $p(s_i, s_{-i}) \geq 1.33$, hence, $p_i(S) + p_j(S) \geq 2.66$, therefore $\forall\ k : s_k \neq s_i + 1 \wedge s_k \neq s_i + 2$. But then, in the best case, $\dfrac{\sum_{s_k \geq s_i+3} p_k(S)}{|\{k \mid s_k \geq s_i + 3\}|} = \dfrac{n-5}{m-2} = \dfrac{n-5}{n-3} < 1$, so S cannot be P1. □

6.7.5. Proof (Proposition 6.3.5). ($\Rightarrow$) Assume $S = (s_i, s_{-i})$ is NE$_\mathrm{p}$. Then, by Claim 6.3.1, S is P1. Also, $\forall\ i : p_i(s_i, s_{-i}) \geq p_i(s_i^*, s_{-i})$, hence, in particular, $\forall\ i : p_i(s_i - 1, s_{-i}) \leq p_i(s_i, s_{-i}) \geq p_i(s_i + 1, s_{-i})$. ($\Leftarrow$) Assume $S = (s_i, s_{-i})$ is P1 and P4. Suppose $S = (s_i, s_{-i})$ is not NE. Then $\exists\ i, l : l = s_i^* \rightarrow p_i(s_i, s_{-i}) < p_i(s_i^*, s_{-i})$. If $s_i^* = s_i - 1$, $s_i^* = s_i$ or $s_i^* = s_i + 1$ we are done. Otherwise, we have to consider the following (sub)cases:

(1) $|h(l)| = 0$, i.e. location l is empty, $h(l) = \{i \mid s_i = l\}$,

(2) $|h(l)| = 1$, i.e. there is one player in l,

(3) $|h(l)| = 2$, i.e. there are two players in l.

Cases $|h(l)| > 2$ are, by Claim 6.3.3, impossible.

In each of the previous cases we have to consider two possibilities:

(.1) $|h(s_i)| = 1$, and

(.2) $|h(s_i)| = 2$, by Claim 6.3.3 $|h(s_1)| > 2$ is also impossible.

For $m = n - 1$ we further have to consider four cases:

(..1) $d^- = 0$, and $d^+ = 0$, where $d^- = l - \mathtt{max}(\{l' \mid s_{-i} < l\} \cup \{0\})$

(..2) $d^- = 0$, and $d^+ = 1$, where $d^+ = \mathtt{min}(\{l' \mid s_{-i} > l\} \cup \{n\}) - l$

(..3) $d^- = 1$, and $d^+ = 0$,

(..4) $d^- = 1$, and $d^+ = 1$, by Claim 6.3.4, $d^\pm > 1$ are impossible.

We will consider the most representative cases and indicate when the remaining ones behave in an analogous way.

Take case (1.1.1), by P1 we have $p_i(s_i, s_{-i}) \geq 1$. But then $p_i(s_i, s_{-i}) \geq p_i(s_i^*, s_{-i})$ and this cannot be a payoff increasing unilateral deviation. Cases (1.2.1), (2.1.1), (2.2.1), (3.1.1) and (3.2.1) behave analogously.

Take case (2.1.4) and consider the best case scenario in which $h(l+2) = h(l-2) = 1$ then we have $p_i(s_i^*, s_{-i}) = 0.5 + 0.33 + 0.33 > 1$. This might be a payoff increasing unilateral deviation if, for instance, $p_i(s_i, s_{-i}) = 1$. However, if $1 \leq p_i(s_i, s_{-i}) < 1.16$ then it should be the case that $m + 1 > n$, which is in contradiction with the initial setting of the claim. Take case (1.1.2), as S is P1 and P4, by Claim 6.3.3, S is P2, which is impossible. Cases (1.1.3), (1.1.4), (1.2.2), (1.2.3) and (1.2.4) behave analogously.

Take case (2.1.2) by P1 we have $p_i(s_i, s_{-i}) \geq 1$. But then $p_i(s_i^*, s_{-i}) \leq 1$ and this cannot be a payoff increasing unilateral deviation. Cases (2.1.3), (3.1.1), (3.1.2), (3.1.3), and (3.1.4) behave analogously. Another possible scenario $h(l+2) = 1$ and $h(l-2) = 2$ (or vice versa). Then we have $p_i(s_i^*, s_{-i}) = 0.5 + 0.33 + 0.25 > 1$. This might be a payoff increasing unilateral deviation if, for instance, $p_i(s_i, s_{-i}) = 1$. However, if $1 \leq p_i(s_i, s_{-i}) < 1.05$ and S is P4 then $p_k(s_k, s_{-k}) > 1$, for any player k for which $s_k = l + 2$. If so, then it should be the case that $h(l+3) = 0$ and this implies $m + 1 > n$, which is in contradiction with the initial setting of the claim. The final possibility is $h(l+2) = h(l-2) = 2$, then we have $p_i(s_i^*, s_{-i}) = 0.5 + 0.25 + 0.25 = 1$ which cannot produce a payoff increasing unilateral deviation in a P1 profile.

Take case (2.2.4), again $p_i(s_i^*, s_{-i}) = 0.5 + 0.33 + 0.33 > 1$ and this might be a payoff increasing unilateral deviation even if S is P1. But then $h(s_i + 1) = 0$ or $h(s_i - 1) = 0$ and this implies $m + 1 > n$, which is in contradiction with the initial setting of the claim. The remaining cases are similar. □

6.7.6. Proof (Theorem 6.5.5). The first part of the proof is as in Proof 5.3.5. For the probabilistic formulae: Suppose xZy and $M \models_x \sum_{i=1}^n q_i D_a(\varphi_i) \geq q$. Let $E_i = \{v \mid M \models_v \varphi_i\}$ and $E'_i = \{v' \mid M' \models_{v'} \varphi_i\}$. From xZx' and Definition 6.5.4 we get $\exists X' \in W'$ such that $D_a(x, X) \leq D'_a(y, X')$ & $\mathtt{zag}(X, X')$. By IH and

$\texttt{zag}(X, X')$ we obtain $\forall v' \in X' : M' \models_{v'} \varphi_i$. Therefore $X' \subseteq E'_i$. Hence we have: $D'_a(x')(X') \leq D'_a(x')(E'_i)$. And from this we can conclude, as desired, that:

$$D_a(x)(E_i) \leq D'_a(x')(X') \leq D'_a(x')(E'_i)$$

The other direction is similar. □

6.7.7. Proof (Theorem 6.5.6). First we will prove that M' is intersection bisimilar with the initial model M, this is established by Lemma 5.3.5.

The rest of the proof consists in computing the new probability distributions. The new probability distribution for any agent $a \in A$ is constructed in three stages from the previous probability distributions:

Line 20: After this step the following equality holds for any $z \in [w]$:

$$\begin{aligned} D_v(w) &= \sum_{w \in [w]} D_v(w) && \text{(by definition)} \\ &= \sum_{z \in [w]} D_v(z) && \text{(because } z \in [w]) \\ &= D_v(z) && \text{(by definition)} \end{aligned}$$

Line 21: After this step the following equality holds for any $u \in [v]$:

$$\begin{aligned} D_v([w]) &= D_v(z) && \text{(for } z = \texttt{min}([w])) \\ &= D_u(x) && \text{(for } x \in [w], u \in [v] \text{ s.t. } u \approx x) \\ &= D_u(z) && \text{(because } \texttt{min}([w]) = \texttt{min}([x])) \\ &= D_u([w]) && \text{(because } z = \texttt{min}([w])) \end{aligned}$$

Line 22: After this step the following equality holds for any $u \in [v]$:

$$\begin{aligned} D_{[v]}([w]) &= D_z([w]) && \text{(for } z = \texttt{min}([v])) \\ &= D_u([w]) && \text{(for } u \in [v] \text{ by the step 21)} \\ &= D_{[u]}([w]) && \text{(because } [v] = [u]) \end{aligned}$$

We now have to check that the newly constructed D' is indeed a probability distribution and that the M' construction is well defined. For this we need for all $[v], [w] \in M' : D'_{[v]}([w]) \geq 0$. And indeed we have:

$$\begin{aligned} D'_{[v]}([w]) &= \sum_{w \in [w]} D_v(w) && \text{(by definition)} \\ &\geq 0 && \text{(because } \forall\, w \in W : D_v(w) \geq 0) \end{aligned}$$

We also have to check that for all $[v], [w] \in M' : \sum_{[w]\in W'} D'_{[v]}([w]) = 1$.
And indeed we have:

$$\begin{aligned}
D'_{[v]}(W) &= \sum_{[w]\in W} D'_{[v]}([w]) && \text{(by definition)}\\
&= \sum_{[w]\in W} \sum_{x\in[w]} D_{[v]}(x) && \text{(by definition and step 20)}\\
&= \sum_{[w]\in W} \sum_{x\in[w]} D_{v}(x) && \text{(by definition and step 22)}\\
&= \sum_{x\in W} D_{v}(x) && \text{(by additivity)}\\
&= 1 && \text{(as } D_M \text{ is a probability distribution)}
\end{aligned}$$

Finally, we have to show for any $X, Y \subseteq W'$ such that $X \cap Y = \emptyset$ we have

$$D'_{[v]}(X \cup Y) = D'_{[v]}(X) + D'_{[v]}(Y)$$

And indeed we have:

$$\begin{aligned}
D'_{[v]}(X) + D'_{[v]}(Y) &= \sum_{[x]\in X} D'_{[v]}([x]) + \sum_{[y]\in Y} D'_{[v]}([y]) && \text{(by definition)}\\
&= \sum_{[x]\in X} \sum_{x\in[x]} D_{[v]}(x) + \sum_{[y]\in Y} \sum_{y\in[y]} D_{[v]}(y) && \text{(by step 20)}\\
&= \sum_{[x]\in X} \sum_{x\in[x]} D_{v}(x) + \sum_{[y]\in Y} \sum_{y\in[y]} D_{v}(y) && \text{(by step 22)}\\
&= D_v\Big(\bigcup_{[x]\in X} [x] \cup \bigcup_{[y]\in Y} [y]\Big) && \text{(by additivity of } D_v)\\
&= D'_{[v]}(X \cup Y) && \text{(by Steps 21-3)}
\end{aligned}$$

This completes the proof. □

Chapter 7
Implementing Querying Strategies and Probability

7.1 Implementation for Questioning Actions

We already mentioned two implementations that were used to model NE_p in LG_l: Haskell and Alloy Analyzer. Haskell offers the advantages and versatility of functional programming [25] and Alloy Analyzer [52] was used for characterizing NE_p in LG_l. In this section we present and discuss further details about these implementations. We will show how the Alloy code can be used to define relevant properties of game profiles. And we will illustrate the Haskell implementation for querying local properties of strategy profiles.

A final important aspect in designing good questioning strategies under uncertainty is an account of probabilities. In the final part of this section we present and document the implementation behind probabilistic DELq. One essential element in this extension is an adequate notion of behavioral equivalence and an algorithm for minimizing probabilistic models, defined in the previous chapter.

The connection between questioning theory and implementation tools has proved once more to be a fruitful one.

7.2 Implementation and Illustrative Examples

In this section we will briefly describe the Haskell implementation for local game properties as list comprehension, explain its main functionality and use the code to produce some illustrative examples.

7.2.1 Haskell Implementation

We start by importing basic Haskell functionality for list manipulation, line 3, and some further functionality for generating various combinatorial functions, line 2.

For instance, all possible strategy profiles in the location game can be generated by taking the cartesian product, line 6, of as many lists of all available locations as there are players in the location game.

```
1  module LocGame where
2  import CombinatoricsGeneration
3  import List
4
5  allgames p l = cartProd $ take p $ repeat [1..l]
6  cartProd (set:sets) =
7    let cp = cartProd sets in [x:xs | x <- set, xs <- cp]
8  cartProd [] = [[]]
9
10 cangames p l = [x | x <- allgames p l, y<-[1..p-2],
11   x!!y >= (maximum (take (y) x)), x!!y <= (minimum (drop (y+1) x))]
```

As already explained in the text, the main task is to generate a search space that contains only relevant profiles while still allowing for an efficient way of finding Nash equilibria for the location game.

The next block of code gives the implementation of the main four local properties discussed as list comprehension over the canonical profiles.

```
13 p1games p l = nub [x | x <- cangames p l, minimum (map
14   (\y -> (payf x y (fromIntegral l) (fromIntegral p)))
15   [0..(fromIntegral p)-1]) >= 1 ]
16
17 p2games p l = nub [x | x <- cangames p l, maximum
18   (map (\y -> (hght x y))  [0..l]) <=2 ]
19
20 p3games p l = nub [x | x <- cangames p l, maximum
21   (map (\y -> (lftd x y)) [0..(fromIntegral p)-1]) <=1, maximum
22   (map (\y -> (rghd x y (fromIntegral l)))
23   [0..(fromIntegral p)-1]) <=1 ]
24
25 p4games p l = nub [x | x <- cangames p l,
26   dycg x (fromIntegral l) (fromIntegral p) == True ]
```

The rest of the code computes the payoff following the previously explained definitions.

As an illustration of how this implementation approach works we present a representative example in which local properties are used to restrict the relevant search space for Nash equilibria profiles. The example also illustrates the fact that a Nash equilibrium for pure strategies is not always guaranteed to exist in the location game on a line.

```
*LocGame> cangames 3 5
[[1,1,1],[1,1,2],[1,1,3],[1,1,4],[1,1,5],[1,2,2],[1,2,3],[1,2,4],
 [1,2,5],[1,3,3],[1,3,4],[1,3,5],[1,4,4],[1,4,5],[1,5,5],[2,2,2],[2,2,3],
 [2,2,4],[2,2,5],[2,3,3],[2,3,4],[2,3,5],[2,4,4],[2,4,5],[2,5,5],[3,3,3],
 [3,3,4],[3,3,5],[3,4,4],[3,4,5],[3,5,5],[4,4,4],[4,4,5],[4,5,5],[5,5,5]]
```

```
*LocGame> p1games 3 5
[[1,1,1],[1,1,4],[1,1,5],[1,2,2],[1,2,3],[1,2,4],[1,2,5],[1,3,3],[1,3,4],
 [1,3,5],[1,4,4],[1,4,5],[1,5,5],[2,2,2],[2,2,3],[2,2,4],[2,2,5],[2,3,3],
 [2,3,4],[2,3,5],[2,4,4],[2,4,5],[2,5,5],[3,3,3],[3,3,4],[3,3,5],[3,4,4],
 [3,4,5],[4,4,4],[4,4,5],[5,5,5]]

*LocGame> p2games 3 5
[[1,1,2],[1,1,3],[1,1,4],[1,1,5],[1,2,2],[1,2,3],[1,2,4],[1,2,5],[1,3,3],
 [1,3,4],[1,3,5],[1,4,4],[1,4,5],[1,5,5],[2,2,3],[2,2,4],[2,2,5],[2,3,3],
 [2,3,4],[2,3,5],[2,4,4],[2,4,5],[2,5,5],[3,3,4],[3,3,5],[3,4,4],[3,4,5],
 [3,5,5],[4,4,5],[4,5,5]]

*LocGame> p3games 3 5
[[1,2,4],[1,3,4],[1,3,5],[2,2,4],[2,3,4],[2,3,5],[2,4,4],[2,4,5]]

*LocGame> p4games 3 5
[]
```

We end this section with additional relevant code output. This includes examples of representative profiles satisfying P1 to P4 properties for the following locations and players values $l = p + 2, l = p + 3$, and in general $p < l$:

```
*LocGame> nelg 5 3                        *LocGame> nelg 6 4
[]                                        [[2,2,5,5]]

*LocGame> nelg 7 5
[[2,2,3,5,6],[2,2,3,6,6],[2,2,5,6,6],[2,3,5,6,6]]

*LocGame> nelg 8 5
[[2,2,3,6,6],[2,2,3,6,7],[2,2,4,7,7],[2,2,5,7,7],[2,3,6,7,7],[3,3,6,7,7]]

*LocGame> nelg 10 9                                 *LocGame> nelg 10 2
[[2,2,3,5,5,7,7,9,9],[2,2,4,4,6,6,8,9,9]]           [[5,5],[5,6],[6,6]]

*LocGame> nelg 11 5
[[2,2,4,9,9],[2,2,5,9,9],[2,3,6,9,9],[2,3,6,9,10],[3,3,6,9,9],
[3,3,6,9,10],[3,3,7,10,10],[3,3,8,10,10]]

*LocGame> nelg 11 6
[[2,2,3,6,9,9],[2,2,3,6,9,10],[2,2,5,6,9,9],[2,2,5,6,9,10],[2,2,5,7,9,10],
[2,2,5,7,10,10],[2,3,5,6,9,9],[2,3,5,6,9,10],[2,3,5,7,9,10],[2,3,5,7,10,10],
[2,3,6,6,9,9],[2,3,6,6,9,10],[2,3,6,7,9,10],[2,3,6,7,10,10],[2,3,6,9,10,10],
[3,3,6,6,9,9],[3,3,6,6,9,10],[3,3,6,7,9,10],[3,3,6,7,10,10],[3,3,6,9,10,10]]

*LocGame> nelg 11 10
[[2,2,3,5,5,7,7,9,10,10],[2,2,4,4,6,6,8,8,10,10]]
```

As discussed in the main text, using local properties allows for faster and correct searching strategies using queries to an oracle, but without breaking the threshold of efficiency. An efficient solution also will have to use a characterization of Nash equilibrium by local properties but additionally will have to make essential use of fragments of strategy profiles.

7.2.2 Alloy Analyzer Implementation

We proceed now towards presenting the Alloy Analyzer implementation that provides a characterization of Nash equilibrium by local properties and tests this by logical entailment in a specified scope.

We start by importing predefined Alloy modules for arithmetic operations over integer numbers line 1, used for computing payoff values, and predefined operations over orders, used to compare location signatures, line 2, and player signatures, line 3, in a strategy profile.

Next we introduce the basic signatures needed to represent and reason about location games. The signatures for locations, line 5, respectively for players, line 7, are atomic signatures without component fields representing the fact that they do not have any relevant internal structure.

```
1  open util/integer
2  open util/ordering [Location]
3  open util/ordering [Player]
4
5  sig Location {}
6
7  sig Player {}
8
9  sig Game {
10   Locations : seq Location,
11   Players : seq Player,
12   Choice : Player -> one Location
13 }
```

The game signature, lines 9 to 13 has some internal structure represented by three fields: a sequence of locations, a sequence of players and a function from players to locations encoding choices made in the game, for each player.

The local properties defined and studied in the main text are going to be introduced as facts over the previously described game signatures. The first property, line 15, captures the canonical profiles. Any game is isomorphic up to permutations of players to some canonical game. The constraints over game signatures that are implementing this condition make use of the order relation over players and their choices represented by the `Locations` field. Intuitively this says that the player's choices are ordered weakly increasing.

```
15 fact {  --canonical games
16 all g : Game | all i,j : Player |
17   lt[i,j] => lte [g.Choice[i],g.Choice[j]]
18 }
```

The following blocks of code are dedicated to game constraints implementing the local properties P1 to P4. For the sake of conceptual clarity we split them in two clusters. The first will contain static properties which describe the characteristics of a favorable strategy profile instance, and the second one will capture dynamic properties or legal ways in which a profile can behave in terms of changes in the field representing choice functions.

The first fact, line 20, implements property P1. Intuitively, this requires that in any game signature any player has a payoff greater then or equal with a unit. This relies on a definition of the payoff values in the game as fractional numbers which will be introduced later. Because the payoff can be a fractional number and Alloy only has predefined functionality for integers the payoff comparisons will require utilities which are slightly more elaborate.

```
20 fact { --P1 : Minimum Payoff
21   all g : Game | all x : Player |
22   payoff_gt_3d[payoff_n_tot[g,x],1,payoff_d_tot[g,x],1]
23   or (div[payoff_n_tot[g,x],payoff_d_tot[g,x]] = 1 &&
24       rem[payoff_n_tot[g,x],payoff_d_tot[g,x]] = 0)
25 }
26
27 fact p2 { --P2 : Maximum Height
28   all g : Game | all i,j,k : Player |
29   i != j && i != k && j != k && g.Choice[i] == g.Choice[j] =>
30   g.Choice[k] != g.Choice[j]
31 }
32
33 fact p3 { --P3 : Maximum Distance
34   all g : Game | all x,y : g.Locations.inds -
35   {x : Int | some y : Location | some z : Player |
36   g.Locations.idxOf[y] == x && g.Choice[z] = y} | x != y-1
37 }
```

The second fact, line 27, implements property P2. Intuitively, this requires that never more than two players occupy the same location in a game profile. This is a straightforward translation of the first order formula expressing the tower height property. Only quantification over players and their choices together with equality and inequality are needed in order to express this fact.

The third fact, line 33, implements property P3. Intuitively, this says that the maximum distance between occupied locations in a game is strictly smaller than two. This is not a straightforward translation form the first order formula expressing this property. Although such a translation is possible it is much easier encoded as a property of locations. For this the indexes of locations in the sequence are used to perform simple arithmetic on their values. First the empty locations in a game are selected by means of a set comprehension expression. Next the condition that no difference between any two numbers in this set is one is imposed. Here and in other places later on it is important to have index values as integers in order to perform basic arithmetical operations, the ordering of locations alone would not be enough for this.

```
39  fact p4 { --P4 : Proximal Move
40   all x,y : Game | all z : Player |
41   not udev [x, y, z]
42  }
```

The next fact, line 39, implements property P4. This is the only property describing the dynamics of the game. Intuitively, it says that no player has an incentive for proximal move or unilateral deviation with one unit from his choice in the considered strategy profile. This being a dynamic property, is implemented in Alloy Analyzer in declarative style, by a predicate taking as input the precondition game, the postcondition game and the relevant player. The following three predicates provide an implementation for this as follows:

```
44  pred devleft [g,g' : Game, p : Player] {
45    g.Locations == g'.Locations
46    g.Players == g'.Players
47    g'.Choice[p] == prev[g.Choice[p]]
48    all x : Player | p != x => g'.Choice[x] == g.Choice[x]
49    payoff_gt_3d[payoff_n_tot [g', p], payoff_n_tot [g, p],
50                 payoff_d_tot [g', p], payoff_d_tot [g, p]]
51  }
52
53  pred devright [g,g' : Game, p : Player] {
54    g.Locations == g'.Locations
55    g.Players == g'.Players
56    g'.Choice[p] == next[g.Choice[p]]
57    all x : Player | p != x => g'.Choice[x] == g.Choice[x]
58    payoff_gt_3d[payoff_n_tot [g', p], payoff_n_tot [g, p],
59                 payoff_d_tot [g', p], payoff_d_tot [g, p]]
60  }
61
62  pred udev [g,g' : Game, p : Player] {
63    devleft [g,g',p] or devright [g,g',p]
64  }
```

The predicate at line 44 defines a left unit unilateral deviation with incentive. It specifies the fact that the `Locations` and `Players` fields in the pre and post game signatures remain unchanged and the fact that the choice of the player given as a parameter is changed from its initial location to the previous one in the order, line 47. In the same time, this change is unilateral, that is, all the other choices in the strategy profile remain unchanged, line 48.

The predicate at line 53 defines a right unit unilateral deviation with incentive. It specifies the fact that the `Locations` and `Players` fields in the pre and post game signatures remain unchanged and the fact that the choice of the player given as a parameter is changed from its initial location to its successor in the order, line 56. In the same time, this change is unilateral, that is, all the other choices in the strategy profile remain unchanged, line 57.

An additional condition in both predicates is that the change comes with an incentive, that is the payoff value in the postgame is greater than the payoff value in the pregame, lines 49 and 58.

The last predicate in the block, line 62, puts together the previous ones to define a left or right unit unilateral deviation with incentive. This is afterward directly used to define the P4 property.

```
66 fun payoff_n_tot [g : Game, i : Player] : Int {
67   let th = tower_height[g, where[g, i]],
68   thl = tower_height_left_limit[g, i],
69   thr = tower_height_right_limit[g, i],
70   dl = left_dis_int_half[g,where[g,i]],
71   dr = right_dis_int_half[g,where[g,i]],
72   fl = rem[left_dis_int[g,where[g,i]],2],
73   fr = rem[right_dis_int[g,where[g,i]],2] |
74   let oone =  add[1,add[dl,dr]] |
75         #left_of_occ[g,where[g,i]] != 0 &&
76         #right_of_occ[g,where[g,i]] != 0 =>
77         add[ mul[oone, mul[add[th,thl],add[th,thr]]],
78                 add[mul[fl,mul[th,thr]],mul[fr,mul[th,thl]]]
79         ]
80         else
81         (#left_of_occ[g,where[g,i]] == 0 =>
82         add[ mul[(1+left_dis_int[g,where[g,i]]+dr),add[th,thr]],
83                 mul[fr,th]]
84         else
85         (#right_of_occ[g,where[g,i]] == 0 =>
86         add[ mul[(1+right_dis_int[g,where[g,i]]+dl),add[th,thl]],
87                 mul[fl,th]]
88         else
89         add[1, add[dl,dr]]
90         ))
91 }
```

The definition of both local properties P1 and P4 have made use of the payoff values for specific choices in the game. So we have to introduce the way in which this value is computed. Due to the fact that Alloy Analyzer is designed to handle only integer numeric types and the payoff is a fractional number we will have to represent the payoff value as a pair of integers.

The function at line 66 takes a game signature and a player and returns the integer representing the numerator of the fraction that computes the payoff value for the given player in the given game. The computation proceeds according to the game definition by considering four relevant cases and using the splitting of location payoff units according to the distances on the line and number of stacked players and their proximal neighbors.

The function at line 93 takes a game signature and a player and returns the integer representing the denominator of the fraction that computes the payoff value for the given player in the given game. Again the computation proceeds according to the game definition by considering four relevant cases and using the tower heights of the relevant locations in the game.

```
93  fun payoff_d_tot [g : Game, i : Player] : Int {
94    let th = tower_height[g, where[g, i]],
95    thl = tower_height_left_limit[g,i],
96    thr = tower_height_right_limit[g,i] |
97      #left_of_occ[g,where[g,i]] != 0 && #right_of_occ[g,where[g,i]] != 0
98      => mul[th, mul[ add[th,thl], add[th,thr] ]]
99      else
100     (#left_of_occ[g,where[g,i]] == 0 =>
101     mul[th, add[th,thr] ]
102     else
103     (#right_of_occ[g,where[g,i]] == 0 =>
104     mul[th, add[th,thl] ]
105     else
106     th
107     ))
108 }
```

Computing the fractional payoff value is only one aspect of analyzing the game from a static perspective. Another important aspect is comparison between payoff values, this can analyze the game from a dynamic perspective.

The function at line 110 takes four integers as arguments, representing, in order, the nominator of the first fraction, the nominator of the second fraction, the denominator of the first fraction and the denominator of the second fraction, and returns `true` if the first fraction is greater then the second. The fractions are representing payoff values and the comparison proceeds up to the third decimal value. Note that for a maximum column height of two the first two decimals should already be enough in order to decide the highest value.

The computation uses predefined alloy functionality for integer arithmetic like multiplication, integer division and division reminder. These were imported in the beginning from the `util/integer` module and are used to encode the result of the fractional division as a rational number. This is needed because Alloy does not have a predefined data structure for floating numbers.

For instance, when comparing $\frac{500}{350}$ with $\frac{500}{351}$ the result will be $1.42857 > 1.42450$ and will be obtained by considering the first three decimal positions. This provides enough precision to be meaningful in the context of the location game.

```
110 pred payoff_gt_3d[n1,n2,d1,d2 : Int] {
111   div[n1,d1] > div[n2,d2] ||
112   (div[n1,d1] = div[n2,d2] &&
113    div[mul[10,rem[n1,d1]],d1] > div[mul[10,rem[n2,d2]],d2] ) ||
114   (div[n1,d1] = div[n2,d2] &&
115    div[mul[10,rem[n1,d1]],d1] = div[mul[10,rem[n2,d2]],d2] &&
116    div[mul[10,rem[mul[10,rem[n1,d1]],d1]],d1] >
117    div[mul[10,rem[mul[10,rem[n2,d2]],d2]],d2]) ||
118   (div[n1,d1] = div[n2,d2] &&
119    div[mul[10,rem[n1,d1]],d1] = div[mul[10,rem[n2,d2]],d2] &&
120     div[mul[10,rem[mul[10,rem[n1,d1]],d1]],d1] =
121     div[mul[10,rem[mul[10,rem[n2,d2]],d2]],d2] &&
122     div[mul[10,rem[mul[10,rem[mul[10,rem[n1,d1]],d1]],d1]],d1] >
123     div[mul[10,rem[mul[10,rem[mul[10,rem[n2,d2]],d2]],d2]],d2])
124 }
```

The following code blocks are dedicated to various computation for payoff value. There is nothing remarkable about the functions, they proceed as expected to capture the game definition. We include them here for the sake of completeness but we will only briefly describe them without too much details.

```
126 fun where [g : Game, p : Player] : Location {
127   g.Choice[p]
128 }
129
130 fun tower_height [g : Game, l : Location]: Int {
131   #{x : Player | g.Choice[x] = l}
132 }
```

The function at line 126 takes a player and a game and returns its chosen location. At line 130, the function takes a location and a game signature and returns the number of players that have chosen the location. The function at line 134 takes a player and a game and returns the tower height of its right limit, similarly, line 139, takes a player and returns the tower height of its left limit.

```
134 fun tower_height_right_limit [g : Game, i : Player]: Int {
135   where[g,i] = right_limit[g, where[g,i]] => 0
136   else tower_height[g,right_limit[g,where[g,i]]]
137 }
138
139 fun tower_height_left_limit [g : Game, i : Player]: Int {
140   where[g,i] = left_limit[g, where[g,i]] => 0
141   else tower_height[g,left_limit[g, where[g,i]]]
142 }
```

Next function, at line 144, takes a location and returns half of its left distance as an integer number, similarly, line 148, takes a location and a game and returns half of its right distance as an integer number.

```
144 fun left_dis_int_half [g : Game, l : Location] : Int {
145   div[left_dis_int[g, l],2]
146 }
147
148 fun right_dis_int_half [g : Game, l : Location] : Int {
149   div[left_dis_int[g, l],2]
150 }
```

The entire left distance is computed at line 152, the function takes a location and a game and returns its left distance as a set of locations including limit locations. Analogously, line 158, takes a location and returns its right distance as set of locations including limits.

```
152 fun left_dis [g : Game, l : Location] : Location {
153   no left_of_occ[g, l] => { x : Location | gt[x,left_limit[g, l]]
154     && lt[x,l]}+left_limit[g, l]
155   else { x : Location | gt[x,left_limit[g, l]] && lt[x,l]}
156 }
157
158 fun right_dis [g : Game, l : Location] : Location {
159   no right_of_occ[g, l] => { x : Location | lt[x,right_limit[g, l]]
160     && gt[x,l]}+right_limit[g, l]
161   else { x : Location | lt[x,right_limit[g, l]] && gt[x,l]}
162 }
```

In some contexts it is useful to consider the distance without the ending locations. The function at line 164 takes a location and returns its left distance as integer excluding limit locations. Analogously for the right side, the function at line 168 takes a location and returns its right distance as integer again by excluding the limit locations.

```
164 fun left_dis_int [g : Game, l : Location] : Int {
165   #(left_dis[g, l] - l)
166 }
167
168 fun right_dis_int [g : Game, l : Location] : Int {
169   #(right_dis[g, l] - l)
170 }
```

In the case of a location game on a line there are two locations with a special behavior because they are the line endpoints. The function at line 172 takes a location and returns its left limit or the least location i.e. the left endpoint of the line or the minimal location. Symmetrically, the function at line 177, takes as input a location and a game and returns its right limit or the greatest location, i.e. the right endpoint of the line or the maximal location.

```
172 fun left_limit [g : Game, l : Location] : Location{
173   no left_of_occ[g, l] => min[Location]
174   else max[left_of_occ[g, l]]
175 }
176
177 fun right_limit [g : Game, l : Location] : Location{
178   no right_of_occ[g, l] => max[Location]
179   else min[right_of_occ[g, l]]
180 }
```

Another auxiliary function, line 182, takes a location and returns the occupied locations to its left. Symmetrically, at line 186, the function takes a location and returns the occupied locations to its right. Analogously the functions at lines 190 and 194 take as input the signatures of a location and a game and return the free locations to the left and to the right, respectively.

```
182 fun left_of_occ [g : Game, l : Location] : Location {
183   { x : Location | lt[x,l] && some {y : Player | g.Choice[y] = x} }
184 }
185
186 fun right_of_occ [g : Game, l : Location] : Location {
187   { x : Location | gt[x,l] && some {y : Player | g.Choice[y] = x} }
188 }
189
190 fun left_of_free [g : Game, l : Location] : Location {
191   { x : Location | lt[x,l] && no {y : Player | g.Choice[y] = x} }
192 }
193
194 fun right_of_free [g : Game, l : Location] : Location {
195   { x : Location | lt[x,l] && no {y : Player | g.Choice[y] = x} }
196 }
```

The final predicate, line 198, used to capture the game dynamics records arbitrary unilateral deviation with incentive. This is completely analogous to the predicate characterizing the P4 property.

The difference is that this time the change of choice or the deviation in location is an arbitrary one, it is not restricted to proximal locations but can be any location in the game. This allows the predicate to be used in the characterization of a Nash equilibrium strategy profile.

```
198 pred ne [g,g’ : Game, p : Player] {
199   g.Locations == g’.Locations
200   g.Players == g’.Players
201   g’.Choice[p] ! = g.Choice[p]
202   all x : Player | p != x => g’.Choice[x] == g.Choice[x]
203   payoff_gt_3d[payoff_n_tot [g’, p], payoff_n_tot [g, p],
204                payoff_d_tot [g’, p], payoff_d_tot [g, p]]
205 }
```

Based on this characterization the entailment between the local properties and Nash equilibrium can be tested using a finite scope in Alloy Analyzer.

There are some additional constraints that do not correspond directly to the local properties discuss so far but are needed in order to make the signatures behave in the desired way. We include then here for the sake of completeness.

The first one, line 207, requires that players and locations are always part, i.e. fields, in a game. The second one, line 210, requires that games always have the same component fields. This is useful for comparing games.

```
207 fact{
208   all x : Player | some g : Game | x in g.Players.elems
209   all x : Location | some g : Game | x in g.Locations.elems
210   all x, y : Game | x.Locations==y.Locations && x.Players == y.Players
211 }
212
```

```
213 fact{
214   all g : Game | all x,y : Location | gt[x,y] =>
215     g.Locations.idxOf[x] > g.Locations.idxOf[y]
216   all g : Game | not g.Locations.hasDups
217   all g : Game | all x,y : Player | gt[x,y] =>
218     g.Players.idxOf[x] > g.Players.idxOf[y]
219   all g : Game | not g.Players.hasDups
220 }
221
222 run {} for 5 but 4 Player, 10 int
```

Finally, a requirement that makes many of the calculations easier is that the order over the Player and Location signatures and the sequence indexing are bijective, line 213. In this way sequence indexes can be used to perform basic arithmetic operations on locations in the game.

This useful correspondence is also illustrated in Figure 7.1 which is the Alloy representation of the example in Figure 6.1 already discussed on page 164.

For instance the run at line 222 checks if the predicates implementing local properties, are consistent, that is, it tries to generate an example satisfying all the local properties. More concretely, an instance obtained by running a predicate is an assignment that makes all the facts of the model true. These can be explicit facts, in our implementation the fact paragraphs at lines 15, 20, 27, 33, 39, 207 and 213, but also implicit facts contained in the declarations used for the defined signatures and their component fields, like for instance the requirement at line 12, specifying the fact that the players choose a unique location.

The variables assigned in an instance contain the sets associated with the signatures, the relations associated with the fields and the arguments of the predicates. The output details for running the predicate are included in the following output block. The run specifies a limited scope in which Alloy tries to find such an example. The scope is a muti-dimensional space of test cases in which each dimension is a bound for the variables in the constraint. In this case the scope specifies a bound of five for the signature representing game locations, and the number of players is limited to four so that the game conditions are captured.

A final clause specifies the maximum integer value, in this case ten as needed in the previously explained payoff computing function.

When the analysis finds an instance satisfying the specifications this can be visualized as a graph, as illustrated below in Figures 7.2 and 7.1.

The provided output contains useful information about the structure of the signatures considered and about the syntactic structure of the clause generated and fed to the SAT solver:

```
> Executing "Run run$1 for 5 but 10 int, 4 Player"
  Sig this/Player scope <= 4
  Sig ordering/Ord scope <= 1
  Sig open$3/Ord scope <= 1
  Sig this/Location scope <= 5
```

```
   Sig this/Game scope <= 5
   Sig this/Location forced to have exactly 5 atoms.
   Sig this/Player forced to have exactly 4 atoms.
   Sig this/Location == [[Location$0], [Location$1], [Location$2],
[Location$3], [Location$4]]
   Sig this/Player == [[Player$0], [Player$1], [Player$2], [Player$3]]
   Sig this/Game in [[Game$0], [Game$1], [Game$2], [Game$3], [Game$4]]
   Sig ordering/Ord == [[ordering/Ord$0]]
   Sig open$3/Ord == [[open$3/Ord$0]]
   Solver=sat4j Bitwidth=10 MaxSeq=5 SkolemDepth=1 Symmetry=20
   1407216 vars. 375 primary vars. 6253223 clauses. 1262515ms.
   Instance found. Predicate is consistent. 8212566ms.
```

The output specifies first the variables used to formulate the constraint, these can be explicitly defined signatures like the location and players in the game or predefined signatures for orders and sequences.

For each signature used there is a bound that defines the multi-dimensional space of test cases in which an exhaustive search for assignments of variables satisfying the constraint is going to be performed.

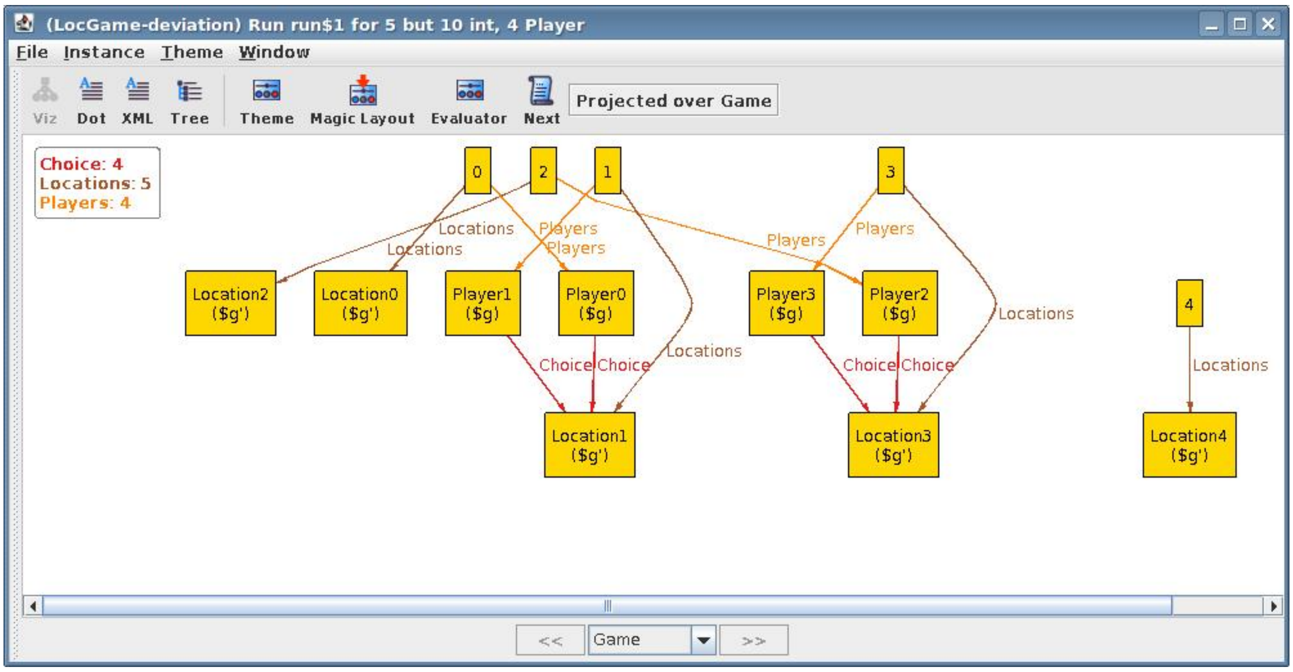


Figure 7.1: An instance of the LG model projected over the Game signature

The way Alloy Analyzer works can be regarded as a constraint solver for the relational logic behind the Alloy syntax. The concrete steps are rather those of a compiler: first the constraint is formulated using the signatures and fields describing the model and the defined predicates, next this is translated in a boolean formula over the specified scope, this translation uses standard techniques like skolemization and the fact that the scope is finite to convert formulae containing

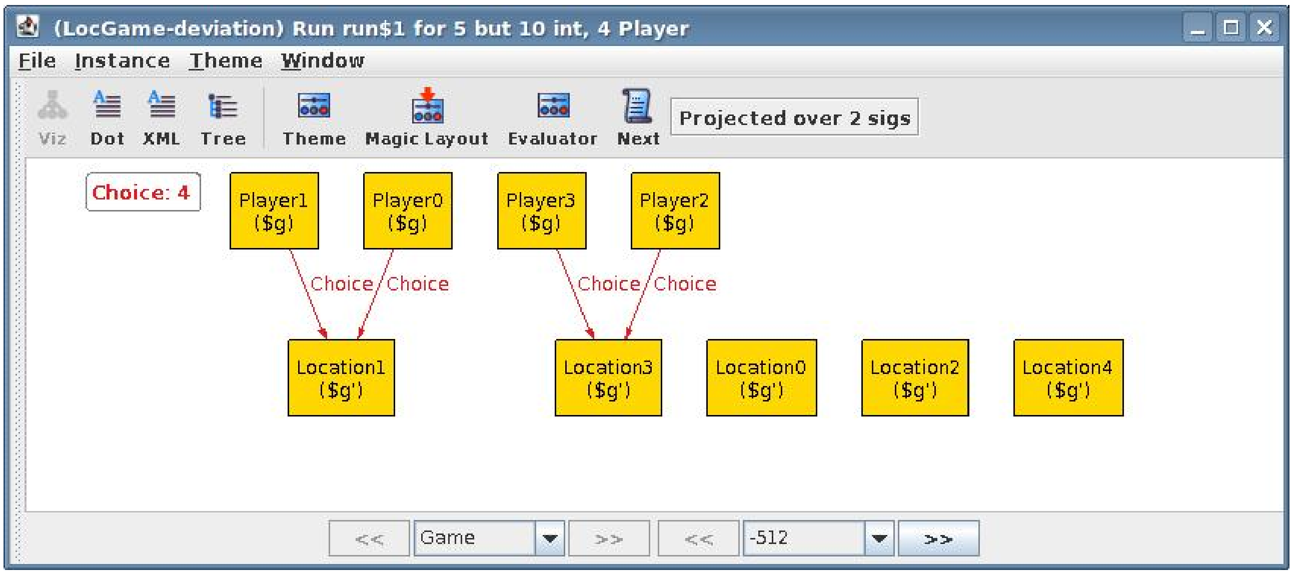


Figure 7.2: An instance of the LG model projected over Game and Int signatures

quantifiers to disjunctions over the range of the scope. Also, various strategies to improve performance by reducing the search space based on symmetries between all possible variable assignments are also employed at this stage.

The final step is to feed the obtained boolean formula to a 'off the shelf' SAT-solver that tries to find a satisfying assignment, which is then translated back in the Alloy relational logic and used to construct and visualize the relational structure representing the instance. The last three lines in the previous output contain information about the SAT-solver used, the number of variables and clauses, and the solving parameters and total time. If needed some of these parameters can be changed as the analysis requires. See [52] for a detailed description of the analysis process and the theoretical foundations of its logical background.

Besides running a predicate in order to find an example, Alloy Analyzer can also check an assertion in order to find a counterexample. In this case the analysis will also perform an exhaustive search inside the multi dimensional space of test cases specified by the scopes of signatures and relevant parameters. The difference is now that the searching involves a refutation, that is, it tries to find an assignment that makes the facts of the model true but the assertion false. If no such instance is found then the assertion is valid within the scope.

For instance the following assertion can be used to justify the entailment from Corollary 6.3.6 within a limited scope.

```
assert char {
  all g, g' : Game, p : Player | not ne[g,g',p]
}

check char for 5 but 4 Player, 10 int
```

In this case the scopes bounds are specified as in the case of the previous predicate as can be seen in the following output:

```
$ Executing "Check char for 5 but 10 int, 4 Player"
   Sig this/Player scope <= 4
   Sig ordering/Ord scope <= 1
   Sig open$3/Ord scope <= 1
   Sig this/Location scope <= 5
   Sig this/Game scope <= 5
   Sig this/Location forced to have exactly 5 atoms.
   Sig this/Player forced to have exactly 4 atoms.
   Sig this/Location == [[Location$0], [Location$1], [Location$2],
[Location$3], [Location$4]]
   Sig this/Player == [[Player$0], [Player$1], [Player$2], [Player$3]]
   Sig this/Game in [[Game$0], [Game$1], [Game$2], [Game$3], [Game$4]]
   Sig ordering/Ord == [[ordering/Ord$0]]
   Sig open$3/Ord == [[open$3/Ord$0]]
   Solver=sat4j Bitwidth=10 MaxSeq=5 SkolemDepth=1 Symmetry=20
   1527795 vars. 389 primary vars. 6833561 clauses. 707222ms.
   No counterexample found. Assertion may be valid. 171107ms.
```

The last three lines contain again information about the SAT solver parameters, and variables and clauses used. These are also as explained before. The only new content is in the last line, this time the goal is to find counterexamples for the assertion. The analyzer reports that no counterexample has been found for the assertion. This is not a result that implies that the assertion is valid for any scope, it just implies validity inside the bounded scope considered and gives some confidence about the general case as many of the remaining models might share the same properties.

7.3 Implementing Probabilistic Extensions

In this section we will introduce the implementation behind the probabilistic extension for DELQ discussed in the previous chapter. Some outstanding features of the implementation are: modeling the dynamics of epistemic updates via questioning actions in a probabilistic setting, intuitive display of issue-epistemic models and questioning action models both enriched with probabilistic components and for the results of update operations, model checking of issue-epistemic probabilistic formulae in issue-epistemic probabilistic structures. The main module `QPRO.lhs` is a literate Haskell program [55, 68], building on previously explained issue-epistemic functionality from the `DELQ.lhs` module, and using specific functionality for probabilistic functional programming from [27].

We briefly describe the module implementing probabilistic extensions for DELQ and include some representative illustrations of situations that can be modeled with the probabilistic extensions and we refer to the electronic version of this thesis for the rest of the code and further illustrations.

The following modules are part of the main `QPRO.lhs` module:

`DELQ.lhs` The module that defines the data structures and the standard functionality for the dynamic epistemic logic of questions. These definitions are behind the theoretical aspects introduced in Chapter 2, where their intuitive meaning is introduced and discussed extensively. The details of the implementation itself have been minutely presented in the previous section.

`Probability` The module that defines data structures and functionality for probabilistic functional programming. The full details of the definitions and functions contained in this module are presented and illustrated in [27]. In this section we will only focus on presenting the details of the part containing functionality that is relevant for DELQ. The main components containing DELQ specific functionality are the definition of a probability distribution, functions that generate probability distributions over a state space, functionality for testing and retrieving probabilistic values for specified events in a distribution. Such functions are used to enrich issue-epistemic structures already defined with a new probabilistic component.

`QPRO.lhs` The module that contains data structures and the core functionality for defining and working with probabilistic issue-epistemic models. The previous DELQ-specific data structures are enriched with intuitively adequate probability distributions for both action structures and models of possible worlds. Also corresponding upgrade operations between such structures are defined in a way that handles the new probabilistic aspects in an adequate way for both epistemic considerations but also taking into account technical aspects concerning the new issue relation.

7.3.1 The `Probability.lhs` Module

We will introduce and explain the bare minimum required to understand the part of the `Probability` module that was used for the extension of standard DELQ functionality with probabilistic features. The rest of the code does not have a direct use in our `QPRO.lhs` module and therefore it will be skipped here. The full code is available, explained and illustrated in [27].

The module starts with importing standard *Haskell* functionality available in `Prelude`, lines 231-236, basic and more advanced list functionality, a module to generate and work with random numbers, standard monadic definitions and functions and input-output classes and display utilities.

```
230 module Probability where
231 import List (sort,sortBy,transpose)
232 import ListUtils
233 import qualified Random
234 import Monad
235 import Foreign (unsafePerformIO)
236 import Show
```

The code starts by introducing a couple of auxiliary definitions that set-up the abstract framework for working with probabilities in general. We will also insert along the way comments about how this general framework is instantiated for the particular case of issue-epistemic structures.

The first definition is for an abstract event type, line 238, this can be a subset of the probability state space, implemented as a function from an arbitrary type `a` to a boolean value. This works as a selection function, separating states in which the event occurs from those in which it does not. In the setting of an IEM, the state space will be the domain of the model or an equivalence class in the information partition of an agent and the selection function performs model checking. An event will be represented as a subset thereof, separated by a formula true at the state, or a corresponding disjunction of nominals.

A probability values are represented as `Float` numbers, line 242, and the probability type is introduced by a corresponding constructor in line 240. A probability distribution is then defined using `unD` in line 244 as a list of pairs in which the first element is an arbitrary type `a` and the second is a probability representation. The new type `Dist`, line 244, is used for constructing probability distributions. These are going to be added as a new component in PIMs.

```
238 type Event a = a -> Bool
239
240 newtype Probability = P ProbRep
241
242 type ProbRep = Float
243
244 newtype Dist a = D {unD :: [(a,ProbRep)]}
245
246 instance Monad Dist where
247   return x = D [(x,1)]
248   d >>= f  = D [(y,q*p) | (x,p) <- unD d, (y,q) <- unD (f x)]
249   fail _   = D []
250
251 instance Functor Dist where
252   fmap f (D d) = D [(f x, p) | (x,p) <- d]
```

The basic monadic laws: return, binding and fail are introduced for probability distributions in lines 246-249. Also a probability distribution can be mapped over and is an instance of the `Functor` class, line 251.

```
254 mkD :: [(a,ProbRep)] -> Dist a
255 mkD = checkD . D
256
257 onD :: ([(a,ProbRep)] -> [(a,ProbRep)]) -> Dist a -> Dist a
258 onD f = D . f . unD
259
260 normBy :: Num b => (a -> a -> Bool) -> Dist a -> Dist a
261 normBy f = onD $ accumBy f . sort
```

Several auxiliary functions for constructing and working with probability distributions follow in the code. We only mention here the ones that are most relevant for adding probabilities to issue-epistemic structures.

The function `mkD` takes a list of pairs of states and probabilities and returns a probability distribution after checking some specific properties of the probability values, like the fact that they add up to one, or signals an error if this is not the case, line 254. The function `onD`, line 257, is used to transform a probability distribution by a given function on distributions. The function `normBy`, line 260, is used to normalize an input distribution by a given grouping function.

```
263 type Spread a = [a] -> Dist a
264
265 uniform :: Spread a
266 uniform = shape (const 1)
267
268 enum :: [ProbRep] -> Spread a
269 enum ps xs = mkD $ zip xs ps
270
271 enumPC :: [ProbRep] -> Spread a
272 enumPC ps = enum (map (/100) ps)
```

Various operations on probability distributions are available in the module like, for instance, taking the product of independent distributions, printing a probability distribution, etc. Since they are not directly relevant for our purpose here we do not include them, and refer to [27] for all details.

Some relevant functions are included in the final code block. They are centered around the `Spread` datatype, line 263. It contains a generator for a probability distributions of various sorts used in standard applications, normal distributions, linear, uniform, etc. The probability generators that we will use more often are the uniform distribution and the enumeration distribution.

The first one, line 266, creates a uniform distribution over a given domain by spreading evenly probability values over all entities. This will be used for instance for creating a uniform probability distribution over an equivalence class inside an information partition, to reflect either issue equivalence between events of epistemic uncertainty between worlds.

Another useful function generates a probability distribution from a list of arbitrary values and a list of states, line 268. This is particularly useful to model specific scenarios that have predefined probabilistic parameters. Finally, the function defined in line 271 is creating a percentage-based probability distribution from a list of numeric values given as input.

In order to be useful in practice for modeling realistic scenarios of issue-epistemic interaction probability distributions have to be suitable for various transformations and their domain and evolving values have to be available for testing at various stages of the dynamic evolution. In line 274 a probabilistic transition type is defined as a function that takes an input value to a probability distribution. Extracting and mapping the domain of a probabilistic distribution is also a functionality that comes in handy in various contexts, its definition is given in line 276 by mapping over the first elements in the distribution.

Last but not least, a test function capable of retrieving the probability values for a given event describing the issue or epistemic situation under consideration in the intended scenario to be modeled is also provided in the module. It is defined as an infix operator with priority 8, line 279, and takes as input an event, i.e. subset of the domain together with a probability distribution over the domain and retrieves its occurrence probability, see line 281.

```
274 type Trans a = a -> Dist a
275
276 extract :: Dist a -> [a]
277 extract = map fst . unD
278
279 infix 8 ??
280
281 (??) :: Event a -> Dist a -> Probability
282 (??) p = P . sum . filter (p . fst) . unD
```

With all these in place the code can deal extremely efficiently with representing and solving standard problems in probability domains. For instance a problem like the following one only takes half a line of code to be solved:

```
> (==[Q,K,A]) ?? select 3 [A,K,Q,A,K]
6.7%
```

> “Given a deck of five cards containing one queen, and two of both aces and kings, what is the probability of drawing a queen, a king and an ace, in this order and without putting them back in the deck?”

Even more complex probabilistic scenarios and famous puzzles like, for instance, the *Monty Hall* problem or the *Riddle of the Quiz-Master* are easily and elegantly handled by the explained code. We refer to [27] for a complete list of such illustrations and further applications.

To proceed towards our main purpose we will continue with showing how the functionality presented so far can be put to work in the context of our dynamic logic of questions. We do this in the next section by constructing line by line both static probabilistic issue structures and upgrade action model that enable probabilistic dynamics in multi-agent contexts.

7.3.2 The `QPRO.lhs` Module

The module preamble starts by making the specific DELQ functionality available, lines 2-9. This has been already explain in previous chapters of this thesis (see Chapters 2 and 3). On top of the explained DELQ functionality, probabilistic specific functionality is made available by importing the `Probability` module which was already discussed and further standard *Haskell* modules, lines 10-13.

```
1  module QPRO where
2  import Syntax
3  import Structures
4  import BinaryRel
5  import Semantics
6  import Upgrade
7  import Shortcuts
8  import Display
9  import DELQ
10 import Probability hiding (choose)
11 import Monad (liftM)
12 import List
13 import qualified Data.Set as Set
```

The main task ahead will be to bridge the issue-epistemic structures and probabilistic patterns from previous sections. This will be achieved in stages, by adding state-dependent probability distributions to issue-epistemic models, and by enriching model transformations with upgrade rules for probabilistic actions. All these will use the code patterns introduced and discussed so far.

```
15 worldindex m w = (elemIndices w (sort (dom m)))!!0
16
17 reornameq :: EIM (Integer,Integer) -> EIM Int
18 reornameq m@(Eim w a q k v p) =
19   Eim w' a' k' q' v' p'
20   where
21     w' = foldr (++) [] (map (\x -> (elemIndices x (sort w))) (sort w))
22     a' = a
23     k' = foldr (++) [] (map (\x -> (zip3
24          (replicate (length (relK x m)) x) (
25          map (worldindex (m)) (map fst (relK x (m)))) (
26          map (worldindex (m)) (map snd (relK x (m)))))) a)
27     q' = foldr (++) [] (map (\x -> (zip3
28          (replicate (length (relQ x m)) x) (
29          map (worldindex (m)) (map fst (relQ x (m)))) (
30          map (worldindex (m)) (map snd (relQ x (m)))))) a)
31     v' = zip w' (map snd v)
32     p' = foldr (++) [] (map (\x->(elemIndices x (sort p))) (sort p))
```

After a product upgrade the domain of an IEM consists of pairs of worlds and events, however, many of the utilities introduced before are easier to keep track of as atomic states. The function in line 15 returns the world-index after sorting the domain. This is later used in the `reordnameq` function which takes an issue-epistemic model and returns it reordered and renamed, lines 17-32.

Line 34 introduces and defines the structure that will be used as an illustrative example in this section, it is the result of a questioning upgrade, as defined previously, using the structures `m0` and `q0` already introduced in previous sections, also the function is setting the actual world to 0.

```
34 mp = (actual (reornameq (upgradeq m0 q0)) 0)
35
36 actual :: EIM Int -> Int -> EIM Int
37 actual m@(Eim dom agts accs eqqs val act) w = (Eim dom agts accs eqqs val act1)
38     where
39     act1 = [w]
40
41 qpart :: EIM Int -> Agent -> [[Int]]
42 qpart m a = rel2partition (dom m) (relQ a m)
43
44 kpart :: EIM Int -> Agent -> [[Int]]
45 kpart m a = rel2partition (dom m) (relK a m)
46
47 qcell :: EIM Int -> Int -> Agent -> [Int]
48 qcell m w a = head (filter (elem w) (qpart m a))
49
50 kcell :: EIM Int -> Int -> Agent -> [Int]
51 kcell m w a = head (filter (elem w) (kpart m a))
52
53 pqpart :: EIM Int -> Int -> Agent -> Dist Int
54 pqpart m w a = uniform (qcell m w a)
55
56 pkpart :: EIM Int -> Int -> Agent -> Dist Int
57 pkpart m w a = uniform (kcell m w a)
```

The `actual` function at lines 36-39, sets the set of actual states to a singleton given as input. This is important for working with questions with mutually exclusive answers represented by a cover instead of a partition.

The pair of functions line 41 and line 44 take an epistemic model and an agent, and return the issue respectively the information partition of the agent. Even more specific, the following pair of functions at line 47 and line 50 take as input an issue-epistemic model an agent and a world and return the equivalence class to which the world belongs to in the issue or information partition, respectively.

The first encounter with a probability distribution takes place in the pair of functions from line 53 and line 56, these take as input an issue-epistemic model an agent and a world and return a uniform probability distribution over the agent's issue respectively information cell containing the world. This is the simplest way to create a distribution, and we will use it hereafter for illustrative purposes, more ways of generating probabilities can be used if needed, for instance by enumeration of arbitrary values, by a normal distribution, etc. as discussed before.

The following output illustrates how these functions work on the concrete example introduced before at line 34.

```
*QPR> qpart mp a
[[0,5],[1,3],[2,6],[4,7]]

*QPR> qcell mp 1 a
[1,3]
```

```
*QPR> pqpart mp 1 a
1  50.0%
3  50.0%
```

Changing probabilities via issue-epistemic dynamics is done by extracting and performing computations on relevant values. The function at line 59 provides a translation by turning a probability into a float number. In issue-epistemic contexts it is most often of importance to be able to manipulate values across cells of equivalence relations. The functions introduced at lines line 62 and line 66 provide this utility for each of the relevant binary relations by retrieving the cell's probability distribution as floating number values.

```
59 p2fl :: Probability -> ProbRep
60 p2fl p@(Probability.P n) = n
61
62 flqpart :: EIM Int -> Int -> Agent -> [Float]
63 flqpart m w a = map (\x -> p2fl
64   ((??) (==x) (pqpart m w a))) (qcell m w a)
65
66 flkpart :: EIM Int -> Int -> Agent -> [ProbRep]
67 flkpart m w a = map (\x -> p2fl
68   ((??) (==x) (pkpart m w a))) (kcell m w a)
```

The last step in building a probabilistic issue-epistemic model is done by lifting the probability distribution to the whole domain in a way that is consistent with both the issue and information partitions. The pair of functions starting from lines 70 respectively 76 take an issue-epistemic model, a world and an agent and use the preexisting issue or information structure to generate automatically a probability distribution on the domain. This probability distribution is generated in such a way that it remains intuitively adequate for the agent's local perspective, but this is only an empirical requirement that provides an adequate description. However, it is also possible to generate other kinds of probability distributions.

```
70 distwq :: EIM Int -> Int -> Agent -> Dist Int
71 distwq mg w a = enum (take (length (dom mg \\ qcell mg w a))
72   (repeat 0) ++
73   (flqpart mg w a)) ((dom mg \\ qcell mg w a) ++ qcell mg w a)
74
75 distwk :: EIM Int -> Int -> Agent -> Dist Int
76 distwk mg w a = enum (take (length (dom mg \\ kcell mg w a))
77   (repeat 0) ++
78   (flkpart mg w a)) ((dom mg \\ kcell mg w a) ++ kcell mg w a)
```

The following output continues the series of intuitive illustrations for the way in which the previously explained functions work in the context of the concrete issue-epistemic model now enriched with probabilistic components.

In the process of adding probabilistic components we make crucial use of the testing and display utilities discussed before, put to work now on various components inside the model under consideration.

```
*QPR> (??) (==1) (pqpart mp 1 a)
 50.0%
*QPR> p2fl ((??) (==1) (pqpart mp 1 a))
0.5
*QPR> flqpart mp 1 a
[0.5,0.5]
```

One noticeable feature of the setting used so far is that all the worlds in the same issue partition cell have a positive probability while all the worlds outside the locally relevant partition cell receive a zero probability. Throughout the information or issue cells the values are uniformly distributed and this is done consistently for all worlds in the model and all agents. This is adequate for settings in which all questioning actions are distinguished but the framework allows for more variation to obtain an adequate model for indistinguishable questioning and resolution actions as discussed before.

```
*QPR> distwq mp 1 a
1 50.0% 3 50.0% 0  0.0% 2  0.0% 4  0.0% 5  0.0% 6  0.0% 7  0.0%

*QPR> distwq mp 5 b
0 50.0% 5 50.0% 1  0.0% 2  0.0% 3  0.0% 4  0.0% 6  0.0% 7  0.0%
```

It is time to put all these together in a function that initializes a probabilistic issue-epistemic model, lines 79-84 do exactly this: for any issue-epistemic structure given as input a new one is generated with an additional probabilistic structure which is intuitively adequate in the sense just discussed. We use here this distribution for illustrative purposes but it is not the only possibility to consider for an initial probability distribution, many other candidates can be empirically adequate in different contexts, for instance, an enumeration distribution.

```
79 initPM :: EIM Int -> PIM Int
80 initPM mg@(Eim dom agts accs eqqs val act) = Pim
81   dom agts accs eqqs prob val act where
82     { prob = (foldr (++) [] (map (\y -> (zip3 (take
83     (length dom) (repeat y)) dom  (map (\x ->
84     distwk mg x y) dom))) agts ))}
85
86 data PIM state = Pim
87   [state]
88   [Agent]
89   [(Agent,state,state)]
90   [(Agent,state,state)]
91   [(Agent,state,Dist state)]
92   [(state,[Prop])]
93   [state]
94   deriving (Eq,Show)
```

Next, the data structure for probabilistic issue epistemic models is defined from line 86 to line 94: previous components in the structures remain unchanged, as they were explained several times in previous sections, the only new component is a list of triples containing an agent label a state name and a probability distribution over the states in the domain of the model.

```
96  relKp :: Agent -> PIM  state -> Rel state
97  relKp a m@(Pim _ _  acc _ _ _ _) =
98    [ (x,y) | (ag,x,y) <- acc, a == ag ]
99
100 relQp :: Agent -> PIM  state -> Rel state
101 relQp a m@(Pim _ _ _ eqq _ _ _) =
102   [ (x,y) | (ag,x,y) <- eqq, a == ag ]
```

Two auxiliary functions used to retrieve the binary relations for issue, line 100, and for uncertainty, line 96, but now from a probabilistic model, are introduced next. These are useful for converting the components of a probabilistic structure in an easier to show list of numeric values. This display conversion is performed by the function between the lines 104-110.

Finally, these can be displayed on the screen in a readable fashion by the function on line 112. The standard DELQ components are displayed in an analogous way to what we presented before, for the new probabilistic components the display functions from the `Probability` module are employed to display the probability distribution for each world in the domain.

```
104 showS5qp :: (Ord state, Show state) => PIM state-> [String]
105 showS5qp m@(Pim dom ags acc eqq prb val act) =
106   [show dom] ++ [show val] {-map show val-} ++
107   map show [(a, (rel2partition dom) (relKp a m)) | a <- ags]
108   ++ [show act] {-[" "] -}  ++
109   map show [(a, (rel2partition dom) (relQp a m)) | a <- ags]
110   ++ map show prb
111
112 dppq :: (Ord state, Show state) => PIM state-> IO()
113 dppq = putStrLn . unlines . showS5qp
```

The next obvious step at this stage is to add probabilistic components to issue-epistemic action structures. The data structure for these extended event models is defined at lines 115-122. It has the customary DELQ components as discussed before plus two new probabilistic components.

The first probabilistic component is a list of pairs of pairwise inconsistent issue-epistemic formulae and probability distributions on the domain of states representing events. It replaces a precondition function for event execution by an occurrence probability for events. This is similar with structures used in [94] but now using a language with issue modalities and nominals which allow an expressive level adequate to model complex questioning actions.

```
115 data PQM state = Pqm
116   [state]
117   [Agent]
118   [(Agent,state,state)]
```

```
119   [Formq]
120   [(Formq,Dist state)]
121   [(Agent,state,Dist state)]
122   deriving (Eq,Show)
```

The second probabilistic component is a list of triples in which the first component is an agent, the second is a state, which might be any epistemic event, a question, a message, and so forth, and a probability distribution over the events. This represents the subjective probability of the agents with regard to what subset of the events takes place in a dynamic action.

The function `initPAq`, lines 124-131, is used to generate automatically probabilistic action models. It takes as input a list of agents, a list of events and a list of propositional formulas, respectively, and returns a probabilistic update model.

The following triple of auxiliary functions, at line 133, line 136 and line 139, are used to retrieve the three relevant probabilistic components from a given issue-epistemic structure: the set of mutually inconsistent formulae representing preconditions, the list of pairs of formulae and probability values representing occurrence probabilities and the list of triples of agents, events, and probability distributions on the domain representing subjective expectations.

```
124 initPAq :: (Num state, Enum state) =>
125   [Agent] -> [state] -> [Formq] -> (PQM state)
126 initPAq ags events precs =
127   (Pqm events ags accs precs precprobs probs) where
128     accs = [(ag,st1,st2) | ag <- ags,
129             st1 <- events, st2 <- events]
130     precprobs   = [(prec,(uniform events)) | prec <- precs]
131     probs = [(ag,ev,(uniform events)) | ag <- ags, ev <- events]
132
133 precs :: PQM state -> [Formq]
134 precs pam@(Pqm _ _ _ precs _ _) = precs
135
136 precprobs :: PQM state -> [(Formq,Dist state)]
137 precprobs pam@(Pqm _ _ _ _ precprobs _) = precprobs
138
139 probabilities :: PQM state -> [(Agent,state,Dist state)]
140 probabilities pam@(Pqm _ _ _ _ _ precs) = precs
```

The basic unit to work with during the computations involved in a dynamic upgrade of probability measures will be the individual probabilistic value that a given agent assigns at a local state to another world or set of worlds or events to be the case. The function defined between lines 142 and 146 performs the task of extracting this value for further manipulation.

```
142 prob_ag_evt1_evt2 :: Eq state =>
143   PQM state -> Agent -> state -> state -> Probability
144 prob_ag_evt1_evt2 m a e1 e2 = (??) (==e2) (trd33 (
145   (filter (\y->(snd33 y == e1)) (filter (\x -> (fst33 x) == a)
146   (probabilities m)))!!0))
```

```
147
148 pim2eim :: PIM state -> EIM state
149 pim2eim m@(Pim dom agt acc eqq prb val act) =
150   Eim dom agt acc eqq val act
```

In certain contexts it becomes useful to detach a model from its probabilistic content before further processing it, the function at line 148 provides this utility. In order to compute the local perspective for a dynamic probabilistic action it is important to select the list of preconditions that are true at the given state, the function starting at line 152 does exactly this.

```
152 trueprecs :: Ord state =>
153   PQM state -> PIM state -> state -> [Formq]
154 trueprecs qm m w = filter (\x ->
155   models (pim2eim m) w x) (precs qm)
156
157 statepre :: PIM Int -> PQM Int
158   -> Int -> Int -> Probability
159 statepre pim pam state event | trueprecs pam pim state ==
160   [] = Probability.P 0.0
161                              | otherwise =
162   (??) (==event) (snd ((filter (\x -> (fst x) ==
163   (trueprecs pam pim state)!!0) (precprobs pam))!!0))
```

The events represented by preconditions which turn out false at the given world will be assigned a zero probability by the `statepre` function, lines 157-163.

```
*QPR> prob_ag_wrd1_wrd2 mpp a 1 2
 12.5%
*QPR> statepre mpp sample 4 2
 50.0%
*QPR> prob_ag_evt1_evt2 sample a 1 2
 50.0%
```

The previous and following outputs illustrate the most relevant of the functions discussed until now using our working example so far.

```
*QPR> statepre mpp sample 3 1
 50.0%
*QPR> statepre mpp sample 4 2
 50.0%
*QPR> statepre mpp sample 1 1
  0.0%
```

Finally, we have come to the point in which we have all the necessary ingredients to introduce the probabilistic update operation. The function `product2`, line 164, does this pointwise starting from seven arguments: an agent, a probabilistic issue-epistemic structure, a probabilistic questioning action model, and two pairs of a world and an event. The result of the pointwise products is put together as a sum over the relevant set of world event pairs in the function starting from line 169. As in the case of probabilities for events, we have an auxiliary function used to retrieve the local probability for both the objective occurrence, line 174, and subjective expectation, line 178 and line 184.

```
164 product2 a m q w e w’ e’=
165   (p2fl (prob_ag_wrd1_wrd2 m a w w’)) * (
166    p2fl (statepre m q w’ e’)) * (
167    p2fl (prob_ag_evt1_evt2 q a e e’))
168
169 sigma a m q w e = sum (map sum (map (\y -> (map
170   (*(p2fl (prob_ag_wrd1_wrd2 m a w y)))) (map
171   (\x -> (p2fl (prob_ag_evt1_evt2 sample a e x))*(p2fl
172   (statepre mpp sample y x))) (dom_pqm sample))) (dom_pim mpp)))
173
174 prob_w_w a m q (w, e) (w’, e’) | sigma a m q w e == 0 = 0
175                                | otherwise =
176   (product2 a m q w e w’ e’) / (sigma a m q w e)
177
178 prob_ag_wrd :: PIM Int -> Agent -> Int ->
179   [(Agent, Int, Dist Int)]
180 prob_ag_wrd m a w = filter (\y->(snd33 y == w)) (
181   filter (\x -> (fst33 x) == a) (precprobsw m))
182
183 prob_ag_wrd1_wrd2
184   :: (Eq t1) => PIM t1 -> Agent -> t1 -> t1 -> Probability
185 prob_ag_wrd1_wrd2 m a w1 w2 = (??) (==w2) (trd33 ((filter
186   (\y->(snd33 y == w1)) (filter (\x -> (fst33 x) == a)
187   (precprobsw m)))!!0))
```

Probabilistic Product Upgrade All the functionality introduced so far allows us to give the final contribution of this section: a function performing probabilistic product update between probabilistic issue epistemic structures and corresponding probabilistic questioning action models.

The function `probProdUpd`, starting from line 189, takes as input two structures and returns the result of performing an update on the probabilities according to the computations explained before leading to an intuitive model of dynamic evolution of both information and issues.

The resulting model is constructed componentwise as follows: the new domain contains pairs of worlds and events such that the probability value resulting from taking the product of the probabilities of the two elements in the pair is not zero, line 193; the function assumes identical agent sets in the two combined structures, line 195, and takes this set to be same in the resulting model; the new accessibility relation is constructed in the standard way from the old ones, line 196; the new issue relation inherits the previous structure between the surviving worlds, line 199; the new probability distributions are computed using the functions explained before in this section, line 204; the valuation function preserves the old values of the surviving worlds, line 205; and the actual world set selects pairs containing the actual elements of the starting models, line 206.

```
189 probProdUpd :: PIM Int -> PQM Int -> PIM (Int,Int)
190 probProdUpd pem@(Pim dom agts accs eqqs prob val act)
191  pam@(Pqm events agts1 accs1 precs precprobs probs) =
192  Pim dom' agts' accs' eqqs' prob' val' act' where
193    dom' = [ (w,e) | w <- dom, e <- events,
194                  p2fl (statepre pem pam w e) > 0.0]
195    agts' = agts
196    accs' = [ (ag1,(w1,s1),(w2,s2)) | (ag1,w1,w2) <- accs,
197              (ag2,s1,s2) <- accs1, ag1 == ag2, elem (w1,s1) dom',
198              elem (w2,s2) dom' ]
199    eqqs' = [ (ag1,(w1,s1),(w2,s2)) |
200              (ag1,w1,w2) <- eqqs, s1 <- events, s2 <- events,
201              elem (w1,s1) dom', elem (w2,s2) dom' ]
202    prob' = [(ag, (w,e), enum (map (prob_w_w ag pem pam (w,e))
203             (dom')) dom') | ag <- agts1, w <- dom, e <- events,
204             elem (w,e) dom']
205    val' = [((w,e),list) | (w,e) <- dom', (w',list) <- val, w==w']
206    act' = nub [(w,e) | (w,e) <- dom', w <- act]
207
208 precprobsw :: PIM state -> [(Agent,state,Dist state)]
209 precprobsw m@(Pim _ _ _ _ prb _ _) = prb
210
211 dom_pim m@(Pim dom agt acc eqq prb val act) = dom
212 dom_pqm q@(Pqm dom agt acc frm pre prb) = dom
213
214 mpp = initPM mp
215 sample = initPAq [a,b] [1,2] [p,q]
216 mppp = probProdUpd mpp sample
```

Finally tree more auxiliary functions are introduced providing the utility of retrieving needed components from a probabilistic structure: the list of probability distributions, line 208, the list of states in the starting probabilistic issue-epistemic model, line 211, and the list of events in the starting probabilistic action structure, line 212. Some concrete illustration of how the componentwise computations work in order to produce the components in the new structure from the components of the starting structures are provided in the following outcomes.

```
*QPR> map (\x -> prob_ag_wrd1_wrd2 mpp a 1 x) (dom_pim mpp)
[ 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%]
*QPR> map (\x -> prob_ag_evt1_evt2 sample a 1 x) (dom_pqm sample)
[ 50.0%, 50.0%]
*QPR> map (\x -> statepre mpp sample 4 x) (dom_pqm sample)
[ 50.0%, 50.0%]

*QPR> display 1 (map (\y -> (map (\x ->
   prob_ag_wrd1_wrd2 mpp a y x) (dom_pim mpp))) (dom_pim mpp))
[ 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%]
[ 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%, 12.5%]
...
```

At line 214 and following two concrete probabilistic structures are built both for possible worlds and events, together with their probabilistic product upgrade. We end the section by providing the complete display of their composition.

```
*QPRO> dppq mpp
[0,1,2,3,4,5,6,7]
[(0,[]),(1,[]),(2,[p]),(3,[p]),(4,[q]),(5,[q]),(6,[p,q]),(7,[p,q])]
(a,[[0,1,2,3,4,5,6,7]])
(b,[[0,2,5,6],[1,3,4,7]])
[0]
(a,[[0,5],[1,3],[2,6],[4,7]])
(b,[[0,5],[1,3],[2,6],[4,7]])
(a,0,0 12.5% 1 12.5% 2 12.5% 3 12.5% 4 12.5% 5 12.5% 6 12.5% 7 12.5% )
(a,1,0 12.5% 1 12.5% 2 12.5% 3 12.5% 4 12.5% 5 12.5% 6 12.5% 7 12.5% )
(a,2,0 12.5% 1 12.5% 2 12.5% 3 12.5% 4 12.5% 5 12.5% 6 12.5% 7 12.5% )
(a,3,0 12.5% 1 12.5% 2 12.5% 3 12.5% 4 12.5% 5 12.5% 6 12.5% 7 12.5% )
(a,4,0 12.5% 1 12.5% 2 12.5% 3 12.5% 4 12.5% 5 12.5% 6 12.5% 7 12.5% )
(a,5,0 12.5% 1 12.5% 2 12.5% 3 12.5% 4 12.5% 5 12.5% 6 12.5% 7 12.5% )
(a,6,0 12.5% 1 12.5% 2 12.5% 3 12.5% 4 12.5% 5 12.5% 6 12.5% 7 12.5% )
(a,7,0 12.5% 1 12.5% 2 12.5% 3 12.5% 4 12.5% 5 12.5% 6 12.5% 7 12.5% )
(b,0,0 25.0% 2 25.0% 5 25.0% 6 25.0% 1  0.0% 3  0.0% 4  0.0% 7  0.0% )
(b,1,1 25.0% 3 25.0% 4 25.0% 7 25.0% 0  0.0% 2  0.0% 5  0.0% 6  0.0% )
(b,2,0 25.0% 2 25.0% 5 25.0% 6 25.0% 1  0.0% 3  0.0% 4  0.0% 7  0.0% )
(b,3,1 25.0% 3 25.0% 4 25.0% 7 25.0% 0  0.0% 2  0.0% 5  0.0% 6  0.0% )
(b,4,1 25.0% 3 25.0% 4 25.0% 7 25.0% 0  0.0% 2  0.0% 5  0.0% 6  0.0% )
(b,5,0 25.0% 2 25.0% 5 25.0% 6 25.0% 1  0.0% 3  0.0% 4  0.0% 7  0.0% )
(b,6,0 25.0% 2 25.0% 5 25.0% 6 25.0% 1  0.0% 3  0.0% 4  0.0% 7  0.0% )
(b,7,1 25.0% 3 25.0% 4 25.0% 7 25.0% 0  0.0% 2  0.0% 5  0.0% 6  0.0% )
```

In order to display a questioning probabilistic structure we will also need a corresponding display function for it, given in line 218. This also uses an auxiliary function to convert a binary relation into a partition, line 224.

```
218 showS5qpq :: (Ord state, Show state) => PQM state-> [String]
219 showS5qpq q@(Pqm dom ags acc frm pre prb) =
220   [show dom] ++ [show frm] ++ map show pre {-map show val-} ++
221   map show [(a, (rel2partition dom) (relKpq a q)) | a <- ags]
222   ++ map show prb
223 
224 relKpq :: Agent -> PQM  state -> Rel state
225 relKpq a m@(Pqm _ _ acc _ _ _) =
226   [ (x,y) | (ag,x,y) <- acc, a == ag ]
227 
228 dppqq :: (Ord state, Show state) => PQM state-> IO()
229 dppqq = putStrLn . unlines . showS5qpq
```

Finally, the `dppqq` function at line 228 puts all the components together to create the resulting probabilistic model which constitutes the function's output.

```
*QPRO> dppqq sample
[1,2]
[p,q]
(p,1 50.0% 2 50.0% )
```

```
(q,1 50.0% 2 50.0% )
(a,[[1,2]])
(b,[[1,2]])
(a,1,1 50.0% 2 50.0% )
(a,2,1 50.0% 2 50.0% )
(b,1,1 50.0% 2 50.0% )
(b,2,1 50.0% 2 50.0% )
```

The listed components are, line by line: the set of events, the precondition formulae, the objective occurrence probability, agents' uncertainty and agents' probabilistic subjective beliefs about what is going on.

```
*QPRO> dppq (probProdUpd mpp sample)
[(2,1),(2,2),(3,1),(3,2),(4,1),(4,2),(5,1),(5,2),(6,1),(6,2),(7,1),(7,2)]
[((2,1),[p]),((2,2),[p]),((3,1),[p]),((3,2),[p]),((4,1),[q]),((4,2),[q]),
 ((5,1),[q]),((5,2),[q]),((6,1),[p,q]),((6,2),[p,q]),((7,1),[p,q]),((7,2),[p,q])]
(a,[[(2,1),(2,2),(3,1),(3,2),(4,1),(4,2),(5,1),(5,2),(6,1),(6,2),(7,1),(7,2)]])
(b,[[(2,1),(2,2),(5,1),(5,2),(6,1),(6,2)],[(3,1),(3,2),(4,1),(4,2),(7,1),(7,2)]])
[(0,1),(0,2)]
(a,[[(2,1),(2,2),(6,1),(6,2)],[(3,1),(3,2)],[(4,1),(4,2),(7,1),(7,2)],[(5,1),(5,2)]])
(b,[[(2,1),(2,2),(6,1),(6,2)],[(3,1),(3,2)],[(4,1),(4,2),(7,1),(7,2)],[(5,1),(5,2)]])
(a,(2,1),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(2,2),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(3,1),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(3,2),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(4,1),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(4,2),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(5,1),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(5,2),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(6,1),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(6,2),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(7,1),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(a,(7,2),(2,1)  8.3% (2,2)  8.3% (3,1)  8.3% (3,2)  8.3% (4,1)  8.3% (4,2)  8.3%
         (5,1)  8.3% (5,2)  8.3% (6,1)  8.3% (6,2)  8.3% (7,1)  8.3% (7,2)  8.3% )
(b,(2,1),(2,1) 16.7% (2,2) 16.7% (5,1) 16.7% (5,2) 16.7% (6,1) 16.7% (6,2) 16.7%
         (3,1)  0.0% (3,2)  0.0% (4,1)  0.0% (4,2)  0.0% (7,1)  0.0% (7,2)  0.0% )
(b,(2,2),(2,1) 16.7% (2,2) 16.7% (5,1) 16.7% (5,2) 16.7% (6,1) 16.7% (6,2) 16.7%
         (3,1)  0.0% (3,2)  0.0% (4,1)  0.0% (4,2)  0.0% (7,1)  0.0% (7,2)  0.0% )
(b,(3,1),(3,1) 16.7% (3,2) 16.7% (4,1) 16.7% (4,2) 16.7% (7,1) 16.7% (7,2) 16.7%
         (2,1)  0.0% (2,2)  0.0% (5,1)  0.0% (5,2)  0.0% (6,1)  0.0% (6,2)  0.0% )
(b,(3,2),(3,1) 16.7% (3,2) 16.7% (4,1) 16.7% (4,2) 16.7% (7,1) 16.7% (7,2) 16.7%
          (2,1)  0.0% (2,2)  0.0% (5,1)  0.0% (5,2)  0.0% (6,1)  0.0% (6,2)  0.0% )
(b,(4,1),(3,1) 16.7% (3,2) 16.7% (4,1) 16.7% (4,2) 16.7% (7,1) 16.7% (7,2) 16.7%
          (2,1)  0.0% (2,2)  0.0% (5,1)  0.0% (5,2)  0.0% (6,1)  0.0% (6,2)  0.0% )
(b,(4,2),(3,1) 16.7% (3,2) 16.7% (4,1) 16.7% (4,2) 16.7% (7,1) 16.7% (7,2) 16.7%
          (2,1)  0.0% (2,2)  0.0% (5,1)  0.0% (5,2)  0.0% (6,1)  0.0% (6,2)  0.0% )
(b,(5,1),(2,1) 16.7% (2,2) 16.7% (5,1) 16.7% (5,2) 16.7% (6,1) 16.7% (6,2) 16.7%
          (3,1)  0.0% (3,2)  0.0% (4,1)  0.0% (4,2)  0.0% (7,1)  0.0% (7,2)  0.0% )
(b,(5,2),(2,1) 16.7% (2,2) 16.7% (5,1) 16.7% (5,2) 16.7% (6,1) 16.7% (6,2) 16.7%
          (3,1)  0.0% (3,2)  0.0% (4,1)  0.0% (4,2)  0.0% (7,1)  0.0% (7,2)  0.0% )
(b,(6,1),(2,1) 16.7% (2,2) 16.7% (5,1) 16.7% (5,2) 16.7% (6,1) 16.7% (6,2) 16.7%
          (3,1)  0.0% (3,2)  0.0% (4,1)  0.0% (4,2)  0.0% (7,1)  0.0% (7,2)  0.0% )
(b,(6,2),(2,1) 16.7% (2,2) 16.7% (5,1) 16.7% (5,2) 16.7% (6,1) 16.7% (6,2) 16.7%
          (3,1)  0.0% (3,2)  0.0% (4,1)  0.0% (4,2)  0.0% (7,1)  0.0% (7,2)  0.0% )
(b,(7,1),(3,1) 16.7% (3,2) 16.7% (4,1) 16.7% (4,2) 16.7% (7,1) 16.7% (7,2) 16.7%
          (2,1)  0.0% (2,2)  0.0% (5,1)  0.0% (5,2)  0.0% (6,1)  0.0% (6,2)  0.0% )
(b,(7,2),(3,1) 16.7% (3,2) 16.7% (4,1) 16.7% (4,2) 16.7% (7,1) 16.7% (7,2) 16.7%
          (2,1)  0.0% (2,2)  0.0% (5,1)  0.0% (5,2)  0.0% (6,1)  0.0% (6,2)  0.0% )
```

The result of the probabilistic upgrade is as expected: new states are now pairs of worlds and events as the precondition product mechanism dictates, the valuation of old worlds is inherited, new uncertainties are constructed by combining the old ones, the new issue relation is constructed from the old one and also processes the probabilistic input, and the new probability distribution takes a local product of both objective and subjective previous probabilities.

7.3.3 Issue-Probabilistic Minimal Model

The first step in the minimization algorithm is to ensure atomic harmony at propositional level. In order to do this the function at line 231 retrieves the valuation component in a PIM. Next, using the retrieved valuation, the function at line 234 tests two worlds for propositional equivalence. Based on this test the block of any world in the model is computed by the function at line 238. This block equivalence induces a partition of the domain based on propositional equivalence as constructed in line 241

```
231  pval :: PIM t -> [(t, [Prop])]
232  pval m@(Pim _ _ _ _ _ val _) = val
233
234  propeq :: (Eq a) => a -> a -> PIM a -> Bool
235  propeq w1 w2 m = snd ((filter (\x -> (fst x == w1)) (pval m))!!0) ==
236    snd ((filter (\x -> (fst x == w2)) (pval m))!!0)
237
238  propart :: (Eq t) => t -> PIM t -> [t]
239  propart w m = [x | x <- dom_pim m, propeq x w m ]
240
241  ppart :: (Eq a) => PIM a -> [[a]]
242  ppart m = nub (map (\x -> propart x m) (dom_pim m))
```

The first running example in this section is introduced in the next output block and the working of the `ppart` function in illustrated immediately after.

This is a very simple model with only one agent which however already contains all the needed ingredients. The most noteworthy feature is the logical relation between worlds 0 and 4.

```
*QPRO> dpq $ m5i
[0,1,2,3,4]
[(0,[]),(1,[p]),(2,[p]),(3,[p]),(4,[])]
(a,[[0,3],[1,4],[2]])
[0,1,2,3,4]
(a,[[0,3],[1],[2,4]])

*QPRO> ppart $ initPM m5i
[[0,4],[1,2,3]]
```

Each world in the domain will belong to an equivalence class under the propositional partition. The function at line 244 retrieves for any world w taken as an argument the partition cell or block of w.

The two binary accessibility relations in the model induce an image for each world based on the block to which it belongs. The functions at lines 247 and 251 take the same arguments and retrieve the K-image and the Q-image for the given world, respectively, using a cell-by-cell construction.

```
244 cell :: (Eq a) => [[a]] -> a -> [a]
245 cell part w = head (filter (elem w) part)
246
247 kacc :: (Eq a) => PIM a -> [[a]] -> a -> Agent -> [[a]]
248 kacc m@(Pim _ _ acc _ _ _ _) part w agt =
249   nub [cell part x | (agt',y,x) <- acc, agt' == agt, y == w ]
250
251 qacc :: (Eq a) => PIM a -> [[a]] -> a -> Agent -> [[a]]
252 qacc m@(Pim _ _ _ eqq _ _ _) part w agt =
253   nub [cell part x | (agt',y,x) <- eqq, agt' == agt, y == w ]
```

We can already notice that the image under the K relation for connected worlds is the same and is different for unconnected worlds:

```
*QPRO> kacc (initPM m5i) (ppart (initPM m5i)) 0 a
[[0,4],[1,2,3]]
*QPRO> kacc (initPM m5i) (ppart (initPM m5i)) 1 a
[[1,2,3],[0,4]]
*QPRO> kacc (initPM m5i) (ppart (initPM m5i)) 2 a
[[1,2,3]]
*QPRO> kacc (initPM m5i) (ppart (initPM m5i)) 3 a
[[0,4],[1,2,3]]
*QPRO> kacc (initPM m5i) (ppart (initPM m5i)) 4 a
[[1,2,3],[0,4]]
```

The same happens for the issue relation, now with world 1 as unconnected world:

```
*QPRO> qacc (initPM m5i) (ppart (initPM m5i)) 0 a
[[0,4],[1,2,3]]
*QPRO> qacc (initPM m5i) (ppart (initPM m5i)) 1 a
[[1,2,3]]
```

```
*QPRO> qacc (initPM m5i) (ppart (initPM m5i)) 2 a
[[1,2,3],[0,4]]
*QPRO> qacc (initPM m5i) (ppart (initPM m5i)) 3 a
[[0,4],[1,2,3]]
*QPRO> qacc (initPM m5i) (ppart (initPM m5i)) 4 a
[[1,2,3],[0,4]]
```

So far there is no way to distinguish worlds 1 and 4, reflecting the variance under standard bisimulation of the intersection modality.

Next we will make use of the function defined at line 255 to tests if two worlds have the same K-image and Q-image in a given issue-epistemic structure. If this is the case then the two worlds are behaviorally indistinguishable from each other and they will satisfy the same modal formulae with epistemic and questioning modalities and any combinations thereof.

```
255 sameKQ :: (Ord t) => PIM t -> [[t]] -> t -> t -> Bool
256 sameKQ m@(Pim _ agts acc eqq _ _ _) part w v =
257   and [and [sort (kacc m part w agt) == sort (kacc m part v agt)
258        | agt <- agts],
259   and [sort (qacc m part w agt) == sort (qacc m part v agt)
260        | agt <- agts]      ]
```

We can see in the following output that in this way many worlds can be distinguished, however, we can also see that worlds 0 and 4 can not.

```
*QPRO> sameKQ (initPM m5i) (ppart (initPM m5i)) 2 3
False
*QPRO> sameKQ (initPM m5i) (ppart (initPM m5i)) 1 2
False
*QPRO> sameKQ (initPM m5i) (ppart (initPM m5i)) 1 3
False
*QPRO> sameKQ (initPM m5i) (ppart (initPM m5i)) 0 4
True
```

In order to have a correct algorithm we have to do more work, in particular, we have to consider the image under the intersection of the two modalities.

The following quadruple of functions implement the double splitting step in the birelational partition refinement algorithm.

```
262 img :: (Eq t) => PIM t -> t -> Agent -> [t]
263 img m@(Pim dom _ acc eqq _ _ _) w agt = nub [ x | x<-dom, (agt,w,x) `elem` acc]
264
265 primg :: (Eq t) => PIM t -> t -> Agent -> [t]
266 primg m@(Pim dom _ acc eqq _ _ _) w agt =
267   nub [ x | x <- dom, (agt,w,x) `elem` eqq, w <- img m w agt ]
268
269 img2 :: (Eq t) => PIM t -> t -> Agent -> [t]
270 img2 m@(Pim dom _ acc eqq _ _ _) w agt = nub [ x | x<-dom, (agt,w,x) `elem` eqq]
271
272 primg2 :: (Eq t) => PIM t -> t -> Agent -> [t]
273 primg2 m@(Pim dom _ acc eqq _ _ _) w agt =
274   nub [ x | x <- dom, (agt,w,x) `elem` acc, w <- img m w agt ]
```

At line 262 the function takes a probabilistic issue-epistemic model, a world and an agent and returns the image for the given world in the given model under the relation indexed to the given agent.

The relational image of a world obtained in the way just described can be further process as a list of worlds. The function at line 265 computes the preimage under the second relation starting from the image under the first relation of the given world in the given model for the given agent.

In a similar fashion, but in the other direction the pair of functions at lines 269, respectively, line 272 compute the preimage of the image for a world in a model using a reversed alternation of the two relations.

As a final step, the function at line 276 takes the intersection of the two preimages computed by the previously explained functions.

```
276 intImg :: (Ord a) => PIM a -> a -> Agent -> [a]
277 intImg m w a = Set.toList (
278   Set.intersection (Set.fromList (primg m w a))
279   (Set.fromList (primg2 m w a)))
280 
281 inters :: (Eq t) => PIM t -> t -> Agent -> [t]
282 inters m@(Pim dom _ acc eqq _ _ _) w agt =
283    [ x | x <- dom, (agt,w,x) `elem` acc, (agt,x,w) `elem` eqq]
284 
285 sameInters :: (Ord t) => PIM t -> [[t]] -> t -> t -> Bool
286 sameInters m@(Pim _ agts acc eqq _ _ _) part w v =
287   and [and [sort (kacc m part w agt) == sort (kacc m part v agt)
288        | agt <- agts],
289        and [sort (qacc m part w agt) == sort (qacc m part v agt)
290        | agt <- agts],
291        and [ intImg m w agt == intImg m v agt | agt <- agts]
292   ]
```

The function at line 281 retrieves the intersection set image for a world in a model for an agent given as parameters of the function.

The function at line 285 tests not only for K-image and Q-image equivalence but also for intersection set image equivalence, thus providing the correct way of further splitting the partition during the refinement.

We can see now that this also induces various images for worlds in the domain. In special, we can see that worlds' 0 and 4 intersection images are different:

```
*QPRO> inters (initPM m5i) 0 a
[0,3]
*QPRO> inters (initPM m5i) 1 a
[1]
*QPRO> inters (initPM m5i) 2 a
[2]
*QPRO> inters (initPM m5i) 3 a
[0,3]
*QPRO> inters (initPM m5i) 4 a
[4]
```

```
*QPRO> sameInters (initPM m5i) (ppart (initPM m5i)) 0 4
False
```

We can use this insight to write a correct split function, line 294, that will separate the relevant blocks in the partition in an adequate way at each step in the refinement process. In contrast, the partition split function at line 303 considers only the KQ-image and captures only issue-epistemic formulae.

```
294 split :: (Ord t) => PIM t -> [[t]] -> [[t]]
295 split m part = splInters m part part
296   where
297   splInters m part [] = []
298   splInters m part (block:blocks) =
299     newblocks ++ (splInters m part blocks)
300       where
301       newblocks = cf2part block (\ x y -> sameInters m part x y)
302
303 splitMinus :: (Ord t) => PIM t -> [[t]] -> [[t]]
304 splitMinus m part = splInters m part part
305   where
306   splInters m part [] = []
307   splInters m part (block:blocks) =
308     newblocks ++ (splInters m part blocks)
309       where
310       newblocks = cf2part block (\ x y -> sameKQ m part x y)
```

The next output block reflects the difference between the two split functions. The first one fails to distinguish worlds 0 and 4, and because of this it will fail in general to provide an adequate formal mechanism for working with formulae containing intersection modalities.

The second one distinguishes worlds 0 and 4 and will therefore provide a general and correct refinement mechanism which will preserve the truth values both modal and intersection formulae in the language.

```
*QPRO> splitMinus (initPM m5i) (ppart (initPM m5i))
[[0,4],[1],[2],[3]]

*QPRO> split (initPM m5i) (ppart (initPM m5i))
[[0],[4],[1],[2],[3]]
```

We can now use this split function in a standard design pattern that implements the refinement process capturing intersection, line 312.

The function at line 315 computes the minimal model under intersection split, as the least fixpoint in the refinement process.

```
312 refint :: (Ord t) => PIM t -> [[t]] -> [[t]]
313 refint m = lfp (split m)
314
315 mintMod :: PIM Int -> PIM [Int]
316 mintMod m@(Pim dom agts acc eqq prb val act) =
317   (Pim dom' agts acc' eqq' prb' val' act')
```

```
318    where
319      dom' = sort (refint m (ppart m))
320      f    = cell dom'
321      val' = (nub . sort) (map (\ (x,y) -> (f x, y)) val)
322      acc' = (nub . sort) (map (\ (x,y,z) -> (x, f y, f z)) acc)
323      eqq' = (nub . sort) (map (\ (x,y,z) -> (x, f y, f z)) eqq)
324      prb' = foldr (++) [] (map (newdist m) agts)
325      act' = nub (sort (map f act))
```

We can now complete our exposition with a concrete illustration:

```
*QPRO> dpq  m7i
[0,1,2,3,4,5,6]
[(0,[]),(1,[p]),(2,[p]),(3,[p]),(4,[]),(5,[]),(6,[])]
(a,[[0,3,5],[1,4,6],[2]])
[0,1,2,3,4,5,6]
(a,[[0,3,5],[1],[2,4,6]])

*QPRO> dpq (pim2eim (mintMod (initPM m7i)))
[[0,5],[1],[2],[3],[4,6]]
[([0,5],[]),([1],[p]),([2],[p]),([3],[p]),([4,6],[])]
(a,[[[0,5],[3]],[[1],[4,6]],[[2]]])
[[0,5],[1],[2],[3],[4,6]]
(a,[[[0,5],[3]],[[1]],[[2],[4,6]]])
```

We start from the model `m7i` in which worlds 0 and 5 , respectively 4 and 6 are bisimilar under intersection but 0 and 4 are not.

We then apply the minimization algorithm to `m7i` and we obtain the expected result, illustrated as a new model in which states are the partition blocks computed by the birelational splitting algorithm.

We can further refine this process by converting the partition blocks which are represented as list of states back to regular states. The function at line 327 performs this conversion, and the function at line 343 combines the minimization and the conversion to bisimulate the initial model into a minimal one.

```
327  convert :: (Num t, Ord t, Eq t1, Enum t) => PIM t1 -> PIM t
328  convert (Pim dom agts accs eqqs prb val act) =
329   Pim dom' agts accs' eqqs' prb' val' act'
330     where
331      f     = apply (zip dom [0..])
332      dom'  = sort (map f dom)
333      val'  = sort (map (\ (x,y) -> (f x,y)) val)
334      accs' = map (\ (x,y,z) -> (x, f y, f z)) accs
335      eqqs' = map (\ (x,y,z) -> (x, f y, f z)) eqqs
336      prb' = map (\ (x,y,z) -> (x, f y, (enum
337        (map p2fl (map (\x -> (==x) ?? (head ( map trd33 (filter
338        (\v -> snd33 v == y) prb)))) (extract (head ( map trd33 (filter
339        (\v -> snd33 v == y) prb) )))))  (map f (extract (head
340        ( map trd33 (filter (\v -> snd33 v == y) prb) ) ) ) ) ))  prb
341      act'  = nub ( sort ( map f act))
342
343  bisint :: PIM Int -> PIM Integer
```

```
344 bisint = convert . mintMod
```

The following output illustrates the results of the `convert` and `bisint` functions. So far the probabilistic component has not been considered. After the issue-epistemic components have been worked out the computation proceeds by taking issue-epistemic cell additivity to construct the new probability distributions. We will shortly illustrate this aspect in the following sections.

```
*QPRO> dpq (pim2eim (convert (mintMod (initPM m7i))))
[0,1,2,3,4]
[(0,[]),(1,[p]),(2,[p]),(3,[p]),(4,[])]
(a,[[0,3],[1,4],[2]])
[0,1,2,3,4]
(a,[[0,3],[1],[2,4]])

*QPRO> dpq (pim2eim (bisint (initPM m7i)))
[0,1,2,3,4]
[(0,[]),(1,[p]),(2,[p]),(3,[p]),(4,[])]
(a,[[0,3],[1,4],[2]])
[0,1,2,3,4]
(a,[[0,3],[1],[2,4]])
```

But before we proceed to introducing and explaining the probabilistic component inside the model minimization algorithm we still have to introduce and explain some auxiliary functions that were already used so far in the birelational partition refinement algorithm.

First, the function at line 346 is used to convert from a characteristic function to a partition. Next the function at line 353 computes the closure of a binary relation by taking again the least fixed point of the expansion of the given relation which is computed by the function at line 358.

```
346 cf2part :: [t] -> (t -> t -> Bool) -> [[t]]
347 cf2part [] r = []
348 cf2part (x:xs) r = xblock : cf2part rest r
349   where
350   (xblock,rest) = (x : filter (r x) xs, filter (not . (r x)) xs)
351
352 closure ::  Ord state => [(Agent,state,state)] ->
353             [Agent] -> [state] -> [state]
354 closure rel agents xs = lfp f xs
355  where f = \ ys -> (nub.sort) (ys ++ (expand rel agents ys))
356
357 expand :: Ord state => [(Agent,state,state)] ->
358   [Agent] -> [state] -> [state]
359 expand rel agents ys = (nub . sort . concat)
360   [ alternatives rel ag state | ag <- agents, state <- ys ]
```

The `expand` function used the accessible states from a given state to construct the next set of accessible states, the alternatives are computed by the function at line 363. Finally, the function at line 369 takes a function represented as a list of pairs and applies it to the second argument.

```
362 alternatives :: Eq state => [(Agent,state,state)] ->
363   Agent -> state -> [state]
364 alternatives rel ag current =
365   [ s' | (a,s,s') <- rel, a == ag, s == current ]
366
367 type State = Int
368
369 apply :: Eq a => [(a,b)] -> a -> b
370 apply [] _ = error "argument␣not␣in␣list"
371 apply ((x,z):xs) y | x == y    = z
372                    | otherwise = apply xs y
```

All these functions have an ancillary utility and were used during the previously presented model minimization algorithm.

7.3.4 The Probabilistic Component

The remaining functions implement the probabilistic part in the minimization process in Algorithm 3. All the functions will, in one way or another, process the probabilities existing in an issue-epistemic structure, therefore, the first step will be to have a function that extracts the probability component, line 374.

```
374 prb :: PIM t -> [(Agent, t, Dist t)]
375 prb m@(Pim dom agts acc eqq prb val act) = prb
376
377 prbAg :: PIM t -> Agent -> [(Agent, t, Dist t)]
378 prbAg mg ag =
379   filter (\x -> ((fst33 x) == ag)) (prb mg)
```

In some contexts information about the probability assigned by a specific agent is needed. The function at line 377 takes a PIM and an agent and filters out and returns the agent's probability distribution over the domain as a list of triples containing the agent a world and a distribution over the domain.

```
*QPRO> prb (initPM m7i)
[(a,0,0  33.3% 3 33.3% 5 33.3% 1  0.0% 2  0.0% 4  0.0% 6  0.0% ),
 (a,1,1  33.3% 4 33.3% 6 33.3% 0  0.0% 2  0.0% 3  0.0% 5  0.0% ),
 (a,2,2 100.0% 0  0.0% 1  0.0% 3  0.0% 4  0.0% 5  0.0% 6  0.0% ),
 (a,3,0  33.3% 3 33.3% 5 33.3% 1  0.0% 2  0.0% 4  0.0% 6  0.0% ),
 (a,4,1  33.3% 4 33.3% 6 33.3% 0  0.0% 2  0.0% 3  0.0% 5  0.0% ),
 (a,5,0  33.3% 3 33.3% 5 33.3% 1  0.0% 2  0.0% 4  0.0% 6  0.0% ),
 (a,6,1  33.3% 4 33.3% 6 33.3% 0  0.0% 2  0.0% 3  0.0% 5  0.0% )]
```

For contexts that require an even more precise computation the function at line 381 takes a PIM and an agent and returns the distribution triple for the specified world that is of interest. This provides the first query-ready format of a distribution, i.e. the distribution is stripped of its first and second elements in the triplet and is therefore suitable to receive predefined probabilistic functions.

```
381 askProb_a_w :: PIM Int -> Agent -> Int -> Dist Int
382 askProb_a_w mg ag w = (trd33 . head) (prob_ag_wrd mg ag w)
```

```
383
384 qq :: PIM Int -> Int -> Int -> Agent -> Probability
385 qq mg w1 w2 ag = (??) (`elem` (cell (rel2partition (dom (pim2eim mg))
386   (relK ag (pim2eim mg))) w2 ) ) (askProb_a_w mg ag w1)
387
388 qqq :: PIM Int -> Int -> [Int] -> Agent -> Probability
389 qqq mg w1 cell ag = (??) (`elem` cell) (askProb_a_w mg ag w1)
```

A first level of probabilistic querying is performed by the function at line 384, which takes a model, a world and an agent and returns the probability for the partition cell under the epistemic relation of the world by taking (i.e. querying) the prior probabilities and summing them up. This uses the predefined testing functionality for the probability of event occurence, in this case the test is the event of 'being inside the information partition cell'.

A second level of probabilistic querying is performed by the function at line 388 which lifts the functionality of the `qq` function from the level of worlds to the level of partition blocks given as the third argument.

```
391 wpdist :: PIM Int -> Int -> Agent -> Dist [Int]
392 wpdist mg w ag = enum (map p2fl (map (\x->(qqq mg w x ag))
393   ( (dom (pim2eim (mintMod mg)))))) (dom (pim2eim (mintMod mg)))
394
395 bpdist :: PIM Int -> [Int] -> Agent -> Dist [Int]
396 bpdist mg bk ag = enum (map p2fl (map (\x->(qqq mg (head bk) x ag))
397   ( (dom (pim2eim (mintMod mg)))))) (dom (pim2eim (mintMod mg)))
398
399 newdist :: PIM Int -> a -> [(a, [Int], Dist [Int])]
400 newdist mg ag = zip3 (repeat ag) (dom (pim2eim (mintMod mg)))
401   (map (\x -> (bpdist mg x a)) (dom (pim2eim (mintMod mg))))
```

Three more auxiliary functions, first, at line 391 taking as input a model, a world and an agent the probability distribution for a world over the blocks in the minimized model is computed. This provides the first level of correspondence from the minimization algorithm.

Next, at line 395 the function takes a model and a block and an agent and returnes the probability distribution for the minimal element/world in the block over the blocks in the minimized model. This provides the second level of abstraction in the minimization algorithm.

```
*QPRO> wpdist (initPM m7i) 0 a
[0,5] 66.7%   [3] 33.3%   [1]  0.0%   [2]  0.0% [4,6]  0.0%

*QPRO> bpdist (initPM m7i) [1] a
[4,6] 66.7%   [1] 33.3% [0,5]  0.0%   [2]  0.0%   [3]  0.0%
```

Finally, the function at line 399 takes a PIM and an agent and builds the new probability distribution in the minimized model.

We have now all the needed ingredients to apply the probabilistic minimization algorithm for a concrete illustrative example.

We start from a probabilistic model in which some worlds are behaviorally equivalent and the probability is constructed in the previously described way as a uniform distribution over the epistemic partition.

```
*QPRO> dppq (initPM m7i)
[0,1,2,3,4,5,6]
[(0,[]),(1,[p]),(2,[p]),(3,[p]),(4,[]),(5,[]),(6,[])]
(a,[[0,3,5],[1,4,6],[2]])
[0,1,2,3,4,5,6]
(a,[[0,3,5],[1],[2,4,6]])
(a,0,0  33.3% 3 33.3% 5 33.3% 1  0.0% 2  0.0% 4  0.0% 6  0.0% )
(a,1,1  33.3% 4 33.3% 6 33.3% 0  0.0% 2  0.0% 3  0.0% 5  0.0% )
(a,2,2 100.0% 0  0.0% 1  0.0% 3  0.0% 4  0.0% 5  0.0% 6  0.0% )
(a,3,0  33.3% 3 33.3% 5 33.3% 1  0.0% 2  0.0% 4  0.0% 6  0.0% )
(a,4,1  33.3% 4 33.3% 6 33.3% 0  0.0% 2  0.0% 3  0.0% 5  0.0% )
(a,5,0  33.3% 3 33.3% 5 33.3% 1  0.0% 2  0.0% 4  0.0% 6  0.0% )
(a,6,1  33.3% 4 33.3% 6 33.3% 0  0.0% 2  0.0% 3  0.0% 5  0.0% )
```

We end this section by an illustration of the input and output of the computation for minimizing under itersimulation and probabilistic cell additivity as previously described in Algorithm 3:

```
*QPRO> dppq (mintMod (initPM m7i))
[[0,5],[1],[2],[3],[4,6]]
[([0,5],[]),([1],[p]),([2],[p]),([3],[p]),([4,6],[])]
(a,[[[0,5],[3]],[[1],[4,6]],[[2]]])
[[0,5],[1],[2],[3],[4,6]]
(a,[[[0,5],[3]],[[1]],[[2],[4,6]]])
(a,[0,5],[0,5] 66.7%   [3] 33.3%   [1] 0.0% [2] 0.0% [4,6] 0.0% )
(a,  [1],[4,6] 66.7%   [1] 33.3% [0,5] 0.0% [2] 0.0%   [3] 0.0% )
(a,  [2],[2]  100.0% [0,5]  0.0%   [1] 0.0% [3] 0.0% [4,6] 0.0% )
(a,  [3],[0,5] 66.7%   [3] 33.3%   [1] 0.0% [2] 0.0% [4,6] 0.0% )
(a,[4,6],[4,6] 66.7%   [1] 33.3% [0,5] 0.0% [2] 0.0%   [3] 0.0% )
```

A final processing step converts the blocks back to integer numbers.

```
*QPRO> dppq (convert (mintMod (initPM m7i)))
[0,1,2,3,4]
[(0,[]),(1,[p]),(2,[p]),(3,[p]),(4,[])]
(a,[[0,3],[1,4],[2]])
[0,1,2,3,4]
(a,[[0,3],[1],[2,4]])
(a,0,0  66.7% 3 33.3% 1  0.0% 2  0.0% 4  0.0% )
(a,1,4  66.7% 1 33.3% 0  0.0% 2  0.0% 3  0.0% )
(a,2,2 100.0% 0  0.0% 1  0.0% 3  0.0% 4  0.0% )
(a,3,0  66.7% 3 33.3% 1  0.0% 2  0.0% 4  0.0% )
(a,4,4  66.7% 1 33.3% 0  0.0% 2  0.0% 3  0.0% )
```

Chapter 8

Conclusions and Outlook

8.1 General Conclusions

The dynamic calculi of questions developed in this thesis show how dynamic-epistemic logic can incorporate a wide range of questioning and 'issue management' actions beyond mere information handling. Our contribution is showing how this can be done precisely, leading to complete dynamic logics that fit well with the general DEL methodology and connects to existing systems.

Moreover, we have indicated how these systems can be used to explore properties of issue management beyond what is found in traditional approaches of questions, including complex questioning and resolution actions, genuine multi agent settings, explicit dynamics of raising and solving issues, and temporal protocols for inquiry and questioning games.

Let us have a final retrospective overview of the main topics that emerged from the entire storyline of the thesis and draw the final conclusions. This will be our starting point for the emerging agenda of further research topics and also for comparisons with alternative approaches in light of the desiderata that motivated our approach from the very beginning.

In Chapter 2 we have shown how dynamic logics of questions can analyze various aspects of private and public inquiry. We extend the standard epistemic models with equivalence relations for questions. We introduce a static logical language to describe such structures by means of corresponding modalities, most important being the resolution modality which uses the intersection of the two equivalence relations for information respectively questioning partitions.

The main contributions contained in Chapter 2 are the following:

- A rich system of dynamic issue-management actions,
- Complete dynamic logics for questioning in DEL style,
- Extension to privacy and product update for questions,
- Extension to temporal protocols for inquiry.

These systems fit entirely within the methodology of dynamic-epistemic logic, and they seem to form a natural complement to what already exists in this area, making the questioning an explicit entity that drives and guides public announcements and other informational events.

In Chapter 3 we complemented this theoretical perspective with an implementation for our logic of questions. This was literate *Haskell* program that extends the previous implementation for epistemic model checking from *DEMo* [107, 108] with questioning specific functionality.

The main new extensions that we provide in this chapter are:

- A richer, more expressive language that includes formulas with intersection modalities describing the interaction between questioning and knowledge,
- Model checking utility for resolution, questioning and epistemic formulae,
- A general and extensible implementation for complex questioning and resolution dynamic actions that emerge in this framework.

We also showed how the implementation is useful by analyzing some paradigmatic examples of questioning scenarios in epistemic settings.

What we did in chapters 2 ans 3 provides a setting in which questions are analyzed and understood in their intricate conceptual, logical, and practical interdependence and essential connection with knowledge and information dynamics. This connection is made explicit within a language that can describe both aspects and their intersection with adequate modalities. We have studied issue-knowledge resolution both at a static level, expressed by an intersection modality, and a dynamic one, by modeling the intersection between relations.

We return to a theoretical approach in Chapter 4 by defining and investigating games with questioning moves. We first look at strategic games with two players and question-answer moves. We then extend this basic approach to settings with more players, sequential moves, and oracles encoding interactions between imperfectly informed agents or limitations in external information sources or measurement instruments and experimental procedures.

The most noteworthy contributions in this chapter are:

- Definitions for games with questioning moves and their solution concepts,
- We give a tripartite interpretation for questioning moves and use it show first why questioning phenomena in multi-agent contexts are more complex than traditionally understood and second to distinguish games with questioning moves from games with informative actions.
- We analyze illustrative examples and present an inexistence result for Nash equilibrium with pure strategies in questioning games with oracles,

- We identify the difraction property and show why it is important for describing strategic abilities in extensive questioning games.

In Chapter 5 we return to implementation by presenting and discussing the Haskell scripts behind the questioning games introduced in the previous chapter. These also extend basic epistemic functionality from [109] to include strategic aspects specific for a game theoretic approach of questioning actions.

Some of the most noticeable new functionalities include the following:

- Our implementation provides 'expressive harmony' for questioning moves in our games by linking the semantic level based on partitions of the domain with a corresponding syntactic level using disjunctions of nominals,
- We compute complete game matrices for games with questioning moves,
- We give an algorithm for minimizing issue-epistemic models using a notion of behavioral equivalence that is adequate for the questioning language.

What we did in chapters 4 and 5 provides a formal setting in which questions arise in an interactive multi-agent environment in which the epistemic aspects plays an important role. We have also studied the strategic aspects that emerge in such a setting both in strategic games with questioning moves and in questioning games in extensive form providing model for long term interactive inquiry.

Chapter 6 approaches the topic of designing questioning strategies in problem solving from a theoretical perspective. We take again solving games as our point of departure and a rich test case representative for a more general theory. In this context we investigate the problem of solving the location game played on a line. Our solution concept of choice will be Nash equilibrium with pure strategies. We also discuss the general relevance of this approach for designing querying strategies in problem solving by using oracles of operational properties to solve a principal problem using efficiently available sources of information.

The most noticeable contributions contained in this chapter are:

- We give a characterization of NE by means of local properties in the game,
- We use an approach based on querying an oracle of local properties and matching of strategy profile fragments to design questioning strategies that solve the game in an efficient way.

In the final section we provide a minimization algorithm for probabilistic issue models based on partition refinement solving the birelational coarsest partition problem and in accordance with the adequate notion of behavioral equivalence for probabilistic issue models and probabilistic questioning actions.

In Chapter 7 we present a *Haskell* implementation illustrating how queries of local properties in the game can search for equilibrium strategy profiles using list comprehension. This assumes the existence of oracles of local properties and uses this to search for Nash equilibria in pure strategies.

The result needed to show that the query strategies are correct is the following:

- We give an *Alloy Analyzer* implementation for building coutermodels for the location game and checking assertions about logical entailment within a predetermined scope between facts expressing local properties and NE.

What we did in chapters 6 and 7 provides a setting in which the connection between a questioning theory and the process of designing efficient querying strategies is investigated in the context of a concrete example. The interactive location game studied and the solution concept investigated have also an independent interest and relevance. Besides this we provide a bridge between a theory of questions and known search heuristics using backtrack oracles.

The final Chapter 8, shows how our approach gives rise to a coherent research agenda with a broader scope, and points to some further directions for future research and comparisons with alternative approaches.

8.2 Future Research and Outlook

So far in this thesis, we have shown how dynamic logics of questions can analyze various aspects of private and public inquiry. These systems fit entirely within the methodology of dynamic-epistemic logic, and they seem to form a natural complement to what already exists in this area, making the questions explicit that drive public announcements and other informational events. In line with this first finding, many lines of investigation open up:

Further general issues that emerge from our research agenda are the following:

Further types of questions: Wh-questions. One obvious next step in developing our approach would be an extension to more complex types of questions beyond propositional ones. The next step would be to study objectual or wh-questions in our framework. This will require an extension to a modal setting for first order logic provides more expressive power but is undecidable. However there are other possible extension which could consider various well behaved subsystems: like the fragment with monadic predicates, alternation free fragments, guarded fragments or query languages with safe recursion.

Questions, decisions and information. Another connection, with both information theory and decision theory, defines the 'value of a question' as its role in resolving decision problems by means of the information contained in the available answers. These aspects were introduced and discussed, for instance, in

[113, 112, 111]. This connection can become even more relevant in a setting that makes explicit use of both an information partition and issue-structure for the answers considered relevant by an agent.

Quantitative approaches. Another interesting direction for future research is the relevance of questioning actions to notions of entropy of questions in multi-agent contexts [113, 112, 110, 111] when this measure does not presuppose that all information is always available to be directly accessed. We have considered contexts in which agents have conflicting epistemic interests and incentive to deceive or withhold information. Therefore, it is for such reasons desirable to have a notion of entropic value of a question that can serve as a reference for designing questioning mechanisms that capture both epistemic and strategic aspects.

There are still more fundamental limits to the information that can become available by questioning. In quantum physics experiments there is a fundamental limitation with regard to the information that can be obtained by questions and measurements. A more general notion of entropy that considers such informational limits might be also relevant for such contexts. The ultimate test of the value of these definitions has to be assessed by weather they arise as natural answers to a number of useful questions about questioning scenarios in interactive situations and weather they are useful generalizations that also capture traditional notions as particular cases.

Further agent attitudes: beliefs and preferences. We have studied the interaction of questions with knowledge. But of course, agents' *beliefs* are just as important, and we can also merge the preceding analysis with dynamic logics of belief change. In fact, in addition to conveying hard information, asking a question can also be a subtle way of influencing beliefs of agents. For instance, we said earlier that not all questions impart knowledge that the speakers does not know the answer. But we might say that, barring further information, they induce a defeasible belief of the audience that this is the case. Thus, our question dynamics might be added to the *DEL*-style belief logics of [10, 96].

Beyond beliefs, questions can also affect other agent attitudes. For instance, a question can give us information about other agents' goals and *preferences*, and indeed, "Why" questions explicitly concern such reasons for behavior. Just as information dynamics does not stop at purely informational attitudes, but also extends to the way in which agents evaluate situations and actions, the same extension makes sense for questions. This would come out concretely by adding question dynamics to the preference logics of [33, 66].

There are also formal analogies between our question update operation and the 'ceteris paribus' preferences from [98].

Update, inference, and syntactic awareness dynamics. While DEL has been largely about observation-based semantic information, some recent proposals have extended it to include more finely grained information produced by inference or introspection. One can interpret the effect of asking a question as making agents aware that something is an issue. Raising an issue makes agents aware that some proposition is important. In that case, we can think of a finer dynamics of questions, where they increase some current set of 'relevant propositions' whose truth value needs to be determined. This would work well in the syntactic approach to inferential and other fine-grained information in [115, 114, 95], with questions providing one reason for their acts of 'awareness promotion'. The latter take would also fit well with Hintikka's emphasis on the combination of questions and deductions as driving inquiry. In a dynamic perspective, merging semantic observational information and inferential syntactic information will become even more natural when questions come into play. Instead of using the phrase "Observation enables realization" we can consider a weaker slogan "Questioning and observation enables partial realization": $[p?][p \to q!]\langle +q \rangle Iq$ With these preconditions, a formula can be promoted into the access set even if it was not yet announced. This would also work in the setting that uses both formulas and rules of inference. If some premises of a rule have been announced already and the rest are already part of the current issue, the rule can be applied.

Structured issues and agenda dynamics. Surely, both in conversation and in general investigation, the *agenda* of relevant issues is much more delicate than just some equivalence relation, see also [26, 31, 72]. The primary fact seems to be rather that we are usually maintaining a 'structured agenda' – and it is this agenda that gets modified by successive events of either resolving old questions, or raising new ones. If we are to have any realistic logical account of, say, the development of research programs, we need to understand this more finely-grained dynamics. Moreover, there are already models that allow for this sort of dynamics. Both [33, 66] consider, essentially, 'priority graphs' of ordered relevant propositions (first proposed and studied in [3]) that can be used for this purpose. Priority graphs can encode a structured family of issues, and they allow for a larger repertoire of inserting or deleting questions. Being good at research seems to imply being able to ask good questions just as much as giving clever answers.

Multi-agent behavior over time. We have already indicated that, just as with assertions, questions make most sense in the context of some longer temporal process of inquiry and discovery. Our study of *protocols* was one step in this direction. Another long-term perspective where this makes eminent sense are *learning* scenarios, where asking successive local questions would be a natural addition to the usual input streams of answers (cf. [56]) contributing to one unchanging grand question which global hypothesis about the actual history is the correct one.

Bibliography

[1] Thomas Ågotnes, Johan van Benthem, Hans van Ditmarsch, and Ştefan Minică. Question-Answer Games. In *LOFT, Toulouse, France*, 2010.

[2] Thomas Ågotnes and Hans van Ditmarsch. What will they say?–public announcement games. In *Logic, Game Theory and Social Choice 6, Tsukuba, Japan*, 2009.

[3] H. Andréka, M. Ryan, and P.Y. Schobbens. Operators and laws for combining preference relations. *Journal of logic and computation*, 12(1):13, 2002.

[4] Krzysztof R. Apt. *Strategic Games.* ILLC, University of Amsterdam, 2010.

[5] L. Aqvist. A new approach to the logical theory of questions. *Tübingen*, 1965.

[6] Xabier Arrazola and Maria Ponte, editors. *Proceedings of the Second ILCLI International Workshop on Logic and Philosophy of Knowledge, Communication and Action, LoKCA-10, Donostia - San Sebastian, Spain, November 3-5.* University of the Basque Country Press, 2010.

[7] R.J. Aumann. Agreeing to disagree. *The annals of statistics*, 4(6):1236–1239, 1976.

[8] A. Baltag. Logics for insecure communication. In *Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, pages 111–121. Morgan Kaufmann Publishers Inc., 2001.

[9] Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of the 7th conference on Theoretical aspects of rationality and knowledge*, TARK '98, pages 43–56, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[10] Alexandru Baltag and Sonja Smets. From conditional probability to the logic of doxastic actions. In *Proceedings of the 11th conference on Theoretical aspects of rationality and knowledge*, pages 52–61. ACM, 2007.

[11] Y. Bar-Hillel and R. Carnap. Semantic information. *The British Journal for the Philosophy of Science*, 4(14):147–157, 1953.

[12] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational complexity*, 2(2):97–110, 1992.

[13] N.D. Belnap, T.B. Steel, and H. Schleichert. *The logic of questions and answers*, volume 20. Yale University Press London, 1976.

[14] R. Blutner. Questions and answers in an orthoalgebraic approach. 2009.

[15] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Logical foundations of peer-to-peer data integration. In *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*, pages 241–251, 2004.

[16] L. Carter, L. Stockmeyer, and M. Wegman. The complexity of backtrack searches. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 449–457. ACM, 1985.

[17] I. Ciardelli, J. Groenendijk, and F. Roelofsen. Might and free choice in inquisitive semantics. In *Proceedings of Semantics and Linguistic Theory*, 2009.

[18] I. Ciardelli and F. Roelofsen. Inquisitive logic. *Journal of Philosophical Logic*, pages 1–40, 2009.

[19] José A. Moinhos Cordeiro, Boris Shishkov, Alexander Verbraeck, and Markus Helfert, editors. *CSEDU 2010 - Proceedings of the Second International Conference on Computer Supported Education, Valencia, Spain, April 7-10, 2010 - Volume 2*. INSTICC Press, 2010.

[20] R.T. Cox. Of inference and inquiry, an essay in inductive logic. *The maximum entropy formalism*, pages 119–167, 1978.

[21] C. Daskalakis, P.W. Goldberg, and C.H. Papadimitriou. The complexity of computing a nash equilibrium. *Communications of the ACM*, 52(2):89–97, 2009.

[22] Cédric Dégremont. *The Temporal Mind - Observations on the logic of belief change in interactive systems*. PhD thesis, ILLC, University of Amsterdam, 2010.

[23] L. Demey. Agreeing to Disagree in Probabilistic Dynamic Epistemic Logic. Master's thesis, 2010.

[24] H. Ditmarsch, W. Hoek, and B. Kooi. *Dynamic epistemic logic*. Springer, 2007.

[25] K. Doets and J. van Eijck. *The Haskell Road to Logic, Math and Programming*. Citeseer, 2004.

[26] S. Enqvist. Interrogative belief revision in modal logic. *Journal of philosophical logic*, 38(5):527–548, 2009.

[27] M. Erwig and S. Kollmansberger. FUNCTIONAL PEARLS: Probabilistic functional programming in Haskell. *Journal of Functional Programming*, 16(01):21–34, 2006.

[28] R. Fagin, J.Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities* 1. *Information and computation*, 87(1-2):78–128, 1990.

[29] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A.A. Kalyanpur, A. Lally, J.W. Murdock, E. Nyberg, J. Prager, et al. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79, 2010.

[30] M.M. Flood. Mastermind strategy. *Journal of Recreational Mathematics*, 18(3):194–202, 1985.

[31] E. Genot. The best of all possibleworlds: Where interrogative games meet research agendas. *Belief Revision meets Philosophy of Science*, pages 225–252, 2011.

[32] E.J. Genot. The game of inquiry: the interrogative approach to inquiry and belief revision theory. *Synthese*, 171(2):271–289, 2009.

[33] Patrick Girard. *Modal Logic for Belief and Preference Change*. PhD thesis, Stanford University, 2008.

[34] V. Goranko and G. van Drimmelen. Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science*, 353(1-3):93–117, 2006.

[35] Valentin Goranko and Wojciech Jamroga. e-mail communication. ESSLLI, Copenhagen, Denmark, 2010.

[36] G. Gottlob, E. Grädel, and H. Veith. Datalog lite: A deductive query language with linear time model checking. *ACM Transactions on Computational Logic (TOCL)*, 3(1):42–79, 2002.

[37] J. Groenendijk. Questions and answers: Semantics and logic. In *Proceedings of the 2nd CologNET-ElsET Symposium. Questions and Answers: Theoretical and Applied Perspectives*, pages 16–23. OTS, 2003.

[38] J. Groenendijk. Inquisitive semantics: Two possibilities for disjunction. *Logic, Language, and Computation*, pages 80–94, 2009.

[39] J. Groenendijk and F. Roelofsen. Inquisitive semantics and pragmatics. In *Proceedings of SPR*, volume 9. Citeseer, 2009.

[40] Jeroen Groenendijk and Martin Stokhof. *Questions*, chapter 19, pages 1055–1125. In van Benthem and ter Meulen [101], 1997.

[41] Y. Hamami. The Interrogative Model of Inquiry meets Dynamic Epistemic Logics. Master's thesis, 2010.

[42] C.L. Hamblin. Questions aren't statements. *Philosophy of Science*, 30(1):62–63, 1963.

[43] D. Harrah. On completeness in the logic of questions. *American Philosophical Quarterly*, 6(2):158–164, 1969.

[44] D. Harrah. The logic of questions. *Handbook of philosophical logic*, 2:715–764, 1984.

[45] J.C. Harsanyi. Games with incomplete information played by "bayesian" players, i-iii. part i. the basic model. *Management science*, 14(3):159–182, 1967.

[46] S. Hart, A. Heifetz, and D. Samet. "Knowing whether", "knowing that", and the cardinality of state spaces. *Journal of economic theory*, 70(1):249–256, 1996.

[47] Xiangdong He, John F. Horty, and Eric Pacuit, editors. *Proceedings of Second International Workshop on Logic, Rationality, and Interaction, LORI-II, Chongqing, China, October 8-11, 2009*, volume 5834 of *Lecture Notes in Computer Science*. Springer, 2009.

[48] J. Hintikka. *Socratic epistemology: explorations of knowledge-seeking by questioning.* Cambridge Univ Pr, 2007.

[49] J. Hintikka, I. Halonen, and A. Mutanen. Interrogative logic as a general theory of reasoning. *Studies in Logic and Practical Reasoning*, 1:295–337, 2002.

[50] Jaakko Hintikka and Gabriel Sandu. *Game-theoretical semantics*, chapter 6, pages 361–410. In van Benthem and ter Meulen [101], 1997.

[51] Tomohiro Hoshi. *Epistemic Dynamics and Protocol Information.* PhD thesis, Stanford University, 2009.

[52] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis.* MIT Press, 2006.

[53] W. Jamroga and T. Ågotnes. Constructive knowledge: what agents can achieve under imperfect information. *Journal of Applied Non-Classical Logics*, 17(4):423–475, 2007.

[54] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2):185–219, 2004.

[55] Simon Jones. Haskell 1998 language report, available online at: `http://www.haskell.org/onlinereport`.

[56] Kevin T. Kelly. *The Logic of Reliable Inquiry.* Oxford University Press, USA, 1996.

[57] Kevin T. Kelly and C. Glymour. Convergence to the truth and nothing but the truth. *Philosophy of Science*, 56(2):185–220, 1989.

[58] K.H. Knuth. Intelligent machines in the 21st century: foundations of inference and inquiry. *Phil. Trans. Roy. Soc. Lond. A*, 361(1813):2859–2873, 2003.

[59] K.H. Knuth. Lattice duality: The origin of probability and entropy. *Neurocomputing*, 67:245–274, 2005.

[60] K.H. Knuth. Toward question-asking machines: the logic of questions and the inquiry calculus. In *10th International Workshop on Artificial Intelligence and Statistics, Barbados.* Citeseer, 2005.

[61] Phokion Kolaitis. Relational Databases, Logic, and Complexity. In *Sino-European Winter School in Logic, Language and Computation, Guangzhou, China*, 2010.

[62] B. Kooi. Yet another Mastermind strategy. *ICGA Journal*, 28(1):13–20, 2005.

[63] B.P. Kooi. Probabilistic dynamic epistemic logic. *Journal of Logic, Language and Information*, 12(4):381–408, 2003.

[64] S. Kraus and D. Lehmann. Knowledge, belief and time. *Theoretical Computer Science*, 58(1-3):155–174, 1988.

[65] T. Kubinski. The logic of questions. *La Philosophie contemporaine, Florence: La Nuova Italia Editrice*, pages 185–189, 1968.

[66] Fenrong Liu. *Changing for the Better: Preference Dynamics and Agent Diversity.* PhD thesis, ILLC, University of Amsterdam, 2008.

[67] Carsten Lutz. Complexity and succinctness of public announcement logic. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pages 137–143, New York, NY, USA, 2006. ACM.

[68] Simon Marlow. Haskell 2010 language report, available online at: `http://www.haskell.org/onlinereport/haskell2010/`.

[69] Ştefan Minică. Characterizing and Computing Pure Nash Equilibria in the Location Game on a Line. SELLC, Sino-European Winter School in Logic, Language and Computation, Guangzhou, China, 3-18 December, 2010.

[70] Ştefan Minică. Extensive Questioning Games. In Arrazola and Ponte [6], pages 315–331.

[71] Ştefan Minică. Implementing Dynamic Epistemic Questioning. In Cordeiro et al. [19], pages 97–105.

[72] E.J. Olsson and D. Westlund. On the role of the research agenda in epistemic change. *Erkenntnis*, 65(2):165–183, 2006.

[73] M.J. Osborne and A. Rubinstein. *A course in game theory.* The MIT press, 1994.

[74] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *Society for Industrial and Applied Mathematics Journal of Computing*, 16(6):937–989, 1987.

[75] Z. Pawlak. *Rough sets: Theoretical aspects of reasoning about data.* Springer, 1991.

[76] M. Peliš and O. Majer. Logic of questions from the viewpoint of dynamic epistemic logic. *The Logica Yearbook*, pages 157–172, 2009.

[77] A. Pietarinen and G. Sandu. Games in philosophical logic. *Nordic Journal of Philosophical Logic, Epistemilogy, and the Unity of Science*, 4:143–174, 1999.

[78] A.V. Pietarinen and G. Sandu. If logic, game-theoretical semantics and the philosophy of science. *Logic, Epistemilogy, and the Unity of Science*, pages 105–138, 2004.

[79] J. Plaza. Logics of public communications. *Synthese*, 158(2):165–179, 2007.

[80] S. Rahman and T. Tulenheimo. From games to dialogues and back: towards a general frame for validity. *Games: Unifying Logic, Language and Philosophy; Logic, Epistemilogy, and the Unity of Science*, 15:153–208, 2006.

[81] Oded Shmueli. Decidability and expressiveness of logic queries. In *PODS*, pages 237–249. ACM, 1987.

[82] A. Sureka and P.R. Wurman. Using tabu best-response search to find pure strategy Nash equilibria in normal form games. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1023–1029. ACM, 2005.

[83] B. ten Cate and C. Shan. Question answering: From partitions to Prolog. *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 75–116, 2002.

[84] B. ten Cate and C. Shan. Axiomatizing Groenendijks Logic of Interrogation. *Questions in dynamic semantics*, pages 63–82, 2007.

[85] Balder D. ten Cate. *Model Theory for Extended Modal Languages*. PhD thesis, ILLC, University of Amsterdam, 2005.

[86] The inquisitive website. `sites.google.com/site/inquisitivesemantics`.

[87] A.M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[88] C. Unger and G. Giorgolo. Interrogation in dynamic epistemic logic. *Proceedings of the 13th ESSLLI Student Session*, pages 195–202, 2008.

[89] J. van Benthem. Reflections on epistemic logic. *Logique & Analyse*, 133(134):5–14, 1991.

[90] J. van Benthem. Games in Dynamic-Epistemic Logic. *Bulletin of Economic Research*, 53(4):219–248, 2001.

[91] J. van Benthem. *One is a lonely number: on the logic of communication*. Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 2003.

[92] J. van Benthem. The information in intuitionistic logic. *Synthese*, 167(2):251–270, 2009.

[93] J. van Benthem. *Logical dynamics of information and interaction*. Cambridge University Press, 2011.

[94] J. van Benthem, J. Gerbrandy, and B. Kooi. Dynamic Update with Probabilities. *Studia Logica*, 93(1):67–96, 2009.

[95] J. van Benthem and F.R. Velázquez-Quesada. Inference, promotion, and the dynamics of awareness. *ILLC Amsterdam. To appear in Knowledge, Rationality and Action*, 2009.

[96] Johan van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-Classical Logics*, 17(2):129–155, 2007.

[97] Johan van Benthem, Jelle Gerbrandy, Tomohiro Hoshi, and Eric Pacuit. Merging frameworks for interaction. *Journal of Philosophical Logic*, 38:491–526, 2009.

[98] Johan van Benthem, Patrick Girard, and Olivier Roy. Everything else being equal: A modal logic for *ceteris paribus* preferences. *Journal of Philosophical Logic*, (38):83–125, 2009.

[99] Johan van Benthem and Fenrong Liu. Dynamic logic of preference upgrade. *Journal of Applied Non-Classical Logics*, 14(2), 2004.

[100] Johan van Benthem and Ştefan Minică. Toward a Dynamic Logic of Questions. In He et al. [47], pages 27–41.

[101] Johan van Benthem and Alice ter Meulen, editors. *Handbook of Logic and Language.* The MIT Press, Elsevier, 1997.

[102] H. van Ditmarsch. Knowledge games. *Bulletin of Economic Research*, 53(4):249–273, 2001.

[103] H. van Ditmarsch and B. Kooi. The secret of my success. *Synthese*, 153(2):339–339, 2006.

[104] Hans van Ditmarsch and Jan van Eijck. Dynamic-Epistemic Logic. Course Notes and Haskell Code, ESSLLI, Hamburg, August 2008: `http://homepages.cwi.nl/~jve/courses/esslli08/`.

[105] J. van Eijck and C. Unger. The epistemics of presupposition projection. In *Proceedings of the Sixteenth Amsterdam Colloquium*, pages 235–240. Citeseer, 2007.

[106] J. van Eijck and C. Unger. *Computational semantics with functional programming.* Cambridge University Press, 2010.

[107] Jan van Eijck. Dynamic epistemic modelling. *manuscript, CWI, Amsterdam*, 2005.

[108] Jan van Eijck. DEMO – a demo of epistemic modelling. In *Interactive Logic. Selected Papers from the 7th Augustus de Morgan Workshop, London*, volume 1, pages 303–362, 2007.

[109] Jan van Eijck, Lakshmanan Kuppusamy, and Floor Sietsma. Demo light for composing models. `http://www.cwi.nl/~jve/software/demolight`, 2011.

[110] R. van Rooij. Quality and quantity of information exchange. *Journal of Logic, Language and Information*, 12(4):423–451, 2003.

[111] Robert van Rooij. Questioning to resolve decision problems. *Linguistics and Philosophy*, 26(6):727–763, 2003.

[112] Robert van Rooij. Questions and relevance. In *Questions and Answers, Proceedings 2d CoLogNET ElsNET Symposium, ILLC Amsterdam*, pages 96–107, 2005.

[113] Robert van Rooij. Comparing questions and answers: A bit of logic, a bit of language, and some bits of information. *Formal Theories of Information*, pages 161–192, 2009.

[114] Fernando Velázquez. *Small Steps in Dynamics of Information.* PhD thesis, Institute for Logic, Language and Information, University of Amsterdam, 2010.

[115] F.R. Velázquez-Quesada. Inference and update. *Synthese*, 169(2):283–300, 2009.

[116] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *Proceedings of the National Conference on Artificial Intelligence*, pages 345–351. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.

[117] A. Wiśniewski. *The posing of questions: logical foundations of erotetic inferences.* Kluwer Academic Pub, 1995.

[118] A. Wiśniewski. Erotetic search scenarios. *Synthese*, 134(3):389–427, 2003.

[119] A. Wiśniewski. Erotetic search scenarios, problem solving, and deduction. *Logique & Analyse*, 185-188:139–166, 2004.

Abstract

The dissertation presents an approach to questions and broader questioning phenomena inside the paradigm of dynamic epistemic logic. The main topics that emerge in the thesis are concerned with: defining a rich repertoire of questioning actions, developing a theoretical framework and logics to reason about interrogative actions, provide implementation tools for analyzing realistic scenarios of questioning in inquiry. In this way we provide a setting in which questions can be understood and analyzed in their intricate conceptual, logical and practical interdependence and strategic connection with knowledge and information dynamics.

Chapter 2 shows how dynamic logics of questions can analyze various aspects of private and public inquiry. We introduce a resolution modality based on the intersection between the two equivalence relations representing the information and questioning partitions. This provides a rich system of dynamic issue-management actions leading to complete dynamic logics for questioning in DEL style. These systems fit entirely within the methodology of dynamic-epistemic logic, and they seem to form a natural complement to what already exists in this area.

Chapter 3 complements the theoretical perspective with an implementation. This extends previous implementations for epistemic model checking with questioning specific functionality. The main new utilities are model checking utility for questioning, resolution, and epistemic formulae, and a general and extensible implementation for complex questioning and resolution dynamic actions that emerge in this framework. We also show how the implementation is useful by modeling and analyzing some paradigmatic examples of questioning scenarios.

Next we study the strategic aspects that emerge in DELQ both in games with questioning moves and in long term interactive inquiry.

Chapter 4 defines and investigates games with questioning moves. We first look at strategic games with two players. We then extend this basic approach to a more realistic setting with sequential moves, and oracles encoding interactions between imperfectly informed agents or limitations in external information sources or measurement instruments and experimental procedures. We give a tripartite interpretation for questioning moves and use it to show first why questioning phenomena in multi-agent contexts are more complex than traditionally understood and second to distinguish games with questioning moves from games with informative actions. We analyze illustrative examples and present an inexistence result for Nash equilibrium with pure strategies in questioning games with oracles. We also identify the difraction property and show why it is important for describing strategic abilities in extensive questioning games.

Chapter 5 provides an implementation for questioning games extending basic epistemic functionality to include strategic aspects specific for a game theoretic approach of questioning actions. The implementation provides 'expressive harmony' for questioning moves by linking the semantic level based on partitions of the domain with a corresponding syntactic level using disjunctions of nominals. In this way, we can compute complete game matrices for games with questioning moves and we give an algorithm for minimizing issue-epistemic models using a notion of behavioral equivalence that is adequate for the questioning language.

Next we investigate the connection between a questioning theory and the process of designing efficient querying strategies. In this way we provide a bridge between a theory of questions and known search heuristics using backtrack oracles.

Chapter 6 approaches the topic of designing questioning strategies in problem solving from a theoretical perspective. We take again solving games as our point of departure and a rich representative test case. In this context we investigate the problem of finding Nash equilibria in the location game played on a line. We also discuss the general relevance of this approach for designing querying strategies in problem solving by using oracles of operational properties to solve a principal problem using efficiently available sources of information. We give a characterization of NE by means of local properties in the game. We use an approach based on querying an oracle of local properties and matching of strategy profile fragments to design questioning strategies that solve the game in an efficient way. In the final section we provide a minimization algorithm for probabilistic issue models based on partition refinement solving the birelational coarsest partition problem and in accordance with the adequate notion of behavioral equivalence for probabilistic issue models and probabilistic questioning actions.

In Chapter 7 we present a *Haskell* implementation illustrating how queries of local properties in the game can search for equilibrium strategy profiles using list comprehension. This assumes the existence of oracles of local properties and uses this to search for Nash equilibria in pure strategies. Next, we give an *Alloy Analyzer* implementation for building countermodels for the location game and checking assertions about logical entailment within a predetermined scope between facts expressing local properties and NE.

Chapter 8 shows how our approach gives rise to a coherent research agenda with a broader scope, and points to some further directions for future research.

Acknowledgments

There are many people I wish to acknowledge for their direct or indirect contribution to this thesis or for making this text possible at all.

First of all, I am very grateful to my three supervisors who guided and shaped this thesis in many ways. I thank Johan van Benthem for interesting discussions, extended and useful comments on various draft manuscripts, long emails full of insightful thoughts and much appreciated clarifications. He was a permanent source of fresh and stimulating ideas while also providing constant support, knowledgeable supervision and active encouragement for articulating and developing my own ideas. His generous willingness to share his extensive knowledge was the reason to start my research in DEL in the first place and instrumental for the attempt to enrich it with an interrogative twist in this thesis. I can only hope that in the future I will find the time and wisdom to follow up on more of the the many interesting ideas Johan shared with me over the years.

Hans van Ditmarsch and Jan van Eijck joined in the supervision process in the last two years and their dedication and support were pivotal in enhancing my knowledge and fueling my interest. In 2008 at ESSLLI in Hamburg I attended the DEL course taught by Jan and Hans in which they were using Haskell and DEMO. It was during the next coffee break when I knew that my research will never be the same again. Soon after that I discovered myself thinking about DELq in DEMO code and, for better or worse, I kept doing it for the rest of this thesis. I owe to Jan my initial interest in implementations and much of what I learned about them since. His passion for neat code and deep thinking are truly outstanding and inspirational.

I am thankful to Hans for his perceptive and enthusiastic supervision on any PAL or DEL topic I explored. He also shaped my interest in games with epistemic moves. I am indebted to him and Thomas Ågotnes for intellectual engagement and many interesting discussions from which the questioning games emerged. I also greatly appreciate their generosity as co-authors, advice and encouragement, challenging joint work, enduring collaboration and many fruitful ideas.

Next I want to thank Krzysztof Apt for many critical comments and valuable feedback about various subjects in the thesis. I have learned a lot from our interaction over the years, the very topic of Chapter 6 emerged from an exercise in the 'Strategic Games' lecture notes and many of the subsequent developments would not have been possible without his constructive criticism and ineluctable demands for more general and interesting results.

I thank Gabriel Sandu for useful advice and inspiring supervision as a visiting student in Helsinki and also for awakening, perhaps unintended, my interest in interrogative logic for the first time. I also admired from a very early stage Alexandru Baltag's work on integrating questions inside DEL, I thank him for making DEL not just an interesting research field but in so many occasions also a fun and enjoyable subject. I also thank the other members of the thesis committee Paul Egré, Jeroen Groenendijk, Robert van Rooij and Frank Veltman for the honor of agreeing to read and asses the manuscript.

I have also benefited from comments on parts of the text or discussions of some topics in the thesis from many people over the years: Lorenz Demey, Viktoria Denisova, Valentin Goranko, Meiyun Guo, Yacin Hamami, Wesley Holliday, Thomas Icard, Wojtek Jamroga, Lena Kurzen, Minghui Ma, Alexandru Marcoci, Eric Pacuit, Floris Roelofsen, Mehrnoosh Sadrzadeh, Katsuhiko Sano, Sunil Simon, Sonja Smets, Cristina Unger and Fernando Velázquez-Quesada.

ILLC was also a great place for study and broadening my intelectual horizon. I have learned a lot from the excellent teachers whose lectures I attended: Paul Dekker, Peter van Emde Boas, Ulle Endriss, Dick de Jongh, Benedikt Löwe, Sebastiaan Terwijn, Jouko Väänänen and Yde Venema. I also thank George Barmpalias and Jan Jaspars for interesting and useful teaching activities.

I thank the dynamic seminar co-organizers and the seminar yearbook coeditors: Cédric Dégremont, Davide Grossi, Alexandru Marcoci, Sonja Smets, Ben Rodenhäuser, and Fernando Velázquez-Quesada and all the other members in the dynamic logic group who were always a stimulating and enjoyable presence. I thank Lena Kurzen for sharing an office in the JK building and so many others for sharing the workspace in Science Park: Pietro Galliani, Umberto Grandi, Joel Uckelman, Lucian Zagan, however, any list would be inevitably incomplete. The entire ILLC community has made living and working enjoyable and interesting.

I thank the ILLC office for help with administrative and organizational issues: Karin Gigengack, Tanja Kassenaar, Ingrid van Loon, Marco Vervoort, and I am grateful to Peter van Ormondt also for helping with the dutch samenvatting.

Finally, I thank my family for their invaluable support over the years and Camelia for always finding a way to make life so much more meaningful.

Amsterdam
October, 2011.

Ştefan Minică