A Strategic Analysis of Multi-agent Protocols

Sieuwert van Otterloo

A Strategic Analysis of Multi-agent Protocols

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

A Strategic Analysis of Multi-agent Protocols

Thesis submitted in accordance with the requirements
of the University of Liverpool for the degree of Doctor in
Philosophy

by

Sieuwert Maarten van Otterloo

Defended November 25 2005

Promotors: Prof. Dr. Wiebe van der Hoek
Prof. Dr. Michael Wooldridge
University of Liverpool
Department of Computer Science
Liverpool L69 7ZF
United Kingdom

Co-promotor: Prof. Dr. Johan van Benthem
Institute for Logic Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
The Netherlands

Contents

Acknowledgments

This dissertation has benefited enormously from comments, suggestions and discussions with other people, and on these pages I would like to mention as many of these people as I can remember, in a more or less arbitrary order.

Since I started my PhD by moving to Liverpool, it seems appropriate to start with the people from this city. I am very grateful to my supervisors, Wiebe van der Hoek and Mike Wooldridge. A special mention should go to Peter McBurney, who was not only my advisor but has also encouraged and stimulated me continuously. I would like to thank many colleagues who have helped me to find my way in Liverpool and in research: Hanno Hildmann, Steve Phelps, Dirk Walther, Ian Blacoe, Laera Loredana, Rafael Bordini, Carmen Pardavilla and of course my office mates Benjamin Hirsch, Justin Wang, Qin Xin, Claudia Nalon, Mari Carmen Fernandez-Gago, Mark Roberts and Nivea de Carvalho Ferreira. I feel greatly indebted to the many people that contributed to the great atmosphere in the department of Computer Science, including Thelma Williams, Ken Chan and the other members of the technical staff, Katie Atkinson, Trevor Bench-Capon, Aiman Badri, Celia Casado-Castilla, Alison Chorley, Andrea Kam, Christian Guensel, Catherine Atherton, Shaheen Fatima, Tim Miller and Adele Maggs. I would like to thank Frank Wolter and Mark Ryan for acting as examiners in my PhD defense.

In Amsterdam I have benefited from support and encouragement from many sources, and from a great research atmosphere. First of all I would like to give a special mention to Johan van Benthem for his critical and stimulating comments, and for giving me the opportunity to work in the ILLC, and my office mates Aline Honingh, Merlijn Sevenster and Yoav Seginer. I would like to thank Olivier Roy, Fenrong Liu, Clemens Kupke, Nick Bezhanishvili, Jelle Zuidema, Reut Tsarfaty, Maricarmen Martinez Baldares, Balder ten Cate, Ulle Endriss, Joost Joosten, Chiaki Ohkura, Scott Grimm and Eric Pacuit for many pleasant conversations, not only during lunch time. I wish to thank Peter van Emde Boas, Marjan Veldhuizen, Igrid van Loon, Rens Bod, Benedikt Löwe and Robert van Rooij for

Chapter 1

Introduction

1.1 Multi-agent Protocols

Multi-agent protocols are sets of rules that specify how agents can interact with each other. For example, an auction has strict bidding rules and is thus an example of a protocol. Elections form another example. These activities have in common that they can be done in real life, without use of computers or networks. However, one can also imagine auctions and elections in which computer programs participate, perhaps even in competition with humans. These examples of multi-agent protocols are already widely used in all kinds of settings. The next examples illustrate situations in which multi-agent protocols are useful.

- Everyone is familiar with the problem of dividing a cake fairly among several people. Assume that a round birthday cake is to be shared fairly among a certain number of guests. If 'fair' means into equally sized and shaped parts and one can use ruler and compass, then this becomes a mathematical problem. One can also see this situation as a social science problem, by using another notion of fairness. One can require *envy-freeness*, which means that nobody should be envious of somebody else. Everyone should judge his or her own piece at least as good as the other pieces. This is achievable for two agents by using so-called 'cut and choose' protocols: one agent splits the cake into two parts, the other agent chooses who gets which part. The fact that both agents play an active role in this protocol makes it easier for agents to accept this protocol. What makes the protocol fair is the fact that the cutter has an incentive to cut as fair as possible. In case of larger sets of agents, more elaborate procedures exist [17].

- Suppose you want to raise money for a good cause, and a sponsor has given you a car in order to help you to do so. Should you auction this car, or start a lottery? A lottery is the traditional way to raise money, at least in The Netherlands. However, auctions of different types have become increasingly

popular in other domains (consider eBay, or the distribution of mobile phone network licenses), and a lot is known about the good theoretical properties of auctions [63]. Recent work by Goeree and others [39] indicates that a lottery, despite being random, is a better way to raise money than an auction. The authors do not only give mathematical arguments. In one example where parents are asked to donate money to their children's school, they note that social arguments also play a role:

> "Some parents may be offended when told they contributed nothing because they lost the auction, or, in other words, because their contributions were not high enough." [39, p.3]

- On auction websites such as eBay, many buyers prefer to place their bids at the very last moments. This is called *sniping*. These snipers choose to bid in the last minute even when the auction lasts a week [76]. Possibly they do this in order to avoid bidding wars with other bidders, who in the face of competition want to spend more money than they originally planned. Another explanation is that the snipers somehow enjoy to surprisingly outbid other people. Either way, some bidders refuse to behave as prescribed by auction-theory textbooks. To some bidders and sellers sniping seems unfair, but others feel it is justified by the amount of pleasure they derive from it.

  > "A lot of people who do not snipe feel it is unfair, but it happens to be my absolute favorite way to win at auctions." *Marcia Collier* [38]

The need to understand multi-agent protocols is growing, because these protocols are used in new and possibly surprising environments. What worked well in real life may work differently on the Internet or with computer programs as participants. The stakes are also getting higher. The Internet is not only used for buying low cost commodities such as books, but also for flights, cars and houses. A new research field focused on understanding and designing protocols, sometimes called 'social software' [82], is therefore emerging.

Agents

It is difficult to come up with a universally accepted definition of the word *agent* [124]. Nevertheless, every researcher should know the meaning of the words he or she uses, or risk talking nonsense. I use the word *agent* to mean a decision making entity, and thus I accept humans, computer programs or even organisations as agents. The word has been used in this way within the English language for centuries, for instance in the following quotation.

> "Nor are we to be meer instruments moved by the will of those in authority..but are morall Agents." *Samuel Bolton, 1646* [80]
> (original spelling)

In a more recent tradition, agents are seen as software programs with artificial intelligence-like abilities and properties, such as reflexes, mobility, intelligence and emotions. The English science-fiction writer Douglas Adams for instance describes how agents might function when a spaceship tries to recover from a meteorite hit.

> Small modules of software – agents - surged through the logical pathways, grouping, consulting, re-grouping. They quickly established that the ship's memory, all the way back to its central mission module, was in tatters. [2]

I am not concerned with designing or dissecting actual agents, but with *protocols* for agents. Thus, where other researchers see the construction of agents, or even 'intelligent agents', as a long term, yet unattained research goal, in my view there are already agents and protocols. The focus is not on the internal workings of these individual agents, but on the way protocols function when used by agents. Since protocols are formal objects, they can be studied formally, without experiments or empirical investigations.

Protocols

A protocol is different from an algorithm because it gives agents a choice of actions. Agents have the freedom to bid whatever they think an item is worth, or to vote for whatever they think is best. All agents together determine what the outcome of the protocol is. In an ideal world, the protocol allows all participants to reach an outcome that is 'optimal' in some sense: the protocol should be fair, efficient, democratic, or otherwise meet some expectation. There are often many different protocols for a certain problem. One can for instance sell a house by asking all interested parties to submit a bid in a closed envelop, open all envelops at the same time, and sell the house to the highest bidder. Alternatively, one could have an open-cry auction, in which bidding continues until no agent wants to bid higher than the current bid. Another option would be to have a lottery, or an essay contest. It is often not immediately obvious which protocol is best. Selecting the best protocol for a certain task is thus a relevant and sometimes difficult problem.

Traditional versus Computational Approaches

What we call multi-agent protocols has been already studied under different names by other disciplines than computer science. For example, economists and

game theorists have studied the properties of auctions [63]. Social choice scientists have been working on voting protocols and fair division protocols [17]. These related fields also have their own traditional applications. Fair division protocols are can be used for cake cutting, but also for divorce settlements [17]. Economists' examples are often more trade-oriented or money-oriented. Thus, an opportunity exists here to take results and insights from computer science, and use them to get improvements in applications outside the traditional scope of computer science. We hope that knowledge about multi-agent protocols can be used to design better solutions for the example scenarios described in this chapter.

One can distinguish 'traditional' approaches from computer science (and AI) approaches by the fact that the computer scientists emphasise computational properties. In the traditional approach one determines whether some solution to a protocol problem exists. The computer scientists are also interested in the question whether something can be computed efficiently. This emphasis on computation can lead to interesting insights. Even though it has been proven that no voting scheme is completely immune to manipulation [36], one can show that in certain schemes it is very hard to compute how one should manipulate the voting in order to get a required outcome [24]. In these schemes, it is unlikely that someone can manipulate an election.

In this dissertation computational arguments are also used, but in a different way. By comparing the complexity of different protocol verification problems, one can determine what kind of goals are hard for agents to achieve, and which properties are hard to verify. This leads to more insight into the causes of the difficulty of protocol verification of design. For instance it is often assumed that the interaction between agents makes games and protocols difficult. Sometimes however also models with only one agent can have interesting computational properties, which is a surprising result.

Logical Approach

In the examples of cake-cutting and charity auctions, it has become clear that the properties that one wants protocols to have can be very diverse. In the second example, parents actually wanted to contribute, while in the first example fairness meant that agents would not choose to swap pieces. In some cases, for example money-based auctions, one can reduce the quest for the right protocol to a numerical problem: the problem is reduced to computing the optimal parameters, or finding the right side payments. If this is not possible, for instance because money is not available in the protocol, one must capture these properties in some other precise way, before one can test protocols for these properties. Logical languages are very suitable for this task: one can describe complicated properties in short logical formulas. Logics are also very expressive: one can state both the presence and the absence of certain properties. Different logics are thus used as specification languages: the formulas express what one wants or does not want

from a protocol.

1.2 Protocol Problems

Throughout the dissertation different example problems are used to illustrate the issues at hand. For instance chapter 4 contains several protocols that can be used in the following situation.

> Three agents Alice, Bob and Caroline (or A, B and C) have to select one of the alternatives x, y and z. They are looking for a suitable voting protocol to select exactly one of these three alternatives as the outcome. The protocol should be democratic, so that any majority can enforce any outcome.

In chapter 4, the focus is on what outcomes can be guaranteed by agents and coalitions of agents. Thus, the issue of *effectivity* is studied. In this chapter, we find many solutions for this problem. The logical approach of chapter 4 cannot be used for distinguishing these many solutions. Therefore, in chapter 5 and 6 more expressive logics are studied that can find subtle differences between the different solution to this problem.

The following two problems are more basic than the voting problem stated above, and using the logic from chapter 4 one can again find protocols that solve these problems. In chapters 4 and 5 it is shown how solutions for these problems differ from each other.

joint decision problem A decision p is taken if either Alice or Bob think that p should be the case. If both agents do not want p, it should be rejected.

independent decision problem Alice can decide whether a should hold or not, and Bob can decide whether b should hold or not.

Chapter 7 is concerned with the knowledge that agents have at the end of a protocol, and how this knowledge depends on the strategies that are used. One example problem that can be analysed using the techniques from this chapter is the following problem.

> In a TV quiz show the quiz master asks a candidate the following question: Which day of the week comes directly after Tuesday? Is it a) Monday, b) Wednesday, c) Friday or d) Saturday. The candidate has no clue whatsoever about the days of the week, and replies: 'I am not sure. Can I do fifty-fifty?'. The quiz master has to remove two options that are not the answer, so he says: 'The answer is not Monday and neither Friday'. Does the candidate now know the answer?

In chapter 8, agents are concerned about their privacy, and use random strategies in order to hide their preferences.

> Alice needs to buy one box of breakfast cereals every week. Each week she can either buy Allgrain (A), Barley (B) or Cornflakes (C). She likes A better than B and B better than C. However, Alice knows that the shop is watching her shopping behaviour closely, and she does not want the shop to know for sure what her preferences are. Therefore, she buys a different brand every day.

The techniques presented in this chapter allow one to calculate the optimal (random) strategies that agents should use if they are concerned about keeping their preferences private.

1.3   Outline of this Dissertation

This first chapter is a formula-free introduction. The last chapter is also written in plain English, and presents some conclusions. The chapters in the middle have a high density of mathematical notation. The first two 'middle' chapters are introductory.

- Chapter 2 contains definitions in the area of logic. It contains definitions of propositional logic, which is the basic logic of which all other logics are extensions, modal logic and epistemic logic.

- Chapter 3 is a concise introduction and overview of game theory. It defines the concepts of game theory that are used later on.

The remainder of this dissertation, the so-called 'content'-part, describes original research that I have conducted over the last three years. It can be divided roughly in two parts. The first three chapters deal with logics that are interpreted over extensive games of perfect information. In these games it is assumed that agents have perfect information about the current state of the world. They know what other agents have done, but are uncertain about what other agents will do. In order to express properties of such situations, different logics are examined.

- Chapter 4 describes a logic for reasoning about extensive games called EFL. This logic deals with reasoning about whether coalitions of agents can achieve certain goals, without help of the other agents. Game-theoretically this is a simple situation, and one can therefore efficiently check properties in this logic. A complete proof system is also given, together with a procedure to automatically constructs protocols for given properties. The main result of this chapter has been accepted for ESSLLI 2005 [107].

- Chapter 5 introduces logics that are more expressive than EFL. First of all the logic EFLS can be used to express more subtle properties involving side effects of using certain strategies. One way to look at this logic is by saying

that it deals with the situation where agents are initially not aware of the preferences of other agents, a situation that is not normally considered in game theory. This logic can thus be used to express more properties of game forms, but has the same model checking complexity as EFL.

A second logic introduced in this chapter, called EFLS, allows one to reason about agents that want other agents to be able to do something. This is an example of reasoning about preferences on the whole play of the game, not just on the outcome. One can apply this logic to reason about polite agents that want to give other agents the ability to choose. This chapter extends work presented at the AAMAS 2004 conference in New York [113].

- Chapter 6 reasons about game forms and preferences explicitly. It uses a special logic for reasoning about preferences. As an example we study the backward induction solution concept in this chapter, in which agents use their knowledge of each others preferences in order to anticipate each others choices. This chapter is exceptional because, unlike the other chapters of this thesis, it is not the sole work of Sieuwert van Otterloo, but is based on joint, yet unpublished, work done in pleasant cooperation with Olivier Roy and Johan van Benthem at the ILLC in Amsterdam.

These three chapter can be seen as an attempt to understand game forms using more and more precise languages. For the logic EFL many protocols appear to be the same. The next two chapters offer logics that give more detailed views, so that one can discover subtle differences in protocols.

Chapters 7 and 8 deal with the case where agents are not fully aware of all aspects of the current situation. They have imperfect information about certain facts. Since information is very important to agents, they might act in order to get more information. In other situations agents act so that others obtain no information.

- Chapter 7 discusses *knowledge condition games*. In this new type of games, agents act in order to achieve a certain knowledge situation: they want to know that others do not know that something happened. Two variants with different computational complexity are introduced, and some tractable variants are described. The work has been presented incrementally at GTDT in New York [110], at the first Knowledge and Games Workshop in Liverpool [111] and at the European workshop on Multi-Agent Systems [115], and have finally been accepted for publication in the Journal of Logic, Language and Information [114].

- In chapter 8, a similar problem is treated in a different way. We assume that certain agents want to keep their preferences secret. Using techniques from information theory, we determine what strategy agents must use in

order to maximize the uncertainty of an observer. This chapter is based on
a paper presented at the AAMAS 05 conference in Utrecht [108].

A recurrent theme is this dissertation is the idea that agents can have complex
goals in a protocol, and that nondeterministic strategies can be used for achieving
these goals(see for instance the situation on page 89 where Bob makes Alice unable
to decide, or the quiz master example on page 134 where nature does not favour
the candidate). Normally in game theory agents have preferences over outcomes,
and the goal is to achieve a certain outcome. A complex goal on the other
hand depends on the whole game, including properties of the strategies used and
other outcomes than the actual one. For instance in chapter 5 agents care about
whether other agents can achieve outcomes or not. In knowledge condition games
coalitions act in order to make sure certain knowledge is achieved in the end, and
in chapter 8 agents care about how predictable their strategy is.

It is easiest to use strategies that recommend single best actions for any sit-
uation. These strategies are called pure strategies and are often sufficient for
reaching simple goals. In this dissertation we often use nondeterministic strate-
gies. These strategies can recommend multiple actions and are thus potentially
more powerful. In knowledge condition games and the privacy games of chapter 8,
agents can deliberately use these strategies to become unpredictable. In the logics
EFLS and EFLN these nondeterministic strategies are used because in many case
several actions are equally good. One cannot know beforehand which action an
agent will take in this case, so we model this uncertainty using nondeterministic
strategies.

1.4 Conclusion

Multi-agent protocols have not been discovered recently. The term can be used
to describe common situations, such as auctions, voting and cake cutting. These
protocols can be studied from different disciplines, such as game theory, eco-
nomics and social science. Furthermore one can test these protocols for many
different properties, for instance envy-freeness. The different frameworks and
logical languages defined in this dissertation make it possible to formally analyse
these multi-agent protocols, and to test them on many different properties. The
precision of a logical approach makes it possible *in principle* to use the computer
to find the right protocol for any situation. In this dissertation, it is determined
in which cases this is also practical. This is done by looking at the complexity of
these computing problems.

Chapter 2

Logic

2.1 Introduction

Logic is one of the oldest disciplines of science. It has been studied more or less continuously from Aristotle to the present day. For example, it was an important part of the Medieval academic curriculum: together with grammar and rhetoric, logic formed the 'trivium', the relatively simple arts that one should master before one could move on to the more advanced arts of the Quadrivium (arithmetic, astronomy, geometry, and music) [122].

Given the rich history of logic, it is not possible to give a complete overview of the area. The goal of this chapter is merely to provide the necessary definitions of propositional, modal and epistemic logic that will subsequently be used throughout this dissertation. For readers not so familiar with logic, this chapter can serve as a concise introduction. For other readers, this chapter introduces the notational conventions that I use in the remainder of this dissertation. First propositional logic is defined. Then in section 2.3 modal logic and epistemic logic are defined. In section 2.4 we discuss the different ways in which theorem proving, satisfiability and model checking can be used for protocol verification.

A logic typically consists of three elements: the logical language, the semantics and the proof system. A logical language is a set $\mathcal{L}$ of formulas. The semantics is a relation between formulas and models, that says when a formula is true on a model. If a formula ϕ is true on a model M we write $M \models \phi$, otherwise we write $M \not\models \phi$. We are often interested in formulas that are true in any model, and these formulas are called *valid formulas*, *validities* or *tautologies*. In order to indicate that ϕ is a tautology, we write $\models \phi$. If a formula ϕ holds in at least one model, the formula is called *satisfiable*.

The final typical element of a logic is the *proof system*. Such a proof system $\mathcal{S}$ consists of axioms and derivation rules and allows one to formally derive formulas.

2.1.1. DEFINITION. Let $\mathcal{L}$ be a logical language. A proof system $\mathcal{S}$ is a pair $(\mathcal{A}, \mathcal{R})$ where $\mathcal{A} \subseteq \mathcal{L}$ and $\mathcal{R} \subseteq \mathcal{L}^*$.

description	language	proof system
propositional logic	$\mathcal{L}_p$	$\mathcal{S}_p$
standard modal logic	$\mathcal{L}_\square$	$\mathcal{S}_\square$
epistemic logic	$\mathcal{L}_K$	$\mathcal{S}_K$

Figure 2.1: Proof Systems for different logics

The set $\mathcal{A}$ is called the set of axioms of $\mathcal{S}$, and $\mathcal{R}$ is the set of reasoning rules. If a proof system $\mathcal{S}$ proofs a formula ϕ, then we write $\mathcal{S} \vdash \phi$. The notion of proof that we use here is that of a finite list of statements $\mathcal{S} \vdash \phi_1$, $\mathcal{S} \vdash \phi_2$, $\ldots \mathcal{S} \vdash \phi_n$ such that each formula ϕ_i is either an axiom of $\mathcal{S}$, or $(\phi_{(m_1)}, \ldots, \phi_{(m_n)}, \phi_i)$ is a reasoning rule of $\mathcal{S}$, with $m_1, \ldots, m_n < i$. Thus, axioms count as self-evident, and the reasoning rules allow one to derive a formula from formulas proven before.

2.1.2. DEFINITION. Let $\mathcal{L}$ be a a logical language. A proof system $\mathcal{S}$ is *sound* if $\mathcal{S} \vdash \phi$ implies $\models \phi$. It is *complete* if $\models \phi$ implies $\mathcal{S} \vdash \phi$.

All logics in this thesis make use of a symbol $\neg$ for negation of a formula. In such logics, a formula ϕ is called *consistent* if its negation cannot be proven: $\mathcal{S} \not\vdash \neg \phi$. In a complete proof system (for a logic where negation is defined in the classical way), every consistent formula is satisfiable.

Ideally, a proof system should be sound *and* complete. Furthermore, the axioms and rules should not be arbitrary sets, but one should be enumerable in an automatic fashion: a mechanical procedure should be able to generate all axioms. In practice, this means that the sets of axioms and reasoning rules consist of a finite number of patterns, so that any formulas can be inserted in the open places of the pattern. This constraint ensures that one can effectively generate all proofs, which means that there is a procedure to find all proofs and thus all theorems.

Table 2.1 lists the languages and proof systems that are defined in this chapter.

2.2   Propositional Logic

Propositional logic is a logic for reasoning on a sentence level: It explains how complex sentences follow from simple sentences. We assume that there is a set P of basic or atomic propositions. These represent sentences that cannot be broken down in smaller sentences. In the next example this set contains the atomic propositions p and q, that capture 'It rains' and 'The weather is good'. These atomic propositions are combined using logical connectives or operators, which stand for some semantical relation between facts.

sentence	proposition
It rains	p
It does not rain	$\neg p$
The weather is good	q
It rains and the weather is good	$p \wedge q$
It rains or the weather is good	$p \vee q$
If it rains then the weather is good	$p \rightarrow q$
It rains if and only if the weather is good	$p \leftrightarrow q$
It rains if and only if the weather is not good	$p \nabla q$
Contradiction	$\bot$

It is convenient to have so many operators available, so that one can concisely and naturally express complex formula structures. At the same time, it is cumbersome to deal with all the different operators in all theorems and proofs. It is also redundant, because many operators can be expressed in terms of each other. For instance, $p \leftrightarrow q$ is equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$. The common solution to this dilemma is to treat some of these connectives as fundamental, and others as abbreviations for something expressed using the fundamental connectives. In the next subsection, a variant of propositional logic with a minimal set of fundamental operators is presented, and a proof system for this logic is developed. In the following subsection, it is shown in detail how propositional logic with all operators reduces to this language.

2.2.1 Minimal Propositional Logic

The language of minimal propositional logic has a set of basic operators that is minimal in the following sense: Every function from truth values to a truth value can be expressed by composing the basic operators, but none of the basic operators can be expressed as a composition of the other basic operators. One can choose such a minimal set in different ways. Two well-known minimal sets are $\{\vee, \neg\}$ and $\{\wedge, \neg\}$, see for instance [53, p.71]. Our approach is based on the basic operators $\bot$ (the constant 'false' that is never true) and $\rightarrow$ (implication). An argument for this particular choice would be that implication plays an important role in the proof system defined for modal logic.

2.2.1. DEFINITION. Let P be a set of atomic propositions and $p \in P$ an element of P. Minimal propositional logic $\mathcal{L}_p(P)$ consists of formulas ϕ generated by the grammar

$$\phi ::= p \mid \phi \rightarrow \phi \mid \bot$$

Parentheses indicate how certain formulas are constructed, and can be used, for instance, to make a distinction between $(p \rightarrow q) \rightarrow r$ and $p \rightarrow (q \rightarrow r)$. If no parentheses are given then the second reading is intended: $p \rightarrow q \rightarrow r$ should be read as $p \rightarrow (q \rightarrow r)$.

This language is interpreted in the following way. A model M for this logic is a subset of P. The atomic propositions in M are assumed to be true, the ones that are not in M are false. The next definition determines when $M \models \phi$ for any formula ϕ.

2.2.2. DEFINITION. Let $M \subseteq P$ be a model. The satisfaction relation $\models$ for minimal propositional logic is defined recursively by the following three rules:

$$M \models \bot \qquad\qquad\qquad\qquad \text{never}$$
$$M \models p \text{ where } p \in P \qquad\qquad \text{iff } p \in M$$
$$M \models \phi \rightarrow \psi \qquad\qquad\qquad \text{iff } M \models \phi \text{ implies } M \models \psi$$

Like many logics, propositional logic is closed under *uniform substitution*. This means that if one has a valid formula in which p occurs, and one replaces all occurrences of p for any other formula ϕ, one again has a valid formula. For example, since $\phi = p \vee \neg p$ is valid, the formula $\psi = \neg q \vee \neg\neg q$ is also valid. A formula ψ that is obtained from ϕ by uniform substitution is called an *instance* of ϕ.

In the remainder of this section we define a proof system $\mathcal{S}_p$ for the language $\mathcal{L}_p$. The next three formulas serve as the axioms for this proof system.

$$A_1 = \phi \rightarrow (\psi \rightarrow \phi)$$
$$A_2 = (\phi \rightarrow (\psi \rightarrow \xi)) \rightarrow ((\phi \rightarrow \psi) \rightarrow (\phi \rightarrow \xi))$$
$$A_3 = ((\phi \rightarrow \bot) \rightarrow (\psi \rightarrow \bot)) \rightarrow (\psi \rightarrow \phi)$$

We have defined axioms as sets of formulas, and thus A_1 to A_3 are sets of formulas. To be precise, $A_1 = \{\phi \rightarrow (\psi \rightarrow \phi) | \phi, \psi \in \mathcal{L}_p\}$, but it is hoped that the notation without set brackets is more readable. We can write $\phi \in A_i$ to indicate that ϕ has the stated form. If $\phi \in A_i$ we say that ϕ is an instance of the axiom scheme A_i.

Suppose that $\mathcal{L}$ is a logical language in which $\rightarrow$ has its usual interpretation. If both ϕ and $\phi \rightarrow \psi$ are validities in this logic, then ψ must be a validity as well. This fact forms the basis of the reasoning rule *Modus Ponens*. The set $MP_\mathcal{L}$ that expresses this rule is the following.

$$MP_\mathcal{L} = \{(\phi, \phi \rightarrow \psi, \psi) | \phi, \psi \in \mathcal{L}\}$$

A more traditional way of presenting this rule is the following.

$$MP_\mathcal{L} = \frac{\phi \qquad\quad \phi \rightarrow \psi}{\psi}$$

2.2.3. DEFINITION. The standard proof system $\mathcal{S}_p$ for minimal propositional logic consists of the three axioms A_1, A_2, A_3 and the rule Modus Ponens.

The system $\mathcal{S}_p$ is sound and complete for minimal propositional logic. A proof of this claim is beyond the scope of this dissertation, but proofs for similar systems can be found in logic textbooks, for instance [53].

2.2.2 Full Propositional Logic

Logical connectives can be seen as functions that take as input a number of truth values, and return a truth value. There are two truth values, and thus two corresponding truth constants: $\bot$ (false) and $\top$ (true). There are two one-place functions, namely the identity, which does not have a connective, and negation, for which the notation $\neg$ is used. A simple counting argument can be used to show that there are $2^4 = 16$ different two-argument functions. Only a few of these are commonly used as connectives, namely $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implication), $\leftrightarrow$ (double implication) and ∇ (exclusive or). In this section, a version of propositional logic based on these connectives is presented. This logic is called *full* propositional logic.

A formula of the form $\neg\phi$ is called a negation. Similarly we call $\phi \vee \psi$ a disjunction, $\phi \wedge \psi$ a conjunction, $\phi \nabla \psi$ an exclusive disjunction, $\phi \rightarrow \psi$ an implication and $\phi \leftrightarrow \psi$ a double implication. The two constants $\top$ and $\bot$ can be called verum and falsum. Negation is assumed to be the strongest binding connective, so that $\neg q \wedge r$ is the same formula as $(\neg q) \wedge r$. For all other connectives, operators that appear further to the right in the expression bind stronger. Thus, $p \wedge q \vee r$ is the same formula as $p \wedge (q \vee r)$.

2.2.4. Definition. Let P be a set of atomic propositions and $p \in P$ an element of P. Full propositional logic $\mathcal{L}_p^f(P)$ consists of formulas ϕ generated by the rule

$$\phi ::= p \mid \bot \mid \top \mid \phi \rightarrow \phi \mid \phi \leftrightarrow \phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \phi \nabla \phi$$

This full language is interpreted in the following way. The model M is again a subset of the set of all propositions P.

$M \models \top$	always
$M \models \bot$	never
$M \models p$ where $p \in P$	iff $p \in M$
$M \models \neg\phi$	iff not $M \models \phi$
$M \models \phi \vee \psi$	iff $M \models \phi$ or $M \models \psi$ (or both)
$M \models \phi \wedge \psi$	iff $M \models \phi$ and $M \models \psi$
$M \models \phi \nabla \psi$	iff $M \models \phi$ or $M \models \psi$ but not both
$M \models \phi \rightarrow \psi$	iff $M \models \phi$ implies $M \models \psi$
$M \models \phi \leftrightarrow \psi$	iff $M \models \phi$ and $M \models \psi$ or neither

It is not hard to show that under this interpretation the following formulas hold

on any model M.

$$M \models \top \leftrightarrow (\bot \to \bot)$$
$$M \models \neg\phi \leftrightarrow (\phi \to \bot)$$
$$M \models (\phi \lor \psi) \leftrightarrow ((\phi \to \bot) \to \psi)$$
$$M \models (\phi \land \psi) \leftrightarrow ((\phi \to \psi \to \bot) \to \bot)$$
$$M \models (\phi \nabla \psi) \leftrightarrow ((\phi \to \psi) \to (\psi \to \phi)) \to \bot)$$
$$M \models (\phi \leftrightarrow \psi) \leftrightarrow ((\phi \to \psi) \to (\psi \to \phi) \to \bot) \to \bot)$$

One can thus define all other operators in terms of the two connectives $\bot$ and $\to$.

Conjunction and disjunction have as a feature that changing the order and nesting of these operators does not change their truth value: $\phi \lor \psi$ is equivalent to $\psi \lor \phi$, and $(\phi \lor \psi) \lor \chi$ is equivalent to $\phi \lor (\psi \lor \chi)$. These properties make it possible to apply these operators to finite sets. Thus, we define a disjunction of a set by

$$\bigvee \{\phi_0, \phi_1, \ldots, \phi_n\} = \phi_0 \lor \phi_1 \lor \ldots \lor \phi_n$$

Similarly we define the conjunction of a set as

$$\bigwedge \{\phi_0, \phi_1, \ldots, \phi_n\} = \phi_0 \land \phi_1 \land \ldots \land \phi_n$$

A useful property of these operators is that any propositional logic formula is equivalent to a conjunction of disjunctions of possibly negated atomic propositions.

2.2.5. DEFINITION. A formula ϕ is in *conjunctive normal form* iff it has the form $\phi = \bigwedge_i \bigvee_j \psi_{ij}$, where ψ_{ij} is of the form $\psi_{ij} = \neg a$ or $\psi_{ij} = a$, for some atomic proposition $a \in P$.

2.2.6. DEFINITION. A formula ϕ is in *disjunctive normal form* iff it has the form $\phi = \bigvee_i \bigwedge_j \psi_{ij}$, where ψ_{ij} is of the form $\psi_{ij} = \neg a$ or $\psi_{ij} = a$, for some atomic proposition $a \in P$.

Two example propositional logic formulas, one in conjunctive and one in disjunctive normal form, are the following.

$$\begin{array}{ll}\text{Conjunctive normal form} & (p \lor q) \land (\neg p \lor \neg q) \\ \text{Disjunctive normal form} & (p \land \neg q) \lor (\neg p \land q)\end{array}$$

For logics other than propositional logic, one can also define the notions of conjunctive and disjunctive normal form in a similar way. For instance for modal logic (defined in the next section), a formula is conjunctive form is a formula of the form $\phi = \bigwedge_i \bigvee_j \psi_{ij}$ where ψ_{ij} is either a or $\neg a$, for some formula a that is either an atomic proposition, or of the form $\Box_X \phi$.

2.3 Modal Logic

Modal logic can be seen as an extension of propositional logic with operators for expressing 'modal' concepts, such as provability, knowledge or belief. It has a long history (see [12, pp. 37–48] for a brief exposition) and many applications in logic, mathematics, computer science and artificial intelligence. It originates in the study of necessity. Consider the following pair of statements:

> If it does not rain then the weather is good $(\neg p \to q)$
> If it rains then the weather is not good $(p \to \neg q)$

At my time and place of writing, it rains and the weather is not good. Therefore, both these sentences are true in my current situation. We assume that people do not like rain, so rain is not called good weather. Many philosophers feel that, given this assumption, a sentence such as the first is true merely by accident, whereas the second sentence is necessarily true. In order to express this difference a new symbol is needed. Here, we use the symbol $\square$ in front of a formula in order to express necessity. Thus, if M expresses the current state of the world, and w my current time and place, then the following hold.

$$M, w \models (\neg p \to q)$$
$$M, w \models \square(p \to \neg q)$$

The dual of the box $\square$ is the diamond $\Diamond$. It expresses that something is possible, and can be defined as $\Diamond\phi = \neg\square\neg\phi$. One can for instance say that it is possible that it rains and the weather is not good ($\Diamond(\neg p \wedge \neg q)$), but that it is impossible that it rains and the weather is good at the same time $\neg\Diamond(p \wedge q)$. One can define modal logic formally in the following way. The set P is again a set of atomic propositions, and the set Δ contains the different modalities that we allow. Thus, if Δ contains only one element, we get basic modal logic. If Δ contains multiple elements, we get a multi-modal logic with multiple different modal operators.

2.3.1. DEFINITION. Suppose the finite sets Δ and P are given, and let $X \in \Delta$ and $p \in P$ be typical elements. Multi-modal logic $\mathcal{L}_\square(P)$ consists of formulas ϕ generated by the rule

$$\phi ::= p \mid \square_X \phi \mid \phi \to \phi \mid \bot$$

If the set of modalities Δ is a singleton $\Delta = \{X\}$ then the subscript X can be omitted and we have single-agent modal logic. The notation $\Diamond_X \phi$ is used as a shorthand for $\neg\square_X\neg\phi$.

It took a while before logicians discovered a good way to interpret the new operators. One of the reasons is that one can read the operator in different ways. The next table shows a provability reading, a temporal reading, ethical reading, doxastic reading and an epistemic reading. For many readings an alternative notation is sometimes used, so that one can mix these different readings without chance of confusion.

meaning	notation	reference
ϕ can be proven	$\Box\phi$	[16]
ϕ is always true in the future	$\Box\phi$	page 24
ϕ ought to be true	$\Box\phi$ or $O\phi$	[70]
A believes ϕ	$\Box_A\phi$ or $B_A\phi$	[71]
A knows ϕ	$\Box_A\phi$ or $K_A\phi$	[32]

In the last two examples, the logical language contains multiple modal operators, one for each agent. These logics are thus multi-modal logics, whereas in the other examples we have single-agent modal logic.

A general semantics for modal logics was finally found around 1960, and is for a large part due to Saul Kripke, and is therefore called Kripke semantics [12].

2.3.2. DEFINITION. A Kripke model M is a tuple $M = (\Delta, W, \{R_X\}_{X\in\Delta}, P, \pi)$, where Δ is a set of agents, W is a set of worlds, $\{R_X\}_{X\in\Delta}$ is a collection of binary accessibility relations R_X between worlds, one for each modality $X \in \Delta$, P is a set of atomic propositions and π is a function $\pi : W \to 2^P$.

The function π is typically called an *interpretation function*. The statement $\Box_X\phi$ is interpreted as saying that ϕ is true in all possible worlds. This semantics is therefore called the *possible world semantics*. Which worlds are possible is determined by the accessibility relation R_X.

2.3.3. DEFINITION. Suppose that $M = (\Delta, W, \{R\}_\Delta, P, \pi)$ is a Kripke model, $w \in W$, $p \in P$ and $X \in \Delta$.

$$
\begin{aligned}
&M, w \models p && \text{iff} && p \in \pi(w) \\
&M, w \models \bot && && \text{never} \\
&M, w \models \phi \to \psi && \text{iff} && M, w \models \phi \text{ implies } M, w \models \psi \\
&M, w \models \Box_X\phi && \text{iff} && \forall v : (w, v) \in R_X \ \Rightarrow \ M, v \models \phi
\end{aligned}
$$

Different operators $\Box_X$ can have different accessibility relations R_X and thus satisfy different properties. There are some properties that hold for all modal logics. Other formulas are only true under some readings of modal logic. One important principle is that of necessitation.

2.3.4. LEMMA. *Let $\Box_X\phi \in \mathcal{L}_\Box$. If ϕ is valid, then $\Box_X\phi$ is valid. [12]*

It is perhaps interesting to remark that this rule preserves logical truth, but not actual truth. Thus, it is not the case that if ϕ is true in a situation, then $\Box_X\phi$ is true in that situation. The theorem only claims that validity is preserved, which is a weaker statement. This lemma can therefore not be used to introduce an axiom along the lines of $\phi \to \Box_X\phi$, but it can be turned into a reasoning rule. This rule is called *Necessitation*.

$$Nec_{\mathcal{L}} = \{(\phi, \Box_X\phi) | \Box_X\phi \in \mathcal{L}\}$$

Or expressed in the more traditional format:

$$Nec_{\mathcal{L}} = \frac{\phi}{\Box_X \phi}$$

Another principle of any normal modal logic is distribution of the box operator. From the truth of $\Box_X(p \rightarrow q)$ one can derive $\Box_X p \rightarrow \Box_X q$. This can be used to formulate the axiom *Distribution* or K.

$$K = \Box_X \phi \rightarrow \Box_X(\phi \rightarrow \psi) \rightarrow \Box_X \psi$$

An important question is of course whether one can formulate a proof system for modal logic. In such a proof system one could reuse all axioms of propositional logic, but this is not normally done. It is more convenient to assume familiarity with propositional logic, and to allow any propositional logic tautology as an axiom. Thus, the following is an axiom in our proof system for modal logic.

$$\text{prop} = \tau \quad \text{where } \tau \text{ is an instance of a propositional logic tautology}$$

This axiom allows one to substitute any number of atomic propositions by arbitrary modal logic formulas. One cannot only use this rule to derive the tautology $p \rightarrow p$, but also to derive $\Box_X p \rightarrow \Box_X p$.

One can of course question the legitimacy of this axiom. Is it not too easy to allow any tautology in a proof? Does this not lead to uncheckable proofs? The answer is 'no'. One can test whether a propositional logic formula is valid (and thus provable) by checking all different models: there is a finite number of atomic propositions in any propositional logic formula, and thus a finite number of models. Furthermore one can convert proofs that use this axiom by replacing each usage of this axiom by the corresponding $\mathcal{S}_p$ proof. Thus, at least in theory, such an axiom can be allowed.

2.3.5. DEFINITION. Let $\mathcal{L}_\Box$ be the language of modal logic. The proof system $\mathcal{S}_\Box$ for this language has prop and K as axioms, and Modus Ponens and Necessitation as reasoning rules.

2.3.6. THEOREM. *Suppose that $\phi \in \mathcal{L}_\Box$ is valid. Then $\mathcal{S}_\Box \vdash \phi$.*

This property is called weak completeness, and a proof of some version of this theorem can be found in most modal logic books, for instance the one by Blackburn et al [12, p.194] or Meyer and Van der Hoek [71, p.18]. Below we sketch the general idea, which uses maximally consistent sets.

2.3.7. DEFINITION. Suppose a logic is given with language $\mathcal{L}$ and with a proof system $\mathcal{S}$ that uses Modus Ponens as a reasoning rule (and possibly other rules as well). A set of formulas $S \subseteq \mathcal{L}$ is *maximally consistent* if the following conditions are all met: $\bot \notin S$, all instances of axioms $\phi \in \mathcal{A}$ are in S, if $\phi, \phi \rightarrow \psi \in S$ then $\psi \in S$, and for any formula ϕ either $\phi \in S$ or $\neg\phi \in S$.

For each consistent formula ϕ one can find a maximally consistent set S so that $\phi \in S$ [12]. If the proof system $\mathcal{S}$ is sound, then for any pointed model M, w, the set $\{\phi \mid M, w \models \phi\}$ is maximally consistent.

In order to prove the theorem, consider a consistent formula ϕ with atomic propositions from the set P. It is necessary to show that there is a model M such that $M, w_0 \models \phi$. Such a model $W = (\Sigma, W, R, P, \pi)$ can be constructed using maximally consistent sets as the worlds as the model: $W = \{w \mid w \text{ is a maximally consistent set}\}$. The relation R_X is then defined such that $(v, w) \in R_X$ if $\Diamond_X \phi \in v$ for all $\phi \in w$, and $\pi(w) = \{p \in P \mid p \in w\}$. For w_0 one can take any maximally consistent set that contains ϕ. This construction defines one large model M called the *canonical model* for a logic, such that any consistent formula holds in some world of this model. This technique to use maximally consistent sets as possible worlds not only works for plain modal logic, but can often be adapted for modal logics with a different interpretation [12, p.194].

The multi-modal logic presented here is the weakest possible version of a modal logic with a possible worlds semantics. The next table lists some formulas that one might expect to hold for certain readings of the $\Box$ operator, but that cannot be proven in $\mathcal{S}_\Box$.

4	$\Box_X \phi \to \Box_X \Box_X \phi$	What is necessary (for X) is necessarily necessary
D	$\Box_X \phi \to \neg \Box_X \neg \phi$	What is necessary (for X) is possible
T	$\Box_X \phi \to \phi$	What is known to X is true

The names $4, D$ and T are the standard names for these axioms [12, p.192]. One can make these formulas valid by putting constraints on the relations R_X. This idea is used in order to get an epistemic reading in the next section.

It is often important to determine whether two models satisfy the same formulas, and for this purpose the notion of bisimulation can be used. Two Kripke models are *bisimilar* if one can find for any world in one model a corresponding world in the other model. Such a relation between worlds is called a bisimulation. A formal definition of this concept for single-agent modal logic would be the following.

2.3.8. DEFINITION. Let $M_1 = (\{X\}, W_1, R_1, P, \pi_1)$ and $M_2 = (\{X\}, W_2, R_2, P, \pi_1)$ be single-agent Kripke models. A non-empty relation $S \subseteq W_1 \times W_2$ is a *bisimulation* if the following conditions hold

- If $(w_1, w_2) \in S$ then $\pi(w_1) = \pi(w_2)$

- If $(v_1, w_1) \in R_1$ and $(v_1, v_2) \in S$ then there is a world w_2 such that $(v_2, w_2) \in R_2$ and $(w_1, w_2) \in S$

- If $(v_2, w_2) \in R_2$ and $(v_1, v_2) \in S$ then there is a world w_1 such that $(v_1, w_1) \in R_1$ and $(w_1, w_2) \in S$

For logic with more than one agent one has to use a set of relations S_X as bisimulation. For simplicity reasons this extension has been omitted.

A bisimulation matches worlds that behave similarly with respect to modal logic formulas. Thus, if $M_1, w_1 \models \phi$ and $(w_1, w_2) \in S$ then also $M_2, w_2 \models \phi$. In order to decide whether two models are equivalent one thus has to find a bisimulation between the models, or prove that no such relation exists.

2.3.1 Epistemic Logic

Epistemic logic is an extension of propositional logic with operators that express that a proposition is known. It originates from philosophy [48], but has found applications in computer science and artificial intelligence [71, 32]. It is one of the best known modal logics.

In epistemic logic, the operator $\Box$ is usually written K. In the case of multiple operators these are written K_X instead of $\Box_X$. The X indicates which agent's knowledge one is talking about. Thus, $K\phi$ denotes that ϕ is known in the single agent case, and $K_A\phi$ means that A knows ϕ. Instead of $\Diamond$ we use M. The operators M and M_X express that something is considered possible, the dual of knowledge.

The following statements about knowledge make use of these operators.

It rains	p
Alice know it rains	$K_A p$
Bob does not know that it rains	$\neg K_B p$
Bob thinks it is possible that it rains	$M_B p$
Alice or Bob knows that it rains	$K_A p \vee K_B p$
Bob does not know that Alice knows that it rains	$\neg K_B K_A p$
Alice knows that she thinks it is possible that it rains	$K_A M_A p$

2.3.9. DEFINITION. Suppose the finite sets Σ and P are given, and let $X \in \Sigma$ and $p \in P$ be typical elements. Epistemic logic $\mathcal{L}_K$ consists of formulas ϕ generated by the rule

$$\phi ::= p \mid K_X \phi \mid \phi \rightarrow \phi \mid \bot$$

For epistemic logic we identify each modality with an agent. Since the notation Σ is used for the set of all agents in this dissertation, we here use Σ for the set of modalities, rather than Δ as we did on page 15.

A relation R is an *equivalence relation* over W if for any worlds $v, w, x \in W$ it is the case that $(v, v) \in R$ (reflexivity), if $(v, w) \in R$ then $(w, v) \in R$ (symmetry) and if $(v, w) \in R$ and $(w, x) \in R$ then $(v, x) \in R$ (transitivity). If a relation is an equivalence relation we often use the symbol $\sim$ for this relation. Furthermore we write $w \sim_X v$ instead of $(w, v) \in \sim_X$.

2.3.10. DEFINITION. A Kripke model $M = (\Sigma, W, \{R_X\}_{X \in \Sigma}, P, \pi)$ is an *epistemic model* if each relation R_X is an equivalence relation over W.

The next table lists some examples of formulas that hold over epistemic models M.

$$\models K_X\phi \rightarrow \phi \qquad \text{Truth}$$
$$\models K_X\phi \rightarrow K_X K_X\phi \qquad \text{Positive Introspection}$$
$$\models \neg K_X\phi \rightarrow K_X \neg K_X\phi \qquad \text{Negative Introspection}$$

These validities can be used as axioms in a reasoning system for epistemic logic.

$$T = K_X\phi \rightarrow \phi$$
$$4 = K_X\phi \rightarrow K_X K_X\phi$$
$$5 = \neg K_X\phi \rightarrow K_X \neg K_X\phi$$

For epistemic logic one also has a proof system.

2.3.11. DEFINITION. The proof system $\mathcal{S}_K$ for $\mathcal{L}_K$ is defined as

$$(\text{prop} \cup K \cup 4 \cup 5 \cup T, MP \cup Nec)$$

This proof system is again sound and complete with respect to the given semantics based on epistemic models. A proof can be found in epistemic or modal logic text books such as [12, p.194]. The common name for this proof system is $S5$ or $S5_n$ where n is the number of agents. This semantics is widely used because it is simple and seems realistic for rational agents, but has also received criticism. First of all there is the omniscience problem: all agents know all tautologies, so every agent knows all mathematical theorems. This seems unrealistic in the case of human agents, or artificial agents with a limited computing capacity. There have been attempts to model knowledge while avoiding omniscience [32]. A second potential objection is that it is not certain that humans have full introspection over all their knowledge. Humans do not always know what they know, and especially it seems doubtful that they have negative introspection. This is for instance stated in Rumsfeld's famous remark:

> "There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know." *(D. Rumsfeld, USA secretary of defense, Feb. 12, 2002)*[92]

Philosophically there are many arguments against introspection [123]. In the context of protocols, with a finite number of states and possibilities, it does not seem to be a problem. It is not unreasonable to assume that agents have introspection over a small, well-known domain such as the possible outcomes of a protocol. It does not follow from this assumption that agents know everything about the whole world, or are completely aware of omissions on their knowledge in general.
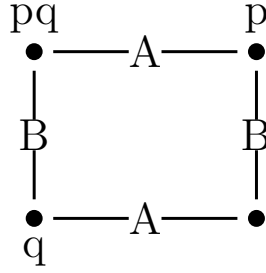
Figure 2.2: Rain in Liverpool and Amsterdam: Model M_1

2.3.2   Common Knowledge

If one reasons about the knowledge of multiple agents, it is natural to consider cases where agents collectively know something. In order to make this possible several notions of group knowledge have been defined. The first of these notions allows one to say that 'Everbody knows …' and is denoted $E\phi$. This notion can be defined by means of a conjunction over all agents.

$$M, w \models E\phi \; \Leftrightarrow \; M, w \models \bigwedge_{X \in \Sigma} K_X \phi$$

This operator can also be defined using an accessibility relation, like the K operators. Define R_E as the union of all single agent relations: $R_E = \bigcup_{X \in \Sigma} \sim_X$. The everybody knows operator can be interpreted in the following way.

$$M, w \models E\phi \quad \text{iff} \quad \forall v : (w, v) \in R_E \text{ implies } M, v \models \phi$$

The relation R_E is not transitive, and thus it is not an equivalence relation. Therefore, the principles of positive and negative introspection do not hold for 'Everybody knows'. The formula $E\phi \rightarrow EE\phi$ is not valid, and a counterexample is presented in figure 2.2. This illustrated model M_1 has two agents, A and B. A is in Liverpool, and can thus observe the weather in Liverpool, and B is in Amsterdam, and can see the weather in Amsterdam. The atomic proposition p indicates that it rains in Liverpool, and q that it rains in Amsterdam. Suppose that in the actual situation s it rains in both Amsterdam and Liverpool. The following formulas hold.

$M_1, s \models K_A p \wedge K_B q$	A and B know that it rains in their city
$M_1, s \models K_A(p \vee q)$	A knows that it rains in Liverpool or Amsterdam
$M_1, s \models \neg K_A K_B q$	A does not know that B knows it rains in Liverpool
$M_1, s \models E(p \vee q)$	Everybody knows that it rains somewhere
$M_1, s \models \neg EE(p \vee q)$	Not everybody knows that everybody knows it rains

The conclusion of this example is that given what everybody knows, one cannot conclude anything about what agents know about each other's knowledge. Since

one often does want to reason about this higher order knowledge, a stronger notion of group knowledge is useful. Lewis therefore introduced the notion of *common knowledge* [65]. Intuitively something is common knowledge if everybody knows it, everybody knows that everybody knows it, everybody knows everybody knows everybody knows it, etcetera. Common knowledge turns out to be a more powerful notion than 'everybody knows'. First of all it is hard to obtain, but on the other hand it can be a necessary condition in order to coordinate [71] or to make linguistic conventions work [65]. Other phenomena, such as the pricing of TV commercials, can also be explained by the need of advertisers to achieve common knowledge instead of plain knowledge [21].

The notation $C\phi$ is used to convey that ϕ is common knowledge. It can be defined in the following way. Let $\sim_C$ be the smallest equivalence relation such that for all X we have $\sim_X \subseteq \sim_C$. We can interpret $C\phi$ in the following way.

$$M, w \models C\phi \quad \text{iff} \quad \forall v : w \sim_C v \text{ implies } M, v \models \phi$$

The following formulas are valid under this interpretation.

$$\begin{aligned}
&\models C\phi \rightarrow CC\phi \\
&\models \neg C\phi \rightarrow C\neg C\phi \\
&\models C\phi \rightarrow C(\phi \rightarrow \psi) \rightarrow C\psi \\
&\models C\phi \rightarrow E\phi \\
&\models \phi \rightarrow E(\phi \rightarrow E\phi) \rightarrow C\phi
\end{aligned}$$

The common knowledge operator C thus satisfies positive and negative introspection, which means that it behaves as a knowledge operator.

One can show that if a formula ϕ holds in every world w of a model M, then ϕ is common knowledge in M. Thus, if we present a model of a certain situation in which there is no state where ϕ does not hold, then we have implicitly assumed that ϕ is common knowledge. In most examples in this dissertation, common knowledge is not introduced in the language. However, common knowledge of at least the protocol is assumed everywhere in this dissertation, and in some cases also the preferences of agents are common knowledge.

2.4   Theorem Proving, Satisfiability and Model Checking

Theorem Proving, Satisfiability and Model Checking are three different problems that one can formulate for a logic. In this section, it is explained how these problems are relevant for multi-agent protocols.

Theorem proving, the problem of finding derivations that prove a given theorem, has always been one of the main uses of logic. Automated theorem proving has received much attention in the field of AI. As described by for instance

MacKenzie [66], automated theorem proving has important applications in the verification of hardware and software.

A related problem to theorem proving is satisfiability. In the satisfiability problem one has a formula ϕ such that $\neg\phi$ cannot be proven, and one would like to find a model M such that $M \models \phi$. In many cases one can use the same method, for instance a tableau-based method [96], for theorem proving and satisfiability: If ϕ is satisfiable a model is produced by the method, otherwise the method returns a proof for $\neg\phi$.

Model checking is the problem of verifying whether a formula ϕ holds on a given model M, and is also widely used for verification of systems [53]. For many logics, including propositional logic, model checking is substantially easier than satisfiability. Intuitively this can be explained by the fact that in order to solve a satisfiability problem, one has to find a model, whereas for model checking one has been given one specific model.

Model checking can be used for verification of computer hardware and programs, in the following way. The system to be checked is translated in a logical model M, and the property that one would like to verify is translated in a formula ϕ. The following set of correspondences illustrates the correspondence between the original verification problem and the model checking problem.

$$\frac{\text{protocol}}{\text{property}} \Leftrightarrow \frac{\text{model}}{\text{formula}}$$

What is meant here is that protocols correspond to models, and properties with formulas. The double-headed arrow indicates that one can go back and forth from the informal description on the lefthand side to the formal description of the situation on the righthand side.

Model checking was first done for properties involving time [22, 10]. One reason that model checking has become popular is the invention of *symbolic model checking* [69]. Using this technique the model that is checked is not stored explicitly, but represented in a symbolic way, using for example ordered binary decision diagrams. The model can thus have more states than one can store explicitly in the memory of the computer that is used. The use of symbolic model checking has made it possible to check systems with billions of states, instead of only millions of states. Recent work aims to develop model checking techniques for epistemic properties [104, 101, 56].

For some logics one can use theorem proving for system verification. This is possible if one can translate the system into a formal structure (for instance a tree or a graph) and then describe this structure using a formula ϕ_1. The property to be verified is represented by a formula ϕ_2, and one uses the theorem prover to prove that $\vdash \phi_1 \rightarrow \phi_2$. This results into a proof that any system of the given structure has the desired property. Schematically one can display this in the following way.

$$\frac{\text{protocol}}{\text{property}} \quad \Leftrightarrow \quad \frac{\phi_1}{\phi_2}$$

Hence both the protocol and the property correspond to a formula. This method of verification by theorem proving is only possible for logics in which one can express the structure of a system. For some logics, such as Pauly's coalition logic [85] (discussed in section 3.4), this is possible because the logic has a modal operator that corresponds to exactly one step in the protocol. Thus, one can construct, for each game tree T describing a protocol, a formula ϕ_1 that describes this game tree. For other logics, such as Van Benthem's logic for process models [99], also discussed in section 3.4, this is not possible. In this logic game trees cannot be described in detail, and hence no suitable formula ϕ_1 can be found.

Another long term goal in computer science is the automatic design of systems. In this case the user indicates the properties that a system, protocol or program must have, and the computer constructs a system. One can always use the AI-heuristic 'generate and test' in order to solve this problem, but this heuristic is not very efficient. In general this problem is more difficult than verification, just as satisfiability is more difficult than model checking: again one has to construct something, instead of just computing a yes-no answer.

In economics, the classical name for the problem of finding a suitable auction or procedure is *mechanism design* [83]. This problem has been picked up by computer science researchers, and several researchers are now active in the area called *automated mechanism design* or *computational mechanism design* [29, 25]. For all logics in which models can be seen as representing a system, the mechanism design problem can be reduced to satisfiability checking. Unfortunately satisfiability checking is often a difficult problem, and hence mechanism design using logical methods is often hard as well.

Linear Temporal Logic

A logic that is often used successfully in combination with model checking is *linear temporal logic LTL*. A formula of this logic is interpreted over a sequence of states. Such a sequence represents the different states a system can go through during a computation. The different operators of LTL can be used to refer to states after the current state.

2.4.1. DEFINITION. The logic LTL contains formulas ϕ generated by the following rule. In this rule, p is a typical element of P

$$\phi ::= p \mid \bot \mid \phi \rightarrow \phi \mid \phi \mathcal{U} \phi \mid \bigcirc \phi$$

The sequences of states that are considered for LTL are not always finite. Therefore, it is more convenient to use functions $f : \mathbb{N} \to 2^P$ as sequences. These functions assign subsets of true atomic propositions to each natural number. Such a sequence is called a history or a run. Each formula ϕ is interpreted over a pair f, n where f is such a function, and $n \in \mathbb{N}$ indicates the current state.

$$
\begin{aligned}
&f, n \models \bot && \text{never} \\
&f, n \models p \text{ where } p \in P && \text{iff } p \in f(n) \\
&f, n \models \phi \to \psi && \text{iff } f, n \models \phi \text{ implies } f, n \models \psi \\
&f, n \models \bigcirc \phi && \text{iff } f, n+1 \models \phi \\
&f, n \models \phi \mathcal{U} \psi && \text{iff } \exists m : m \geq n f, m \text{ and } \models \psi \text{ and } \forall m > k > n : f, k \models \phi
\end{aligned}
$$

The 'until' operator $\phi \mathcal{U} \psi$ indicates that at some point in the future ψ is true, and until that time ϕ holds. The next operator $\bigcirc \phi$ expresses that ϕ is true at the next state. A commonly defined shortcut is the 'sometimes' operator $\Diamond \phi = \top \mathcal{U} \phi$ that indicates that ϕ is true somewhere in the future. One can define an 'always' operator $\Box \phi$ that indicates that ϕ holds forever from now on, with the following definition: $\Box \phi = \neg \Diamond \neg \phi$.

A system to be verified is not modeled as a single run but as a set of runs. Since these runs can be infinite, such a set must be specified implicitly. The most common way to do this is to define a labeled graph (V, E, π) where π is a function from V to 2^P. A possible *path* in this graph is a sequence $w : \mathbb{N} \to V$ such that $(w(n), w(n+1)) \in E$ for all $n \in \mathbb{N}$. Any possible path w defines a possible run $f : n \mapsto \pi(w(n))$. A shorthand notation for this is $f = \pi(w)$, since $f(n)$ is constructed by first computing $w(n)$, and then applying π to the result. Let $\mathcal{W}$ be the set of all possible paths, and $\mathcal{R}$ the set of all possible runs.

A model checker for LTL takes a description of a system and an LTL formula ϕ. From the description it computes a set of possible runs $\mathcal{R}$, and then it checks whether for all runs $r \in \mathcal{R}$ it is the case that $r, 1 \models \phi$. If so the model checker returns true, otherwise the model checker returns the run r such that $r, 1 \models \neg \phi$. An LTL model checker thus returns counter-examples that can be very informative for system designers. A well-known model checker for LTL is *SPIN* [51].

Branching Time Temporal Logic

Computation tree logic is a temporal logic that is suitable for reasoning about models that have multiple possible futures [23]. The formulas can express whether certain events happen in all possible futures, or only in some possible future. This has turned out to be an important feature in the verification of for instance concurrent computer systems.

2.4.2. Definition. The logic CTL contains formulas ϕ generated by the following rule. In this rule, p is a typical element of P.

$$\phi ::= p \mid \phi \rightarrow \phi \mid \bot \mid \forall \psi \mid \exists \psi$$
$$\psi ::= \Box \phi \mid \phi \mathcal{U} \phi$$

The formulas ϕ generated by the grammar rules stated above are called *state formulas* because they are interpreted over states. The formulas ψ are called *path formulas*, because they are interpreted over paths. As the name indicates this logic can be interpreted over tree structures, where the nodes of the tree are states of a system, and the edges indicate how the system can go from one state to the next. Typically such a tree is constructed by considering all possible paths through a labeled graph (V, E, π), combined in a set $\mathcal{W}$. Each path w is a finite sequence $w(1)w(2)\ldots w(n)$ or infinite sequence $w(1)w(2)\ldots$ of states $w(n)$. Thus the notation $w = v\ldots \in \mathcal{W}$ is used to indicate that w is an infinite path that starts in v. CTL formulas ϕ can be interpreted over a set of paths combined with an interpretation function $\pi : V \rightarrow 2^P$ and a current state $v \in V$. The path formulas are however interpreted over a set of paths combined with an interpretation function $\pi : V \rightarrow 2^P$ and a current path $w \in \mathcal{W}$.

$$
\begin{aligned}
&\mathcal{W}, \pi, v \models \bot && \text{never} \\
&\mathcal{W}, \pi, v \models p \text{ where } p \in P && \text{iff } p \in \pi(v) \\
&\mathcal{W}, \pi, v \models \phi_1 \rightarrow \phi_2 && \text{iff } \mathcal{W}, \pi, v \models \phi_1 \text{ implies } \mathcal{W}, \pi, v \models \phi_2 \\
&\mathcal{W}, \pi, v \models \forall \psi && \text{iff } \forall w = v\ldots \in \mathcal{W} : \mathcal{W}, \pi, w \models \psi \\
&\mathcal{W}, \pi, v \models \exists \psi && \text{iff } \exists w = v\ldots \in \mathcal{W} : \mathcal{W}, \pi, w \models \psi \\
\\
&\mathcal{W}, \pi, w \models \Box \phi && \text{iff } \forall n \geq 1 : \mathcal{W}, \pi, w(n) \models \phi \\
&\mathcal{W}, \pi, w \models \phi_1 \mathcal{U} \phi_2 && \text{iff } \exists m \geq 1 : \mathcal{W}, \pi, w(m) \models \phi_2 \text{ and} \\
&&& \quad \forall k : \ m > k \geq 1 \ \Rightarrow \ \mathcal{W}, \pi, w(k) \models \phi_1
\end{aligned}
$$

Both CTL and LTL can be model checked in polynomial time in terms of the number of states of the system [90]. This makes the verification of systems feasible. Unfortunately, the number of states of a system can increase exponentially with the number of components, or the amount of memory cells, that a system has. The conclusion that model checking is tractable, is thus perhaps somewhat misleading, since model checking is intractable when the input is measured in the size of the description of the system. Summarizing and interpreting the current state of the art, one could say that model checking using LTL and CTL is feasible, but not (yet) very tractable for real world systems. In practice, techniques like symbolic model checking and other heuristics can be used to make model checking for ever larger systems feasible.

2.5 Computational Complexity

In the field of *computational complexity*, researchers contemplate why certain problems are hard to solve for computers [81, p. v]. A *problem* in this context has to be defined precisely. A problem is function $f : \text{Prob} \to \text{Sol}$, that takes *problem instances* $d \in \text{Prob}$ to their solution $f(d) \in \text{Sol}$. In the case of a *decision problem*, the solution space consists of only two answers: $\text{Sol} = \{0, 1\}$ where 0 means 'no' and 1 means 'yes'. An *algorithm* for problem f is a mechanical procedure that takes an input d and produces its answer $f(d)$.

To give an example of a problem in this sense, consider satisfiability. For each logic, its satisfiability problem is a logical problem of interest to complexity theorists. Satisfiability is usually phrased as a decision problem f such that $f(\phi) = 1$ iff ϕ is satisfiable.

Since there are many different computer architectures, one can have many different ideas of what counts as a mechanical procedure. However, it turns out that many of these architectures are equivalent with respect to the computational resources that are required to solve a problem, and therefore it does not matter which architecture or notion of algorithm one chooses. A common choice is to consider *Turing machines*, introduced by Alan Turing, because these machines are very convenient from a theoretical perspective [81, p. 19]. In practice, algorithms for Turing machines can be converted to programs for actual computers, that have a comparable efficiency.

Turing machines that compute f, do not work on problem instances simpliciter, but on *representations* thereof. Every problem instance can be represented by a string of symbols. In practice, it is sufficient to consider only two distinct symbols, 0 and 1, to represent any object. A two-valued variable is called a *bit*, and the size $\|d\|$ of an object d can thus be measured in the number of *bits* one needs to encode the object. For instance formulas could be encoded by using a sequence of say 8 bits to encode each symbol in the formula. In that case a formula $p \vee q$ would be 24 bits in size. The exact number of bits needed in a representation is not always important. It is enough to know that "any 'finite' mathematical object can be represented by a finite string over an appropriate alphabet" [81, p. 26]. We simply assume a "reasonably succinct representation" [81, p. 26] is used for objects such as formulas.

One computational resource we are interested in is the time it takes for an algorithm to compute the answer to a problem, and this is called the *computation time*. The question is how the computation time (measured in steps of the head of the Turing machine) depends on the size of the input. In particular, one would like to find monotone functions $b : \mathbb{N} \to \mathbb{R}$ that provides an upper bound on the computation time. If for all $d \in \text{Prob}$ larger than a certain fixed length c, the algorithm can compute $f(d)$ in less than $b(\|d\|)$ steps, then we say that the running time of the algorithm is bounded by b.

Similarly one can consider bounds on the amount of bits that an algorithm

needs as working memory in order to compute its answer. Thus, if there is a constant c such that for instances d with $\|d\| > c$, the algorithm can compute $f(d)$ using less than $b(\|d\|)$ bits of memory, then we say that the space needed by the algorithm is at most b.

This notion of a bound is overly precise as it greatly depends on the machine architecture chosen: for instance sorting a list can have a bound $b(x) = 3 \cdot x^2 + 167$ on one machine and $b'(x) = 3.5 \cdot x^2 + 14$ on another machine. Therefore, the bound functions are put into equivalence classes. A bound b' is in the same equivalence class as b if there are constants $c, e \in \mathbb{R}$ such that for all inputs d with $\|d\| > e$ we have that $b'(\|d\|) \leq c \cdot b(\|d\|)$. We write $\mathcal{O}(b(\|d\|))$ to denote the equivalence class of $b(\|d\|)$. It is not hard to see that both bounds given above sit in the same equivalence class $\mathcal{O}(\|d\|^2)$. They are called quadratic bounds. Similarly one has linear bounds $\mathcal{O}(\|d\|)$ and exponential bounds $\mathcal{O}(2^{\|d\|})$.

An algorithm runs in *polynomial time* if there is some bound b on its running time that is in $\mathcal{O}(x^n)$ for some n. Similarly an algorithm can be in *polynomial space* if there is a bound b on the memory needed that is in $b = \mathcal{O}(x^n)$ for some n.

In figure 2.3 a graph is displayed. If one sees such a graph as a network of roads between cities, a question of practical importance would be what the shortest path is between two nodes of the graph. Thus, the path-finding problem would be a function f that takes (V, E, v_1, v_2, n) as input, and returns a path $w_1 \ldots w_m$ with $m \leq n$, $w_1 = v_1$ and $w_m = v_2$, if such a path exists. Otherwise it should return 'no'. This problem can be solved in polynomial time, for instance using Dijkstra's algorithm [27].

The model checking problem for propositional logic is the function g such that $g(M, \phi) = 1$ if and only if $M \models \phi$. This function can be computed in time $\mathcal{O}((\|M\| + \|\phi\|)^2)$ and thus this problem can be solved in polynomial time. Only a small amount of memory is needed, bounded by $\|\phi\|$, and thus the problem is also in polynomial space. In general a Turing machine can only use a polynomial amount of space in polynomial time, so all polynomial time problems are in polynomial space.

Algorithms can be classified into classes of algorithms whose bounds are in the same equivalence class. These classes are called *complexity class*es. The following complexity classes are well-known and turn out to be relevant for the results in this dissertation. The class P contains problems that can be solved in polynomial time. The class $PSPACE$ contains problems that need a polynomial amount of memory.

Problems can also be divided into the same complexity classes. A problem belongs to a class $\mathcal{C}$ if there is an algorithm in $\mathcal{C}$ that solves the problem. Thus, the problem of finding the shortest path between two points in a graph is in P because there is a P algorithm that solves this problem.

The class P is often called that class of *tractable* of problems or problems that can be solved *efficiently*. In this dissertation we follow this convention and indeed
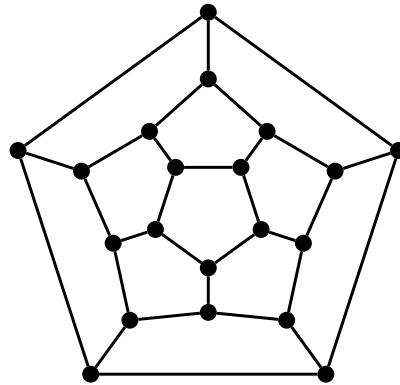
Figure 2.3: A graph with a Hamiltonian cycle

uses these words as meaning 'solvable in polynomial time'. The term *intractable* means that the problem is not in the class P.

Nondeterministic Computation

There are many decision problems f such that $f(d) = 1$ if there exists some object w that satisfies certain criteria. A good example is the problem, for a given graph, to decide whether this graph has a *Hamiltonian cycle*. A Hamiltonian cycle visits each node of the graph, but does not use the same edge twice. In figure 2.3, Hamilton's original problem is displayed. In this particular graph there is a Hamiltonian cycle, and the reader is invited to find one as an exercise (a possible answer is displayed in figure 2.4 on page 29). The reader will most likely experience that finding a Hamiltonian cycle is harder than verifying that a given path is a Hamiltonian cycle.

Suppose that the problem $f : \text{Prob} \to \{0, 1\}$ is defined such that $f(d) = 1$ iff d is a graph that has a Hamiltonian cycle. Suppose also that we have a problem $g \in \text{P}$ that checks whether a path w is a Hamiltonian cycle on d. The relation between f and g is that $f(d) = 1 \Leftrightarrow \exists w : g(d, w) = 1$.

The Hamiltonian cycle w is called a witness for d, since the existence of a path w is evidence for the fact that $f(d) = 1$. We have assumed that $g \in \text{P}$, and hence that we have a polynomial time algorithm for g. A very naive algorithm for solving f would be the following. Guess some value w, and check whether $g(d, w)$. If you are very lucky in guessing w, then this algorithm works in polynomial time as well. If you are not a good guesser, this algorithm is not efficient.

Any problem f for which there exist a polynomial time function g that checks witnesses, is called solvable in *nondeterministic polynomial time* [26]. The class of these problems is called *NP*. There are many problems that have practical relevance in this class [35]. A logical example of an NP problem is the proposi-

tional logic satisfiability problem f. In order to determine whether a formula ϕ is satisfiable, one can guess a model $M \subseteq P$ such that $M \models \phi$. If such a model is guessed correctly, then we know that $f(\phi) = 1$. If no lucky guess can be made at all, then ϕ is not satisfiable and thus $f(\phi) = 0$.

Reductions and Completeness

It is often possible to translate an instance of one problem into an instance of another problem. Such a translation is called a reduction in the context of complexity theory. Suppose that f and f' are two problems. A *reduction r* from f to f' is a function such that $\forall d : \ f(d) = f'(r(d))$. If a reduction r exists that is relatively easy to compute, then solving an instance of f cannot be harder than solving an instance of f. In order to determine $f(d)$ one first computes the reduced problem $r(d)$ and then uses the algorithm, if one is known, for $f'(r(d))$. If the reduction function r is in P, then we call f *reducible in polynomial time* to f' [27].

A problem f is called $\mathcal{C}$-hard if all problems in the complexity class $\mathcal{C}$ can be reduced in polynomial time to problem f. If the $\mathcal{C}$-hard problem f is itself a member of class $\mathcal{C}$, then f is called $\mathcal{C}$-*complete*. Such a problem can be said to be a representative for all problems of this class. Consider for instance the *satisfiability problem f* for propositional logic. This decision problem can be defined by saying that $f(\phi) = 1$ if $M \models \phi$ for model M.

2.5.1. Theorem (Cook's theorem). *Deciding whether a propositional logic formula ϕ is satisfiable is NP-complete.*

This was the first theorem to be proven NP-complete, presented in 1971 [81, p. 176]. The proof contains a general method how a reduction function r can be found for any NP decision problem that has a verification method g. Such a general proof has to be given once for each class. In order to prove that another problem in NP is NP-complete, it suffices to select a known NP-complete problem, and then give a specific reduction function r from the complete problem to the next problem.

In many NP-completeness proofs it is convenient not to use the general satisfiability problem in a reduction argument, but to give a reduction from 3-CNF formulas. A formula ϕ is in 3-CNF if it is in conjunctive normal form, and each disjunction contains exactly three literals. Such a formula thus has the following form.

$$\phi = \bigwedge_i (\pm a_i \vee \pm b_i \vee \pm c_i)$$

The sign $\pm$ can be either a negation or nothing, and $a_i, b_i, c_i \in P$ are atomic propositions. The satisfiability problem for 3-CNF formulas is called *3SAT* and this problem is NP-complete [81, p. 183].

For the class PSPACE, we also use a logical problem in the reduction arguments. We use the satisfiability problem for a quantified boolean formula, since this problem is PSPACE-complete.

2.5.2. DEFINITION. A *quantified boolean formula* ϕ is a formula of the form

$$\forall x_1 \exists x_2 \forall x_3 \ldots \exists x_{n-1} \forall x_n \phi_q$$

such that ϕ_q is a propositional logic formula with $\{x_1, \ldots x_n\}$ as atomic propositions.

An example formula is $\forall p \exists q (p \vee \neg q) \wedge (\neg p \vee q)$. This formula is true if for all truth values of p one can find a truth value for q such that the propositional logic formula holds. Intuitively, $\forall p \phi$ is true if ϕ holds regardless what truth value one chooses for p, and similarly $\exists p \phi$ holds if ϕ is true for some truth value for p. Formally, we can define the following interpretation for quantified boolean formulas, which is an extension of the interpretation of propositional logic given on page 12. Let $S \subseteq P$ be a set of atomic propositions, ϕ a quantified boolean formula, and $\phi_q \in \mathcal{L}_p$ a propositional logic formula.

$$
\begin{aligned}
S &\models \forall x \phi && \text{iff } (S \cup \{x\}) \models \phi \text{ and } S \models \phi \\
S &\models \exists x \phi && \text{iff } (S \cup \{x\}) \models \phi \text{ or } S \models \phi \\
S &\models \phi_q && \text{iff } S \models \phi_q \text{ in propositional logic}
\end{aligned}
$$

The *QBF* decision problem f is defined such that for any quantified boolean formula ϕ we have

$$
\begin{aligned}
f(\phi) &= 1 \text{ if} && \emptyset \models \phi \\
f(\phi) &= 0 && \text{otherwise}
\end{aligned}
$$

2.5.3. THEOREM (STOCKMEYER AND MEYER). *The QBF decision problem is PSPACE-complete.*

The completeness proof for this problem was presented in 1973 [81, p. 487].

The satisfiability problem for propositional logic is of course a special case of the QBF problem, where we allow only existential quantifiers. Instead of considering whether $p \rightarrow q$ is satisfiable, one can consider the QBF problem of deciding whether $\exists p \exists q (p \rightarrow q)$ holds. Similarly one can consider other restrictions on the number of quantifier series in a QBF problem. The following table lists a few variants.

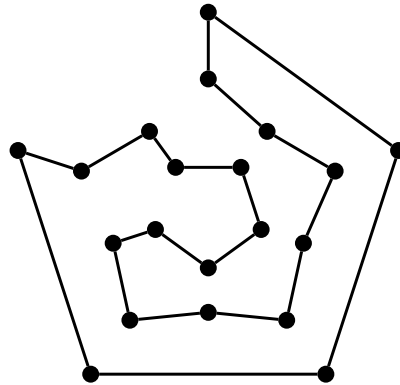|	
$\exists p_1 \ldots \exists p_n \ \phi_q$	satisfiability, in class $\Sigma_1 P$ or NP
$\forall p_1 \ldots \forall p_n \ \phi_q$	tautology, in class $\Pi_1 P$ or co-NP
$\exists p_1 \ldots \exists p_n \ \forall q_1 \ldots \forall q_n \ \phi_q$	$\Sigma_2 P$
$\forall p_1 \ldots \forall p_n \ \exists q_1 \ldots \exists q_n \ \phi_q$	$\Pi_2 P$
$\ldots$	

Figure 2.4: A solution to Hamilton's problem

The classes $\Sigma_n P$ and $\Pi_n P$ are defined using oracles. A Turing machine with an oracle is a machine that is allowed to ask certain difficult questions to a supernatural being (an oracle). The oracle returns the right answer to these questions in one time step. The class $\Sigma_2 P$ contains problems that can be verified by a Turing machine that has an oracle for some NP-complete problem. The problems given above are again complete for these classes: any $\Sigma_2 P$ problem can be reduced to a QBF problem of the form $\exists p_1 \ldots \exists p_n \, \forall q_1 \ldots \forall q_n \, \phi_q$ [81, p. 428].

Open Problem

Intuitively it seems easier to verify a problem than to solve it. For instance it is easy to see that the path given in figure 2.4 is a Hamiltonian cycle, whereas it is less easy to find such a path. It is therefore widely believed [35] that not all NP problems can be solved in polynomial time, and thus that no NP-complete problems can be solved in polynomial time. If we assume that this is the case, then NP-complete problems are *intractable*, and the same holds for the classes $\Sigma_2 P$ and PSPACE. Unfortunately whether $P \neq NP$ is one of the most famous open problems in computer science [26].

In this dissertation we show for several problems that they are NP-complete, $\Sigma_2 P$ complete or PSPACE-complete, and we use this as evidence for the intractibility of these problems. Of course this is only partial evidence, since it is still possible that all problems in NP, contrary to popular belief, are tractable.

Chapter 3

Game Theory

3.1 Overview

One can define game theory as the area of mathematics that is about games. In this case, game theory is older than most people think. In the sixteenth century the mathematician and physician Jerome Cardano wrote his *Book on Games of Chance* [78]. The book opens with a statement describing the various forms that games can take.

> Games depend either on agility of body, as with a ball; or on strength, as with discus and in wrestling; or on industriously acquired skill, as at chess; or on chance, as with dice and with knucklebones; or on both, as fritillus. [78, p. 185]

As Cardano indicates, certain games can depend on both skill and luck at the same time. Nowadays Cardano's book is classified as being about probability theory, as opposed to game theory. The reason for this is that his book is concerned with calculating the probabilities of certain events, but does not consider various strategies and the influence of an opponents' strategy.

The origin of game theory is therefore better placed in the first half of the twentieth century. One of the first mathematical papers that focused on the strategic aspects of games was *Zur Theorie der Gesellschaftspiele* by John von Neumann [118]. The central question of this paper is about the optimal strategies for players in a parlour game.

> n Players, $S_1, S_2, \ldots, S_n$, are playing a given parlour game G. How should one of those players, S_m, play, in order to get a most beneficial result? [118, p. 295]

John von Neumann also co-authored the first book on game theory, *Game theory and economic behaviour*, which appeared in 1944 [74]. The title already

Figure 3.1: Jerome Cardano

indicates that game theory is not merely about recreational games, but can be applied to economics. Von Neumann and Morgenstern see a game as an optimisation problem in which multiple parties simultaneously try to optimize their own outcome. According to them, this is 'nowhere dealt with in classical mathematics' [74, p.11]. This influential book introduced game theory to a wide audience, summarized results that were 'already known, but lacked formal proof' [74, p.6] and gave many game-theoretic terms their meaning.

The following list contains some of the terms introduced by Von Neumann and Morgenstern.

A game: A description of a set of interactions between agents. The description should include which agents can participate, what these agents can do, and what these agents try to achieve.

A play: A specific sequence of interactions between agents. A game consists of many possible plays.

A player: An entity that can make decisions in a certain game.

A move: An action that one can choose, a possibility.

A choice: An action that one has chosen.

A solution: 'plausibly a set of rules for each participant which tell him how to behave in every situation which may conceivably arise.' [74, p31]

The words 'game' and 'play' still have the same meaning in most of the literature. The word 'player' is nowadays often replaced by the synonym 'agent'. A possible reason for this change of terminology is that the word 'player' reminds people of recreational games, whereas game theorists often consider less leisurely situations. An agent can be a human player, but also an organisation or a computer program.

The word 'solution' should be used with caution. Not every game has a unique solution, so the phrase 'the solution of a game' is misleading. A solution is always a set of rules for all players. A set of rules for a single player can be called a *strategy*, and if it is a good set of rules it can be called an 'optimal' or 'rational' strategy.

In order to decide what the best set of rules for an agent is, one must take into account what information the agent has. First of all, it is important to know whether the agent knows exactly which game it is playing. Even the game of chess has several variants, and one can imagine a player who is not sure what the current variant is. Furthermore, in chess an opponent can try to win at all cost, or an opponent can be trying to draw. These two opponents may require different strategies. Von Neumann and Morgenstern boldly state that

> ... we cannot avoid the assumption that all subjects of the economy under consideration are completely informed about the physical characteristics of the situation in which they operate and are able to perform all statistical, mathematical, etc., operations which this knowledge makes possible. [74, p.30].

They call this assumption *complete information*. Another question is whether a player can observe or remember every aspect of the current situation. If this is the case we say that a game has *perfect information*, but if some aspects are hidden the game has *imperfect information*. Chess is a good example of a perfect information game, whereas poker is an imperfect information game.

In games of imperfect information, one has to consider the question of whether a player can remember his own observations, and its own previous actions. If the description of the game indicates that a player can remember both observations and previous actions, then a game has *perfect recall*. If a player can remember all its previous observations the game has *perfect memory*. An interesting but perhaps artificial example of a game with *imperfect recall* was constructed by Von Neumann and Morgenstern when they argued that one can treat teams of players with the same objective as single players. 'Bridge is a two-player game, but the players 1 and 2 do not play it themselves' [74, p. 53]. The four real participants of a bridge game become 'agents', acting on behalf of the two absent players.

Von Neumann and Morgenstern focus on games with two players in which the preferences of the players are exactly opposite. They have less to say about what they call *general games*, which are games with more than two players. The main problem of solving these general games is that in these games, the outcome depends on the possible co-operations between two players. Game theory is nowadays split in two almost unrelated parts: in *cooperative* game theory it is assumed that agents can make binding agreements between each other, and these agreements are enforcable [46]. In this case the exact strategies that are used are not so important. The important aspect is which outcome the agents

should collectively aim for, and how they should split the profits of their collaboration. Throughout this thesis it is not assumed that agents can make binding agreements, and this is called non-cooperative game theory.

The problem of how to treat general games was addressed by John Nash in 1951 [73]. Nash showed that a solution of a game should be a set of strategies such that when all agents use these strategies, no player has any incentive to use another strategy. He called such a solution an equilibrium, and nowadays it is called a *Nash equilibrium*.

The Nash equilibrium is a solution concept that allows the same game to have many solutions. A *solution concept* is a general rule that for each game predicts which strategies are good. Many researchers have argued that the Nash equilibrium allows more solutions than it should. Various refinements have thus been proposed. One of the first was Selten's *subgame perfect (Nash) equilibrium* [79]. This concept makes most sense when applied to perfect information extensive games. Every decision point of such a game can be seen as the starting point of some game, and these games are called the subgames of the original game. Selten argued that a solution should not only be an equilibrium of the original game, but also of every subgame. All finite perfect information extensive games have a subgame perfect equilibrium, and this equilibrium can efficiently be calculated by a procedure named *backward induction*. The procedure is sometimes also called *Zermelo's algorithm*, since Zermelo applied the same procedure in 1913 to analyse chess. Other refinements exist, for instance the trembling hand perfect equilibrium [93], the proper equilibrium [72] and the sequential equilibrium [62].

Even though Von Neumann and Morgenstern limited themselves to the study of complete information games, it was only a matter of time before *incomplete information* games were considered. Harsanyi proved in 1967 that incomplete information games can in certain cases be reduced to complete, imperfect information games [47]. He did not assume that all agents know all other's preferences. Instead he assumed that each agents' preferences depended on the type of the agent. The number of types was limited, and a probability distribution for the types should be commonly known. In that case the incomplete information game can be considered an imperfect information game where in the first move the types of all agents are determined at random according to the given probability distribution. After that the game would proceed as normal.

The applications of game theory have not been limited to the economic realm. Early on it was already realized that game theory could be applied to political and military conflict situations. An early and influential book applying game theory to political science is *The Strategy of Conflict* by Thomas C. Schelling [88]. This book contains various ideas. First of all Schelling showed using experiments that people are able to coordinate their actions. They can do this because real world problems have so-called focal points, even if these points are no longer present in the abstract models of game theory. For example, Schelling asked people to try to meet each other on a given day in a given city, without giving them a specific time

and place, and without allowing them to communicate to each other. One might expect that this is not possible, since there are so many possibilities. However, Schelling's subjects were remarkably successful in meeting each other. Many of them were able to select the same time and place, by reasoning about which points were most obvious to the average person. For instance for a meeting in New York, people often choose to meet at 12.00 at Central Station. Schelling also discusses the rationality of promises and threats. An equilibrium that is not subgame perfect, can for instance contain an unreliable threat. In that case a player threatens to do something in a certain situation, even though it is not rational to really do this action if the subgame in which the action can be done is reached.

More surprising applications were found in biology. This may seem strange, as animals or plants are not usually ascribed rationality or intelligence. The players are thus not assumed to reason about their strategies, but to repeat behaviour that has been successful in the past. The Nash equilibrium can also be applied in these circumstances. In classical game theory attention is focused on the solutions itself, whereas the process by which a solution is reached is ignored. In evolutionary game theory it is also studied how certain strategies are replaced by others, using dynamic systems theory. John Maynard Smith is one of the originators of this field [68]. Evolutionary game theory is a maturing and popular research area [37, 120].

3.2 Strategic Games

Games can be presented in different forms. A very natural but detailed form is as an extensive game. In this form there is a number of decision points in each play of the game, and the outcome is determined by all these decisions. This model is very detailed. Often a less detailed perspective is taken, and thus games are studied in strategic or normal form. In this form, each agent has a number of strategies available at the beginning of the game, and each agent independently picks a strategy. We can calculate the payoff of the game directly, without going into details which actions have been played. The general definition for an n-agent normal form game is the following. We let Σ be the set of all agents, and assume that $\Sigma = \{1, 2, \ldots, n\}$ for some $n > 0$. Thus, in this chapter, and in chapter 8, we use the natural numbers as labels for agents. This is necessary to simplify some of the definitions.

3.2.1. DEFINITION. A *strategic game* G is a tuple $(\Sigma, \{S^X\}_{X \in \Sigma}, \mathfrak{U})$ where $\Sigma = \{1, \ldots, n\}$ is a set of agents, for each $X \in \Sigma$ the set S^X is a set of strategies for agent X, and $\mathfrak{U}^X : (S^1 \times \ldots \times S^n) \to \mathbb{R}$ is a utility function for agent X.

The utility function $\mathfrak{U}^X$ takes a strategy for each agent as input, and return a real number for agent X, which represents that agent's utility. The notation $\mathfrak{U}^X(\vec{s})$

denotes the Xth element of the vector $\mathfrak{U}(\vec{s})$, and thus represents the utility of agent X when the strategy profile $\vec{s}$ is used. One can see $\mathfrak{U}$ as a function that returns a vector of real numbers, one for each agent.

In a strategic game, each agent tries to maximize its utility. They can choose any strategy from their set of strategies, and these sets can be infinite. One can combine the strategies that agents have chosen in a so-called *strategy vector*. A tuple $\vec{s} = (s_1, s_2, \ldots, s_n)$ is a strategy vector for game G if $G = (\Sigma, \{S^X\}_{X \in \Sigma}, \mathfrak{U})$ and for all agents X we have $s_X \in S^X$. In order to manipulate these strategy vectors, two constructs are needed. The construct s_{-j} denotes the vector s with the jth element removed. Thus, $(a, b, c)_{-2} = (a, c)$. The construct $[s, x]$ is used to denote the vector s with x inserted in an appropriate place.

For example $[(a, c), d] = (a, d, c)$, or $[(a, c), d] = (a, c, d)$, depending on what is appropriate. Determining what the appropriate place is can be difficult, therefore the construction $[s, x]$ can only be used if s is of the form s_{-X} for some agent X. For example $[(a, b, c)_{-2}, d] = (a, d, c)$. In practice this means that these constructs can be used to replace one strategy of a strategy vector by another strategy. The combination $[s_{-j}, t_j]$ or, depending on author's preferences, (s_{-j}, t_j) is standard in game theory [79, p. 7].

In many situations, every agent X has a finite number of basic actions m_i to choose from. The total *utility* of a strategy somehow depends on the *payoff* of each action. The number of strategies can still be infinite. The payoff of each action is typically given in the form of a matrix A. We first present the case for two agents, and then extend this to an arbitrary number of agents.

Two player games

3.2.2. DEFINITION. An $m \times n$ *bi-matrix* is a function A such that for each vector (a, b) where $a \in \{1, \ldots, m\}$ and $b \in \{1, n\}$, and for each $X \in \{1, 2\}$, the function A returns a real number $A^X(a, b) \in \mathbb{R}$.

The next table shows how a 2×3 bi-matrix is usually displayed. This matrix can be used to define a game where agent 1 has two actions, and agent 2 has three actions.

$$\begin{pmatrix} A^1(1,1), A^2(1,1) & A^1(1,2), A^2(1,2) & A^1(1,3), A^2(1,3) \\ A^1(2,1), A^2(2,1) & A^1(2,2), A^2(2,2) & A^1(2,3), A^2(2,3) \end{pmatrix}$$

In a pure strategy game, the strategy of both agents consists of a single action. We can use the bi-matrix A given above to define a pure strategy game $G = (\{1, 2\}, (\{1, 2\}, \{1, 2, 3\}, T), \mathfrak{U})$. The set of agents would be $\{1, 2\}$, the strategy set of agent 1 would be $\{1, 2\}$, and the strategy set of agent 2 would be $\{1, 2, 3\}$, and the utility function would be defined by $\mathfrak{U}(a, b) = (A^1(a, b), A^2(a, b))$.

In a mixed strategy game, a strategy consists of a probability for each action. Thus a mixed strategy game G based on the bi-matrix A would be $G =$

$(\{1, 2\}, (S^1, S^2), \mathfrak{U})$, where $S^1 = \{(a, b) | a, b \in [0, 1], a + b = 1\}$ and $S^2 = \{(a, b, c) | a, b, c \in [0, 1], a + b + c = 1\}$. An example strategy for player 1 would be $(0.25, 0.75)$. If the agent follows this strategy it should take action 1 twenty-five percent of the time, and action 2 seventy-five percent of the time. The utility function $\mathfrak{U}$ returns the expected payoff, and is defined as

$$\mathfrak{U}((a, b), (c, d, e)) =$$
$$(acA^1(1, 1) + adA^1(1, 2) + aeA^1(1, 3) + bcA^1(2, 1) + bdA^1(2, 2) + beA^1(2, 3),$$
$$acA^2(1, 1) + adA^2(1, 2) + aeA^2(1, 3) + bcA^2(2, 1) + bdA^2(2, 2) + beA^2(2, 3))$$

Multi-player games

3.2.3. DEFINITION. An $m_1 \times m_2 \ldots \times m_n$ *multi-matrix* is a function A such that for each vector $a_1 a_2 \ldots a_n$ where $a_Y \in \{1, \ldots, m_Y\}$ for all $Y \in \{1, \ldots, n\}$, and for each $X \in \{1, \ldots, n\}$, the function A returns a real number $A^X(a_1 i_2 \ldots a_n) \in \mathbb{R}$.

The term $A(a_1, a_2 \ldots a_n)$ denotes a vector $v \in \mathbb{R}^\Sigma$ such that $v_1 = A^1(a_1 a_2 \ldots a_n)$, $v_2 = A^2(a_1 a_2 \ldots a_n)$ etcetera. A *bi-matrix* is a multi-matrix where $n = 2$. The notation $\mathbb{R}^\Sigma$, which is for instance used by Gamut [34, p. 84], denotes the set of all functions $f : \Sigma \to \mathbb{R}$. The set Σ is often finite and is assumed to have some kind of natural ordering, such as the set $\{1, 2, 3\}$. If this is the case then the elements $f \in \mathbb{R}^\Sigma$ can be seen as tuples. Each element f would correspond to the tuple $(f(1), f(2), f(3))$. Thus, the set $\mathbb{R}^{\{1,2,3\}}$ is isomorphic to $\mathbb{R}^3$.

For a given multi-matrix A one can in fact define different games. The simplest type of game is the *pure strategy game*. In this game, the strategy of each agent X consists of a single action a_X and the payoff is then $A(a_1 \ldots a_n)$. This definition does not allow agents to play randomly.

In a mixed strategy game, the strategy of an agent is a probability distribution over the available actions. The utility is the expected (weighed average) value of A. This type of game is defined below. The shorthand $A_i^X(\vec{s})$ denotes the expected payoff of action i for agent X when the other agents use strategies from $\vec{s}$. It can be defined in the following way. Define the set $V_i^X = \{\vec{v} | \forall Y \in \Sigma \setminus \{X\} : v_Y \in S^Y, v_X = i\}$. Thus, this set contains the pure strategy profiles in which agent X selects action i. For instance if A is a 2×2 multi-matrix we have $V_2^1 = \{(2, 1), (2, 2)\}$.

3.2.4. DEFINITION. For any multi-matrix A, agent X and action i, and vector $\vec{s}$ of mixed strategies s_Y for each agent Y, we define the expected payoff of action i for agent X by:

$$A_i^X(\vec{s}) = \sum_{(v_1 \ldots v_n) \in V_i^X} ((s_1)_{v_1} \cdots (s_{X-1})_{v_{X-1}} (s_{X+1})_{v_{X+1}} \cdots (s_n)_{v_n}) A^X(\vec{v})$$

In this definition, the element v_1 denotes a possible action for player 1, s_1 is the strategy of player 1, and therefore $(s_1)_{v_1}$ denotes the probability that player 1 will play action 1. Although this definition is hard to read, in practice it is not hard to see how this expected payoff is computed. To give an example, let A be again a 2×2 multi-matrix, assume that the first player plays action 1 with ninety percent probability, and that the second player plays the first action with forty percent probability. Then the expected payoff of action 2 for agent 1 is computed in the following way.

$$A_2^1(((0.9, 0.1), (0.4, 0.6))) = 0.4A^1(2, 1) + 0.6A^1(2, 2)$$

The following set $\mathbf{P}^m$ is used in the definition of mixed strategies. It contains vectors that sum up to one. These vectors can be seen as specifying probabilities for all actions.

$$\mathbf{P}^m = \{x \in [0, 1]^m \mid \sum_i x_i = 1\}$$

3.2.5. DEFINITION. Let A be an $m_1 \times m_2 \ldots \times m_n$ multi-matrix. The *mixed strategy game* $M_m(A)$ of A is a tuple $(\Sigma, \{S^X\}, \mathfrak{U})$ where $\Sigma = \{1, 2, \ldots, n\}$, the strategy sets are $S^X = \mathbf{P}^{m_X}$, and $\mathfrak{U}^X(\vec{s}) = \sum_i s_i^X A_i^X(\vec{s})$.

Recall that the notation $\mathfrak{U}^X(\vec{s})$ denotes the Xth element of the vector $\mathfrak{U}(\vec{s})$. It represents the utility of agent X when the strategy profile $\vec{s}$ is used.

The fact that agents can play mixed strategies is explicitly defined in this definition of a mixed strategy game. We assume that all agents are equipped with random number generators (coins, dice or whatever) so that they can randomize their behaviour exactly as specified in their strategy. This definition of a mixed strategy game is such that each mixed strategy game is in fact a strategic game.

For the next definition we need the function argmax that returns all inputs that maximize a given function. $\text{argmax}_x f(x) = \{x \mid \neg \exists y : f(x) < f(y)\}$ We use the function argmax to define what a 'good' strategy is: A good strategy is a strategy that returns a maximal utility. The function b^X returns the best response strategies for agent X for a given game and strategy vector.

3.2.6. DEFINITION. Let $(\Sigma, \{S^X\}_{X \in \Sigma}, \mathfrak{U})$ be a game and $\vec{s} \in (\prod_X S^X)$ a strategy profile. The best response $b(\vec{s}) = b^1(\vec{s}) \times \cdots \times b^n(\vec{s}))$ is defined by

$$b^X(\vec{s}) = \text{argmax}_t \mathfrak{U}^X((s_{-X}, t))$$

The set $b(\vec{s})$ thus contains the strategy vectors t such that t_X is optimal if all opponents Y use the strategy s_Y. We could assume that the strategy of the opponents is fixed. The set $b^X(\vec{s})$ is the set of best decisions for agent X.

When playing a game an agent cannot always predict what strategy the other agents use, because the other agents might want to change their strategy once

they learn that X uses a strategy in $b_X(\vec{s})$. The notion of a best response is therefore not a solution concept in itself. One can however search for fixed points in the best response function, and this is called a Nash equilibrium.

3.2.7. DEFINITION. Let $(\Sigma, \{S^X\}_{X \in \Sigma}, \mathfrak{U})$ be a game and $\vec{s} \in (\prod_X S^X)$ a strategy profile. The vector $\vec{s}$ is a *Nash equilibrium* iff $\vec{s} \in b(\vec{s})$.

Every mixed strategy game has at least one Nash equilibrium [73]. There has been some discussion in the literature whether the notion of a Nash equilibrium needs to be refined. Several refinements have been proposed [72], but none of them has the appealing simplicity of the Nash equilibrium.

A special class of games for use in logic are the *win-loss games*. In these games the utility functions only takes two values, which can be associated with winning and losing. Typically these values are 1 and 0. The utility function can then be specified by stating what the winning positions are. These sets can then be specified by stating a formula, so that a position is winning if it makes the formula true.

Another special class are the *constant-sum games*. We define this property only for games with exactly two players. A game $(\{A, B\}, \{S^A, S^B\}, \mathfrak{U})$ is a constant-sum game if there is a constant $c \in \mathbb{R}$ such that for any strategy profile (σ_A, σ_B) it is the case that $\mathfrak{U}^A((\sigma_A, \sigma_B)) + \mathfrak{U}^B((\sigma_A, \sigma_B)) = c$. If the constant c is 0 then we call it a *zero-sum game*. The next bi-matrix A defines a constant-sum game where the constant is 2.

$$\begin{pmatrix} 1,1 & 0,2 & 2,0 \\ 2,0 & 1,1 & 0,2 \end{pmatrix}$$

Examples of strategic games can be found in many game theory text books. All classical examples can be found in the primer by Osborne and Rubinstein [79], but more playful examples are given by Binmore [11]. Textbooks on evolutionary game theory such as Gintis' [37, 120] also make much use of strategic form games. In each case the examples of strategic games are often presented in the form of finite multi-matrices, and this can give readers the false impression that strategic games are always finite, small and simple. In chapter 8, it is shown that there are many other strategic games that do match definition 3.2.1, but are not pure or mixed strategy games.

3.3 Extensive Games

In an extensive game the agents have to make sequences of choices that ultimately lead to an outcome. There are multiple decision points and at each decision point one of the agents has to decide what to do next. The rules of the game specify exactly which sequences of actions are legal. We represent these rules in a very simple way, by listing all sequences that are allowed.
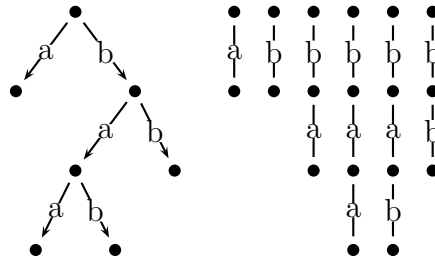
Figure 3.2: A game tree and a set of sequences

3.3.1. DEFINITION. A non-empty set H of sequences is a *sequence set* if for any sequence h and action a it is the case that $ha \in H$ implies $h \in H$. For any sequence set H and $h \in H$ we define the set of next actions $A(H, h) = \{a | ha \in H\}$ and the set of terminal sequences $Z(H) = \{h \in H | A(H, h) = \emptyset\}$.

Another term for a *sequence set* could be a non-empty *prefix-closed* set.

If H is a sequence set then one can define a graph $G = (V, E)$ by defining $V = H$ and $E = \{(h, ha) | ha \in H\}$. This graph is a tree with the empty sequence ϵ as root. Such a tree is often called a game tree. In figure 3.2 a game tree and the corresponding set of sequences is displayed.

Extensive games can be played as perfect information games. In this case every agent can distinguish all sequences, and thus the agent can select the action that is best for that specific decision point.

3.3.1   Perfect Information

In order to play an extensive game, one must know which agent can influence which decision. Therefore, we augment the game tree with a function *turn* that returns the agent that is in control of a certain history.

3.3.2. DEFINITION. A *game form* F is a tuple $F = (\Sigma, H, turn)$, where Σ is a finite set of agents, H is a finite sequence set and *turn* is a function *turn* : $H \setminus Z(H) \to \Sigma$.

A game form by itself is often not what one needs. One typically want a game form with a utility function (if you are a game theorist) or a game tree annotated with atomic propositions (if you are a logician). In certain cases you might want both. We use the word 'interpreted' to indicate a structure that is labeled with atomic propositions. If a structure contains utilities it is called an *extensive game*, otherwise a *game form*.

name	contains
(extensive) game form	tree
interpreted game form	tree, atomic propositions
extensive game	tree, utility function
interpreted game	tree, utility function, atomic propositions

The word 'extensive' emphasizes that we deal with games in which the order of moves is explicitly present. We omit it if it is not necessary, and thus we speak of game forms rather than extensive game forms. An *extensive game form* is thus synonymous to *game form*, and so are *interpreted extensive game form* and *interpreted extensive game*.

3.3.3. DEFINITION. An *extensive game* F is a tuple $F = (\Sigma, H, turn, \mathfrak{U})$, such that $(\Sigma, H, turn)$ is a game form and $\mathfrak{U} : Z(H) \times \Sigma \to \mathbb{R}^{\Sigma}$.

The function $\mathfrak{U}$ is called a utility function. It returns the utility for each agent and each agent tries to maximize its utility.

3.3.4. DEFINITION. Let $(\Sigma, H, turn)$ be a game form. A *pure strategy* σ for agent X in game form F is a function σ with domain $\{h \in H | turn(h) = X\}$ such that $\sigma(h) \in A(H, h)$.

The notion of a strategy can be extended to strategies for coalitions $\Gamma \subseteq \Sigma$. A pure *coalition strategy* σ_{Γ} for Γ is a function f with domain $\{h \in H | turn(h) \in \Gamma\}$ such that $\sigma(h) \in A(H, h)$.

3.3.5. DEFINITION. Let $F = (\Sigma, H, turn)$ be a game form. A *behavioural strategy* σ for agent $X \in \Sigma$ in game form $(\Sigma, H, turn)$ is a function σ with domain $\{h \in H | turn(h) = X\}$ such that for each h, $\sigma(h)$ is a probability distribution over $A(H, h)$.

There is a difference between mixed strategies and behavioural strategies [79]. The concept of mixed strategies applies to strategic games. A mixed strategy is itself a probability distribution. A behavioural strategy is a function that returns probability distributions for nodes of an extensive game. For imperfect information games without perfect recall the two kinds of strategies are not equivalent [79].

3.3.6. DEFINITION. Let $F = (\Sigma, H, turn)$ be a game form. A *nondeterministic strategy* σ for agent X in game form F is a function σ with domain $\{h \in H | turn(h) = X\}$ such that $\sigma(h)$ is a non-empty subset of $A(H, h)$.

The notion of a strategy for a coalition $\Gamma \subseteq \Sigma$ can also be introduced for behavioural and nondeterministic strategies.

For each game form F, we define $S_p^X(F)$ to be the set of pure strategies of agent X in F, the set $S_b^X(F)$ to be the set of behavioural strategies of X in F, and

$S_n^X(F)$ the set of nondeterministic strategies.   The notion of behavioural strategy
is a more general notion than that of a pure strategy. For any pure strategy σ
one can find a behavioural strategy σ' by defining $\sigma'(h)(a) = 1$ if $a = \sigma(h)$, and 0
otherwise. Thus, one action gets probability one and the other actions get probability zero. For each behavioural strategy σ one can define a nondeterministic
strategy σ' by defining $\sigma'(h) = \{a \in A(H,h)|\sigma(h)(a) > 0\}$. Thus, σ' returns that
actions that have a nonzero probability in σ. A nondeterministic strategy is a less
detailed description of a behavioural strategy, in which the exact probabilities are
omitted. We use nondeterministic strategies when the exact probabilities are not
important.

Extensive games can be reduced to strategic games. This observation was
already made by Von Neumann and Morgenstern [74]. Below, we do this for both
pure strategy and behavioural strategy games.

3.3.7. DEFINITION. Let $(\Sigma, H, turn, \mathfrak{U})$ be an extensive game. The corresponding pure strategy strategic game is $(\Sigma, \{S_p^X\}_{X \in \Sigma}, \mathfrak{U}')$, where $\mathfrak{U}'(\vec{s})$ is defined by
$\mathfrak{U}'(\vec{s}) = \mathfrak{U}(r(\vec{s}, \epsilon))$ where

$$r(\vec{s}, h) = \begin{cases} h & \text{iff } h \in Z(h) \\ r(\vec{s}, (h, s_X(h))) & \text{iff } turn(h) = X \end{cases}$$

3.3.8. DEFINITION. Let $(\Sigma, H, turn, \mathfrak{U})$ be an extensive game. The corresponding behavioural strategy strategic game is $(\Sigma, \{S_b^X\}_{X \in \Sigma}, \mathfrak{U}')$. The function $\mathfrak{U}'(\vec{s})$
is defined by $\mathfrak{U}'(\vec{s}) = \sum_{h \in Z(h)} p(\vec{s}, h) \cdot \mathfrak{U}(h)$ where

$$p(\vec{s}, \epsilon) = 1$$
$$p(\vec{s}, ha) = \vec{s}(h)(a) \cdot p(\vec{s}, h)$$

A Nash equilibrium of an extensive game is defined as a Nash equilibrium of
the corresponding strategic game. It should be clear from the context whether
the corresponding strategic game is the pure strategy game or the behavioural
strategy game. In figure 3.3, a small extensive game E_1 is displayed. In this game
two PhD students Alice (A) and Bob (B) have the choice of cleaning their shared
office (action c), or to ignore the mess (action i). Since Alice arrives first, she has
to decide first what she will do. If she does not clean the office, Bob is faced with
the same choice. A clean office is worth 2 utility units, but cleaning the office
costs an agent 1.

Each agent has two pure strategies called σ_c and σ_i, listed in the next table.

$_A\backslash^B$	σ_c	σ_i
σ_c	$(1, 2)$	$(\mathbf{1}, \mathbf{2})$
σ_i	$(\mathbf{2}, \mathbf{1})$	$(0, 0)$
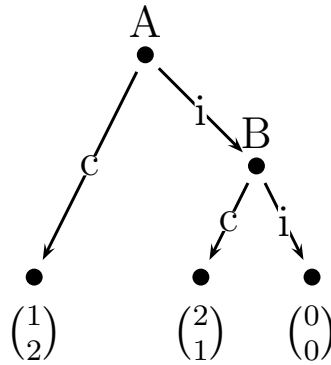
Figure 3.3: To clean or not to clean: Game E_1

The two Nash equilibria of this game are indicated in bold. In the lower left equilibrium, A ignores the problem and B cleans the room (payoff $(2, 1)$). In the other equilibrium B plans to ignore the problem and A cleans the room (payoff $(1, 2)$). Are both of these Nash equilibria equally good? Many people tend to say 'no'. The reasoning is as follows. Each decision node of an extensive game can be seen as the starting point of a smaller extensive game. Such a game is called a subgame of an original game. One would expect each agent to act rationally in each subgame. If an equilibrium has such a property, it is called a subgame perfect equilibrium.

3.3.9. DEFINITION. Let $F = (\Sigma, H, turn)$ be a game form, and $h \in H$. The sub-'game form' of F starting at h is defined as $\text{subg}(F, h) = (\Sigma, H', turn')$ where $H' = \{h' | h \cdot h' \in H\}$ and $turn'$ is the same as $turn$ but with the domain restricted to H'

This definition can be extended to interpreted game forms, games and interpreted games in a straightforward way. The set of all subgames of a given game is defined as the set $\text{allsub}((\Sigma, H, turn, \mathfrak{U})) = \{\text{subg}((\Sigma, H, turn, \mathfrak{U}), h) | h \in H\}$.

3.3.10. DEFINITION. Let $G = (\Sigma, H, turn, \mathfrak{U})$ be an extensive game, f a function that reduces extensive games to strategic games. The strategy profile $\vec{s}$ is a *subgame perfect equilibrium* if for all subgames G' of G it is the case that $\vec{s}$ is a Nash equilibrium of $f(G')$.

The example game of figure 3.3 has one subgame perfect Nash equilibrium in which agent B cleans the office. The other Nash equilibrium is not subgame perfect, since it is not optimal for agent B to choose the action ignore if A does not clean the room.

3.3.11. THEOREM (KUHN). *Every perfect information game has at least one subgame perfect equilibrium in pure strategies [64].*
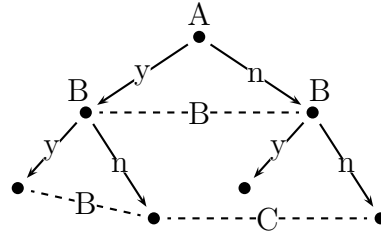
Figure 3.4: An imperfect information game form F_I

If the utility function $\mathfrak{U}$ of a game G is such that for any agent X and history $h \in Z(h)$ we have that $h \neq h' \Rightarrow \mathfrak{U}^X(h) \neq \mathfrak{U}^X(h')$ then the optimal action a in a certain situation is always unique, and thus the subgame perfect equilibrium is unique.

3.3.2   Imperfect Information

In imperfect information games it is possible that an agent X does not see the difference between histories h and h'. Thus when agent X is in the situation represented by h, it considers it possible that it might be in the situation h'. We use equivalence relations $\sim_X$ to store this information, and write $h \sim_X h'$ to indicate this lack of information. For instance in a game of Poker, agent X may not know the hand of cards that an opponent Y holds. If the only difference between situations h and h' would be the hand of cards of Y, then $h \sim_X h'$.

3.3.12. DEFINITION. An *imperfect information game form* F is a tuple $F = (\Sigma, H, turn, \sim)$, where Σ is a finite set of agents, H is a non-empty, prefix-closed set of finite sequences, *turn* is a function $turn : H \setminus Z(H) \to \Sigma$, for each $X \in \Sigma$ the relation $\sim_X \subseteq H \times H$ is an equivalence relation between states. Furthermore $\sim_X$ has to satisfy the following condition: if $turn(h) = X$ and $h' \sim_X h$ then also $turn(h') = X$ and $A(H, h) = A(H, h')$.

An example imperfect information game form is displayed in figure 3.4. In this figure, the lack of information is indicated by dashed lines. Hence agent B cannot distnguish the histories y and n, and thus has no information what action agent A has chosen: $y \sim_B n$.

It often happens that an agent X has to make decisions in situations h and h' that it cannot distinguish, i.e. $h \sim_X h'$. Since X cannot see a difference between these situations, strategies for X must prescribe the same behaviour in both situations. The definitions of the different strategies thus have to be modified.

3.3.13. DEFINITION. A strategy σ for X in game form $F = (\Sigma, H, turn, \sim, \pi)$ is *uniform* if for all $h \sim_X h'$ it holds that $\sigma(h) = \sigma(h')$.

This modification can be applied to all kinds of strategies: pure, behavioural and nondeterministic. For imperfect information games we only consider uniform strategies, even if the word uniform is not mentioned.

An example of a uniform pure strategy σ_B for agent B in game form F_I would be $\sigma_B(y) = \sigma_B(n) = y$. The strategy $\sigma_B(y) = y$ and $\sigma_B(n) = n$ would not B uniform, and thus this would not be an acceptable strategy.

For games of imperfect information we use a different notion of a subgame.

3.3.14. DEFINITION. Let $F = (\Sigma, H, turn)$ be a game form, and $h \in H$. Let $F' = (\Sigma, H', turn', \sim')$ where $H' = \{h'|h \cdot h' \in H\}$ and $turn', \sim'$ are the same as $turn, \sim$ but with their domain restricted to H' instead of H. The structure F' is a *subgame form* iff for all $j \in H'$ and $X \in \Sigma$ it is the case that $j \sim_X j'$ implies $j' \in H'$.

Any decision node h' in a subgame should only be indistinguishable from histories that are also in the subgame. This means that all agents 'know' that they are in the subgame, and anything that is outside the subgame does not influence the agents' decisions. Some imperfect information games have subgames, but others do not have any because certain information is private from start to end. The notion of a subgame perfect equilibrium is therefore not often used on imperfect information games. Instead, people use extensions such as the sequential equilibrium or the trembling hand perfect equilibrium [79].

It is often reasonable to suppose that agents remember information. This means that if two histories h, h' can be distinguished, then any pair of extensions of these histories can also be distinguished: $h \nsim_X h'$ implies $h \cdot h_2 \nsim_X h' \cdot h'_2$. This property is called *perfect memory*. Another useful assumption to make is that agents remember their own decisions. Thus, if $turn(h) = X$ then $ha \sim_X ha'$ implies $a = a'$. If a game form has this property and perfect memory then the game form has *perfect recall*.

Examples

Agent C in game form F_I does not ave perfect memory. The agent can distinguish y and n (thus $y \nsim_C n$), but it cannot distinguish yn and nn (hence $yn \sim_C nn$).

In the game form F_I, agent B can forget its own action, since $yy \sim_B yn$. Thus the game form F_I does not have the property and hence does not have perfect recall.

The assumption of perfect recall makes it easier to compute Nash equilibrium strategies. In fact, for two player constant-sum games with perfect recall, one can find Nash equilibrium strategies in polynomial time. If perfect recall is not assumed, the problem is $\Sigma_2 P$-complete and thus believed to be intractable [57].

3.4   Existing Work on Logic and Games

Logicians and game theorists often work on the same problems, and their respective fields are becoming more and more connected due to the efforts of many researchers in both fields [97].

It is surprising to see that logic and games can be connected in many different ways.

Logic can be used to understand and make transparent the reasoning behind game-theoretic solution concepts. In this case logic is a tool used to understand the assumptions made in game theory. The focus is often on the knowledge that is required for agents in order to ensure that a certain outcome is reached. Examples of work in this direction include Aumann's discovery that a Nash equilibrium can arise without common knowledge [7], or De Bruin's analyses of iterated elimination of dominated strategies [30].

Epistemic logic and its extensions can be used to understand situations that occur in imperfect information games. A typical example is the game of Clue. In order to play this board game one must reason about knowledge of cards, and thus this game lends itself well to modeling using *dynamic epistemic logic* [105]. Probabilistic epistemic logic is useful for modeling games in which probabilities play a role [59]. The dynamic epistemic logic approach can be extended to include even complex game actions such as cheating and deceiving other players [8]. The focus in this area of research is on the imperfect information of players.

The subgame perfect Nash equilibrium, also known as backward induction, is the most popular solution concept for extensive games of perfect information. Modal logicians are always interested in determining the expressivity of modal languages, and at least two authors have thus determined what language one needs to characterise this solution concept. Bonanno [14] has given a characterisation of backward induction using branching time temporal logic. Harrenstein [45] has used a different multi-modal logic.

Since a game form lacks preferences, one cannot ask what agents want in a game form, or which agents will win in a game form: the concept of winning is not defined if there are no preferences. One can however investigate the *effectivity* of coalitions of agents: whether agents can, by choosing the right strategy, ensure that a certain outcome holds. This can be formalized in logic. Pauly's *coalition logic* [85] and Van Benthem's logic for process models [99] do exactly this. These logics are suitable for reasoning about what agents, or coalitions of agents, can achieve by their choice of strategy. This is called effectivity. Since both these logics are closely related to the work in this chapter, they are introduced in more detail in section 3.4.1.

In order to study games instead of game forms, one must introduce the notion of preferences. This leads to the idea of *preference logic*. Our work on preference logic goes back to Von Wright [119]. Von Wright used an intuitive approach, based on identifying likely axioms. Van Dalen [100] subsequently proved several

completeness results for certain basic semantics. Rescher [86] also developed a semantic for Von Wright's language, this time based on the fact that the value of a formula ϕ (in a certain model) should be the average of the utilities of all worlds in the model that satisfy ϕ. This is certainly an interesting idea, but hard to characterize in an axiomatic way. Chisholm and Sosa [20] investigated the philosophical aspects of preference logic further. A recent overview is presented by Hansson [44, 320].

What these sources have in common is that they use a binary construction $\phi P \psi$ in order to express preferences. This is different from the usual modal logic approach, which is normally based on an unary operator $\Box \phi$ (see for instance [45]). The binary approach seems to correspond better with natural language, where one can say things such as "I prefer coffee over tea". The sort of relations that is used to indicate preferences can also be used to express relative likelihood [42]. As a result, the technical results stated in terms of likelyhood can be applied to preferences. Huang [52] also introduces preference logics, in order to model agents with bounded rationality.

Another way to combine logic and games is to interpret a formula as a game between two players, one of which wants to reach a 'true' outcome, the other one a 'false' outcome. This idea has been used in the interpretation of *independence friendly logic* (IF logic). This logic was introduced by Hintikka with the bold goal of replacing first order logic as the primary logic of scientific discourse [49]. This idea has not materialized, partly because IF logic is quite complicated, and has several interesting 'features' (that some call 'bugs'). It has a compositional semantics but it is not the simplest semantics [50], for this logic the falsity conditions are not the mirror image of the truth conditions [31], and one can question whether the interpretation of IF is faithful to game theory [94].

3.4.1 Coalition Logics

Coalition logic [84, p.46] is a logic for reasoning about effectivity in general game frames. The language is very similar to EFL (to be defined on page 58), and thus coalition logic is examined here in detail first.

3.4.1. DEFINITION. Assume that finite sets P, Σ are given. A coalition logic formula ϕ is defined by the following rule. In these rules $p \in P$ and $\Gamma \subseteq \Sigma$.

$$\phi ::= p \mid \phi \to \phi \mid \bot \mid [\Gamma]\phi$$

Coalition logic is interpreted over general game frames, which combine features of strategic games and extensive games. A general frame is similar to a tree, but at each decision node all agents have to select an action, like in a strategic game. The next state depends on the actions chosen by all agents. This semantics is described in detail in Pauly's dissertation [84, p.46]. This class of models is more general than the interpreted game forms that are used for EFL. The interpretation

of a coalition logic formula ϕ over a general game frame G in state s is defined below.

3.4.2. DEFINITION. A *coalition model* M is a tuple $(S, \{E_\Gamma | \Gamma \subseteq \Sigma\}, \pi)$ where S is a set of states, $\pi : S \to P$ an interpretation function, and for each coalition Γ we have $E_\Gamma : S \to 2^{2^S}$ is a function that to each $s \in S$ assigns a set of sets of states. The functions E_Γ must be monotonic, i.e. if $T \in E_\Gamma(s)$ and $T \subset T'$ then $T' \in E_\Gamma(s)$

The following rules define the interpretation of coalition logic formulas over pointed models M, s, where $s \in S$ is a state in the coalition model M

$$
\begin{array}{lll}
G, s \models \bot & & \text{never} \\
G, s \models p & \text{for} & p \in \pi(s) \\
G, s \models \phi \to \psi & \text{iff} & \text{not } G, s \models \phi \text{ or } G, s \models \psi \\
G, s \models [\Gamma]\phi & \text{iff} & \phi^G \subseteq E_\Gamma(s) \\
& & \text{where } \phi^G = \{t \in S | G, t \models \phi\}
\end{array}
$$

This logic does not make a distinction between intermediate states and end states or outcomes. The internal structure of the protocol can therefore be described in coalition logic, and one can use satisfiability for protocol verification.

The fact that coalition logic does take intermediate states into account, means that one should read formulas from EFL and coalition logic in a different way. In chapter 4 we see that the EFL formula $[A]\square p$ expresses that A can enforce that a p outcome is reached. Syntactically the closest coalition logic formula is $[A]p$. This formula means A can make p true in the next state. If the next state is not an outcome state, then this formula does not say anything about which outcomes A can reach. In order to express something about outcomes, one can however use extended coalition logic. This logic is an extension of coalition logic with operators $[\Gamma^*]\phi$ and $[\Gamma^\times]\phi$. The first operator expresses that ϕ can eventually be reached by Γ, and the second operator $[\Gamma^\times]\phi$ expresses that Γ can keep ϕ true in the entire future. The first operator can be used to refer to outcome states, and indeed Pauly introduces a special notation to do so.

$$
[\Gamma^t]\phi \stackrel{\text{def}}{=} [\Gamma^\times]([\emptyset]\bot \wedge \phi)
$$

Proof Theory

There are sound and complete proof systems for several variantes of Coalition logic. The full details are given by Pauly [85, 85]. Here we present as an example a sound and complete proof system for coalition models that have a weakly playable effectivity function.

3.4.3. DEFINITION. An effectivity function $E_\Gamma : S \to 2^{2^S}$ is Γ-*maximal* iff for all T, if $S \setminus T \notin E_{\Sigma \setminus \Gamma}$ then $T \in E_\Gamma$

3.4.4. DEFINITION. An effectivity function $E_\Gamma : S \to 2^{2^S}$ is *superadditive* iff for all $T_1, T_2, \Gamma_1, \Gamma_2$ such that $\Gamma_1 \cap \Gamma_2 = \emptyset$, if $X_1 \in E_{\Gamma_1}$ and $X_2 \in E_{\Gamma_2}$ then $X_1 \cap X_2 \in E_{\Gamma_1 \cup \Gamma_1}$

3.4.5. DEFINITION. An effectivity function $E_\Gamma : S \to 2^{2^S}$ is *weakly playable* if it satisfies the following five conditions: (1) $\emptyset \notin E(\Sigma)$, (2) if $\emptyset \in E(\Gamma)$ and $\Gamma' \subset \Gamma$ then $\emptyset \in E(\Gamma')$, (3) If $\emptyset \notin E(\emptyset)$ then $S \in E(\Gamma)$ for all $\Gamma \subseteq \Sigma$, (4) E is Σ-maximal and (5) E is superadditive.

These following axioms are sound on coalition models with weakly playable effectivity functions.

$$\models \neg[\Sigma]\bot \qquad\qquad (N\bot)$$
$$\models [\Gamma \cup \Gamma_2]\bot \to [\Gamma]\bot \qquad\qquad (\bot)$$
$$\models \neg[\emptyset]\bot \to [\Gamma]\top \qquad\qquad (\top)$$
$$\models \neg[\emptyset]\neg\phi \to [\Sigma]\phi \qquad\qquad (N)$$
$$\models ([\Gamma_1](\phi_1) \land [\Gamma_2](\phi_2)) \to [\Gamma_1 \cup \Gamma_2](\phi_1 \land \phi_2) \qquad\qquad (S)$$
$$\text{where } \Gamma_1 \cap \Gamma_2 = \emptyset$$

There are two reasoning rules for coalition logic. The first is Modus Ponens, the second one is called *Monotonicity*.

$$\frac{\phi \leftrightarrow \psi}{[\Gamma]\phi \leftrightarrow [\Gamma]\psi}$$

The proof system consisting of these two rules and the five axioms is sound and complete on coalition models with weakly playable effectivity functions [84, p. 55].

Coalition logic can be used for reasoning about extensive games. In that case the following formula, valid on extensive game forms, should be added as an axiom.

$$[\Sigma]\phi \to \bigvee_{X \in \Sigma} [X]\phi$$

This formula can be read as saying that if something can be done, it can be done by one of the agents.

Pauly presents several completeness proofs for different classes of models, and for a detailed presentation we refer to his dissertation [84, p. 54]. The most general proof is similar to the standard completeness proof of modal logic based on a canonical model.

For extended coalition logic, a complete axiomatization is also given, and this axiomatisation was later used by Goranko [40, 41] to develop a complete proof system for ATL. These proof systems are more complex than the system given here for EFL, since these proof systems deal with systems with infinite runs. Our logic EFL is only a small fragment of these logics.

3.4.2   Power Level Logic and Bisimulation

Another language for reasoning about what agents can effect has been introduced by Van Benthem [99]. Van Benthem introduces the notation $\{G, X\}\phi$, where X is a single agent and G refers to a strategic game. This operator is interpreted in the following way.

$$M, s \models \{G, X\}\phi \;\Leftrightarrow\; \exists S : \; \rho_G^X s, S \wedge \forall t \in S : M, t \models \phi$$

The relation $\rho_G^X s, S$ is interpreted as saying that agent X has a strategy for playing game G from state s onwards such that all next states are within the set S. Van Benthem remarks that the argument G can be omitted if the game does not change, and this makes the language even more similar to EFL.

This language allows one to write $\{A\}\{B\}\phi$. This does not add to the expressivity of the language, since the second operator can be omitted without changing the meaning of the formula.

$$\models \{A\}\{B\}\phi \leftrightarrow \{A\}\phi$$

Variants on this language, presented in the same paper [99], combine this language with features of coalition logic.

As explained on page 18, the notion of bisimulation is used in standard modal logic in order to decide when two models are the same. For the logic described here, one cannot use the same definition directly. Instead Van Benthem defines the notion of a power bisimulation.

3.4.6. DEFINITION. Suppose two models M and M' with sets of worlds W, W' are given. A binary relation $E \subseteq W \times W'$ is a *power bisimulation* if the following conditions hold.

- If $(x, y) \in E$ then they satisfy the same atomic propositions

- For any agent X, if $(x, y) \in E$ and $\rho_M^X xU$ then there is a set V such that $\rho_{M'}^X yV$ and $\forall v \in V \; \exists u \in U : \; (u, v) \in E$

- Vice versa: For any agent X, if $(y, x) \in E$ and $\rho_M^X yV$ then there is a set U such that $\rho_{M'}^X xU$ and $\forall u \in U \; \exists v \in V : \; (u, v) \in E$

This definition captures the idea that agents have the same abilities in the models M and M'. Two models that are power bisimilar satisfy the same formulas [99].

3.4.3   Alternating-time Temporal Logic

Alternating-time Temporal Logic (*ATL*) is a multi-agent extension of CTL [6]. The language of ATL contains temporal operators similar to CTL, but instead of the quantifiers $\forall$ and $\exists$ that appear in CTL, strategy operators $\langle\!\langle \Gamma \rangle\!\rangle$ are used, where Γ can be any set of agents.

3.4.7. DEFINITION. Let Σ be a set of agents, and P a set of atomic propositions. The logic ATL contains formulas ϕ generated by the following rule. In this rule, p is a typical element of P and $\Gamma \subseteq \Sigma$

$$\phi ::= p \mid \phi \to \phi \mid \bot \mid \langle\!\langle \Gamma \rangle\!\rangle \psi$$
$$\psi ::= \Box \phi \mid \phi \mathcal{U} \phi$$

The meaning of a formula $\langle\!\langle \Gamma \rangle\!\rangle \phi$ is that the agents in Γ can use a strategy such that ϕ holds.

This logic is interpreted over alternating transition systems [6]. These are defined as tuples $(P, \Sigma, Q, \pi, \delta)$. As usual P is a set of atomic propositions and Σ a set of agents. The set Q is a set of states the system can be in, and $\pi : Q \to P$ adds propositions to these states. The function $\delta : Q \times \Sigma \to 2^{2^Q}$ assigns to each agent in each state a set of sets of states. Each agent can choose one set of states, and the next state of the system will be from that set.

An example would be a system where $Q = \{0, 1, 2, 3, 4\}$. Suppose that $\delta(0, X) = \{\{1, 2\}, \{3, 4\}\}$ and $\delta(0, Y) = \{\{1, 3\}, \{2, 4\}\}$. Agent X can now choose $\{1, 2\}$ and Y can choose $\{2, 4\}$. They make these choices simultaneously. The next state of the system will be 2, because that is the only common state in their chosen sets. It is necessary to put some constraints on δ so that a next state can always be chosen.

The interpretation of this logic uses the notion of strategy to interpret the coalition operator $\langle\!\langle \Gamma \rangle\!\rangle$. A strategy for Γ is any function that makes a choice $\sigma_\Gamma(X, q) \in \delta(q, X)$ for any agent $X \in \Gamma$ in any state $q \in Q$. Based on a strategy σ_Γ, one can define the set of possible walks $\mathcal{W}(\sigma_\Gamma)$ through Q so that all choices for agents $X \in \Gamma$ are made as recommended by the strategy. This set of walks is used in the following interpretation of ATL.

$$
\begin{array}{ll}
M, q \models \bot & \text{never} \\
M, q \models p \text{ where } p \in P & \text{iff } p \in \pi(v) \\
M, q \models \phi \to \psi & \text{iff } M, q \models \phi \text{ implies } M, q \models \psi \\
M, q \models \langle\!\langle \Gamma \rangle\!\rangle \phi & \text{iff } \exists \sigma_\Gamma : \forall w = v... \in \mathcal{W}(\sigma_\Gamma) : M, w \models \phi \\
\\
M, w \models \Box \phi & \text{iff } \forall n > 0 : M, w(n) \models \phi \\
M, w \models \phi \mathcal{U} \psi & \text{iff } \exists m > 0 : M, w(m) \models \psi \text{ and} \\
& \quad \forall m > k > 0 : M, w(k) \models \phi
\end{array}
$$

The model checker *Mocha* can be used to verify ATL properties of system specifications [5].

3.4.4   Dynamic Epistemic Logic

Strictly speaking, dynamic epistemic logic is not a game logic because its definition does not make any reference to games at all. It is however frequently applied to game-like situations [105], and it seems to be the right tool to model knowledge change in imperfect information games.

Dynamic epistemic logic is an extension of epistemic logic and contains the usual logical connectives, ordinary modal operators $K_X\phi$, and update operators $[\phi]\psi$.

3.4.8. DEFINITION. Assume that finite sets P, Σ are given. A *dynamic epistemic logic* formula ϕ is defined by the following rule. In these rules $p \in P$ and $X \in \Sigma$.

$$\phi ::= p \mid \phi \rightarrow \phi \mid \bot \mid K_X\phi \mid [\phi]\phi$$

Formulas are interpreted over a pointed epistemic models M, w. The ordinary operators are interpreted in the same way as in epistemic logic. The construct $[\phi]\psi$ is interpreted by first computing an updated model M^ϕ and then determining whether $M^\phi \models \psi$.

3.4.9. DEFINITION. Let $M = (\Sigma, W, \sim, P, \pi)$ be an epistemic model and $w \in W$. Dynamic epistemic logic is interpreted in the following way.

$$
\begin{array}{lll}
M, w \models p & \text{iff} & p \in \pi(w) \\
M, w \models \bot & & \text{never} \\
M, w \models \phi \rightarrow \psi & \text{iff} & M, w \models \phi \text{ implies } M, w \models \psi \\
M, w \models K_X\phi & \text{iff} & \forall (w, v) \in \sim_X : M, v \models \phi \\
M, w \models [\phi]\psi & \text{iff} & M, w \models \phi \text{ implies } M^\phi, w \models \phi
\end{array}
$$

Where $M^\phi = (\Sigma, W', \sim', P, \pi')$ is defined such that $W' = \{w \in W | M, w \models \phi\}$, $\pi' : W' \rightarrow 2^P$ is defined by $\pi'(w) = \pi(w)$ and for all agents X we have $\sim'_X = \sim_X \cup (W' \times W')$

In the interpretation of an update formula $\phi = [\psi]\chi$, the model M is changed. We have that $M, w \models [\psi]\chi$ if and only if $M^\psi, w \models \chi$. The model M^ψ is a model for the situation that you get if you update M with the information ψ. In the case of dynamic epistemic logic, this update is done by removing from M the worlds in which ψ does not hold. The updates in dynamic epistemic logic can be compared to announcements, because if you announce a simple formula p, then everybody knows p afterwards: $W, w \models [p]p$. For more complicated formulas this does not hold, consider for instance $\phi = [p \wedge \neg K_B p](p \wedge \neg K_B p)$. After the announcement, B of course knows that p holds, so ϕ is not a tautology.

A complete proof system for dynamic epistemic logic exists, because one can use the following reduction axioms to reduce any DEL formula to a formula of

epistemic logic.

At	$[\phi]p \leftrightarrow (\phi \rightarrow p)$
PF	$[\phi]\neg\psi \leftrightarrow (\phi \rightarrow \neg[\phi]\psi)$
Dist	$[\phi](\psi_1 \wedge \psi_2) \leftrightarrow ([\phi]\psi_1 \wedge [\phi]\psi_2)$
KA	$[\phi]K_X\psi \leftrightarrow (\phi \rightarrow K_X[\phi]\psi)$

This reduction method can also be used to obtain a complete proof system for dynamic epistemic logic with common knowledge [61], and for more complicated actions involving knowledge and beliefs [9].

Chapter 4

Logics for Protocols

4.1 Introduction

In this chapter, the logic EFL is presented, that can be used for reasoning about multi-agent protocols. The acronym EFL stands for *effectivity logic*, because this logic can be used to express whether coalitions have strategies that are *effective* in achieving certain goals. Thus, the logic contains statements such as $[X]\phi$, meaning that X can achieve ϕ. The statement $[X]\phi$ does not mean that X wants ϕ, but rather that X would have a strategy for ϕ at hand if it would ever need one.

As an example of how a logical approach can be useful for people interested in multi-agent protocols, the following informal situation description is used throughout this chapter.

> Three agents Alice, Bob and Caroline (or A, B and C) have to select one of the alternatives x, y and z. They are looking for a suitable voting protocol to select exactly one of these three alternatives as the outcome. The protocol should be democratic, so that any majority can enforce any outcome x, y or z.

The goal of this chapter is to capture these requirements in logic, and then to find protocols that satisfy these requirements.

In this chapter, we give several examples, and we have tried to give the smallest interesting examples of each phenomenon. The following more basic decision problems are used for these examples.

joint decision problem A decision p can be taken if either A or B thinks that p should be the case. If both agents do not want p, it should be rejected.

independent decision problem An agent A can decide whether a should hold or not, and agent B can decide whether b should hold or not.
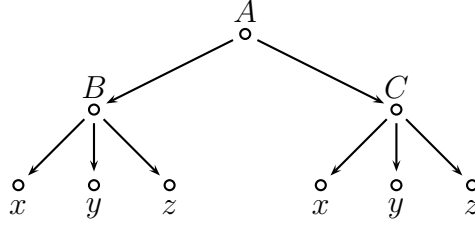
Figure 4.1: A voting protocol F_V

In the next section, section 4.2, the language EFL is defined. Section 4.3 discusses the model checking problem. Bisimulation is discussed in section 4.4, and section 4.5 presents a proof system.

In the second part of this chapter, we look at the different ways in which one can represent protocols. It is shown in section 4.6 that the way in which protocols are represented influences the model checking complexity. This is done by specifying a more efficient way of representing protocols, and proving that the model checking problem becomes harder when this input format is used. Within this chapter we also present many alternative protocols for the example voting problem. The last section, section 4.7, contains conclusions.

4.2   Defining Effectivity Logic

Protocols are modeled in this chapter as (extensive) game forms. In order to use logical formulas for the properties, these game forms are extended with atomic propositions. These atomic propositions are added only to the outcome states of each game form. Such game forms are called interpreted game forms.

4.2.1. DEFINITION. An *interpreted game form* F is defined as a tuple $F = (\Sigma, H, turn, P, \pi)$, so that $(\Sigma, H, turn)$ is a game form, P is a finite set of atomic propositions, and $\pi : Z(H) \to 2^P$ returns the true atomic propositions of any terminal history.

An example game form that represents a protocol for the voting problem is displayed in figure 4.1. The outcomes are marked with propositions x, y, z. In this protocol, A decides whether B or C can decide on the outcome of the protocol.

4.2.2. DEFINITION. Assume that finite sets P, Σ are given. An EFL formula ϕ is defined by the following two rules. In these rules $p \in P$ and $\Gamma \subseteq \Sigma$

$$\phi ::= \Box\psi \mid \phi \to \phi \mid \bot \mid [\Gamma]\psi$$
$$\psi ::= p \mid \psi \to \psi \mid \bot$$

The second line of this definition defines a typical propositional logic formula ψ. These propositional logic formulas are not EFL formulas. They can only appear as $\square\psi$ or $[\Gamma]\psi$. Thus, formulas such as $p \vee \neg p$ and $p \rightarrow q$ are propositional logic formulas, but are not themselves EFL formulas.

Propositional logic is interpreted in the usual way. The logic EFL is interpreted over an interpreted game form $F = (\Sigma, H, turn, P, \pi)$. The definition makes use of pure strategies σ_Γ and updates with these strategies.

4.2.3. DEFINITION. Let $F = (\Sigma, H, turn, P, \pi)$ be an interpreted game form and σ_Γ a pure strategy for coalition Γ. The updated model $F' = Up(F, \sigma_\Gamma)$ is defined as $F' = (\Sigma, H', turn', P, \pi')$ where H' is the unique subset of H such that

- the empty sequence ϵ is a member of H'

- if $h \in H'$ and $turn(h) \in \Gamma$ then $h\sigma_\Gamma(h) \in H'$, but for all other actions b we have $hb \notin H'$

- if $turn(h) \notin \Gamma$ then $ha \in H'$ for any $ha \in H$.

The new elements P', $turn'$ and π' are identical to P and π respectively, except that they are restricted to H'.

The idea behind an update $F' = Up(F, \sigma_\Gamma)$ is that it calculates a reduced game form F', in which no action is taken that is excluded by the strategy σ_Γ. The strategy σ_Γ is only defined for agents $X \in \Gamma$. The other agents are not restricted in any way by the strategy. The notion of an update is used in the following interpretation of EFL.

$$
\begin{aligned}
&F \models \bot && &&\text{never} \\
&F \models \phi \rightarrow \psi && \text{iff} \quad &&\text{not } F \models \phi \text{ or } F \models \psi \\
&F \models \square\phi && \text{iff} \quad &&\forall h \in Z(H) : \pi(h) \models \phi \\
&F \models [\Gamma]\phi && \text{iff} \quad &&\exists\sigma_\Gamma \forall h \in Z(H') : \pi'(h) \models \phi \\
& && &&\text{where } (\Sigma, H', turn', P, \pi') = Up(F, \sigma_\Gamma)
\end{aligned}
$$

Intuitively, the box $\square\phi$ is a universal quantifier. It expresses that ϕ holds in every outcome state. The construction $[\Gamma]\phi$ expresses that Γ has a strategy so that if it uses this strategy, any reachable outcome satisfies ϕ.

The language EFL is expressive enough to express the first two properties that are required for the example protocol. First of all, it can be used to express that a protocol F selects exactly one action.

$$
\begin{aligned}
F &\models \square(x \vee y \vee z) \\
F &\models \square\neg(x \wedge y) \\
F &\models \square\neg(x \wedge z) \\
F &\models \square\neg(y \wedge z)
\end{aligned}
$$

Secondly, one can express that any two agents can enforce any outcome.

$$F \models [AB]x \wedge [AB]y \wedge [AB]z$$
$$F \models [AC]x \wedge [AC]y \wedge [AC]z$$
$$F \models [BC]x \wedge [BC]y \wedge [BC]z$$

It is not hard to verify that the example protocol F_V displayed in figure 4.1 indeed satisfies these formulas. Therefore, there exists a suitable protocol for the example voting problem.

4.3   Model Checking for EFL

The main point of this section is to show that the model checking problem for EFL is tractable. This is not a very deep point. However, it is valuable in practice and it serves as a test case for the notation chosen.

4.3.1. DEFINITION. Let $F = (\Sigma, H, \textit{turn}, \mathfrak{U})$ be an extensive game. For each agent X the value function v^X is defined recursively by $v^X(h) = \mathfrak{U}^X(h)$ if $h \in Z(H)$ and $v^X(h) = \max_{a \in A(H,h)} v^X(ha)$ when $X = \textit{turn}(h)$, and $v^X(h) = \min_{a \in A(H,h)} v^X(ha)$ when $X \neq \textit{turn}(h)$.

4.3.2. LEMMA. *The function v^X can be computed in time $\mathcal{O}(\|F\|)$.*

PROOF. Suppose that we walk through the game tree using a post-order tree walk [27, p.245]. At each node h, we can compute $v^X(h)$ since either h is a leaf, or we have already computed the value of all children ha of h, in which case we take the maximum $\max_{a \in A(H,h)} v^{\textit{turn}(h)}(ha)$ of all children. This walk takes time $\mathcal{O}(\|F\|)$.   □

The value of a game indicates how much payoff agents can expect if they act optimally. If one knows the value of each node, one also knows which moves are good. Suppose that h is a nonterminal history and $\textit{turn}(h) = X$. Intuitively an action a is a 'good' action iff $v^X(ha) = v^X(h)$, and thus knowing the value function helps agents to select the best actions.

4.3.3. THEOREM. *For a given formula $\phi \in$ EFL and interpreted game form F, checking whether $F \models \phi$ takes time $\mathcal{O}(\|F\| \cdot \|\phi\|)$*

PROOF. Assume that a formula ϕ and a model $F = (\Sigma, H, \textit{turn}, P, \pi)$ are given, and furthermore assume that F is represented explicitly by listing all elements of pairs and sets.
   Determining for any propositional logic formula ψ and terminal history s whether $\pi(s) \models \psi$ can be done in time proportional to $\|\psi\|$. For any formula

$\phi = \Box\psi$, determining whether $F \models \phi$ can thus be done in time proportional to $\|H\| \cdot \|\psi\| \leq \|F\| \cdot \|\phi\|$.

Suppose now that $\phi = [\Gamma]\psi$. We can define a game $F' = (\Sigma', H, turn', \mathfrak{U})$ where $\Sigma' = \{\Gamma, \Xi\}$ and $turn'(h) = \Gamma$ iff $turn(h) \in \Gamma$. Otherwise $turn'(h) = \Xi$. The utility function is defined such that $\mathfrak{U}^{\Gamma}(h) = (1, 0)$ iff $\pi(h) \models \psi$. For each terminal history, this takes time at most $\mathcal{O}(\|\phi\|)$, thus computing the whole function takes time $\mathcal{O}(\|F\| \cdot \|\phi\|)$. For this game, one can compute the value function v^{Γ}. If $v^{\Gamma}(\epsilon) = 1$ then $F \models \phi$. Otherwise $F \not\models \phi$. Computing the value function is thus sufficient for determining whether $F \models \phi$ holds. According to lemma 4.3.2 this can be done in time $\mathcal{O}(\|F\|)$.

For other formulas ϕ one can prove the theorem by using induction over the formula structure. The case $\phi = \bot$ is a trivial case. The base cases are formed by $\phi = \Box\psi$ and $\phi = [\Gamma]\psi$, and we have seen that these cases take time at most $\mathcal{O}(\|F\| \cdot \|\phi\|)$, and the induction hypothesis is that this holds for all formulas. In case $\phi = \psi_1 \rightarrow \psi_2$, one can see that determining whether $F \models \phi$ takes time $\mathcal{O}(\|F\| \cdot \|\phi_1\|) + \mathcal{O}(\|F\| \cdot \|\phi_2\|) \leq \mathcal{O}(\|F\| \cdot \|\phi\|)$. Since these are all the cases, we conclude that for any formula ϕ and game form F the theorem holds. $\qquad\square$

Combining logic and game theory can potentially lead to problems with high complexity, but the result given here shows that this is not always the case. This model checking problem is easy due to two factors. First of all the preferences of agents are expressed by means of propositional logic. In this format one cannot express complicated, higher order preferences. Secondly, this logic essentially describes two-player constant-sum games, since one group of agents tries to achieve something under the assumption that the other agents do not cooperate. Two player constant-sum games with perfect information are well understood and computing optimal strategies for such games is not computationally costly.

4.4 Bisimulation

An important question is to decide when two protocols are the same. In general, this is a complicated question, because one can compare protocols with more or less scrutiny. One way out of this dilemma is to use logical equivalence as a deciding factor. Given a suitable logic one can define two protocols to be the same when their corresponding game forms satisfy the same logical formulas. This leaves us with the problem of deciding when two interpreted game forms satisfy the same formulas.

For the logic EFL one cannot apply the notion of bisimulation of standard modal logic directly, because it is not clear what the 'sets of worlds' are that should act as domain of the bisimulation relation. For some logics defined on extensive games one can define a bisimulation between the sets of histories. Each position in the game tree of the first model is matched by the bisimulation to an

equivalent position in the other model. For EFL this idea does not work, because EFL does not use the structure of the game tree directly in its semantics. Two models in which agents move in a completely different order can still satisfy the same EFL formulas. Thus, for EFL one must use a relation similar to the power bisimulation discussed on page 52.

4.4.1. DEFINITION. Suppose that the two models $F = (\Sigma, H, turn, P, \pi)$ and $F' = (\Sigma, H', turn', P, \pi')$ are given. A binary relation $E \subseteq Z(H) \times Z(H')$ is an *outcome bisimulation* if the following conditions hold.

- If $(h, h') \in E$ then $\pi(h) = \pi'(h')$

- For any coalition strategy σ_Γ there exists a strategy σ'_Γ such that the following holds: Let Z be the set of terminal histories of $Up(F, \sigma_\Gamma)$ and let Z' be the set of terminal histories of $Up(F', \sigma'_\Gamma)$. Then $\forall z' \in Z' \; \exists z \in Z : (z, z') \in E$

- Vice versa. For any coalition strategy σ'_Γ there exists a strategy σ_Γ such that the following holds: Let Z be the set of terminal histories of $Up(F, \sigma_\Gamma)$ and let Z' be the set of terminal histories of $Up(F', \sigma'_\Gamma)$. Then $\forall z \in Z \; \exists z' \in Z' : (z, z') \in E$

If two game forms are outcome bisimilar, then they satisfy the same formulas. This is proven in two steps. Below we define which formulas are called *basic* and *simple*. In lemma 4.4.3 we show that two bisimilar models satisfy the same basic and simple formulas. Lemma 4.4.5 can then be used to show that if two models satisfy the same basic and simple formulas, they satisfy the same formulas.

4.4.2. DEFINITION. A formula of the form $\Box\phi$ is *basic*. A formula of the form $[\Gamma]\phi$ is called *simple*.

Basic formulas can be seen as representing global constraints on the possible outcomes, or as powers of the empty coalition. Each simple formula expresses a power of a certain coalition.

4.4.3. LEMMA. *Suppose that the two models $F = (\Sigma, H, turn, P, \pi')$ and $F' = (\Sigma, H', turn', P, \pi')$ are given and that E is a bisimulation between F and F'. Then these models satisfy the same basic and simple formulas.*

PROOF. Suppose that $F \models \Box\phi$. The empty coalition has only one strategy $\sigma_\emptyset$ and this strategy has the property that $Up(F, \sigma_\emptyset) = F$. The same holds for F'. Take Z as the set of terminal histories of F and Z' of F'. Take any state $z' \in z$. The second clause of the bisimulation tells us there a bisimilar state $z \in Z$. Since $F \models \Box\phi$ and $\pi(z) = \pi'(z')$ it follows that $\pi'(z') \models \phi$. Since z' was chosen arbitrarily, it follows that $F' \models \Box\phi$.

Suppose that $F \models [\Gamma]\phi$. This means that there is a strategy σ_Γ such that $Up(F, \sigma_\Gamma) \models \Box\phi$. According to bisimulation one can find a matching strategy σ'_Γ. Take Z, Z' to be the terminal histories of $Up(F, \sigma_\Gamma), Up(F', \sigma'_\Gamma)$ respectively. Take any state $z' \in Z'$. One can find a state $z \in Z$ such that $(z, z') \in E$. Since $\pi(z) \models \phi$ and $\pi'(z') = \pi(z)$ it follows that $\pi'(z') \models \phi$. Since z' was chosen arbitrarily, we conclude that $F' \models [\Gamma]\phi$.

Since the definition of outcome bisimulation is symmetric, one can repeat the argument to show that $F' \models \Box\phi$ implies that $F \models \Box\phi$, and that $F' \models [\Gamma]\phi$ implies $F \models [\Gamma]\phi$. $\qquad\square$

In the next lemma, we use *specific* formulas instead of simple formulas. These specific formulas are simple formulas such that no stronger simple formulas exists.

4.4.4. DEFINITION. Take any set S of formulas and suppose that $\phi_1 = [\Gamma]\psi \in S$ and $\phi_2 = [\Gamma]\chi \in S$. The formula ϕ_2 is *more specific* than ϕ_1 if $\Box(\chi \to \psi) \in S$ and $\Box(\psi \to \chi) \notin S$. The formula ϕ_1 is *specific* if there is no more specific formula in S.

To give an example of a specific formula, take $S = \{[A]a, [A]b, \Box(b \to a)\}$. In this case $[A]b$ is the only specific formula in S, because this formula is more specific than $[A]a$.

The next lemma tells us that it is enough to check only formulas that are simple and specific to ensure that two maximally consistent sets are the same. Since the set of all formulas satisfied by a model is always a maximally consistent set, one can also use this lemma to show that two models satisfy the same formulas.

4.4.5. LEMMA. *Suppose that S and T are maximally consistent sets. Let S' contain all basic and all specific formulas of S and T' all basic and all specific formulas of T. If $S' = T'$ then $S = T$.*

PROOF. Suppose that S and T are maximally consistent sets. Let S' contain all basic and all specific formulas of S and T' all basic and all specific formulas of T. Suppose also that $S' = T'$. Let $\xi = [\Gamma]\psi \in S$. We have to show that $\xi \in T$. If ξ is specific, then $\xi \in S'$, thus $\xi \in T'$ and $\xi \in T$. If not, then there is some 'more specific' formula $[\Gamma]\chi \in S$ so that $\Box(\chi \to \psi) \in S$. This formula itself need not be specific, since there might be an even more specific formula that rules out $[\Gamma]\chi$. However, since P is finite, there is only a finite number of non-equivalent propositional logic formulas. This means there must be a specific formula $[\Gamma]\chi \in S'$ and $\Box(\chi \to \psi) \in S$. Since $S' = T'$ we know that $[\Gamma]\chi \in T'$ and thus $[\Gamma]\chi \in T$. Since $\Box(\chi \to \psi) \in S$ is basic, we can conclude that $\Box(\chi \to \psi) \in S' = T' \subseteq T$. Using the the validity $([\Gamma]\chi \wedge \Box(\chi \to \psi)) \to [\Gamma]\psi$ and the fact that T is maximally consistent, we conclude that $[\Gamma]\psi \in T$.

It is now proven that S and T contain the same simple formulas. Consider now a formula of the form $\neg[\Gamma]\psi$. If $\neg[\Gamma]\psi \in S$, the validity of the axiom DETERMINED,

proven in the next section, can be used to show that $[\Sigma \setminus \Gamma]\neg\psi \notin S$. Since this is a simple formula, we conclude that $[\Sigma \setminus \Gamma]\neg\psi \notin T$. Using DETERMINED again we obtain $\neg[\Gamma]\psi \in T$.

A useful property of maximally consistent sets is that if $\phi \wedge \psi \in S$ then $\phi \in S$ and $\psi \in S$. Moreover if $\phi \vee \psi \in S$ then $\phi \in S$ or $\psi \in S$ (or both). For every formula $\phi \in S$ in conjunctive normal form we can conclude that $\phi \in T$. Since every propositional formula is equivalent to a formula in conjunctive normal form, we may conclude that for any formula ϕ it is the case that $\phi \in S \Leftrightarrow \phi \in T$. Therefore, $S = T$. $\square$

One can also prove the reverse of lemma 4.4.3.

4.4.6. LEMMA. *Suppose that the two models* $F = (\Sigma, H, turn, P, \pi')$ *and* $F' = (\Sigma, H', turn', P, \pi')$ *are given and that these models satisfy the same formulas. Then there is a outcome bisimulation* E *between* F *and* F'.

PROOF. Suppose that $F = (\Sigma, H, turn, P, \pi')$ and $F' = (\Sigma, H', turn', P, \pi')$ are given. Define a relation $E \subseteq Z(H) \times Z(H')$ by stating that zEz' if $\pi(z) = \pi'(z')$. We have to show that this relation is an outcome bisimulation. That the first condition of definition 4.4.1 holds, follows directly from the definition of E. Below we show that the second condition holds. The argument for the third condition is completely parallel to the argument for the second condition.

Take any strategy σ_Γ on F, and compute $Z = Z(Up(F, \sigma_\Gamma))$. Suppose that $P = \{p_0, p_1, \ldots, p_n\}$ and that $Z = \{z_0, \ldots, z_m\}$. Each state $z_j \in Z$ can be completely described by a formula $\psi_j = \bigwedge_{i=0}^{n} \pm_i p_i$ where $\pm_i p_i = p_i$ if $p_i \in \pi(z_j)$, and $\pm_i p_i = \neg p_i$ otherwise. Let $\phi = \bigvee_{j=0}^{m} \psi_j$. It follows that $F \models [\Gamma]\square\phi$. Since F and F' satisfy the same formulas, $F' \models [\Gamma]\square\phi$. Therefore there exists a strategy σ'_Γ such that $Up(F', \sigma'_\Gamma) \models \square\phi$. Now take $Z' = z(Up(F', \sigma'_\Gamma))$ and take any state $z' \in Z'$. Since $\pi'(z') \models \phi$ it must hold that $\pi'(z') \models \psi_j$ for some j. Therefore $\pi'(z') = \pi(z_j)$ and thus $z'Ez_j$ for some $z_j \in Z$, which is what we had to show. $\square$

The notion of outcome bisimulation can thus be used to test whether two protocols have the same properties.

4.5   Completeness

Using the notations from the previous section, one can determine whether two protocols are equivalent. In this section the focus is on the more difficult problem of determining whether there exists a protocol F that satisfies a given property ϕ. In order to do so, a proof system $\mathcal{S}_{EFL}$ is defined, so that one can prove that certain formulas hold for any model. If $\mathcal{S}_{EFL} \vdash \neg\phi$, then there is no model F such that $F \models \phi$. The proof system we present is complete, and thus the opposite also

holds: If $\mathcal{S}_{EFL} \nvdash \neg\phi$, then there exists a model F such that $F \models \phi$. The proof given is constructive, in the sense that it provides a method for constructing such a model.

First the validity of the formulas that are used as axioms is proven. Then the proof system $\mathcal{S}_{EFL}$ is defined, and the completeness proof is given.

The next table lists four axioms that can be written without the coalition operator $[\Gamma]\phi$. These axioms are thus formulas of 'normal' modal logic [12]. For the Greek letter τ one may substitute any instance of any propositional logic tautology that one can obtain using uniform substitution. For instance $[\Sigma]p \vee \neg[\Sigma]p$ is an instance of $p \vee \neg p$. For all other Greek letters one may substitute any propositional logic formula.

$$\text{prop} = \tau \qquad\qquad \text{TAUTOLOGY}$$
$$\text{prop}_2 = \Box\tau \qquad\qquad \text{BOX-TAUTOLOGY}$$
$$S = \Box\phi \rightarrow \Diamond\phi \qquad\qquad \text{SERIALITY}$$
$$K = \Box(\phi \rightarrow \psi) \rightarrow (\Box\phi \rightarrow \Box\psi) \qquad\qquad \text{DISTRIBUTION}$$

All instances of these axioms are valid. For the axiom TAUTOLOGY this follows from the fact that the connectives $\bot, \rightarrow$ are interpreted in the same way as in propositional logic. For the axiom BOX-TAUTOLOGY, one can remark that the definition of $\Box\phi$ uses the semantics of propositional logic. For the axiom SERIALITY, it follows from the fact that each game form must have a non-empty set of outcome states. The DISTRIBUTION axiom is the same as the standard modal logic distribution axiom. Its validity can be shown in the same way. This works because the operator $\Box$ is also defined as a universal operator: $\Box\phi$ holds if ϕ is true in all reachable states. For EFL, the reachable states are all the outcome states.

These axioms are complete for the fragment of EFL in which the construction $[\Gamma]\phi$ is not used. To give a proof sketch: consider the completeness proof of standard modal logic [12]. One can adapt the completeness proof so that no nesting of boxes occurs. In that case, the proof exactly matches the language of EFL without $[\Gamma]\phi$. The remaining axioms for EFL are listed below.

$$C = ([\Gamma]\phi \wedge [\Xi \setminus \Gamma](\phi \rightarrow \psi)) \rightarrow [\Gamma \cup \Xi]\psi \qquad\qquad \text{COMBINATION}$$
$$M = [\Gamma]\phi \leftrightarrow \neg[\Sigma \setminus \Gamma]\neg\phi \qquad\qquad \text{DETERMINED}$$
$$N = [\emptyset]\phi \leftrightarrow \Box\phi \qquad\qquad \text{NOBODY}$$

4.5.1. LEMMA. *All instances of* COMBINATION, *DETERMINED and* NOBODY *are valid*

PROOF. Let F be any interpreted game form.

- Take an instance $([\Gamma]\phi \wedge [\Xi](\phi \rightarrow \psi)) \rightarrow [\Gamma \cup \Xi]\psi$ of COMBINATION so that $\Gamma \cap \Xi = \emptyset$. Take any model F so that $F \models [\Gamma]\phi$ and $F \models [\Xi](\phi \rightarrow \psi)$. It

follows that there are two strategies σ_Γ and σ_Ξ such that $Up(F, \sigma_\Gamma) \models \Box\phi$ and $Up(F, \sigma_\Xi) \models \Box(\phi \to \psi)$. The two strategies σ_Γ and σ_Ξ are two functions with separate domains. One can form a combined strategy $\sigma_{\Gamma \cup \Xi} = \sigma_\Gamma \cup \sigma_\Xi$. If an outcome state is pruned by either σ_Γ or σ_Ξ, then it is also pruned by the combined strategy. Therefore, $Up(F, \sigma_{\Gamma \cup \Xi}) \models \Box\phi \wedge \Box(\phi \to \psi)$ and thus $F \models [\Gamma \cup \Xi]\psi$.

- Take an instance $[\Gamma]\phi \leftrightarrow \neg[\Sigma \setminus \Gamma]\neg\phi$ of DETERMINED. One can define a two player constant-sum extensive game based on the game tree of F between Γ and $\Xi = \Sigma \setminus \Gamma$ so that Γ wins in an outcome s if $s \models \phi$ and Ξ wins if $s \models \neg\phi$. In such an extensive game of perfect information, the two coalitions must have a winning strategy. Therefore, either $F \models [\Gamma]\phi$ or $F \models [\Xi]\neg\phi$.

- Take an instance $[\emptyset]\phi \leftrightarrow \Box\phi$ of NOBODY. First the left to right implication is proven, then we do right to left. Suppose that $F \models [\emptyset]\phi$ This means there is a strategy for the empty coalition $\sigma_\emptyset$ so that $Up(F, \sigma_\emptyset) \models \Box\phi$. The empty coalition has only one strategy (the function with the empty domain), and this strategy $\sigma_\emptyset$ does nothing: $Up(F, \sigma_\emptyset) = F$. Thus, $F \models \Box\phi$. For the right to left implication, assume that $F \models \Box\phi$. It follows from $Up(F, \sigma_\emptyset) = F$ that $F \models [\emptyset]\phi$.

$\square$

One property that one can derive is SPECIFICITY and can be derived using COMBINATION and NOBODY. Another property is MONOTONICITY, which follows from BOX-TAUTOLOGY, COMBINATION and NOBODY.

$$([\Gamma]\phi \wedge \Box(\phi \to \psi)) \to [\Gamma]\psi \quad \text{SPECIFICITY}$$
$$[\Gamma]\phi \to [\Gamma \cup \Gamma_2]\phi \qquad\qquad \text{MONOTONICITY}$$

4.5.2. DEFINITION. The proof system $\mathcal{S}_{EFL}$ consists of the seven axioms TAUTOLOGY, BOX-TAUTOLOGY, SERIALITY, DISTRIBUTION, COMBINATION, DETERMINED, NOBODY given above and the reasoning rule Modus Ponens.

As an example of how this proof system can be used, assume that we have three agents ($\Sigma = \{A, B, C\}$) and three propositions $P = \{a, b, c\}$. We are looking for a protocol that has the following properties.

$$
\begin{array}{ll}
1 & \Box(a \vee b \vee c) \\
2 & \neg[AB]\Box c \\
3 & \neg[AC]\Box b
\end{array}
$$

One can use the proof system $\mathcal{S}_{EFL}$ to show that from these three properties, it follows that $\neg[A]\Box\neg a$. The following derivation proves this property.

4	$[C]\Box\neg c$	2, DETERMINED
5	$[B]\Box\neg b$	3, DETERMINED
6	$[BC]\Box(\neg b \wedge \neg c)$	4, 5, COMBINATION
7	$\Box((\neg b \wedge \neg c) \to a)$	1, BOX-TAUTOLOGY, K
8	$[BC]\Box a$	6, 7, SPECIFICITY
9	$\neg[A]\Box\neg a$	8, DETERMINED

4.5.3. THEOREM. *The proof system $\mathcal{S}_{EFL}$ for* EFL *is sound.*

PROOF. In lemma 4.5.1 it is shown that all axioms are sound. On page 12 it is remarked that the rule modus ponens preserves validity. From these facts it follows that only valid formulas can be derived, which means that the proof system is sound. $\qquad\Box$

The proof system defined here is also complete, and this is proven below in a constructive sense. This proof differs from the standard completeness proof for modal logic, that is sketched on page 17. The proof is a bit more complicated because the semantics of this logic do not refer to single steps in the game trees, but on the possible outcomes that agents can effect.

4.5.4. THEOREM. *The proof system $\mathcal{S}_{EFL}$ is complete for* EFL

PROOF. We have to show that for each consistent formula $\phi \in$ EFL there is a model F such that $F \models \phi$. Let a consistent formula $\phi \in$ EFL be given. Let S be a maximally consistent set so that $\phi \in S$ and let S' contain all basic and specific formulas of S. Below a model F is constructed so that $\forall\psi \in S' : F \models \psi$. Lemma 4.4.5 can then be used to conclude that $F \models \phi$.

The model F we are about to construct is defined recursively using a function $f(C, \mathcal{A}, r)$. The outcome of this function depends on a set of basic and simple formulas C, on a set of active agents $\mathcal{A} \subseteq \Sigma$ and on a representation function $r : \Sigma \to 2^\Sigma$. The set $r(X)$ contains the agents that are represented by agent X. The model F is defined as $F = f(S', \mathcal{A}_0, r_0)$. Initially, all agents are active agents: $\mathcal{A}_0 = \Sigma$, and each agent initially only represents itself: $r_0(X) = \{X\}$. The function r can also be applied to coalitions of agents. This is defined by $r(\Gamma) = \cup_{X \in \Gamma} r(X)$. The pair $\mathcal{A}, r$ can be used to calculate a new set of simple and basic formulas $S(C, \mathcal{A}, r)$ from a given subset C.

$$S(C, \mathcal{A}, r) = \{\Box\psi \mid \Box\psi \in C\} \cup \{[\Gamma]\psi \mid \Gamma \subseteq \mathcal{A}, [r(\Gamma)]\psi \in C\}$$

The game form $f(C, \mathcal{A}, r)$ is defined in the following way. If $\mathcal{A}$ contains exactly one active agent X, then we define a model $f(C, \mathcal{A}, r) = (\Sigma, H, turn, P, \pi)$ where $H = \{\epsilon, \psi \mid [X]\psi \in C, [X]\psi \text{ is specific}\}$. Define $turn(\epsilon) = X$. If $\square\psi_j \in C$, then because of BOX-TAUTOLOGY $\square(\psi \rightarrow (\psi \wedge \psi_j)) \in C$. Using the SPECIFICITY property, we conclude $[X](\psi \wedge \psi_j) \in C$. Repeating this reasoning for any simple formula $\psi_j \in C$, we obtain a formula $[X](\psi \wedge \bigwedge_j \psi_j) \in C$. Let $\pi(\psi)$ be a set of atomic propositions such that $\pi(\psi) \models (\psi \wedge \bigwedge_j \psi_j)$. One can now show that any formula $\chi \in C$ is satisfied by this model.

If $\mathcal{A}$ has two or more members, define $f(C, \mathcal{A}, r) = (\Sigma, H, turn, P, \pi)$ as follows. Take any agent $X \in \mathcal{A}$. Define $turn(\epsilon) = X$, so that this becomes the acting agent of the current situation. The set of options $A(H, \epsilon)$ consists of two parts: $A(H, \epsilon) = E \cup J$. Agent X can thus choose from two different types of actions: formulas from set E or 'joining' an agent from set J.

- The set E consists of all specific choices of agent X: $E = \{\psi_e \mid [X]\psi_e \in C \text{ is specific}\}$. These choices lead to a subgame in which the formula ψ_e holds in all outcomes. This subgame is defined as $f(C', \mathcal{A}, r)$ where

$$C' = \{\square\psi_e, \square\psi, [\Gamma]\chi \mid \square\psi, [\Gamma \cup \{X\}](\chi \wedge \psi_e) \in C\}$$

  This definition ensures that ψ_e holds in the submodel. Axiom COMBINATION ensures that no inconsistent formulas appear in C.

- The set J contains all other active agents: $J = \{Y \in \mathcal{A} \mid Y \neq X\}$. These choices Y lead to the subgames $f(C^Y, \mathcal{A} \setminus \{X\}, r')$ where $C^Y = S(C, \mathcal{A} \setminus \{X\}, r')$, and r' is such that $r'(Y) = \{Y, X\}$ and $r'(Z) = \{Z\}$ for $Z \neq Y$. Intuitively, choosing Y means that agent Y will now make all decisions for agent X.

We must show that $f(C, \mathcal{A}, r)$ satisfies all formulas in C. This is done using induction. The induction hypothesis is that submodels of the current model have this property. The base case is formed by models with one active agent, and this has been done with above.

First, consider basic formulas, of the form $\square\phi \in C$. These formulas are present in each set C' that is used to construct a subgame. Using the induction hypothesis we know that all outcomes of all choices satisfy ϕ, and thus $f(C, \mathcal{A}, r) \models \square\phi$.

Consider $[\Gamma]\psi \in C$ with $X \notin \Gamma$. This formula is also present in any set C' and by induction hypothesis we know that there is thus a strategy in each subgame for Γ to ensure ψ. We can combine these subgame strategies into a strategy σ_Γ for the whole game that guarantees ψ, and thus $f(C, \mathcal{A}, r) \models [\Gamma]\psi$.

Secondly, consider $[\Gamma]\psi \in C$ with $X \in \Gamma$. If $\Gamma = \{X\}$ then there is some specific $\psi_e \in E$ so that $\square(\psi_e \rightarrow \psi) \in C$. This choice leads to a submodel $f(C', \mathcal{A}, r)$. From $\psi_e \in C'$ and the induction hypothesis it follows that there is a strategy σ_X for this submodel that guarantees ψ. Agent X can now use a strategy σ'_X so that
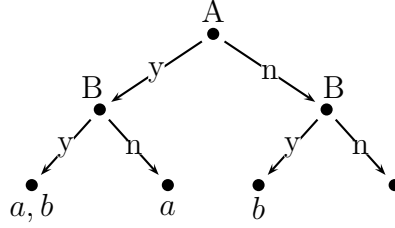
Figure 4.2: A simple game form F_1

$\sigma_X(\epsilon) = \{\psi_e\}$ and within the subgame $f(C', \mathcal{A}, r)$, strategy σ'_X makes the same choices as σ_X. This strategy guarantees ψ and thus $f(C, \mathcal{A}, r) \models [\Gamma]\psi$. If there is more than one agent in Γ, then X can join any of the other agents $Y \in \Gamma$. By induction the coalition $\Gamma \setminus \{X\}$ will have a strategy for guaranteeing ψ in the subgame $f(C', \mathcal{A} \setminus \{X\}, r')$, and thus $f(C, \mathcal{A}, r) \models [\Gamma]\psi$.

The model $F = f(S', \mathcal{A}_0, r_0)$ thus satisfies all formulas $\psi \in S'$. From lemma 4.4.5 it follows that F satisfies all formulas in S and thus $F \models \phi$. $\qquad\square$

Because this proof is constructive, it provides us with a standard method for constructing game trees. These trees have a specific format.
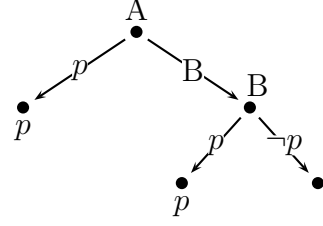
4.5.5. COROLLARY. *For any protocol F there is an equivalent protocol F' in which each agent only moves once, and all agents move in a given order.*

In figure 4.2 an example interpreted game form is shown. In this game form agent A first decides whether a should hold or not. Then agent B can decide whether proposition b should hold or not. A possible story could be that a indicates that A dresses in black, and b indicates that B dresses in black. The next table lists properties that are true for the example F_1.

$$F_1 \models [A]a \wedge [A]\neg a$$
$$F_1 \models [B]b \wedge [B]\neg b$$
$$F_1 \models [B](a \leftrightarrow b) \wedge [B](a \triangledown b)$$

One can conclude that agent B, because it goes second, can control more. This corollary can be illustrated for the example protocol of figure 4.2. According to the proof there should be an equivalent protocol in which agent B moves first. This is indeed the case, and the protocol is illustrated in figure 4.3. One can see that B in this case can choose from four options.

In the construction of the proof, each agent has a choice whether it wants to use one of its abilities (set E) or whether it wants to join a specific agent (set J). In order to illustrate these two possibilities, consider the property $\phi_3 = [A]p \wedge [B]p \wedge [AB]\neg p$. There is only one atomic proposition in this example, so $P = \{p\}$. There are only four distinct formulas that one can express: $p, \neg p, \bot, p \rightarrow p$. Suppose
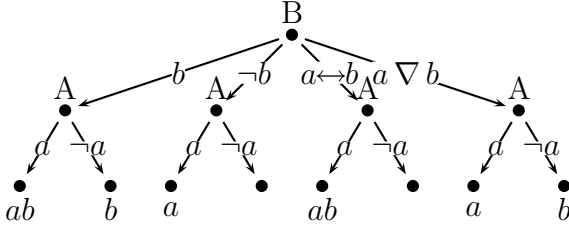
Figure 4.3: Alternative F_2 Figure 4.4: Game form F_3^{AB}

that S is a maximally consistent set containing ϕ_3. The ability $[A]p$ is more specific than $[A](p \to p)$. Thus, agent A has one specific ability p. In the game form F_3^{AB} that is constructed in the proof, agent A has two options. It can use this ability, or it can join agent B. The game form is depicted in figure 4.4. In this protocol agent A and B have exactly the same amount of influence on the outcome, thus one could call this protocol fair.

4.6   Linear Representations

The definition of a game form does not allow one to compactly specify the interpreted game form F_V of figure 4.1 (page 58). A more compact format, similar to the compact formats used in LTL and CTL model checking, is useful. For this purpose we describe here a new way of representing those game forms, called *linear representation*. The idea behind this description method is that a game tree can be summarized by describing a typical path. Consider for instance the solution to the independent decision problem given in figure 4.2. This protocol can be informally described by saying that first agent A chooses whether a should hold, and then agent B decides whether b should hold. Schematically one would like to represent this protocol in the following way

$$R_1 = A \overset{\dddot{}}{\longrightarrow} B \overset{\dddot{}}{\longrightarrow} \dots$$

In this section, such a notation is defined, and used in two ways. First of the notation is used for giving more examples of protocols. Then, it is used to show how the model checking complexity of EFL depends on the way protocols are specified.

In this section we need to make a distinction between a description R of a protocol, and the protocol itself. Suppose that R is a linear representation of a protocol, such as the string R_1 given above. We use $\ulcorner R \urcorner$ to indicate the protocol denoted by R. We hope that R_1 is smaller than the protocol $\ulcorner R_1 \urcorner$ that it represents, and this hope can be expressed as $\|R_1\| < \|\ulcorner R_1 \urcorner\|$.

Examples

In this subsubsection we present a few example linear representations

To start with a simple example, consider the trivial protocol F where agent A can only chose one action, after which an outcome in which p holds is reached. Such a protocol F could be described by the linear representation $R = A \rightarrow \{p\}$. The set $\{p\}$ at the end of the representation is the set of propositions that is true at the end of the protocol. The agent A in front of the arrow indicates that agent A is the agent that can chose. In this protocol agent A can only chose one action. It has no real choice, which makes the protocol trivial.

In order to describe more complicated protocols, two additional constructions can be used. The first one is parallel composition. Suppose that we have two linear representations $R_1 = A \rightarrow \{p\}$ and $R_2 = A \rightarrow \{q\}$. In each of these protocols agent A can only chose one action, but the two protocols have different outcomes. We define R_3 to be the parallel composition of the two linear representations: $R_3 = R_1 || R_2$, or:

$$R_3 = (A \rightarrow \{p\}) || (A \rightarrow \{q\})$$

In the protocol described by R_3, agent A can choose two actions, and end up in either the outcome of $\ulcorner R_1 \urcorner$ or the outcome of $\ulcorner R_2 \urcorner$. Thus parallel composition of protocols gives the starting agents more choice.

The other construction is the use of variables. Take a protocol in which agent A can choose which of the tree propositions $\{p, q, r\}$ is true. The agent is allowed to chose one of these propositions. This protocol can be described by the linear representation $R = A \xrightarrow{v \in \{p,q,r\}} \{v\}$. The symbol v is used here as a variable that can takes the values p, q or r. The protocol $\ulcorner F \urcorner$ thus has three possible outcomes, in which either p, q or r holds.

The follow grammar defines how one can form expressions that denote boolean values, sets, lists and objects. Assume that a set of propositions $P = \{p_0, p_1, \ldots\}$ and a set of agents $\Sigma = \{X_0, X_1, \ldots\}$ are given.

$$
\begin{aligned}
Bool &::= Objct = Objct \mid Bool \vee Bool \\
Set &::= \emptyset \mid \{List\} \mid Set \cup Set \mid Set \cap Set \mid Set \setminus Set \mid \{List \text{ `|' } Bool\} \\
List &::= Objct \mid Objct, List \\
Objct &::= Prop \mid Set \mid Ag \\
Ag &::= X_0 \mid X_1 \mid \ldots \\
Prop &::= p_0 \mid p_1 \mid \ldots
\end{aligned}
$$

All these operators are interpreted in the usual way, except perhaps for $\{x|\phi\}$. The x in this example is a concrete object, not a variable. Thus, if $\ulcorner \phi \urcorner$ holds then $\ulcorner \{x|\phi\} \urcorner = \{\ulcorner x \urcorner\}$, otherwise $\ulcorner \{x|\phi\} \urcorner = \emptyset$.

The denotation $\ulcorner E \urcorner$ of an *Objct* expression E can be computed in polynomial time, since efficient algorithms for all operations exist. In fact the operations used are *polynomial shrinking* (see page 87), which means that the denotation $\ulcorner E \urcorner$ of an expression E is also not bigger than E: $\|\ulcorner E \urcorner\| \leq \|E\|$

The next table gives some examples of *Objct* expressions and their denotations

$$\ulcorner \{p, q, r\} \urcorner \qquad\qquad =\{p, q, r\}$$
$$\ulcorner \{A\} \cup \{B\} \urcorner \qquad\qquad =\{A, B\}$$
$$\ulcorner \{p | A = B\} \urcorner \qquad\qquad =\emptyset$$

Substitution of variables is needed in order to define linear representations. Assume that a set of variables $\mathcal{V}$ is given, and that $v \in \mathcal{V}$. The notation $s[v \setminus x]$ is used to obtained by replacing all occurrences of $v \in \mathcal{V}$ in s by x. Thus, $\{3, v\}[v \setminus 1] = \{1, 3\}$.

A linear representation can now be defined recursively, in three steps. First of all an *Objct* expression that denotes a set of propositions, like $s = \{x, y\}$, is a linear representation of a protocol. Such a set represents a protocol with no choices and only one outcome. It serves as a base case. Since computing the denotation of an expression is a tractable problem, one can always compute $\ulcorner s \urcorner$ in polynomial time $\mathcal{O}(\|s\|^n)$ for some $n \in \mathbb{N}$.

Secondly, one can use an expression of the form $X \xrightarrow{v \in A} R(v)$. Here X is an agent, $v \in \mathcal{V}$ is a variable, $\ulcorner A \urcorner$ is a set of objects. This expression denotes a game form such that agent X can choose any action a from the set $\ulcorner A \urcorner$, after which the protocol proceeds with $R(a)$. An example of this construct is the following voting protocol, in which A decides whether x, y or z holds: $A \xrightarrow{p \in \{x,y,z\}} \{p\}$.

Finally, one can join the options of two different protocol $X \xrightarrow{a \in A} R_1$, $X \xrightarrow{b \in B} R_2$ using the construct $(X \xrightarrow{a \in A} R_1) || (X \xrightarrow{b \in B} R_2)$. In the resulting protocol, agent X can choose either an action from $\ulcorner A \urcorner$ or an action from $\ulcorner B \urcorner$. The following description of the protocol F_3^{AB} displayed in figure 4.4 uses this construct.

$$R_3^{AB} = (A \xrightarrow{p \in \{P\}} \{p\}) || (A \xrightarrow{B} B \xrightarrow{s \in \{\{p\}, \emptyset\}} s)$$

Note that one can represent the same protocol in different ways. At the start of this section as example is given where an agent A can choose whether p, q or r holds. This protocol F was described using variables as $F = \ulcorner R \urcorner$ where $R = A \xrightarrow{v \in \{p,q,r\}} \{v\}$. One can also use parallel composition to describe the same protocol. Thus $F = \ulcorner R' \urcorner$ where

$$R' = (A \to \{p\}) || (A \to \{q\}) || (A \to \{r\})$$

Note that R is a shorter description than R'. The more concise description R is often easier to read, and thus preferred over R'.

In the next two definitions we formally define what we can allow for a representation, and how these representations are translated into game forms. Then we give more example game forms for the voting problem, and give a complexity result.

4.6.1. DEFINITION. Assume a set of variables $\mathcal{V}$ is defined and that the finite sets P, Σ are given. The set of all linear representations is defined recursively as follows.

- If R is a *Objct*-expression, such that R denotes a set of atomic propositions $\ulcorner R \urcorner \subseteq P$, then R is a linear representation

- The construct $X \xrightarrow{v \in S} R$ is a linear representation if the following conditions are met. We demand that $X \in \Sigma$, that $v \in \mathcal{V}$ is a variable, that S is a linear representation such that $\ulcorner S \urcorner$ is a set of objects, and that for all $s_i \in \ulcorner S \urcorner$ we have that $R[v \setminus s_i]$ is a linear representation.

- If $R_0 = X \xrightarrow{v \in S_0} R'_0$ and $R_1 = X \xrightarrow{v \in S_1} R'_1$ are linear representations and $\ulcorner S_0 \urcorner \cap \ulcorner S_1 \urcorner = \emptyset$ then $R_0 || R_1$ is a linear representation.

Using this definition we can now fill in the details in the linear representation of the independent decision problem given above.

$$R_1 = A \xrightarrow{s_1 \in \{\{a\}, \emptyset\}} B \xrightarrow{s_2 \in \{\{b\}, \emptyset\}} s_1 \cup s_2$$

Thus, agent A chooses whether atomic proposition a appears in s_1, agent B chooses whether b appears in s_2, and the final outcome of the protocol is the union of their respective decisions. If one had 100 agents, then this method of specification would be much more efficient than a description in tuples and sets.

The next definition defines a function f^F that translates linear representations into interpreted game forms. This function is defined in the following way.

4.6.2. DEFINITION. Assume the finite sets P, Σ are given, Define $f^F(R) = f_1^F(\epsilon, R)$, where f_1^F is the function defined below. Let h be a sequence of actions.

- If $\ulcorner R \urcorner \subseteq P$ then $f_1^F(h, R) = (\Sigma, \{h\}, \emptyset, P, \pi)$ where $\pi(h) = \ulcorner R \urcorner$.

- Assume $R = X \xrightarrow{s \in S} R'$ is a linear representation. For any $s_i \in \ulcorner S \urcorner$ compute $(\Sigma, H_i, turn_i, P, \pi_i) = f_1^F(hs_i, R'[s \setminus s_i])$. The result $f_1^F(h, R)$ is defined as $f_1^F(h, R) = (\Sigma, \bigcup_i H_i \cup \{h\}, turn, P, \bigcup_i \pi_i)$ where $turn = (\bigcup_i turn_i) \cup \{(h, X)\}$.

- $f_1^F(h, R_0 || R_1) = (\Sigma, H_0 \cup H_1, turn_0 \cup turn_1, P, \pi_0 \cup \pi_1)$ where $(\Sigma, H_i, turn_i, P, \pi_i) = f_1^F(h, R_i)$

The voting protocol F_V has the following linear representation.

$$R_1^A = A \xrightarrow{X \in \{B, C\}} X \xrightarrow{p \in \{x, y, z\}} \{p\}$$

It says exactly what the protocol is : A chooses between B and C, which in turn chooses his favorite alternative.

4.6.3. FACT. For any linear representation R, $f^F(R)$ is an interpreted game form.

PROOF. An induction on R proves this fact. ☐

One of the uses of linear representations is to describe example protocols succinctly. Another use is to show how the complexity of model checking depends on the choice of input format. The linear representation of an interpreted game form can be more compact than a naive representation of a game form. If one specifies the input using linear representation, the EFL model checking problem has a high computational complexity.

4.6.4. THEOREM. *Deciding whether an EFL formula ϕ holds on a linearly represented game form F is PSPACE-complete.*

PROOF. In the proof of theorem 4.3.3 and lemma 4.3.2 it is explained how model checking EFL depends on the ability to do a post-order tree walk. If we can do such a post-order tree walk in polynomial space, then we can model check EFL formulas in polynomial space. An algorithm for such a walk typically uses a stack. On this stack a description of the current node is stored, after which the description of a successor is computed, which is also stored on the stack, etcetera, until a final node is reached. We therefore show the following facts.

- For a linear description R of a terminal node and a propositional logic formula ϕ, one can determine whether $\ulcorner R \urcorner \models \phi$ in polynomial time. This follows immediately from the assumption that one can compute the set of atomic propositions $\ulcorner R \urcorner$ in polynomial time. A naive polynomial time algorithm is to compute $\ulcorner R \urcorner$ and then determine whether $\ulcorner R \urcorner \models \phi$. Hence this takes at most $b(\|R\| + \|\phi\|)$ for some polynomial bound R

- Each linear description of a successor of R is smaller that R itself. This is easy to see. In the case of $R = R_1 \| R_2$, both R_1 and R_2 are smaller than R. In the case of $R = X \xrightarrow{v \in \ulcorner S \urcorner} R'(v)$, it is also clear that $\|R'(v)\| \leq \|R\|$. Thus, each element on the stack needs as most memory $\|R\|$

- The maximal number of descriptions on the stack is also bounded by $\|R\|$, because the maximal depth of the model denoted by R is at most the number of arrows that occur in R.

From the last two facts one can compute that one needs at most $\|R\| \cdot \|R\|$ memory for the stack. Therefore, the total amount of memory that one needs to determine whether ϕ holds on $\ulcorner R \urcorner$ for a linear representation R is less than $\|R\|^2 + b(\|R\| + \|\phi\|)$, and thus is polynomial.

It remains to be proven that the problem is PSPACE-hard. This can be done by reducing the QBF problem described on page 31 to the EFL decision problem.

Assume that a QBF formula $\forall x_1 \exists x_2 \forall x_3 \ldots \forall x_n \phi$ is given. We have to construct an equivalent EFL decision problem, consisting of a representation R and a formula ϕ'. Let $\Sigma = \{X, Y\}$ and $P = \{x_i | 0 \leq i \leq n\}$. The atomic proposition x_0 is a dummy atomic proposition, since it does not appear in ϕ. The representation of the interpreted game form is the following:

$$R = X \xrightarrow{v_1 \in \{x_0, x_1\}} Y \xrightarrow{v_2 \in \{x_0, x_2\}} \ldots Y \xrightarrow{v_{n-1} \in \{x_0, x_{n-1}\}} X \xrightarrow{v_n \in \{x_0, x_n\}} \{v_1, v_2, \ldots, v_n\}$$

Take $\phi' = [Y]\phi$. The agent Y thus makes all existential choices (it tries to pick values for the x_i that make ϕ true), and agent X is used for the universal choices. If $\forall x_1 \exists x_2 \forall x_3 \ldots \exists x_{n-1} \forall x_n \phi$, then $f^F(R) \models \phi'$ and vice versa. $\qquad \square$

It has to be remembered that for some protocols without a lot of structure, the linear representation format is not more efficient. For very irregular protocols all linear representations can be larger that the game form itself. Nevertheless, one can give a linear representation of any protocol.

4.6.5. THEOREM. *For any interpreted game form $F = (\Sigma, H, turn, P, \pi)$ where H consists of sequences of propositions, there is a linear representation R such that $f^F(R) = F$*

PROOF. The set H of any game form F consists of sequences of actions. It does not matter what kind of objects these actions are, as long as they can be distinguished. In the examples throughout this dissertation, we have used natural numbers, propositions and agents as actions. In this proof we restrict the action to be propositions because propositions are part of the *Objct* notation.

For any subset $S \subseteq P$ of propositions, one can find an expression R such that $\ulcorner R \urcorner = S$, by listing all elements of S. For instance if $S = \{p, q\}$ then one can simply take $R = \{p, q\}$.

For any interpreted game form $F = (\Sigma, \{\epsilon\}, turn, P, \pi)$ that has only one outcome, one can take an expression R so that $\ulcorner R \urcorner = \pi(\epsilon)$. This expression R consists of a list of atomic propositions that hold in the single end state ϵ of the interpreted game form F. This simple case can be used as a base case for an inductive proof.

Consider now an interpreted game form $F = (\Sigma, H, turn, P, \pi)$ that has more than one outcome, and assume that for all smaller game forms F' one can construct representation R' such that $f^F(R') = F'$. Let $A(H, \epsilon) = \{a_1, \ldots, a_n\}$ be the set of possible first actions, and let $X = turn(\epsilon)$ be the agent to move first. Define, for each action a_i, the interpreted subgame form $F_i = \mathrm{subg}(F, a_i)$. By induction hypothesis, there is a representation R_i such that $f^F(R_i) = F_i$. Let $v \in \mathcal{V}$ be some variable. One can now create a game form R by using the $||$ construction.

$$R = (X \xrightarrow{v \in \{a_1\}} R_1) || \ldots || (X \xrightarrow{v \in \{a_n\}} R_n)$$
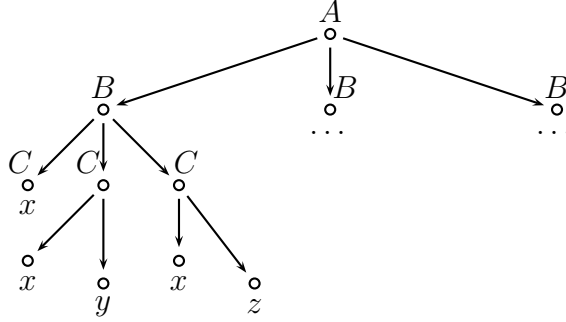
Figure 4.5: A second voting protocol F_{V2}

For this linear representation R, it holds that $f^F(R) = F$. Therefore, by induction, one can find a linear representation R for any interpreted game form F that uses propositions for actions. $\qquad\square$

The new representation format can be used to define more candidate protocols for the example problem defined in the beginning of this chapter. These new protocols all satisfy the requirements of the example, while being very different from protocol F_V. Below we define two more protocols, called $F_{V2} = f^F(R_2^{ABC})$ and $F_{V3} = f^F(R_3^x)$. Part of the game tree of protocol F_{V2} is depicted in figure 4.5.

$$R_2^{ABC} = A \xrightarrow{a \in \{x,y,z\}} (B \xrightarrow{b \in P \setminus \{a\}} C \xrightarrow{c \in \{a,b\}} \{c\}) || (B \xrightarrow{a} \{a\})$$

In F_{V2}, A and B choose from the three possible outcomes. If they choose the same outcome, then that is the final outcome. Otherwise C can choose from the two outcomes they selected.

$$R_3^x = A \xrightarrow{a \in \{x,y,z\}} B \xrightarrow{b \in \{x,y,z\}} C \xrightarrow{c \in \{x,y,z\}}$$
$$\{x | \{a,b,c\} = \{x,y,z\}\} \cup \{a | b = a\} \cup \{c | a = c\} \cup \{c | b = c\}$$

In our third example F_{V3}, the agents A, B, and C vote sequentially for one of the three outcomes; the outcome that gets the most votes is elected. If A, B and C disagree then a pre-determined outcome x is elected.

It is not hard to verify that these three protocols satisfy exactly the same EFL formulas. One can thus conclude that these three protocols are equally fair, and that there is no reason to prefer one of those protocols above the others. This conclusion seems counter-intuitive, because the protocol R_3^x seems biased towards outcome x. If the agents cannot come to an agreement, then outcome x results.

Perhaps EFL is the right tool and the bias mentioned for R_3^x is an irrelevant detail. It is also possible that the logic EFL is not sensitive enough. Both of these viewpoints are valid, depending on the application one has in mind.

Making more use of the new notation, one can make even more complicated protocols. Similar to R_3^x, one can define protocols R_3^y and R_3^z, in which y and respectively z are the default outcomes. One can offer one of the agents the choice of selecting the default outcome.

$$R_4^A = A \xrightarrow{d \in \{x,y,z\}} R_3^d$$

The new protocol $f^F(R_4^A)$ again satisfies the same EFL formulas. If one however thinks that R_3^x is biased towards x, then one must conclude that R_4^A is skewed towards A. The word *skew* is used here to express that a certain agent is treated differently in a significant way from other agents, whereas *bias* means that an outcome is treated in a different way than the other outcomes. To give another example of how one can make more subtle protocols, consider the case where agent A chooses whether B or C selects the default outcome.

$$R_5^A = A \xrightarrow{S \in \{B,C\}} R_4^S$$

The result is that there are indeed many different protocols for the example problem. Many of these can be elegantly described using a linear representation, even when it would have been very hard to draw a picture of the game tree. For EFL however all these protocols are equivalent.

4.7 Conclusion

The logic EFL is a high level logic for reasoning about multi-agent protocols. In this chapter, the problem of finding a good voting protocol has been used as a motivating example for the construction of such logic. Different protocols have been presented and modeled as game forms. Using EFL we have shown that the candidate protocols indeed satisfy the requirements of the problem. Using an effectivity logic such as EFL one can thus reason about the powers of agents and coalitions in protocols.

In order to efficiently discuss multiple protocols, one needs a good way of representing different protocols. The naive way, as a tuple of sets and functions, is rather cumbersome. Another option is to specify protocols by means of a picture of a game tree. However this is also only practical for small protocols. In this chapter, a new input format is defined, called the linear representation. The idea behind this format is that one can specify a protocol by describing a typical execution of the protocol. For the example voting problem, one can elegantly describe many protocol variants using the linear representation.

One big question is whether verification of multi-agent protocols in EFL is tractable. Thus, one would like to know the model checking complexity of EFL. The answer to this question depends on how one wants to specify the input. If one allows protocols to be specified in a linear representation, then this problem is very intractable: It is PSPACE complete. This result does not only say something about EFL, but applies to solving games in general. With a sufficiently advanced notation, determining the winner of a game is an intractable problem. For instance winner determination in the games of Go and Geography is also PSPACE-complete [81, p. 463].

On the other hand, the verification problem becomes tractable if one uses the naive representation for game forms. The interpretation of the logic EFL is defined in terms of winning a perfect information game, and this problem is not too hard. Verification can be done in polynomial time. This supports the conclusion that EFL is indeed simple. One can conclude that verification of properties in EFL is a *feasible* computational problem. Interesting future work is to find out whether techniques that have been used to speed up model checkers for ATEL and CTL can be adapted for EFL.

In many applications one does not already have a protocol. One only has a specification and one would like to find a protocol that meets this specification. This corresponds to the satisfiability problem for EFL: one has an EFL formula ϕ, and would like to know whether there exists an interpreted game form F that satisfies this formula. This problem is also solvable for EFL: a proof system $\mathcal{S}_{EFL}$ has been presented, so that one prove formally which formulas ϕ cannot be satisfied. This proof system is thus complete. If no such proof exists, a method has been sketched that allows one to construct a model. Automated protocol design on the basis of EFL specifications thus seems possible. It is interesting to note that one has a lot of freedom in constructing these models: one can order the agents in any way, and construct a model in which the agents make decisions in this order. This property distinguishes this logic from modal logics that work on the level of single actions.

One open question, concerning the example protocol, is how one can distinguish between all candidate protocols that have been presented in this chapter. All protocols presented are equivalent under EFL. However, it seems that these protocols should behave differently of one considers more complicated properties. In the next chapter, extended logics that work under different assumptions are employed to solve this problem.

Chapter 5

Politeness and Side Effects

5.1 Introduction

Like the previous chapter, this chapter is concerned with reasoning about game forms, which are seen as models for multi-agent protocols. In this chapter we extend the logic of the previous chapter. It is assumed that each agent that participates in a protocol has some private preferences about the outcome of the protocol. The word 'preferences' is used here in a very loose sense, as a synonym for 'goal' or 'desire'. It is also assumed that these preferences are not determined by the protocol. Agents can want whatever they want to want. In voting protocols it is clear that the agent can have its own, private, preferences over outcomes. In an auction this is less clear: auctions are often analysed under the assumption that each agent wants to win at the lowest cost. We assume a more general setting, where agents can also play to lose, or to maximize the amount they pay.

One of the goals of this chapter is to investigate verification of more complicated properties than only the ability to enforce a certain outcome. Three complications that are being discussed are the following.

- Groups of agents can have *coalition preferences*. One can express in the logical languages that A and B together want ϕ. This means that they try to reach a certain goal together and are able to cooperate.

- Agents may be interested in *nested abilities*: an agent can have the ability to enable another agent to achieve something, or to make sure another agents is not able to do something. The wish to give other people the chance to make a decision, is often associated with *politeness*, we use the phrase 'reasoning about politeness' as an informal name for these nested ability goals.

- We are not only interested in knowing what agents can achieve, but also in what way they achieve it. Thus, we would like to know whether an agent

has to spend all his money to win an auction, or whether an agent should vote for the candidate it like best. Thus, the *side effects* of acting in a certain way are also important.

In this chapter, different logical languages are defined, so that we can determine how considering nesting and side effects affects the analysis of protocols. In total four languages are defined. The next table lists the languages and their features.

logic	nesting	side effects
EFL	×	×
EFLS	×	✓
EFLN	✓	×
EFLNS	✓	✓

Chapter Structure

The structure of this chapter is the following. First the logic EFLS is defined in section 5.2, and examples for this logic are given in section 5.3. The next section, section 5.4, contains a theorem stating that for logics of a certain form, the model checking problem is tractable. It is shown that this theorem can be applied to EFLS. Then the question is posed whether there are more expressive or detailed logics based on EFL and EFLS. In section 5.5, first the language EFLN for reasoning about nested abilities is introduced, and we determine the model checking complexity of this logic. Then a more expressive language EFLNS is given for reasoning about both politeness and side effects. The last section, section 5.6 is the conclusion.

5.2   Defining EFLS

5.2.1. DEFINITION. Suppose that Σ and P are finite sets (of agents and atomic propositions respectively). The language EFLS consists of formulas ψ generated by the following rules. In these rules $p \in P$ and $\Gamma \subseteq \Sigma$.

$$\phi ::= p \mid \phi \to \phi \mid \bot$$
$$\psi ::= [\Gamma : \phi]\psi \mid \Box\phi \mid \psi \to \psi \mid \bot$$

This language can be seen as an extension of EFL, in the following way. An EFL formula $[\Gamma]\phi$ is equivalent to the EFLS formula $[\Gamma : \phi]\Box\phi$.

A formula $[\Gamma : \phi]\psi$ should be read as saying 'Assume that Γ uses a strategy that is supporting ϕ. Then ψ follows'. It is assumed that all agents are aware of strategies that are used. The strategy that an agent uses can be said to be 'visible' to other agents. If one wants to express this idea in the most extreme form, one could say that we assume that strategies are visible in the same way as people

can see what clothes other people are wearing on a certain day. In many real life settings, finding out what strategies are used in a strategic setting will probably take a bit more effort, but is not impossible. This visibility assumption is inspired by the assumption of *complete information* in game theory, and is compatible with the idea of a Nash equilibrium. Indeed if one, as a game theorist or as a strategic consultant, intends to publish books and papers about good strategies (whether these are chess strategies, marketing strategies, strategies for penalty taking, or strategies for generating secure random numbers) then such strategies must work even when public. Even if one does not wish to publish strategies, people can often observe what action you take and deduce your strategy from this. Thus it is known what playing styles professional chess players prefer and how professional football players take penalties. Many professional keepers, including Hans van Breukelen, for instance relied on Jan Reker's booklet for this information [117].

The visibility assumption is also inspired by insights from security and cryptography. One can see the definition of specification of a cryptographic algorithm as a protocol, and the implementation details as a strategy within such protocol. For instance the protocol for RSA key generation requires one to choose two prime numbers p and q. An agent has many ways to do this, and common strategies include using the current time and some keyboard input for generating these prime numbers (See Schneier [89] for a dicussion of RSA and implementation details). The implementation details are often public information, since for many security programs one can obtain the source code.

The effectiveness of an implementation should not lie in the fact that its inner details are secret (Thus, one should avoid trying to obtaining security through obscurity [89]). In order to prove that a strategy or algorithm is 'good' or 'safe', one should assume that it is known to all opponents that the strategy is used, and then consider how effective the strategy is. For instance if Microsoft decides to use a certain encryption mechanism in its web server software, then anybody with harmful intentions can buy and study the software, and find out what measures have been taken against attacks. Typically in security one wants to prove that opponents remain ignorant of private data. If they are ignorant even when they know the strategy used, they are certainly ignorant when they do not know the strategy used. This assumption can also be made in the case of imperfect information games, and indeed a similar argument is given on page 132.

The idea that strategies are 'visible' makes the act of deciding to use a strategy similar to publicly announcing that you use the strategy. In complex statements, this idea of an announcement can be used informally when reading formulas. The following examples illustrate how formulas of this logic can be read.

$$[A:q]\Box p$$

This example formula expresses that if A is trying to achieve q, then as a side effect p will hold for every possible outcome.

$$[A:q][B:r]\Box r$$

This example formula can be read as expressing that, assuming after A has decided that it wants q, then B can select a strategy such that r becomes true. The order of operators in the formula indicates that B knows the strategy of A, and can use this in its selection of a strategy for r.

The next formula seems to contain contradictory assumptions.

$$[A:q][A:\neg q]\square(\neg q)$$

This formulas expresses that if A wants q, and then it wants $\neg q$, then $\neg q$ is guaranteed. In order for this formula to hold on a model F, it must be the case that A cannot make q true, otherwise it would choose to do so in the first assumption.

For the interpretation of this logic the following definitions are used.

5.2.2. DEFINITION. Let $F = (\Sigma, H, turn, P, \pi)$ be an interpreted game form and $h \in H$. The reduced model $r(H, h)$ is defined as $r(H, h) = (\Sigma, H', turn', P, \pi')$ where $H' = \{h'|h \cdot h' \in H\}$ and $turn', \pi'$ are restrictions of the corresponding elements of F to H'.

The next definition redefines the update function Up so that it works on nondeterministic strategies. The intuition is that in the updated model $Up(F, \sigma_\Gamma)$, the agents in Γ only take actions that are recommended by σ_Γ.

5.2.3. DEFINITION. Let $F = (\Sigma, H, turn, P, \pi)$ be an interpreted game form and σ_Γ a strategy for Γ. Define $Up(F, \sigma_\Gamma) = (\Sigma, H', turn', P, \pi')$ where H' is the greatest subset of H such that $ha \in H'$ implies $h \in H'$ and $turn(h) \in \Gamma \wedge ha \in H'$ implies $a \in \sigma_\Gamma(h)$. The functions $turn', \pi'$ are identical to $turn, \pi$ but restricted to H'.

The next definition defines a strategy $\sigma_\Gamma^e(\phi)$ that is intended to be the least restrictive, or most general, strategy that Γ can use to achieve ϕ.

5.2.4. DEFINITION. Let $F = (\Sigma, H, turn, P, \pi)$ be an interpreted game form, $\Gamma \subseteq \Sigma$ and $\phi \in \mathcal{L}_p$. A history j is a ϕ-effective position (for Γ) iff there is a strategy σ_Γ such that for each terminal history h in $Up(r(H, j), \sigma_\Gamma)$ it is the case that $\pi(h) \models \phi$. The most general ϕ-effective strategy $\sigma_\Gamma^e(\phi)$ is now defined by

$$\sigma_\Gamma^e(\phi)(h) = \begin{cases} \{a|ha \text{ is a } \phi\text{-effective position for } \Gamma\} & \text{if this set is non-empty} \\ A(H, h) & \text{otherwise} \end{cases}$$

The definition above spells out what we consider a rational strategy $\sigma_\Gamma^e(\phi)$ for a coalition Γ that wants to achieve ϕ. The strategy is defined such that it selects actions a that lead to winning positions. If that is not possible, it selects all actions. The idea is that coalition Γ tries to guarantee ϕ in all positions where it can guarantee ϕ. This is similar to the notion of a subgame-perfect strategy. In the definition below we use this strategy for interpreting the logic. Let $F = (\Sigma, H, turn, P, \pi)$ be an interpreted game form. For any formula $\phi \in$ EFLS the relation $F \models \phi$ is defined as follows.

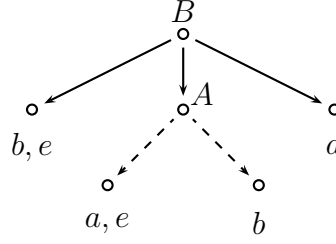Figure 5.1: Alice and Bob eat cake

$$
\begin{array}{lll}
F \models \bot & & \text{never} \\
F \models \phi \to \psi & \text{iff} & \text{not } F \models \phi \text{ or } F \models \psi \\
F \models \Box\phi & \text{iff} & \forall h \in Z(H) : \pi(h) \models \phi \quad \text{where } (\Sigma, H, \mathit{turn}, P, \pi) = F \\
F \models [\Gamma : \phi]\psi & \text{iff} & Up(F, \sigma_\Gamma^e(\phi)) \models \psi
\end{array}
$$

The interpretation of these formulas is similar to that of previous update logics, such as dynamic epistemic logic, discussed in section 3.4.4 on page 54.

In EFLS, formulas of the form $[\Gamma : \phi]\psi$ can be seen as updates. In order to determine whether $F \models [\Gamma : \phi]\psi$, a new model $F' = Up(F, \sigma_\Gamma^e(\phi))$ is computed. This new model represents the situation after Γ has decided to try to achieve ϕ. It holds that $F \models [\Gamma : \phi]\psi$ if and only if $F' \models \psi$. In an update logic, the model can thus be changed by adding new information to it. How the model changes, depends on the update function that is used.

5.3 Examples

5.3.1 Alice and Bob eat Cake

Alice and Bob have a cake, and they have agreed to divide it by means of a "cut-and-choose" protocol [17]. Alice has cut the cake and unfortunately one of the pieces is bigger than the other. Bob can now choose from three options: he can select the big piece, select the small piece, or he can say to Alice 'No, you choose'. If he lets Alice choose, she can either choose the big piece or the small piece. Both agents have common knowledge of this protocol. The interpreted game form protocol corresponding to this situation is displayed in figure 5.1. Proposition a means that Alice gets the biggest piece, b that Bob gets the biggest piece, and e means that something has happened that is embarrassing to Alice and Bob, namely that either Alice or Bob has chosen the biggest piece. In many cultures this is considered impolite. Using EFLS one can express relevant properties of this protocol. First we will provide several EFLS formulas (A stands for Alice, B stands for Bob).
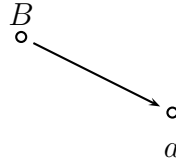
Figure 5.2: Model $Up(M, \sigma_B^e(\neg e))$

- $[B : \neg e]\Box a$ If B does not want either of them being embarrassed, he must take the smallest piece. Our semantics take a pessimistic view, so Bob cannot take the risk of letting A choose. Figure 5.2 shows the updated model $Up(M, \sigma_B^e(\neg e))$.

- $[B : \neg e][A : \neg e]\Box a$ This formula is a consequence of the previous example. It expresses that if B does not want embarrassment and that A does not want embarrassment, then A gets the biggest piece. This may seem strange, since there is an outcome in which $\neg e$ and b are true. However, the order of assumptions is important. The formula expresses that B wishes to guarantee the absence of embarrassment, independently of what A does. Two possible readings of the formula are that he commits himself to his strategy before he learns that A has the same preference, or that he thinks that this goal is so important that he does not wish to rely on A for this property.

- $[AB : \neg e][B : b]\Box b$ In this example, A and B commonly want to avoid embarrassment, and B also prefers b. If this is the case, B can let A choose and then A will take the smallest piece. Figure 5.3 shows the updated model $Up(M, \{A, B\}, \neg e)$.

- $[A : \neg e][B : \neg e][B : b]\Box b$ This formula expresses that if A does not want embarrassment, B does not want embarrassment, and B prefers the biggest piece then B gets the biggest piece. The behaviour of B is influenced by the fact that he knows that A prefers to avoid embarrassment. In this scenario A should try to hide the fact that she has good manners, because it is not in her advantage if B knows this.

This example illustrates that, by using EFLS, one can express consequences of ordering goals in a certain way. There are several interesting side effects mentioned in the above formulas.
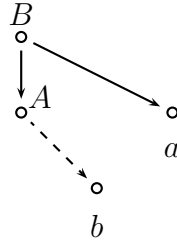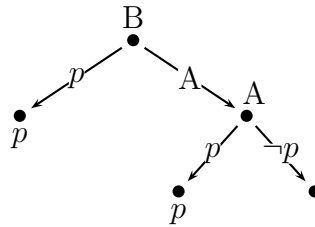
Figure 5.3: Model $Up(M, \{A, B\}, \neg e)$

Figure 5.4: Game form F_3^{BA}

5.3.2   Joint Decision Problem

In figure 4.4, on page 70, an interpreted game form F_3^{AB} is given in which two agents jointly decide whether p should hold or not. If either agent wants to have p it should hold, otherwise p is rejected. In figure 5.4 another protocol is given that satisfies the same EFL formulas. In this protocol, the roles of B and A are reversed. It seems reasonable to assume that agents care who has to give its opinion first, and therefore one would like to have a logic that can distinguish these protocols.

The following statements show that EFLS is such a logic.

$$F_3^{AB} \models [B : \neg p][A : \neg p]\Box\neg p$$
$$F_3^{BA} \not\models [B : \neg p][A : \neg p]\Box\neg p$$

The reason the formula does not hold in the second model is that B, because it moves first in the protocol F_3^{BA}, has an informational disadvantage. When it has to decide it does not know what A will do, and therefore it is not clear that letting A choose helps towards achieving its goal. The logic EFLS is thus more expressive than EFL.

5.4   Model Checking EFLS

The semantics of EFLS is based upon the idea that one can interpret the construct $[\phi]\psi$ by updating a model M with ϕ, and then checking whether ψ holds in the updated model. Such a semantics, familiar from dynamic epistemic logic described on page 54, can be called an update semantics. The goal of this section is to determine the model checking complexity of the logic EFLS. Instead of doing it directly, we prove a more general theorem concerning update semantics of a certain form, and then show that the theorem applies to EFLS.

Below we give a general definition of an update language. This definition is suitable for EFLS, but not general enough for all other update logics. The term 'update language' in this section thus does not refer to all logical languages that use the idea of updates. It refers only to languages to which the given definitions can be applied. In this section we have given this term a specific interpretation, using the following definition. Let $\mathcal{M}$ be a set of models for a logic, $\mathcal{N}_1$ any set of additional information objects, $\mathcal{N}_2$ a set of formulas in another (simpler) language. We assume that two functions f and g are given, such that $f : \mathcal{M} \times \mathcal{N}_1 \to \mathcal{M}$ and $g : \mathcal{M} \times \mathcal{N}_2 \to \{\text{true, false}\}$. Suppose also that for any $n_2 \in \mathcal{N}_2$ and $M \in \mathcal{M}$, one can check in polynomial time whether $g(M, n_2)$ holds. One can, based on these functions f and g, define an update logic $\mathcal{L}_{fg}$ with the following semantics.

5.4.1. DEFINITION. Suppose that $\mathcal{M}$ and $\mathcal{N}$ are given, and assume that $n_1 \in \mathcal{N}_1$ and $n_2 \in \mathcal{N}_2$. The update language $\mathcal{L}_{fg}$ consists of formulas ψ generated by the following rules.

$$\psi ::= [n_1]\psi \mid n_2 \mid \psi \to \psi \mid \bot$$

This language is called an update language, because one can interpret this logic using updates. The function f is used to compute a new model from the current model. The next definitions captures the idea of an update semantics. In the next definitions, $M \in \mathcal{M}$ is a model, $n_1 \in \mathcal{N}_1$, $n_2 \in \mathcal{N}_2$ and $\psi, \xi \in \mathcal{L}_{fg}$.

$$
\begin{array}{lll}
M \models \bot & & \text{never} \\
M \models \psi \to \xi & \text{iff} & \text{not } M \models \psi \text{ or } M \models \xi \\
M \models n_2 & \text{iff} & g(M, n_2) \\
M \models [n_1]\psi & \text{iff} & f(M, n_1) \models \psi
\end{array}
$$

The following formulas are valid under this semantics.

$$\models [n](\phi \to \psi) \to [n]\phi \to [n]\psi$$
$$\models \neg[n]\psi \leftrightarrow [n]\neg\psi$$

These axioms can be compared to the reduction axioms stated for dynamic epistemic logic stated on page 55. The axioms are not identical, and these two axioms

are not sufficient to eliminate all update operators, but they do help to simplify formulas.

This semantics is a generalisation of the semantics of EFLS. For EFLS, the set $\mathcal{N}_1$ consists of pairs (Γ, ϕ), but in this general semantics one can update with anything. The set $\mathcal{N}_2$ consists, in the case of EFLS, of the formulas $\mathcal{N}_2 = \{\Box\phi | \phi \in \mathcal{L}_p\}$. The question is whether such a semantics can be evaluated in polynomial time. If so, then the model checking problem is tractable, and thus this logic can be used in practice for protocol verification.

Whether model checking is tractable, depends on the function f. This function should be easily computable, but it should also not create bigger and bigger models. If a function f has these two properties, it is called polynomial shrinking.

5.4.2. DEFINITION. A function f is *polynomial shrinking* iff $\|f(a,b)\| < \|a\| + \|b\|$ and f can be computed in polynomial time.

If the function f is polynomial shrinking, then the model checking problem is indeed tractable.

5.4.3. THEOREM. *Suppose that f is a polynomial shrinking function, and that $\mathcal{L}_{fg}$ is the corresponding update logic. One can check for given $M \in \mathcal{M}$ and $\psi \in \mathcal{L}_{fg}$ whether $M \models \psi$ within polynomial time.*

PROOF. Suppose that f is a polynomial shrinking function, and that $\mathcal{L}_{fg}$ is the corresponding update logic. In order to prove the theorem, an algorithm and constants a and b must be given, such that the algorithm needs at most time $(\|M\| + \|\psi\|)^a + b$ to determine whether $M \models \psi$, for any inputs M and ψ. The algorithm works recursively on the structure of ψ, so four cases have to be examined. The first two cases are basic cases, in the two other cases we use an induction assumption.

- If $\psi = \bot$, then $M \not\models \psi$. Returning this answer takes constant time, and one can take any b larger than this constant time.

- Suppose that $\psi = n_2 \in \mathcal{N}_2$. It has been assumed that it can be determined in polynomial time whether $g(M, n_2)$. Thus, there are constants c and d such that this takes less time than $(\|M\| + \|\psi\|)^c + d$. Taking $a \geq c$ and $b \geq d$, it follows that this takes less time than $(\|M\| + \|\psi\|)^a + b$.

- Suppose that $\psi = \psi_1 \rightarrow \psi_2$. This means that $\|\psi\| = 1 + \|\psi_1\| + \|\psi_2\|$. In order to determine whether ψ holds, one has to compute whether $M \models \psi_1$ and whether $M \models \psi_2$. Using the induction hypothesis, and supposing that $a \geq 2$, one can show that this takes less than $(\|M\| + \|\psi\|)^a + b$ time.

- Suppose finally that $\psi = [n]\psi_1$. In order to determine whether $M \models \psi$, one has to compute $f(M, n)$ and then check whether $f(M, n) \models \psi_1$. Since

f is polynomial shrinking, there are constants c, d such that computing $f(M, n)$ takes less than $(\|M\| + \|n\|)^c + d$. Furthermore, $\|f(M, n)\| \leq \|M\| + \|n\|$. The induction hypothesis states that determining whether $f(M, n) \models \psi_1$ takes less time than $(\|f(M, n)\| + \|\psi_1\|)^a + b$, which is less than $(\|M\| + \|n\| + \|\psi_1\|)^a + b$. One can assume that $a \geq 2$ and then derive that both parts of this computation can be done in time $(\|M\| + \|\psi\|)^a + b$.

$\square$

This theorem can be used to show that update logics that are based on an update function f with the right properties, have a tractable model checking problem. In the next theorem it is shown that the EFLS update function indeed behaves well (i.e. that it is polynomial shrinking).

5.4.4. THEOREM. *The function $f : (M, (\Gamma, \phi)) \mapsto Up(M, \sigma_\Gamma^e(\phi))$ is polynomial shrinking*

PROOF. The function f takes a coalition $\Gamma \subseteq \Sigma$ and a propositional logic formula $\phi \in \mathcal{L}_p$, calculates the strategy $\sigma_\Gamma^e(\phi)$ and returns the updated model $Up(M, \sigma_\Gamma^e(\phi))$. In order to show that it is polynomial shrinking, we must show two things. First of all that it is computable in polynomial time. Secondly, that the output is smaller than the input. The latter is easy: the reduced model $Up(M, \sigma_\Gamma^e(\phi))$ contains less states than M, and thus it is smaller. It remains to be shown that the function f can be computed in polynomial time. In order to show this, lemma 4.3.2 is used. Define an extensive game $F' = (\Sigma', H, turn', \mathfrak{U})$ where $\Sigma' = \{\Gamma, \Sigma \setminus \Gamma\}$. Thus, it is a two-player game. The function $turn'$ is defined such that $turn'(h) = \Gamma$ iff $turn(h) \in \Gamma$. The function $\mathfrak{U}$ is defined such that $\mathfrak{U}^\Gamma(h) = 1$ if $\pi(h) \models \phi$, and $\mathfrak{U}^\Gamma(h) = 0$ otherwise. It is a constant-sum game, thus $\mathfrak{U}^{\Sigma \setminus \Gamma}(h) = 1 - \mathfrak{U}^\Gamma(h)$. According to lemma 4.3.2, the value function v for this game can be computed in polynomial time. Using the value function, it is not hard to define the strategy $\sigma_\Gamma^e(\phi)$. If $v^\Gamma(h) = 1$ then $\sigma_\Gamma^e(\phi)(h) = \{a | v^\Gamma(ha) = 1\}$. If $v^\Gamma(h) = 0$ then $\sigma_\Gamma^e(\phi)(h) = A(H, h)$. Once one has this strategy, one can use it to compute the model $Up(M, \sigma_\Gamma^e(\phi))$ in linear time: one has to apply this function to every history exactly once. $\square$

A simple model checking program for EFLS has been implemented. The program can be found at `www.csc.liv.ac.uk/~sieuwert/glp`.

The preceding theorem and its proof suggest that one can construct many logics that can be model checked in polynomial time. Should we not search for a more expressive logic than EFLS? At the same time one can wonder whether *polynomial shrinking* is indeed a necessary requirement for the construction of a polynomially model checkable logic. In order to investigate these issues, two conceivable extensions of EFL and EFLS are defined. The first one, EFLN, is an

extension of EFL that allows one to form *nested* abilities. This logic allows one to express interesting properties, but has a very high model checking complexity.

The next logic, EFLNS, is an extension of EFLS that is supposed to capture both nesting and side effects. This logic seems intuitive, but it is hard to give a proper semantics for this language.

5.5 Extensions of EFL

5.5.1 Model Checking EFLN

5.5.1. DEFINITION. Suppose that Σ and P are finite sets (of agents and atomic propositions). The language EFLN consists of formulas ψ generated by the following rules. In these rules $p \in P$ and $\Gamma \subseteq \Sigma$.

$$\phi ::= p \mid \phi \to \phi \mid \bot$$
$$\psi ::= [\Gamma]\psi \mid \Box\phi \mid \psi \to \psi \mid \bot$$

The usual connectives of this logic are interpreted as usual, and $[\Gamma]\phi$ holds if there is a strategy ensuring ϕ.

$$
\begin{array}{llll}
F \models \bot & & \text{never} & \\
F \models \phi \to \psi & \text{iff} & \text{not } F \models \phi \text{ or } F \models \psi & \\
F \models \Box\phi & \text{iff} & \forall h \in Z(H) : \pi(h) \models \phi & \text{where } (\Sigma, H, turn, P, \pi) = F \\
F \models [\Gamma]\phi & \text{iff} & \exists \sigma_\Gamma : Up(F, \sigma_\Gamma) \models \phi &
\end{array}
$$

The following example formulas illustrate how this logic can be used for the 'Alice and Bob eat cake' example.

$$[B]([A]\Box a \wedge [A]\Box b)$$

This example expresses that B can let A choose who gets the biggest piece. This formula does hold for the example: Bob can take the action of letting Alice choose.

$$[B](\neg[A]\Box a \wedge \neg[A]\Box b)$$

The above formula expresses that Bob can make Alice unable to decide. One might think that Bob can satisfy this goal by making a decision himself, but this is not the case. Bob should use a nondeterministic strategy, such as selecting any action, in order to ensure that A cannot determine anything.

$$[B](\neg[B]\Box a \wedge \neg[B]\Box b)$$

This last example sounds strange, because it only makes sense if Bob does not trust himself: It expresses that Bob can get rid of his own abilities. In a game theory setting this can be useful: Bob would like to commit himself so that no-one will try to put pressure on him. This formula does hold in our example: what Bob has to do is to choose either always a, or commit to b. Both these pure strategies do the trick.
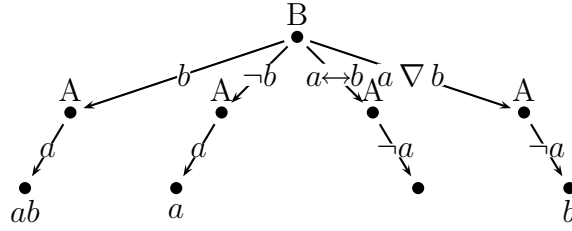
Figure 5.5: A's strategy for letting B decide

Independent Decision Problem

In chapter 4, the independent decision problem was introduced. In this problem two agents A and B can each decide on a certain issue. An agent A can decide whether a should hold or not, and agent B can decide whether b should hold or not. A first protocol for this problem has been given in figure 4.2 on page 69. Using EFL one can conclude that B has additional powers to decide whether a and b should have the same truth value, and we have constructed an equivalent protocol in figure 4.3 on page 70. In EFL these protocols are equivalent, and by enumerating all formulas of the form $[X : \phi][Y : \phi_2]\psi$ one can check that these are also equivalent under EFLS. Since these protocols still look quite different, it would be good to have a logic that can distinguish these protocols. It can be done in the logic EFLN.

Assume that A thinks very highly of agent B, and that A would prefer it if B would decide on the value of both a and b. Whether A can transfer its decision power is expressed by the following formula.

$$\psi = [A]([B](a \wedge b) \wedge [B](a \wedge \neg b) \wedge [B](\neg a \wedge b) \wedge [B](\neg a \wedge \neg b))$$

In the second protocol F_2 this formula holds, since A can use the strategy that is depicted in figure 5.5. On the other hand, this goal cannot be satisfied in game form F_1 in figure 4.2.

5.5.2. THEOREM. *Deciding whether an EFLN formula ϕ holds on an interpreted game form F is PSPACE-complete, even in the case of one agent.*

PROOF. The definition of $F \models \phi$ can be converted into a naive algorithm. For interpreting the construction $[\Gamma]\psi$ one can try all strategies one after another. This may take some time, but does not take much space: applying a strategy gives a smaller model. Therefore, this can be done with an amount of memory that is proportional to the size of the input. Hence the problem is in PSPACE. It remains to be proven that this problem is PSPACE-hard. This can be done by reducing the QBF decision problem of page 31 to the EFLN decision problem. This reduction is explained in general in this proof, and then illustrated using an example. The example is discussed on page 92 and displayed in figure 5.6.

The objective of a QBF problem is to decide for a given formula of the form $\forall x_1 \exists x_2 \forall x_3 \ldots \exists x_{n-1} \forall x_n \phi_q$ whether this formula holds. The formula ϕ_q is a propositional logic formula with only propositions from the set $\{x_i | 0 < i \leq n\}$. Assume that a QBF formula $\forall x_1 \exists x_2 \forall x_3 \ldots \exists x_{n-1} \forall x_n \phi_q$ is given. We have to construct an equivalent EFLN decision problem.

First we construct an interpreted game form $F = (\{X\}, H, turn, P, \pi)$. Define P_q to be the old set of atomic propositions: $P_q = \{x_i | 0 < i \leq n\}$. The set $P = \{q^+, q^- | q \in P_q\}$ contains twice as many atomic proposition: for every old proposition q there is a positive occurrence q^+ and a negative one q^-. The set H is defined as $H = \{\epsilon, p | p \in P\}$. Each terminal run thus consists of one atomic proposition. Naturally, $turn(\epsilon) = X$ and $\pi(p) = \{p\}$.

We can thus construct the required formula ϕ in the following way. Suppose that ϕ_q is in conjunctive normal form. Define a function f in the following way.

$$
\begin{aligned}
f(\neg p) &= \Diamond p^- \\
f(p) &= \Diamond p^+ \\
f(\phi \wedge \psi) &= f(\phi) \wedge f(\psi) \\
f(\phi \vee \psi) &= f(\phi) \vee f(\psi)
\end{aligned}
$$

This function converts a propositional formula ϕ into an EFLN formula. It is used in order to convert the propositional part ϕ_q into EFLN. The next definition defines an EFLN formula ξ_i that expresses that all propositions x_j with $j < i$ have been assigned a value, whereas the propositions x_j with $j > i$ do not have been given a value yet.

$$
\xi_i = \bigwedge_{j \leq i} ((\Diamond x_j^+) \nabla (\Diamond x_j^-)) \wedge \bigwedge_{j > i} ((\Diamond x_j^+) \wedge (\Diamond x_j^-))
$$

The idea used here is that a nondeterministic strategy for the constructed model can be seen as a truth value assignment for the original QBF problem. If the strategy includes x_i^+ then x_i is true in the corresponding assignment, and if x_i^- is included in the strategy then x_i is false in the corresponding assignment. The part $(\Diamond x_j^+) \nabla (\Diamond x_j^-)$ expresses that exactly one of those two actions must be possible, and thus ensures that the assignment is consistent. The formula ξ_i expresses that for $j > i$, both actions must still be possible: $(\Diamond x_j^+) \wedge (\Diamond x_j^-)$. This is necessary because this choice has to be made later.

Next a function g is defined so that $F \models g(\forall x_1 \exists x_2 \forall x_3 \ldots \exists x_{n-1} \forall x_n \phi_q)$ iff the QBF statement $\forall x_1 \exists x_2 \forall x_3 \ldots \exists x_{n-1} \forall x_n \phi_q$ holds. The function g is defined recursively, so that agent X can at each step pick the truth value of exactly one propositional variable x_i.

$$
\begin{aligned}
g(\phi_q) &= f(\phi_q) && \text{if } \phi_q \in \mathcal{L}_p \\
g(\exists x_i \phi) &= [X](\xi_i \wedge g(\phi)) \\
g(\forall x_i \phi) &= \neg [X](\xi_i \wedge \neg g(\phi))
\end{aligned}
$$
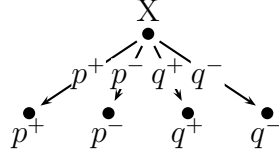
Figure 5.6: The model F_{pq}

The length of the formula $g(\phi)$ is quadratically bounded: $\|g(\phi)\| \leq \|\phi\|^2$. Thus, for a given QBF problem, we have constructed an equivalent EFLN model checking problem in polynomial time. $\square$

To give an example of how the proof works, consider the QBF problem $\forall p \exists q (p \vee \neg q) \wedge (\neg p \vee q)$. The model F_{pq} of the corresponding model checking problem is pictured in figure 5.6. As explained at the end of the proof, the corresponding formula ϕ is rather long, so it is broken down in parts, called ξ_0, ξ_1 and ψ.

$$\xi_0 = (\Diamond p^+ \,\nabla\, \Diamond p^-) \wedge (\Diamond q^+ \wedge \Diamond q^-)$$
$$\xi_1 = (\Diamond p^+ \,\nabla\, \Diamond p^-) \wedge (\Diamond q^+ \,\nabla\, \Diamond q^-)$$
$$\psi = (\Diamond p^+ \vee \Diamond q^-) \wedge (\Diamond p^- \vee \Diamond q^+)$$
$$\phi = \neg[X](\xi_0 \wedge \neg[X](\xi_1 \wedge \psi))$$

One can verify that $F_{pq} \models \phi$, and thus the QBF problem $\forall p \exists q (p \vee \neg q) \wedge (\neg p \vee q)$ has a positive answer. It is interesting to see in this example that one agent alone makes things hard for itself, by denying itself certain rights. Since there is only one agent, this is arguably not even game theory but decision theory. In this framework the way single agents influence their own abilities is the cause of the complexity. The extension to multiple agents comes for free.

5.5.2   Model Checking EFLNS

One advantage of EFLS over EFL is that one can use EFLS to reason about side effects in games. A formula $[\Gamma : \phi]\psi$ expresses that striving towards ϕ has ψ as a side effect. It can be used to express that maximizing profit diminishes social welfare or that knowing A's actions is helpful for B. This feature is not present in EFLN. On the other hand EFLN can be used to express goals for polite and helpful agents. Within EFLN one can say that A wants to help B. A logical next step is therefore to define an even richer language, called EFLNS, that combines the features of EFLS and EFLN. This language is defined in this section, so that we can look at possible interpretations of this language.

5.5.3. DEFINITION. Suppose that Σ and P are finite sets (of agents and atomic propositions), with typical elements $\Gamma \in \Sigma$ and $p \in P$. The language EFLNS consists of formulas ψ generated by the following rules.

$$\phi ::= p \mid \phi \rightarrow \phi \mid \bot$$
$$\psi ::= [\Gamma : \psi]\psi \mid \Box\phi \mid \psi \rightarrow \psi \mid \bot$$

The official reading of a formula $[\Gamma : \phi]\psi$ is that if Γ wants ϕ, then ψ holds. The background assumption is again that if Γ forms a certain plan or adopts a certain strategy, all agents know this strategy immediately (strategies are visible, one might say).

This language is not literally an extension of all logics presented before, but one can easily translate EFL, EFLS and EFLNS formulas into this language. The EFL formula $[\Gamma]\phi$ translates to $[\Gamma : \Box\phi]\Box\phi$. The EFLS formula $[\Gamma : \phi]\psi$ translates to the EFLNS formula $[\Gamma : \Box\phi]\psi$, and the EFLN formula $[\Gamma]\phi$ translates into $[\Gamma : \phi]\phi$. These translations preserve the intuitive meaning of each formula.

We would like to define a semantics for this logic on interpreted game forms M, that is consistent with the semantics of EFL, EFLS and EFLN. Thus, if for some model M it holds that $M \models [\Gamma]\phi$ (using the EFLN semantics), then $M \models [\Gamma : \phi]\psi$ under the EFLNS semantics. Another item on the wish list is that the semantics is an update semantics.

This section contains two results related to possible interpretation of EFLNS. First we give a possible interpretation that is consistent with the semantics of EFLN, and show that under this semantics the new logic is not more expressive than EFLN. Secondly, we show that there is no reasonable update semantics for this language that is consistent with the semantics of EFLN. From these results one can conclude that defining a logic for reasoning about politeness and side effects at the same time is not trivial.

First Interpretation The following rules define an interpretation for EFLNS over interpreted game forms $F = (\Sigma, H, \mathit{turn}, P, \pi)$.

$$
\begin{array}{lll}
F \models \bot & & \text{never} \\
F \models \phi \rightarrow \psi & \text{iff} & \text{not } M \models \phi \text{ or } M \models \psi \\
F \models \Box\phi & \text{iff} & \forall h \in Z(H) : \pi(h) \models \phi \\
F \models [\Gamma : \phi]\psi & \text{iff} & \exists \sigma_\Gamma : Up(F, \sigma_\Gamma) \models \phi \text{ and } Up(F, \sigma_\Gamma) \models \psi
\end{array}
$$

This is a reasonable definition, because it is consistent with EFLN. The update operator $[\Gamma : \phi]\psi$ can be read as saying that it is possible that when Γ uses a strategy for achieving ϕ, then ψ holds. This is similar in spirit to the interpretation of similar formulas in EFLS, but not completely the same. For the logic EFLS, we defined a unique rational strategy $\sigma_\Gamma^e(\phi)$ that is the most general strategy for achieving ϕ. In this semantics we do not use a unique rational strategy, but consider all strategies σ_Γ such that $Up(F, \sigma_\Gamma) \models \phi$.

The main drawback of this semantics is not that it is unreasonable, but that it does not bring us more expressivity than we already had. Everything one can say in EFLNS is under this semantics equivalent to something one could already express using an EFLS formula.

5.5.4. THEOREM. *For every formula $\phi \in$ EFLNS one can find a formula $\psi \in$ EFLN such that for any interpreted game form F the following holds*

$$F \models \phi \iff F \models \psi$$

PROOF. This claim is proven using induction over the structure of the formula $\phi \in$ EFLNS. The base case is provided by formulas $\phi = \bot$ and $\phi = \Box\phi_1$. Both these constructs appear in both languages and are interpreted in the same way, so one can take $\psi = \phi$.

For $\phi = \phi_1 \rightarrow \phi_2$, one can take $\psi = \psi_1 \rightarrow \psi_2$, where $\psi_1, \psi_2 \in$ EFLN are formulas that are equivalent to ϕ_1, ϕ_2 respectively. The induction hypothesis guarantees that these formulas exist.

The difficult case is thus the new construct $[\Gamma : \phi_1]\phi_2$. This formula holds on a model F if Γ has a strategy that satisfies both ϕ_1 and ϕ_2. This can be expressed in EFLN using the formula $\psi = [\Gamma](\psi_1 \wedge \psi_2)$. Again $\psi_1, \psi_2 \in$ EFLS are formulas that are equivalent to ϕ_1, ϕ_2 respectively. The induction hypothesis again guarantees that these formulas exist. $\Box$

Second Interpretation   The previous result shows that if one does not use an update semantics based on a unique 'rational' strategy, it becomes difficult to express side effects. Saying that some strategy for ϕ has ψ as a side effect is not the same as saying that the best or most rational strategy for ϕ has ψ as a side effect.

In order to make this last statement, one would like to have is an update semantics based on an update function f. This function f should return the model $f(F, \Gamma, \phi)$ that one gets when Γ uses the rational or most obvious strategy for obtaining ϕ.

The next theorem shows that one cannot easily find such an update semantics for EFLNS. To be precise we show that there is no non-arbitrary update semantics that is consistent with the interpretation of EFLS. Any update semantics would have to make arbitrary choices about which strategies it considers rational.

5.5.5. FACT. There is no reasonable update semantics for the language EFLNS that is consistent with the interpretation of EFLN.
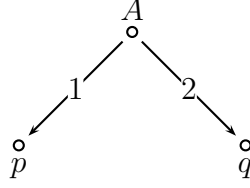
Figure 5.7: A small model F_6

PROOF. An update semantics for EFLNS would have the following form. As a model we again use an interpreted game form $F = (\Sigma, H, turn, P, \pi)$. The symbol f is used for the update function.

$$
\begin{aligned}
F &\models \bot && \text{never} \\
F &\models \phi \rightarrow \psi && \text{iff} \quad \text{not } M \models \phi \text{ or } M \models \psi \\
F &\models \Box\phi && \text{iff} \quad \forall h \in Z(H) : \pi(h) \models \phi \\
F &\models [\Gamma : \phi]\psi && \text{iff} \quad f(F, \Gamma, \phi) \models \psi
\end{aligned}
$$

It is claimed that no suitable function f can be found. This is done by reasoning which properties this function should have, and showing that these properties are contradictory.

Specifically, we look at the behaviour of the function f on an example. In figure 5.7 an interpreted game form F_6 is displayed in which agent A can choose for either p or q. Thus, this model satisfies the following EFLN formula.

$$F_6 \models [A](\Box p \triangledown \Box q)$$

Under the given update semantics for the language EFLNS the following translation of this formula should hold.

$$F_6 \models [A : \Box p \triangledown \Box q](\Box p \triangledown \Box q)$$

Since this is an update semantics, one can apply the following validities for update semantics, that were already stated in section 5.4.

$$
\begin{aligned}
&\models [n](\phi \rightarrow \psi) \rightarrow [n]\phi \rightarrow [n]\psi \\
&\models \neg[n]\psi \leftrightarrow [n]\neg\psi
\end{aligned}
$$

Using these principles one can derive the following.

$$F_6 \models [A : \Box p \triangledown \Box q]\Box p \triangledown [A : \Box p \triangledown \Box q]\Box q$$

And thus exactly one of the following two formulas must hold.

$$F_6 \models [A : \Box p \,\nabla\, \Box q]\Box p$$
$$F_6 \models [A : \Box p \,\nabla\, \Box q]\Box q$$

In the model F_6, the propositions p and q play a symmetric role. All previous logics have a semantics that does not depend on irrelevant properties such as the ordering of propositions or actions. Therefore, one would expect a reasonable semantics not to treat the propositions p or q differently. Any choice for one of the two similar formulas would be arbitrary, and would thus be unreasonable. $\Box$

The word *reasonable* used in the previous fact is of course a vague word. The word has been interpreted in the proof as meaning that any semantics should behave similar in symmetric situations.

The proof is based upon the fact that there are two incompatible strategies for bringing about A's goal $\Box p \,\nabla\, \Box q$. In the interpretation of EFLN, the idea of using the most general strategy was used to solve such dilemmas. Such a strategy would leave the agent with the most freedom, and thus it would be rational for the agent to use such a strategy. Unfortunately, for the example goal $\Box p \,\nabla\, \Box q$ neither effective strategy is more general than the other.

5.6   Conclusion

This chapter defines three new languages EFLS, EFLN and EFLNS that are richer variants of the logic EFL. Using these richer languages one can distinguish protocols that are equivalent for EFL. These languages are thus useful for choosing between protocols.

To illustrate this conclusion with some examples, consider again the following two problems.

joint decision problem A decision p can be taken if either A or B think that p should be the case. If both agents do not want p, it should be rejected.

independent decision problem An agent A can decide whether a should hold or not, and agent B can decide whether b should hold or not.

Both problems can be described in EFL, and different protocols for both problems exist. From the viewpoint of EFL, all these protocols are equivalent. In this chapter, we have seen that the logic EFLS can be used to distinguish two protocols for the joint decision problem.

The logic EFLN on the other hand can distinguish the two solution for the independent decision problems. Thus, the added expressibility of both logics allows us to answer more detailed questions about protocols.

For each of these logics one can determine the complexity of the model checking problem, which indicates whether the language can be used for verification in practice. The next table lists these results.

logic	nesting	side effects	model checking
EFL	×	×	P
EFLS	×	✓	P
EFLN	✓	×	PSPACE-complete
EFLNS	✓	✓	PSPACE-complete

Verification of properties expressed in EFL and EFLS is thus feasible, whereas verification of properties expressed in EFLN can be very difficult. The definition of EFLNS that we have given makes this language translatable into EFLN, so it must have the same model checking complexity. In the previous section an argument is given why a better semantics is hard to define.

Chapter 6
Preference Logics in Extensive Games

This chapter is based on joint research with **Olivier Roy** and **Johan van Benthem**.

6.1   Introduction

The logic EFL presented in chapter 4 provides a high level view of protocols. It can distinguish some protocols, but many protocols that somehow feel different are equivalent according to the logic EFL. One explanation for this result is that agents, according to EFL act without knowledge of the plans of other agents. They search for strategies that lead to success no matter what the other agents do. This approach is in stark contrast with the usual assumptions of game theory. Under the assumption of *complete information*, agents know the preferences of other agents. Therefore, agents can predict what other agents do, and use this to their advantage. In order to provide a logical model that does recognize the importance of agent preferences, we introduce in this chapter a logic based on preference logic.

  Consider the two protocols in figure 6.1. In these games, two agents A and B are faced with the problem of a dirty shared desk. The value of a clean desk is 2 for each agent, but the task of cleaning is valued at utility -1. The game
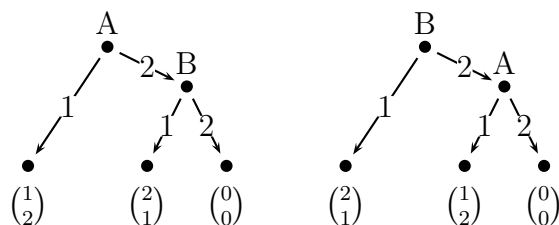
Figure 6.1: Two extensive games

on the left models the situation where Alice arrives first at the office. She can decide to clean the desk, which gives her utility 1. She can also ignore the dirty desk. When Bob arrives and the desk is clean, he experiences utility 2. If Alice has not cleaned the desk, Bob can clean the desk. This gives him utility 1 and Alice utility 2. Bob's other option is to ignore the problem, in which case both agents experience utility 0.

The game on the right is very similar, except that in this game Bob is the first agent to arrive at the office. The roles of the two agents are thus reversed. The central question about these two games is whether these games are equivalent. If that is the case, then apparently either mechanism is a fair way for both agents to jointly decide whether to clean. If however these games are not equivalent, then one agent might have an advantage over the other agent.

The assumption of *complete information* that is common in game theory tells us that agents are not only aware of their own preferences, but also of each other's preferences. Thus, both agents do not only know that they want a clean desk, but they also know that the other agent wants exactly the same. This information can be used by agents to their advantage. The agent that arrives first, Alice in the left game, knows that the other agent prefers a clean desk. If she does not clean the desk, then it is best for Bob to do the cleaning. Since she knows this, she decides to ignore the problem. The rational outcome of the left game is thus that Bob cleans the desk. The rational outcome of the right game is that Alice does the cleaning. Both games are thus not equivalent for the agents, and each agent is motivated to arrive first.

The outcomes of both games that are predicted above are subgame perfect equilibria: The first agent reasons what will happen in the subgame where she does not clean the desk, and uses this information to decide whether she should clean. Such a subgame perfect equilibrium can be computed using a procedure called *backward induction*, and these terms are considered synonyms here. In this chapter, a logic is presented that can capture this reasoning. Since it seems that this preference logic might be interesting in its own right, a completeness proof for this logic is also given.

The structure of this chapter is the following. We define the language of preference logic in section 6.2. In this section we define a semantics for this language, a notion of bisimulation and a proof system. Since this language is defined in what one can call a non-modal-logic style, the competeness proof of the defined proof system is rather hands-on. Therefore, the next section, section 6.3, is used to define what one can call a modern variant of preference logic. For this logic, a completeness proof using standard modal logic techniques can be given. Section 6.4 combines preference logic with a logic for reasoning about game trees. This combined logic is used in section 6.5 to characterize the backward induction solution concept.

6.2   Preference Logic

To have a preference means that one puts "one thing before or above another"[80]. In the context of games, it is important to know the preferences that agents have between the various outcomes. One can model such preferences by giving binary relations between outcome states. A preference relation is thus a set $R = \{(x,y)|\text{outcome } x \text{ is as good as or better than } y\}$. In our logic, one can use $\phi\langle Pref\rangle\psi$ to say that there is a ϕ state x and a ψ state y such that $(x,y) \in R$. If there are multiple agents, the agent X whose preferences are being discussed is indicated with a subscript: $\phi\langle Pref\rangle_X\psi$. The use of preferences is an alternative to the use of utilities. If one has a utility function $\mathfrak{U}$, one can define a preference relation by stating that $x\langle Pref\rangle y$ iff $\mathfrak{U}(x) \geq \mathfrak{U}(y)$. Thus, agents prefer outcomes that have a higher utility over outcomes with a lower utility. On the other hand, if one has a preference relation one can construct a utility function that represents the same structure. This can even be done for probability distributions over outcomes, which arise in mixed strategy games [75]. The utility function one gets is of course not unique: one can apply any linear transformation to the utility function, and still get the same preference relations. This can be seen as an argument against utility functions and thus in favor of preference relations. In mathematical definitions, such as those given in the chapter on game theory of this dissertations, the use of utility functions is notationally more convenient. For logical purposes the use of preference relations makes sense. After all, modal languages are 'languages for talking about relational structures' [12, p. ix].

6.2.1. DEFINITION. Suppose the finite sets Σ and P are given, and let $X \in \Sigma$ and $p \in P$ be typical elements. Preference logic $\mathcal{L}_P$ consists of formulas ϕ generated by the following rule.

$$\phi ::= p \mid \phi\langle Pref\rangle_X\phi \mid \phi \rightarrow \phi \mid \bot$$

A logic for reasoning about preferences with a very similar syntax was already proposed in 1963 [119]. However our interpretation for this language is original. One can introduce other operators by definition in terms of the given operators. Besides the usual logical connectives, one can define the following.

$$E_X\phi \stackrel{\text{def}}{=} \phi\langle Pref\rangle_X\phi$$

$$A_X\phi \stackrel{\text{def}}{=} \neg E_X\neg\phi$$

$$\phi[Pref]_X\psi \stackrel{\text{def}}{=} \neg(\psi\langle Pref\rangle_X\phi)$$

One can think of $\phi[Pref]_X\psi$ as saying that ϕ is strictly preferred by X over ψ. This is some kind of universal quantification: it refers to all states satisfying ϕ and all states satisfying ψ. The operator $\phi\langle Pref\rangle_X\psi$ is the dual of this operator, and says (in preference models) that it is possible that ϕ is as least as good as

ψ. $E_X\phi$ means that there exist circumstances in which ϕ holds, whereas $A_X\phi$ expresses that ϕ always holds.

As an example, the following formula expresses a reasonable property of preferences. It expresses that if p is always better than q and q is always better than r, then q is always better than r.

$$p[Pref]_X q \wedge q[Pref]_X r \rightarrow p[Pref]_X r$$

Whether the principle expressed by this formula holds for a given relation, depends of the constraints that we put on such relation. Three properties of relations turn out to be important. A relation R is *total* if $\forall xy : \; x \neq y \Rightarrow (xRy \vee yRx)$. It is *reflexive* if $\forall x : \; xRx$, and *anti-symmetric* if $\forall xy : \; (xRy \wedge yRx) \rightarrow x = y$. The strict version R^s is the relation $R^s = \{(a,b)|aRb \wedge \neg bRa\}$. A relation R is *strict-transitive* if $\forall xyz : \; xR^s y \wedge yR^s z \rightarrow xR^s z$. In normal circumstances one would expect a preference relation to be total, strict-transitive and reflexive.

These properties reflect reasonable properties that one would expect from a preference relation. If one assumes that preferences follow from an underlying utility relation, these properties should hold. Indeed these three properties are used in the definition of a *preference model*, defined below.

6.2.2. DEFINITION. A *reflexive frame* $\mathcal{F}$ is a tuple $\mathcal{F} = (W, \Sigma, \{\succeq\}_\Sigma)$ such that W is a set of outcomes, Σ is a finite set of agents and $\succeq_X \subseteq W \times W$ is a reflexive relation between worlds for each agent X of A.

6.2.3. DEFINITION. A *minimal preference model* M is a tuple $M = (W, \Sigma, \{\succeq\}_\Sigma, P, \pi)$ such $(W, \Sigma, \{\succeq\}_\Sigma)$ is a reflexive frame, P is a finite set of atomic propositions and $\pi : W \rightarrow 2^P$ assigns propositions to outcomes.

6.2.4. DEFINITION. A *preference model* M is a minimal preference model $M = (W, \Sigma, \{\succeq\}_\Sigma, P, \pi)$ such that each $\succeq_X \subseteq W \times W$ is a strict-transitive, total relation.

In the context of preference modes we refer to elements of W as either outcomes, states or worlds. Intuitively the worlds $w \in W$ are possible outcomes and $w \succeq w'$ means that w is as least as good as w'. An example of a preference model is displayed in figure 6.2. In this figure the preferences of the single agent are indicated by the vertical position of states: Higher states are preferred over lower states. No lines between states are therefore necessary to indicate the preference information.

As is common in modal logic we define a pointed model to be a pair M, w where M is a model with a set of worlds W and $w \in W$. Formulas are interpreted over pointed models M, w in the following way.
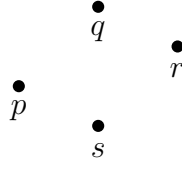
Figure 6.2: A single agent preference model

$$
\begin{array}{lll}
M, w \models \bot & & \text{never} \\
M, w \models p & \text{iff} & p \in \pi(w) \\
M, w \models \phi\langle Pref\rangle_X\psi & \text{iff} & \exists(w', w'') \in \succeq_X : M, w' \models \phi \text{ and } M, w'' \models \psi \\
M, w \models \phi \to \psi & \text{iff} & \text{not } M, w \models \phi \text{ or } M, w \models \psi
\end{array}
$$

Let M be the model displayed in figure 6.2 and let w_p be the world where p holds. Then we can show the following.

$$M, w_p \models p \wedge (p\langle Pref\rangle_X s) \wedge (q\langle Pref\rangle_X r)$$

In standard modal logic, an accessibility relation is used in the interpretation of modal operators. The truth conditions of the modal operator only depend on accessible worlds. The preference operator $\langle Pref\rangle_X$ looks at all worlds, so one might say it uses the universal accessibility relation, in which any world is related to any world. Such an operator is called a *global* operator. In the following lemma it is shown that such an operator can be used to define the operator $E\phi$ that expresses that a world satisfying the formula ϕ exists.

6.2.5. LEMMA. *Let* $M = (W, \Sigma, \{\succeq\}_\Sigma, P, \pi)$ *be a minimal preference model*

$$M, w' \models E_X\phi \Leftrightarrow \exists w \in W : M, w \models \phi$$

PROOF. Suppose that $M, w' \models E_X\phi$. By definition this means that $M, w' \models \phi\langle Pref\rangle_X\phi$. This means that there are states w_1 and w_2 such that, among other things, $M, w_1 \models \phi$. For the reverse direction, suppose that $\exists w \in W : M, w \models \phi$. Because the relation $\succeq_X$ is reflexive, we have $w \succeq w$ and therefore there are worlds $w_1 = w, w_2 = w$ such that $w_2 \succeq_X w_1$, $M, w \models \phi$ and $M, w \models \phi$. We conclude that $M, w' \models \phi\langle Pref\rangle_X\phi$ and thus $M, w' \models E_X\phi$. $\square$

A corollary of this theorem is that $\models E_X\phi \leftrightarrow E_Y\phi$. It is therefore harmless to omit the subscript $_X$ and simply write $E\phi$ instead of $E_X\phi$.

6.2.1 Bisimulation

In this section we define a notion of bisimulation for preference models, and prove that two models are bisimilar if and only if they satisfy the same preference logic formulas.

6.2.6. DEFINITION. Let $M = (W, \Sigma, \{\succeq\}_\Sigma, P, \pi)$ and $M' = (W', \Sigma, \{\succeq'\}_\Sigma, P, \pi')$ be two minimal preference models. A relation $R \subseteq W \times W'$ is a bisimulation iff

- all pairs of related worlds $(w, w') \in R$ satisfy the same atomic propositions: $\pi(w) = \pi'(w')$ and

- for all $v, w \in H$ with $v \succeq_X w : \exists v', w' \in W' : vRv', wRw' \wedge v' \succeq'_X w'$ and

- for all $v', w' \in H'$ with $v' \succeq'_X w' : \exists v, w \in H : vRv', wRw' \wedge v \succeq_X w$.

We say that two pointed models M, w and M', w' are *bisimilar*, iff there exists a bisimulation R between M and M' such that $(w, w') \in R$. Two pointed models M, w and M', w' are *equivalent* iff they satisfy exactly the same formulas: $\forall \phi : M, w \models \phi \leftrightarrow M', w' \models \phi$. The next theorem relates these two notions.

6.2.7. THEOREM. *Let P be finite and let $M, w = (W, \Sigma, \{\succeq\}_{X \in \Sigma}, P, \pi), w$ and $M', w' = (W', \Sigma, \{\succeq'\}_{X \in \Sigma}, P, \pi'), w'$ be two pointed models. The models M, w and M', w' are bisimilar iff they are equivalent .*

PROOF. Let $M = (W, \Sigma, \succeq, P, \pi)$ and $M' = (W', \Sigma, \succeq', P, \pi')$. Suppose there is a bisimulation R between M and M' such that wRw'. We show that that these models are equivalent by induction on the structure of formulas ϕ. The case where $\phi = \bot$ is easy. For any two worlds v, v' we have that $M, v \not\models \bot$ and $M', v' \not\models \bot$. Consider now the case of $\phi = p \in P$. Let v, v' again be arbitrary worlds in W, W' respectively. Suppose we have $M, v \models p$. The first condition of bisimulation tells us that $M', v' \models p$. Because our notion of bisimulation is symmetric, we can repeat the argument with M and M' interchanged for the "only if" part. Assume now the induction hypothesis that for all subformulas ψ of ϕ and all worlds v, v' we have that $M, v \models \psi$ iff $M', v' \models \psi$. It follows, using this hypothesis, that if $\phi = \psi_1 \rightarrow \psi_2$ then $M, v \models \phi$ iff $M', v' \models \phi$. So as a last step we show that $M, v \models \psi_1 \langle Pref \rangle \psi_2$ iff $M', v' \models \psi_1 \langle Pref \rangle \psi_2$. Suppose $M, v \models \psi_1 \langle Pref \rangle_X \psi_2$. This means that there are worlds $x, y \in W$ with $M, x \models \psi_1$, and $M, y \models \psi_2$ and $x \succeq_X y$. Using the definition of bisimulation and the induction hypothesis, we know that there are worlds $x', y' \in W'$ such that $M', x' \models \phi$, and $M', y' \models \psi$ and $x' \succeq'_X y'$. Thus, we know that $M', w' \models \psi_1 \langle Pref \rangle_X \psi_2$. The same argument with M and M' interchanged can be used to show that $M', w \models \psi_1 \langle Pref \rangle_X \psi_2$ only if $M, w' \models \psi_1 \langle Pref \rangle_X \psi_2$.

It remains to be proven that if $\forall \phi : M, v \models \phi \leftrightarrow M', v' \models \phi$, then there is a bisimulation R between M and M' with vRv'. We assume that $\forall \phi : M, v \models \phi \leftrightarrow M', v' \models \phi$. The relation R that is needed is defined in the following way: vRv' iff $\pi(v) = \pi'(v')$. The first condition of bisimulation is thus satisfied. In order to check the second condition, take two worlds $v, u \in W$ with $v \succeq_X u$. One can find formulas ϕ_v and ϕ_u that describe exactly which atoms are true in v and u respectively. This is possible because P is finite. Since $v \succeq_X u$,
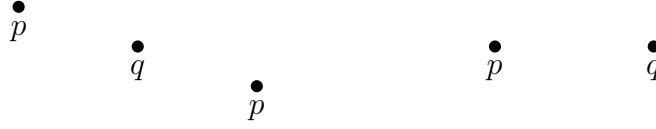
Figure 6.3: Two bisimilar models

we have that $M, w \models (\phi_v \langle Pref \rangle_X \phi_u)$, and thus $M', w' \models (\phi_v \langle Pref \rangle_X \phi_u)$. There must be elements $v' \succeq' u'$ so that $M', v' \models \phi_v$ and $M', u' \models \phi_u$. This proves the second clause of the bisimulation definition. A symmetric argument gives us a proof for the third condition. $\qquad \square$

Using this notion, one can determine whether two models satisfy the same formulas. In figure 6.3 two bisimilar models are displayed. In the right model, two equally good states are displayed, one of them is labeled with proposition p, the other one with proposition q. The left model has three states that are not equally preferred by the single agent of this model. The left p state is better than the q state, which is better than the right p state. (Again the preferences are indicated by the vertical position of the states.)

6.2.2 Proof System

A proof system $\mathcal{S}_P$ for preference logic can be defined in the following way. First we list the axioms, then the reasoning rules. In $AndConv_1$ and $AndConv_2$, it does not matter whether $X = Y$ or not. The symbol $\pm$ stands for a possible negation: $\pm \phi$ can be either $\neg \phi$ or ϕ. In each instance of the axioms $AndConv_1$ and $AndConv_2$, one must make the same choice for this symbol.

Prop	*All propositional tautologies*
E − intro	$\phi \rightarrow E_X \phi$
Exist	$\phi \langle Pref \rangle_X \psi \rightarrow (E_Y \phi \wedge E_Y \psi)$
K_1	$(A_Y(\phi \rightarrow \psi) \wedge \phi \langle Pref \rangle_X \chi) \rightarrow \psi \langle Pref \rangle_X \chi$
K_2	$(A_Y(\phi \rightarrow \psi) \wedge \chi \langle Pref \rangle_X \phi) \rightarrow \chi \langle Pref \rangle_X \psi$
$AndDist_1$	$\phi \langle Pref \rangle_X \psi \rightarrow ((\phi \wedge \xi) \langle Pref \rangle_X \psi \vee (\phi \wedge \neg \xi) \langle Pref \rangle_X \psi)$
$AndDist_2$	$\phi \langle Pref \rangle_X \psi \rightarrow (\phi \langle Pref \rangle_X (\psi \wedge \xi) \vee \phi \langle Pref \rangle_X (\psi \wedge \neg \xi))$
$AndConv_1$	$(\pm(\phi \langle Pref \rangle_Y \psi) \wedge \xi) \langle Pref \rangle_X \chi \leftrightarrow \pm(\phi \langle Pref \rangle_Y \psi) \wedge \xi \langle Pref \rangle_X \chi$
$AndConv_2$	$\chi \langle Pref \rangle_X (\pm(\phi \langle Pref \rangle_X \psi) \wedge \xi) \leftrightarrow \pm(\phi \langle Pref \rangle_Y \psi) \wedge \chi \langle Pref \rangle_X \xi$

The reasoning rules for this proof system are Modus Ponens and Necessitation for A_X. These two rules are listed below.

$$\frac{\phi}{A_X\phi} \qquad \frac{\phi \quad \phi \rightarrow \psi}{\psi}$$

The following rules can be derived in this system.

$$OrDist_1 \qquad (\phi \vee \psi)\langle Pref\rangle_X\chi \rightarrow (\phi\langle Pref\rangle_X\chi) \vee (\psi\langle Pref\rangle_X\chi)$$

$$OrDist_2 \qquad \chi\langle Pref\rangle_X(\phi \vee \psi) \rightarrow (\chi\langle Pref\rangle_X\phi) \vee (\chi\langle Pref\rangle_X\psi)$$

The first rule $OrDist_1$ can be derived using the following instance of $AndDist_1$:

$$(\phi \vee \psi)\langle Pref\rangle_X\chi \rightarrow (((\phi \vee \psi) \wedge \phi)\langle Pref\rangle_X\chi \vee ((\phi \vee \psi) \wedge \neg\phi)\langle Pref\rangle_X\chi)$$

Since $((\phi \vee \psi) \wedge \phi)$ is equivalent to ϕ and $((\phi \vee \psi) \wedge \neg\phi)$ is equivalent to ψ in propositional logic, one can use $Prop$ and K_1 to derive $OrDist_1$. Similarly one can prove $OrDist_2$ using an instance of $AndDist_2$.

It is interesting to consider the nesting of preferences. In order to measure the level of nesting, define the function l

$$
\begin{aligned}
l(\bot) &= 0 \\
l(p) &= 0 \\
l(\phi \rightarrow \psi) &= \max(l(\phi), l(\psi)) \\
l(\phi\langle Pref\rangle_X\psi) &= 1 + \max(l(\phi), l(\psi))
\end{aligned}
$$

The language of preference logic allows for nested preference operators, but it is not clear what such nested formulas express. Does it make sense to say that "to prefer ϕ over ψ is at least as good as χ"? Someone who holds that such sentences are meaningless may decide to exclude them from the logic by restricting the language to $\mathcal{L}_P^{\leq 1} = \{\phi \in \mathcal{L}_P | l(\phi) \leq 1\}$ where nesting of operators is not allowed. We have decided to keep our approach as general as possible on this point, so we did not use this restriction. But, interestingly enough, using axioms $AndConv_1$ and $AndConv_2$ one can show that we can always 'unnest' a nested formula: a formula of nesting $l > 1$ is always equivalent to a formula of nesting $(l - 1)$. This is proven in the next lemma. This equivalence plays a role in the completeness proof, and it shows that even if we can nest preference formulas, such nesting does not add to the expressivity of the language.

6.2.8. LEMMA. *For all formulas $\phi \in \mathcal{L}_P$ with $l(\phi) > 1$ there is a formula χ such that $\mathcal{S}_P \vdash \phi \leftrightarrow \chi$ and $l(\chi) = l(\phi) - 1$*

PROOF. We prove that the theorem holds for $\phi = \psi\langle Pref\rangle_X\xi$. For other formulas ϕ it follows by an induction argument.

Assume $\phi = \psi\langle Pref\rangle_X\xi$. Any formula can be written in disjunctive normal form. We use the notation $dnf(\phi)$ for the disjunctive normal form (see page 14) of any formula ϕ. The formula $\phi \leftrightarrow dnf(\phi)$ is a propositional tautology and thus we can derive $\mathcal{S}_P\vdash\phi \leftrightarrow dnf(\phi)$ and even $\mathcal{S}_P\vdash A_X(\phi \leftrightarrow dnf(\phi))$ for any formula ϕ. For the language $\mathcal{L}_P$ the disjunctive normal form has the following appearance: $dnf(\phi) = \vee_m \wedge_j \pm(\phi^1_{mj}\langle Pref\rangle_{h_{mj}}\phi^2_{mj}) \wedge \phi^p$ where $\pm$ indicates the possible appearance of a negation, and $l(\phi^p) = 0$.

In the next derivation j, k, l and m are indices over formulas (elements of a conjunction or disjunction). The indices X and Y range over agents. The inner indices Y are actually dependent on j, k, l and m, but these arguments have been suppressed. Instead of writing Y_{mj} on the left and Y_{kl} on the right, and Y_{klmn} in the last formula, we have written Y. Also the symbol $\pm$, which indicates a possible negation, should be indexed so that corresponding occurences of this symbol are interpreted correspondingly. These indices are also omitted for readability.

$$\psi\langle Pref\rangle_X\xi$$
$$\Leftrightarrow Prop, Nec_A, K_{1,2}$$
$$[dnf(\psi)]\langle Pref\rangle_X[dnf(\xi)]$$
$$\Leftrightarrow$$
$$[\vee_m \wedge_j \pm(\psi^1_{mj}\langle Pref\rangle_Y\psi^2_{mj})) \wedge \psi^p]\langle Pref\rangle_X[\vee_k \wedge_l \pm(\xi^1_{kl}\langle Pref\rangle_Y\xi^2_{kl})) \wedge \xi^p]$$
$$\Leftrightarrow OrDist_{1,2}$$
$$\vee_m \vee_k[\wedge_j \pm (\psi^1_{mj}\langle Pref\rangle_Y\psi^2_{mj}) \wedge \psi^p]\langle Pref\rangle_X[\wedge_l \pm (\xi^1_{kl}\langle Pref\rangle_Y\xi^2_{kl}) \wedge \xi^p]$$
$$\Leftrightarrow AndConv_{1,2}$$
$$\vee_m \vee_k(\wedge_j \pm (\psi^1_{mj}\langle Pref\rangle_Y\psi^2_{mj})) \wedge (\wedge_l(\xi^1_{kl}\langle Pref\rangle_Y\xi^2_{kl})) \wedge [\psi^p\langle Pref\rangle_X\xi^p]$$
$$\overset{\text{def}}{=} \chi$$

One can see that $l(\chi) = l(\phi) - 1$ by noting that $l(\chi) = \max(l(\psi), l(\xi))$ and $l(\phi) = 1 + \max(l(\psi), l(\xi))$. $\qquad\Box$

6.2.9. THEOREM. *The above proof system is sound: $\mathcal{S}_P\vdash\phi$ implies $\models \phi$*

The validity proofs for each axiom are given below.

- $(E - intro)$ Suppose that $M, w \models \phi$. Since $\succeq_X$ is reflexive, we have that $w \succeq_X w$, and thus $M, w \models \phi\langle Pref\rangle_X\phi$. This is the same as $M, w \models E\phi$.

- $(Exist)$ Suppose $M, w \models \phi\langle Pref\rangle_X\psi$. This means that there are two worlds w' and w'' such that $w' \succeq w''$, and these worlds satisfy $M, w' \models \phi$ and $M, w'' \models \psi$. Since the relation $\succeq_Y$ is reflexive for all agents Y, we obtain

$M, w' \models E_Y \phi$, and $M, w' \models E_Y \psi$. These two conclusions can be combined to derive $M, w' \models E_Y \phi \wedge E_Y \psi$.

- (K_1 - K_2). It is not hard to see that $A_X \phi$ is true iff ϕ holds in every world of the model.

  Suppose that $M, w \models A_Y(\phi \rightarrow \psi)$ and $M, w \models \phi \langle Pref \rangle_X \chi$, for arbitrary agents X and Y. This means that there are $w_1, w_2 \in W$ such that $w_1 \succeq_X w_2$, $M, w_1 \models \phi$ and $M, w_2 \models \chi$. Furthermore for all worlds $w' \in W$ we have $M, w' \models (\phi \rightarrow \psi)$. In particular, this last fact implies that $M, w_1 \models \phi \rightarrow \psi$, from which we get $M, w_1 \models \psi$. Since $w_1 \succeq_X w_2$, we have $M, w \models \psi \langle Pref \rangle_X \chi$. The argument for K_2 is similar.

- ($AndDist_1 - AndDist_2$) Suppose that $M, w \models \phi \langle Pref \rangle_X \psi$. This means that there are w' and w'' such that $M, w' \models \phi$, $M, w'' \models \psi$ and $w' \succeq_X w''$. For every formula ξ we know that either $M, w' \models \xi$ or $M, w' \models \neg \xi$, and hence $M, w' \models (\phi \wedge \xi) \vee (\phi \wedge \neg \xi)$. Thus, $M, w \models ((\phi \wedge \xi)\langle Pref \rangle_X \psi) \vee ((\phi \wedge \neg \xi)\langle Pref \rangle_X \psi)$. The argument for $AndDist_2$ is similar.

- ($AndConv_1 - AndConv_2$) As for the other cases, we only prove soundness for $AndConv_1$, the argument for the other axiom being entirely similar. We also prove soundness for the axiom with a positive occurrence of $\phi \langle Pref \rangle_X \psi$. Suppose that $M, w \models (\phi \langle Pref \rangle_X \psi \wedge \chi)\langle Pref \rangle_X \xi$. This means there are two worlds $w_1, w_2 \in W$ such that $M, w_1 \models \phi \langle Pref \rangle_X \psi \wedge \chi$ and $M, w_2 \models \xi$ and $w_1 \succeq_X w_2$. From these one can derive $M, w_1 \models \phi \langle Pref \rangle_X \psi$ and $M, w_1 \models \chi$. This new fact can then be used to show that $M, w \models \chi \langle Pref \rangle_X \xi$ and $M, w \models \phi \langle Pref \rangle_X \psi$. Finally $M, w \models \phi \langle Pref \rangle_X \psi \wedge \chi \langle Pref \rangle_X \xi$.
  For the reverse implication, suppose that $M, w \models \phi \langle Pref \rangle_X \psi \wedge \chi \langle Pref \rangle_X \xi$. This means firstly that there are two worlds $w_1, w_2 \in W$ such that $M, w_1 \models \phi$, $M, w_2 \models \psi$ and $w_1 \succeq_X w_2$ and secondly that there are two worlds $w_3, w_4 \in W$ such that $M, w_3 \models \chi$, $M, w_4 \models \xi$ and $w_3 \succeq_X w_4$. From these to facts one can derive that $M, w_3 \models \phi \langle Pref \rangle_X \psi \wedge \chi$ and this can be used to conclude $M, w \models (\phi \langle Pref \rangle_X \psi \wedge \chi)\langle Pref \rangle_X \xi$.

6.2.10. THEOREM. *The above proof system is complete:* $\models \phi$ *implies* $\mathcal{S}_P \vdash \phi$

Suppose ϕ is given. Assume that $\neg \phi$ cannot be proven, in other words that ϕ is consistent. We construct a model M with a world w such that $M, w \models \phi$. Let $S = \overline{\{\phi\}}$ be a maximally consistent set containing ϕ. All we need to do is to show that there is a model M, w such that $\forall \psi \in S : M, w \models \psi$. It then follows that $M, w \models \phi$.

Let P be the set of atoms occurring in ϕ. A maximal propositional conjunction ϕ^m is an ordered conjunction of atoms or negated atoms such that every atom in P is mentioned exactly once. Thus, if $P = \{a, b, c\}$ then $a \wedge \neg b \wedge c$ is a maximal propositional conjunction, but $b \wedge a$ is not. The ordering ensures that

equivalent maximally consistent formulas are exactly equal. Let S^{prop} consist of all propositional logic formulas in S, and let $S^{max} = \{\phi^m \langle Pref \rangle_X \psi^m \in S | \phi^m, \psi^m$ are maximal propositional conjunctions$\}$. The proof now proceeds as follows. We first construct a model M such that there is a state w in M so that $M, w \models \phi$ for all $\phi \in S^{prop}$ and such that every state in M satisfies all formulas in S^{max}. Then we show that M, w satisfies all formulas in S.

The model $M = (W, \Sigma, \succeq, P, \pi)$ is defined in the following way. Let $W = \{\phi^m | (E\phi^m) \in S^{max}\}$. So W contains maximal propositional conjunctions. Σ is the set of agents mentioned in S, and this set is finite because only a finite number of agents can be mentioned in a finite formula. We define $\pi(\phi^m) = \{p | \mathcal{S}_P \vdash \phi^m \to p\}$. The preference relation is defined by $\phi^m \succeq_X \psi^m \Leftrightarrow (\phi^m \langle Pref \rangle_X \psi^m) \in S^{max}$. This relation is reflexive: If $(\phi^m \langle Pref \rangle_X \psi^m) \in S^{max}$ then, because of $Exist$, we know $E_X \phi^m = \phi^m \langle Pref \rangle_X \phi^m \in S^{max}$. Similarly for ψ^m.

The world w that we need can be found in the following way. Suppose that $\phi^m \in S^{prop}$ for a maximal propositional conjunction ϕ^m. This implies, using $E - intro$, that $E\phi^m \in S$ and thus there is a world w such that $M, w \models \phi^m$. Since every formula $\phi \in S^{prop}$ is a consequence of ϕ^m, we have that $M, w \models \phi$ for all $\phi \in S^{prop}$.

6.2.11. LEMMA. *Let ϕ and ψ be propositional formulas. $\phi \langle Pref \rangle_X \psi \in S$ iff $M, w \models \phi \langle Pref \rangle_X \psi$*

PROOF. Let ϕ, ψ be propositional formulas and assume $\phi \langle Pref \rangle_X \psi \in S$. We can repeatedly apply axioms $AndDist_1$ and $AndDist_2$ for all the propositions, and use the fact that $\chi \vee \xi \in S$ implies $\chi \in S$ or $\xi \in S$, to obtain two maximal conjunction ϕ^m and ψ^m such that $\mathcal{S}_P \vdash \phi^m \to \phi$ and $\mathcal{S}_P \vdash \psi^m \to \psi$ and $\phi^m \langle Pref \rangle_X \psi^m \in S$. It follows that $\phi^m \langle Pref \rangle_X \psi^m \in S^{max}$. Using axiom $Exist$ we conclude that $E\phi^m \in S^{max}$ and $E\psi^m \in S^{max}$. Therefore, $\phi^m \in W$ and $\psi^m \in W$. From the definition of $\succeq_X$ we obtain that $\phi^m \succeq_X \psi^m$ and this leads us to conclude that $M, w \models \phi^m \langle Pref \rangle_X \psi^m$. Using the soundness of K_1 and K_2 we can derive that $M, w \models \phi \langle Pref \rangle_X \psi$.

For the reverse part, assume that $M, w \models \phi \langle Pref \rangle_X \psi$. From the soundness of $AndDist_1$ and $AndDist_2$ it follows that there are maximal conjunctions ϕ^m and ψ^m such that $M, w \models \phi^m \langle Pref \rangle_X \psi^m$. This implies that $\phi^m \langle Pref \rangle_X \psi^m \in S$, and using K_1 and K_2 we conclude that $\phi \langle Pref \rangle_X \psi \in S$. $\qquad\square$

An induction argument over the level of nesting in sets S can be given to show that $\phi \in S$ iff $M, w \models \phi$ for any formula ϕ with $l(\phi) \leq 1$. To show that this is also the case for higher level formulas, with nested preferences, we can use lemma 6.2.8 to find for any formula ϕ a formula ξ such that $\mathcal{S}_P \vdash \phi \leftrightarrow \xi$ and $l(\xi) \leq 1$. If $\phi \in S$ then $\xi \in S$, and since $l(\xi) \leq 1$ we can find a model M, w such that $M, w \models \xi$. It now follows that $M, w \models \phi$, and we have shown that the given

proof system is complete for minimal preference models.

Now we can make the step from minimal preference models to preference models. We say that an axiom A is sound on a set of reflexive frames S if every instance $\phi \in A$ is true on every model $M = (W, \Sigma, \{\succeq_X\}_{X \in \Sigma}, P, \pi)$ such that $(W, \Sigma, \{\succeq_X\}_{X \in \Sigma}) \in S$. An axiom scheme ϕ is complete for S if for every reflexive frame $(W, \Sigma, \{\succeq_X\}_{X \in \Sigma}) \notin S$ can be extended to a pointed model $M, w = (W, \Sigma, \{\succeq_X\}_{X \in \Sigma}, P, \pi), w$ such that there is an instance ϕ_0 of ϕ with $M, w \models \neg\phi_0$.

Define the following extra axioms.

$$Total \qquad (E_X\phi \land E_X\psi) \to (\phi\langle Pref \rangle_X \psi \lor \psi\langle Pref \rangle_X \phi)$$

$$Trans \qquad (E_X\psi \land \phi\langle Pref \rangle_X \xi) \to (\phi\langle Pref \rangle_X \psi \lor \psi\langle Pref \rangle_X \xi)$$

6.2.12. THEOREM. *Total is a sound and complete axiom scheme for the set of reflexive frames with total preference relations.*

PROOF. Suppose that $M = (W, \Sigma, \{\succeq_X\}_{X \in \Sigma}, P, \pi)$ is a model such that all relations $\succeq_X$ are total and let $w \in W$. Assume that $M, w \models E_X\phi \land E_X\psi$. This means that there are two worlds $x, y \in W$ such that $M, x \models \phi$ and $M, y \models \psi$. From totality we know that either $x \succeq_X y$ or $y \succeq x$. In the first case, we have $M, w \models \phi\langle Pref \rangle_X \psi$ and in the second case $M, w \models \phi\langle Pref \rangle_X \psi$. Either way this leads to $M, w \models (\phi\langle Pref \rangle_X \psi \lor \psi\langle Pref \rangle_X \phi)$. Thus, $Total$ is sound for reflexive frames with total preference relations.

For the second part assume that $(W, \Sigma, \{\succeq_X\}_{X \in \Sigma})$ is a non-total reflexive frame. This means that for some X and $x, y \in W$ it is the case that neither $x \succeq_X y$ nor $y \succeq_X x$. Define $P = \{p, q\}$ and $\pi(x) = \{p\}, \pi(y) = \{q\}$ and for all remaining worlds $z : \pi(z) = \emptyset$. Let $M = (W, \Sigma, \{\succeq_X\}_{X \in \Sigma}, P, \pi)$. This model satisfies $M, x \models (E_X p \land E_X q)$. However, $M, w \models (p\langle Pref \rangle_X q \lor q\langle Pref \rangle_X p)$ does not hold. Thus, $M, x \models (E_X p \land E_X q) \to (p\langle Pref \rangle_X q \lor q\langle Pref \rangle_X p)$ does not hold. Therefore, $Total$ is complete for the class of total reflexive frames. $\square$

6.2.13. THEOREM. *Total are sound and complete axioms for the set of reflexive frames with strict-transitive, total preference relations.*

PROOF. Note that the transitivity axiom implies totality: Take $\phi = \xi$. Thus, if we add the axiom $Trans$ then $Total$ is derivable.

First we prove the soundness of $Trans$ on strict-transitive, total reflexive frames. Suppose that $M = (W, \Sigma, \{\succeq_X\}_{X \in \Sigma}, P, \pi)$ is a model such that all relations $\succeq_X$ are total and strict-transitive, and let $w \in W$. Assume that $M, w \models (E_X\psi \land \phi\langle Pref \rangle_X \xi)$. This means that there are three worlds $x, y, z \in W$ such that $M, x \models \phi$, $M, y \models \psi$, $M, z \models \xi$ and $x \succeq_X z$. In order to obtain a contradiction, assume that $M, w \models \neg(\phi\langle Pref \rangle_X \psi) \land \neg(\psi\langle Pref \rangle_X \xi)$. Since the

model is total, it follows now that $z \succ_X y$ and $y \succ_X x$. From strict-transitivity we obtain that $z \succ_X x$, and this implies that not $x \succeq_X z$. This is a contradiction, so the axiom $Trans$ does hold on this model. Since we have assumed M, w to be an arbitrary total, strict-transitive model, we may conclude that any such model satisfies the transitivity axiom.

For the second part, assume that $(W, \Sigma, \{\succeq_X\}_{X \in \Sigma})$ is a reflexive frame. We can assume that the preference relations are total, otherwise the derivable axiom $Total$ would not hold on some model based on this reflexive frame. Thus, we assume that for some X, the relation $\succeq_X$ is not a strict-transitive relation. This means that for some states $x, y, z \in W$ it is the case that $x \succ_X y$ and $y \succ_X z$, but not $x \succ_X z$. Using totality we can show that $z \succeq_X x$ must hold. Define $P = \{p, q, r\}$ and $\pi(x) = \{p\}, \pi(y) = \{q\}, \pi(z) = \{r\}$ and for all remaining worlds $w : \pi(w) = \emptyset$. Let $M = (W, \Sigma, \{\succeq_X\}_{X \in \Sigma}, P, \pi)$. This model satisfies $M, x \models (E_X q \wedge r\langle Pref \rangle_X p)$, and $M, x \not\models q\langle Pref \rangle_X p$ and also $M, x \not\models p\langle Pref \rangle_X q$. Hence the axiom $Trans$ is not sound on this model.

Therefore, $Trans$ characterizes the class of reflexive frames with strict-transitive, total preference relations. $\square$

Transitivity of a relation is a commonly assumed in modal logic, for instance for reasoning about time [12], and a standard axiom for transitivity is $\Diamond\Diamond\phi \rightarrow \Diamond\phi$. For preference logic we have used the more difficult notion of strict-transitivity. A reasonable question is therefore whether a simpler axiom is not available. Consider for instance the following axiom, which seems to capture transitivity.

$$((\phi\langle Pref \rangle_X \psi) \wedge (\psi\langle Pref \rangle_X \xi)) \rightarrow (\phi\langle Pref \rangle_X \xi)$$

This axiom is unfortunately not valid on preference models. A counter-example is the following instance, which does not hold on the model displayed in figure 6.2 on page 103.

$$((p\langle Pref \rangle_X (q \vee s)) \wedge ((q \vee s)\langle Pref \rangle_X r)) \rightarrow (p\langle Pref \rangle_X r)$$

Another reasonable question is whether there are strict-transitive models that are not transitive. An example of such a model is given in figure 6.4. In this figure the preference relation of a single agent has been indicated using arrows. The reflexive arrows have been omitted.

6.3   An Alternative Preference Logic

The proof given above, although valid, lacks a certain elegance: it does not make use of existing modal logic results. Therefore, in this paragraph we present a different axiomatisation. In order to do so the language of preference logic is slightly adapted.
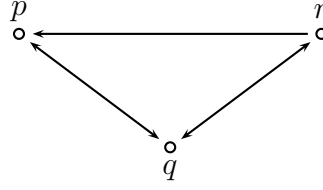
Figure 6.4: A strict-transitive, intransitive preference model

6.3.1. DEFINITION. Suppose the finite sets Σ and P are given, and let $X \in \Sigma$ and $p \in P$ be typical elements. Alternative preference logic $\mathcal{L}_P^2$ consists of formulas ϕ generated by the following rule.

$$\phi ::= p \mid \Diamond_X \phi \mid E\phi \mid \phi \to \phi \mid \bot$$

In this logic, the operator $\phi \langle Pref \rangle_X \psi$ can be defined as $E(\phi \wedge \Diamond \psi)$. Intuitively the meaning of $\Diamond_X \phi$ is that there is a state better for X than the current state, in which ϕ holds. The construct $E\phi$ means that somewhere in the model a state exists in which ϕ holds. For this logic we define the operators $\Box_X \phi = \neg \Diamond_X \neg \phi$ and $A\phi = \neg E \neg \phi$. These operators are thus duals for the two primitive operators.

This logic is interpreted in the following way. Let $M = (W, \Sigma, \succeq, P, \pi)$ be a preference model.

$$
\begin{array}{lll}
M, w \models \bot & & \text{never} \\
M, w \models p & \text{iff} & p \in \pi(w) \\
M, w \models E\phi & \text{iff} & \exists w' \in W : M, w' \models \phi \\
M, w \models \Diamond_X \phi & \text{iff} & \exists (w, w') \in \succeq_X : M, w' \models \phi \\
M, w \models \phi \to \psi & \text{iff} & \text{not } M, w \models \phi \text{ or } M, w \models \psi
\end{array}
$$

The operator $\Diamond_X$ is interpreted in the standard modal logic fashion using the relation $\succeq_X$. The operator $E\phi$ also has a standard interpretation, but based on the universal relation. All worlds are accessible to this operator.

6.3.2. FACT. Alternative preference logic $\mathcal{L}_P^2$ is strictly more expressive than preference logic $\mathcal{L}_P$.

PROOF. The operator $\phi \langle Pref \rangle_X \psi$ can be defined in alternative preference logic, but the operator $\Diamond_X \phi$ is not definable in the old preference logic. This can be shown by looking again at the models in figure 6.3. The formula $E(p \wedge \neg \Diamond q)$, which says that there is a p world for which no equal or better q world exists, is satisfied in one model but not in the other. Since these models satisfy the same $\mathcal{L}_P$ formulas, there cannot be a $\mathcal{L}_P$ formula that is equivalent to $E(p \wedge \neg \Diamond q)$. $\Box$

6.3.1 Proof System

One can define a proof system for this logic quite easily, because the interpretation of both operators is a standard modal logic interpretation. The proof system $\mathcal{S}_P^2$ for $\mathcal{L}_P^2$ has the following axioms:

$$
\begin{array}{ll}
 & \text{Axioms for } \diamond \\
\text{K} & \Box_X(\phi \to \psi) \to (\Box_X\phi \to \Box_X\psi) \\
4 & \diamond_X\diamond_X\phi \to \diamond_X\phi \\
\text{T} & \phi \to \diamond_X\phi \\
 & \text{Axioms for } E \\
\text{K}_E & A(\phi \to \psi) \to (A\phi \to A\psi) \\
4_E & EE\phi \to E\phi \\
\text{T}_E & \phi \to E\phi \\
\text{B}_E & \phi \to AE\phi \\
\text{Incl} & \diamond_X\phi \to E\phi \\
\text{Tot} & (E\phi \wedge E\psi) \to (E(\phi \wedge \diamond_X\psi) \vee E(\psi \wedge \diamond\phi))
\end{array}
$$

The reasoning rules for this logic are Modus Ponens, Necessitation for $\Box_X\phi$ and necessity for $A\phi$. These three rules are listed below.

$$
\frac{\phi}{\Box_X\phi} \qquad \frac{\phi}{A\phi} \qquad \frac{\phi \quad \phi \to \psi}{\psi}
$$

6.3.3. THEOREM. *The proof system $\mathcal{L}_P^2$ is sound and complete for the language $\mathcal{L}_P^2$ on minimal preference models with transitive preference relations.*

PROOF. It is clear that the axioms are sound and that the reasoning rules preserve validity: they are all standard axioms. The logic is complete with respect to the class of reflexive and transitive frames, see [12, p.417]. To see that the logic is complete with respect to the class of reflexive, transitive and linear frames, just notice that *Lin*, the axiom for linearity, is a Sahlqvist formula. Applying the algorithm of the Sahlqvist Correspondence Theorem [12, p.165] one sees that it corresponds to the first order expression of linearity: $\forall x, y(x \succeq y \vee y \succeq x)$. Furthermore, by the canonicity of Sahlqvist formulas, [12, p.322], we know that the canonical model for $\mathcal{L}_P^2$ is linear, and so that $\mathcal{L}_P^2$ is complete with respect to the class of reflexive, transitive and linear frames. $\qquad \Box$

6.4 Finite Tree Logic

In this section a logic is presented that can be used for reasoning about finite trees, such as game trees. Using this logic one can describe games trees. This logic is adapted from Blackburn and Viol [13], as is the completeness proof.

6.4.1. DEFINITION. Suppose the finite sets Σ and P are given, and let $X \in \Sigma$ and $p \in P$ be typical elements. Finite tree logic $\mathcal{L}_T$ consists of formulas ϕ generated by the following rule.

$$\phi ::= p \mid \phi\langle Pref\rangle_X\phi \mid \langle X\rangle\phi \mid \langle \Sigma\rangle\phi \mid \langle \Sigma^+\rangle\phi \mid \phi \to \phi \mid \bot$$

We define the following additional operators.

$$[X]\phi \stackrel{\text{def}}{=} \neg\langle X\rangle\neg\phi$$
$$[\Sigma]\phi \stackrel{\text{def}}{=} \neg\langle \Sigma\rangle\neg\phi$$
$$[\Sigma^+]\phi \stackrel{\text{def}}{=} \neg\langle \Sigma^+\rangle\neg\phi$$

6.4.2. DEFINITION. Let $\{R_X\}_{X\in\Sigma}$ be an indexed set of relations R_X, one for each agent $X \in \Sigma$. We define $R = \bigcup\{R_X\}_{X\in\Sigma}$ as the union of all relations in $\{R_X\}_{X\in\Sigma}$ and R^+ as the transitive closure of R.

6.4.3. DEFINITION. A *proto-model* M is a tuple $M = (W, \Sigma, \{R_X\}_{X\in\Sigma}, P, \pi,)$ such that W is a set of worlds, Σ a finite set of agents, for each $X \in \Sigma$ the relation $R_X \subseteq W \times W$ is a relation between worlds, P is a set of atomic propositions and $\pi : W \to 2^P$ assigns propositions to worlds.

6.4.4. DEFINITION. A *tree model* M is a proto-model $M = (W, \Sigma, \{R_X\}_{X\in\Sigma}, P, \pi)$ such that $R = \cup_X R_X$ defines a finite tree on some subset W^T of W.

The worlds $W \setminus W^T$ are not related by the relation R to any other worlds. These so-called 'loose worlds' are not reachable by actions, but are important in the preferences of agents.

We define the set of terminal nodes $Z(W, R)$ by $Z(W, R) = \{w \in W | \neg\exists w' : wRw'\}$. The reach function $\text{reach}(R, w)$ is the set of points reachable from w and can be defined by stating that $\text{reach}(R, w)$ is the smallest set S such that $w \in S$ and $x \in S \wedge xRy \Rightarrow y \in S$.

6.4.5. DEFINITION. A *preference tree model* M is a tuple $M = (W, \Sigma, \{R_X\}_{X\in\Sigma}, \{\succeq_X\}_{X\in\Sigma}, P, \pi)$ such that $(W, \Sigma, \{R_X\}_{X\in\Sigma}, P, \pi,)$ is a tree model and for each agent $X \in \Sigma$ the relation $(\succeq_X) \subseteq Z(W, R) \times Z(W, R)$ is a reflexive, strict-transitive, total relation.

$M, w \models \bot$		never
$M, w \models p$	iff	$p \in \pi(w)$
$M, w \models \phi \to \psi$	iff	$M, w \models \phi$ implies $M, w \models \psi$
$M, w \models \langle X\rangle\phi$	iff	$\exists w' : wR_Xw'$ such that $M, w' \models \phi$
$M, w \models \langle \Sigma\rangle\phi$	iff	$\exists X, w' : wR_Xw'$ such that $M, w' \models \phi$
$M, w \models \langle \Sigma^+\rangle\phi$	iff	$\exists w' : wR^+w'$ such that $M, w' \models \phi$
$M, w \models \phi\langle Pref\rangle_X\psi$	iff	$\exists(u, v) \in (\succeq_X)$ such that $M, u \models \phi$ and $M, v \models \psi$

As an example of how this logic can be used, consider the protocol displayed in figure 4.1 on page 4.1. This protocol can be described as a preference tree model M_1, if we indicate what the preferences of the agents are. One possible set of preferences has therefore been indicated in the next table.

Agent	Most to least preferred outcomes
A	x, y, z
B	y, x, z
C	z, x, y

It is assumed here that the utility that an agent attaches to a certain outcome, only depends on the atomic propositions that hold on a certain outcome. This is a reasonable assumption, since these atomic propositions are supposed to encode the relevant properties of each outcome. Thus, according to the table, A prefers both x outcomes above both y outcomes, and both y outcomes over both z outcomes. The first two formulas describe the structure of the game, and the last formula refers to the preferences. Let w_A be the root of model M_1, and let $\phi_\perp = [\Sigma]\perp$. The formula $\phi_\perp$ holds in terminal states.

$$M_1, w_0 \models \langle A \rangle (\langle B \rangle (\phi_\perp \wedge x) \wedge \langle B \rangle (\phi_\perp \wedge y) \wedge \langle B \rangle (\phi_\perp \wedge z))$$
$$M_1, w_0 \models \langle A \rangle (\langle C \rangle (\phi_\perp \wedge x) \wedge \langle C \rangle (\phi_\perp \wedge y) \wedge \langle C \rangle (\phi_\perp \wedge z))$$
$$M_1, w_0 \models (y \langle Pref \rangle_A z) \wedge (y \langle Pref \rangle_B \neg y) \wedge (z \langle Pref \rangle_C \neg z)$$

6.4.6. DEFINITION. The proof system $\mathcal{S}_T$ for finite tree logic consists of the rule Modus Ponens and the following axioms.

$$\begin{aligned}
\text{prop} =& \tau \text{ where } \tau \text{ is an instance of a propositional logic tautology} \\
K =& [X]\phi \to ([X](\phi \to \psi) \to [X]\psi) \\
XY =& \langle X \rangle \top \to \neg \langle Y \rangle \top \text{ where } X \neq Y \\
\text{all} =& \langle \Sigma \rangle \phi \leftrightarrow \vee_X \langle X \rangle \phi \\
\text{trans} =& [\Sigma^+]\phi \leftrightarrow [\Sigma](\phi \wedge [\Sigma^+]\phi) \\
L =& \langle \Sigma^+ \rangle \phi \to \langle \Sigma^+ \rangle (\phi \wedge \neg \langle \Sigma^+ \rangle \phi) \\
A_G =& (([\Sigma]\perp \wedge \phi) \vee \langle \Sigma^+ \rangle ([\Sigma]\perp \wedge \phi)) \to \phi \langle Pref \rangle_X \phi
\end{aligned}$$

The following formula is known in dynamic predicate logic as the induction principle.

$$\phi_I = ([\Sigma]\phi \wedge [\Sigma^+](\phi \to [\Sigma]\phi)) \to [\Sigma^+]\phi$$

This formula scheme is valid for finite tree logic. Here we show that this axiom can be derived from the other principles.

6.4.7. THEOREM. *The induction principle can be derived in finite tree logic:*

$$\mathcal{S}_T \vdash \phi_I$$

PROOF. Define α as the negation of ϕ_I: $\alpha = ([\Sigma]\phi \wedge [\Sigma^+](\phi \to \langle\Sigma\rangle\phi)) \wedge \neg[\Sigma^+]\phi$. The Löb axiom L can be formulated as

$$\mathcal{S}_T \vdash [\Sigma^+]([\Sigma^+]\phi \to \phi) \to [\Sigma^+]\phi$$

and axiom *trans* can be expressed equivalently as

$$\mathcal{S}_T \vdash [\Sigma^+]\phi \leftrightarrow ([\Sigma]\phi \wedge [\Sigma^+]\phi)$$

Using this formulation of the *trans* axiom, one can show that

$$\mathcal{S}_T \vdash \alpha \leftrightarrow ([\Sigma]\phi \wedge [\Sigma^+](\phi \to [\Sigma]\phi)) \wedge \neg[\Sigma](\phi \vee [\Sigma^+]\phi)$$

This can be simplified, because $[\Sigma]\phi$ appears twice in this formula.

$$\mathcal{S}_T \vdash \alpha \leftrightarrow ([\Sigma]\phi \wedge [\Sigma^+](\phi \to [\Sigma]\phi)) \wedge \neg[\Sigma][\Sigma^+]\phi$$

And one can bring the negation inside:

$$\mathcal{S}_T \vdash \alpha \leftrightarrow ([\Sigma]\phi \wedge [\Sigma^+](\phi \to [\Sigma]\phi)) \wedge \langle\Sigma\rangle\neg[\Sigma^+]\phi$$

Using *trans* again and Modus Ponens gives

$$\mathcal{S}_T \vdash \alpha \leftrightarrow ([\Sigma]\phi \wedge [\Sigma][\Sigma]\phi \wedge [\Sigma][\Sigma^+](\phi \to [\Sigma]\phi)) \wedge \langle\Sigma\rangle\neg[\Sigma^+]\phi$$

One can now derive three implications from this formula.

$$\mathcal{S}_T \vdash \alpha \to [\Sigma][\Sigma]\phi$$
$$\mathcal{S}_T \vdash \alpha \to [\Sigma][\Sigma^+](\phi \to \langle\Sigma\rangle\phi)$$
$$\mathcal{S}_T \vdash \alpha \to \langle\Sigma\rangle\neg[\Sigma^+]\phi$$

These three statements combine to the following conclusion

$$\mathcal{S}_T \vdash \alpha \to \langle\Sigma\rangle([\Sigma]\phi \wedge [\Sigma^+](\phi \to [\Sigma]\phi)) \wedge \neg[\Sigma^+]\phi)$$

This statement is the same as $\mathcal{S}_T \vdash \alpha \to \langle\Sigma\rangle\alpha$
Using contraposition this leads to $\mathcal{S}_T \vdash [\Sigma]\phi_I \to \phi_I$
From *trans* one can derive that $\mathcal{S}_T \vdash [\Sigma^+]\phi_I \to [\Sigma]\phi_I$
And these can be combined into $\mathcal{S}_T \vdash [\Sigma^+]\phi_I \to \phi_I$
The necessitation rule can be used to derive $\mathcal{S}_T \vdash [\Sigma^+]([\Sigma^+]\phi_I \to \phi_I)$
From L now follows $\mathcal{S}_T \vdash [\Sigma^+]\phi_I$
and thus (with Modus Ponens) $\mathcal{S}_T \vdash \phi_I$ □

This proof system is complete, and the proof for this fact is given below in several steps. It is a straightforward adaptation from Blackburn and Viol [13], except that we have translated notations where necessary. Blackburn and Viol introduced this logic for the finite binary trees that are used in linguistics as parse trees. Our adaptation shows that this logic can also be used for game trees.

6.4.8. DEFINITION. The closure $cl(\Phi)$ of a set of formulas Φ is the smallest set S such that the following hold.

- For all subformulas ψ of formulas $\phi \in \Phi$ we have $\psi \in S$

- If $\langle \Sigma^+ \rangle \phi \in S$ then $\langle \Sigma \rangle \phi \in S$

- If $\phi \in S$ and ϕ is not of the form $\neg \psi$ then $\neg \phi \in S$

If Φ is a finite set, then $cl(\Phi)$ is finite. From now on we assume that Φ is a finite set.

6.4.9. DEFINITION. A set A is a *maximally consistent subset* of some set S if there is a maximally consistent set C such that $A = C \cap S$. The atom set $At(\Phi)$ of a set of formulas Φ consists of all the maximal consistent subsets A of $cl(\Phi)$.

If Φ is finite then also the set of atoms is finite. Furthermore, for every finite set $At(\Phi)$ it is the case that $\mathcal{S}_P \vdash \bigvee_{A \in At(\Phi)} A$. One can think of $At(\Phi)$ as the set of subformulas whose truth we are interested in. A maximally consistent subset is thus that part of a maximally consistent set that we are interested in.

These definitions are used to define a proto-model. This proto-model is not a yet tree, but later on we order the elements of this model in such a way that a tree is formed.

6.4.10. DEFINITION. The proto-model C^Φ is defined as $(At(\Phi), \Sigma, \{R_X\}_{X \in \Sigma}, P, \pi)$ where Σ consists of all agents mentioned in Φ, P of all atomic propositions occurring in Φ, and $\pi(A) = \{p \in P | p \in A\}$. The relations R_X are defined by

$$AR_X B \iff \mathcal{S}_P \nvdash \neg(\bigwedge A \wedge \langle X \rangle(\bigwedge B))$$

It is not hard to show that for the operator $\langle \Sigma \rangle$ a similar condition holds. Using axioms 3, 4 we can show that $ARB \iff \mathcal{S}_P \nvdash \neg(\bigwedge A \wedge \langle \Sigma \rangle(\bigwedge B))$. We define $L_0(\Phi) = \{A \in At(\Phi) | [\Sigma]\bot \in A\}$. Let $S_i = \bigcup_{j \le i} L_i$. If $At(\Phi) \setminus S_i \ne \emptyset$ then L_{i+1} exists and we define

$$L_{i+1} = \{A \in At(\Phi) | A \notin S_i \text{ and } \bigwedge A \wedge [\Sigma^+](\bigvee_{B \in S_i} \bigwedge B) \text{is consistent}\}$$

6.4.11. LEMMA. *Suppose $A \in L_i$ and $\langle X \rangle \phi \in A$. There is a $B \in L_j$ such that $AR_X B$ and $j < i$.*

PROOF. Suppose that $\mathcal{S}_T \vdash \neg(\phi \wedge \langle \Sigma^+ \rangle \neg \phi)$. Using necessitation one can derive that $\mathcal{S}_T \vdash \neg\langle \Sigma^+ \rangle(\phi \wedge \langle \Sigma^+ \rangle \neg \phi)$. The Löb axiom can now be used in contra positive form (thus $\neg b \to \neg a$ instead of $a \to b$). This gives us $\mathcal{S}_T \vdash \neg \phi$. Another way to put

this is to say that if ϕ is consistent ($\mathcal{S}_T \nvdash \neg\phi$), then ($\phi \wedge \langle\Sigma^+\rangle\neg\phi$) is consistent: $\mathcal{S}_T \nvdash \neg(\phi \wedge \langle\Sigma^+\rangle\neg\phi)$.

Suppose now that $\mathcal{A}$ and $\mathcal{B}$ together form a partition of $At(\Phi)$: All atoms appear in $\mathcal{A}$ or $\mathcal{B}$ but not in both. One can show using the result of the previous paragraph that there is an $A_i \in \mathcal{A}$ such that $\bigwedge A_i \wedge [\Sigma^+](\bigvee_{B_j \in \mathcal{B}} \bigwedge B_j)$ is consistent. The proof for this observation is the following.

The disjunction $\bigvee_{A_i \in \mathcal{A}} \bigwedge A_i$ is consistent and therefore (observation 1) $(\bigvee_{A_i \in \mathcal{A}} \bigwedge A_i) \wedge [\Sigma^+] \neg \bigvee_{A_i \in \mathcal{A}} \bigwedge A_i$ is consistent. Since $\mathcal{A}$ and $\mathcal{B}$ form a partition of a set $At(\Phi)$ of *all* maximally consistent subsets, if no set in $\mathcal{A}$ is satisfied then a set of $\mathcal{B}$ must be satisfied: $\neg \bigvee_{A_i \in \mathcal{A}} \bigwedge A_i$ implies $\bigvee_{B_j \in \mathcal{B}} \bigwedge B_j$. This leads to the conclusion that $(\bigvee_{A_i \in \mathcal{A}} \bigwedge A_i) \wedge [\Sigma^+](\bigvee_{B_j \in \mathcal{B}} \bigwedge B_j)$ is consistent and thus for some $A_i \in \mathcal{A}$ we have that $A_i \wedge [\Sigma^+](\bigvee_{B_j \in \mathcal{B}} \bigwedge B_j)$ is consistent. In the construction of the proto-model, the set S_i and $At(\Phi) \setminus S_i$ form a partition of $At(\Phi)$. This proves the observation.

We can use the observation that we have just proven to see that if L_{i+1} exists, then it is non-empty. Every level i the set S_i thus gets bigger. Since $At(\Phi)$ is a finite set, eventually $S_i = At(\Phi)$ and every atom has been assigned to a certain level. For the proof of the lemma, suppose $A \in L_i$ and $\langle X \rangle \phi \in A$. It is clear that $i > 0$ and thus S_{i-1} exists. To obtain a contradiction, suppose that there is no atom $B \in S_{i-1}$ such that $AR_X B$, thus that for all B the formula $\bigwedge A \wedge \langle X \rangle \bigwedge B$ is inconsistent. This means that $\mathcal{S}_T \vdash \bigwedge A \wedge [X] \neg (\bigwedge_{B \in S_{i-1}} \bigwedge B)$. As a shortcut, define $\mathcal{B} = \bigwedge_{B \in S_{i-1}} \bigwedge B$. Since $A \in L_i$, we have that $\bigwedge A \wedge [\Sigma^+]\mathcal{B}$ is consistent. Since we can rewrite $[\Sigma]$ as $\bigwedge_Y [Y]$, we know that $\bigwedge A \wedge \bigwedge_Y [Y]\mathcal{B}$ is consistent and thus that $\bigwedge A \wedge [X]\mathcal{B}$ is consistent. This yields a contradiction. Thus, there must be an atom $B \in S_{i-1}$ such that $AR_X B$. $\square$

This concludes the preparations for the completeness proof, and we can prove the completeness theorem.

6.4.12. THEOREM. *The proof system $\mathcal{S}_T$ is complete.*

PROOF. We can construct a model for any formula ϕ. Take $\Phi = \{\phi\}$. Choose $A \in At(\Phi)$ such that $\phi \in A$. Let $W_0 = \{A\}$. We call a pair $(w, \langle X \rangle \phi)$ of $w \in W_n$ an *unsatisfied demand* if $\langle X \rangle \phi \in w$ but there is no $w' \in W_n$ such that $wR_k w'$ and $\phi \in w'$. As long as there are unsatisfied demands, pick one unsatisfied demand $(w, \langle X \rangle \phi)$ and take a world $w' \in L_m$ in a set L_m such that $wR_X w'$ and $\phi \in W$. The world w' must have a lower level than w: if $w \in L_i$ then $i < m$. Lemma 6.4.11 guarantees that such a w' exists. Define $W_{n+1} = W_n \cup \{w'\}$. The construction process must terminate after a certain number of steps, because we add worlds with strictly lower levels every time. Thus, for some sufficiently large m, there is a world W_m that has no unsatisfied demands.

Let the proto-model $C^\Phi = (At(\Phi), \Sigma, \{R_X\}_{X \in \Sigma}, P, \pi)$ be defined as before. Define the following model $M = (W_m, \Sigma, \{R'_X\}_{X \in \Sigma}, P, \pi')$ where $\{R'_X\}_{X \in \Sigma}, \pi'$ are

the restrictions of $\{R_X\}_{X \in \Sigma}, \pi$ to W_m. One can use induction on the structure of the formula ϕ to prove that $M, w_0 \models \phi$.

It remains for us to define a preference relation over W_m. The axiom A_G establishes that all worlds that are involved in the actions, are used in the preference relation. All other axioms only deal with preferences or only with actions. Since we have proven the preference logic to be complete, any consistent set of preference formulas can be satisfied by a model. This model contains exactly one state for each maximally consistent propositional formula, but we can multiply these states to obtain a bisimilar model in which we have a correspondence between the outcomes of the action relation and the preference worlds. If some world of the preference model cannot be mapped on the action model, a 'loose world' can be created that does not form part of the game tree: Our definition of a tree model allows for worlds that only play a role in the preference model. $\square$

6.5 Backward Induction: An Application

One of the reasons to study preference logics is that they can be used for logical investigations into game theoretic solution concepts. Since the subgame perfect equilibrium that is computed by the backward induction procedure is one of the best known solution concepts, it is a good guinea pig for testing the expressivity of game-related logics. For instance it is used for demonstrating the use of branching time temporal logic by Bonanno [14], and is also modelled by Harrenstein [45].

The subgame perfect equilibrium has been defined in defition 3.3.10 on page 45. Below we rephrase this definition in terms more suitable for logical purposes. We define a solution concept as a relation R_{sol} between the states of an extensive game. The relation contains the moves that are rational: These moves are the recommendations to the players made by the solution concept. Below we give the definition of the relation BI that describes the backward induction solution concept. Let $M = (W, \Sigma, \{R_X\}_{X \in \Sigma}, \{\succeq_X\}_{X \in \Sigma}, P, \pi)$ be a preference tree model. BI is a subset of the relation R. The outcomes of this relation are $Z(W, BI)$, and the reach of this relation from x is reach(BI, x). Intuitively, reach(BI, x) contains the nodes that one can reach using only moves that appear in BI. We define the reachable outcomes as $rz(W, R, w) = Z(W, R) \cap$ reach(R, w). The relation BI is defined in such a way that for all $w' \in W \setminus Z(W)$ it is the case that $rz(W, BI, w') \subseteq Z(W, R)$ and $\|rz(W, BI, w)\| = 1$. Thus, BI only recommends possible moves, and it recommends exactly one move in every node that is part of the tree. Finally, if for some state w it is the case that $wBIw'$ and wR_Xw'', then there is an $x \in rz(W, BI, w')$ such that there is a $y \in rz(W, BI, w'')$ such that $x \succeq_X y$. That is, w' is possibly at least as good as w''. This property makes it rational for agent X to choose the move w' above w''.

Not every game has a unique backward induction relation. Take for instance any game in which all agents value all outcomes equally. In such a game, any relation that chooses one move for each node is a backward induction relation.

If the preference relation is anti-symmetric however, then there is exactly one backward induction relation. Otherwise there might be more. Thus, if no agents values a pair of nodes equally, then there exists a unique subgame perfect equilibrium.

Below we extend finite tree logic with operators $\diamondsuit_{\text{sol}}\phi$ and $\diamondsuit^*_{\text{sol}}\phi$ that refer to recommended moves. This new logic is called *solved game logic* because it is interpreted over solved models.

6.5.1. DEFINITION. Suppose the finite sets Σ and P are given, and let $X \in \Sigma$ and $p \in P$ be typical elements. Solved game logic $\mathcal{L}_{sol}$ consists of formulas ϕ generated by the rule

$$\phi ::= p \mid \phi\langle \mathit{Pref}\rangle_X \phi \mid \langle X\rangle\phi \mid \langle\Sigma\rangle\phi \mid \langle\Sigma^+\rangle\phi \mid \phi \rightarrow \phi \mid \bot \mid \diamondsuit_{\text{sol}}\phi \mid \diamondsuit^*_{\text{sol}}\phi$$

Again one can assume the following derived operators.

$$\begin{aligned}
\Box_{\text{sol}}\phi &\qquad\qquad &\overset{\text{def}}{=}\ \neg\diamondsuit_{\text{sol}}\neg\phi \\
\Box^*_{\text{sol}}\phi &\qquad\qquad &\overset{\text{def}}{=}\ \neg\diamondsuit^*_{\text{sol}}\neg\phi
\end{aligned}$$

The construction $\diamondsuit_{\text{sol}}\phi$ means that one can use a recommended moved to reach a state where ϕ holds. Similarly, $\diamondsuit^*_{\text{sol}}\phi$ means that one can use only recommended moves to reach an *outcome* where ϕ holds.

This logic can be interpreted over solved models. These models are similar to preference tree models, but contain an extra relation that describes a solution to the game.

6.5.2. DEFINITION. A solved model M is a tuple $M = (W, \Sigma, \{R_X\}_{X\in\Sigma}, \{\succeq_X\}_{X\in\Sigma}, P, \pi, R_{sol})$ so that $(W, \Sigma, \{R_X\}_{X\in\Sigma}, \{\succeq_X\}_{X\in\Sigma}, P, \pi)$ is a preference tree model and $R_{sol} \subseteq R$ is a function: for each nonterminal state s there is a unique next state t such that $(s, t) \in R_{sol}$.

The relation R_{sol} indicates a game-theoretic solution to the game, because it contains recommended moves. A solved model is thus a structure that contains a game and recommendations to all players. Such a solved model can be tested for rationality, by checking whether the recommendations are consistent with the preferences of the agents. The interpretation of all operators is the same as for preference tree logic. The new operators $\diamondsuit_{\text{sol}}$ and $\diamondsuit^*_{\text{sol}}$ are interpreted in the following way.

$$\begin{aligned}
M, w &\models \diamondsuit_{\text{sol}}\phi &&\text{iff}\quad \exists w': wR_{sol}w' \text{ such that } M, w' \models \phi \\
M, w &\models \diamondsuit^*_{\text{sol}}\phi &&\text{iff}\quad \exists w' \in rz(W, R_{sol}, w) \text{ such that } M, w' \models \phi
\end{aligned}$$

Note that the operator $\diamondsuit^*_{\text{sol}}$ is interpreted such that it refers to outcome states only. This is done because only outcome states have preferences.

In order to be able to characterize backward induction, we have to put an extra constraint on the models that we consider. In the following definition we define when we call a preference relation functional (with respect to a given interpretation function π).

6.5.3. DEFINITION. Consider a preference relation $\succeq_X$ in the context of a model with some interpretation π. The relation $\succeq_X$ is *functional* if there is a function $f_X : 2^P \to \mathbb{R}$ such that for all worlds v, w we have that $v \succeq_X w$ implies $f_X(\pi(v)) \geq f_X(\pi(w))$.

We call preference models, tree models and solved models *functional* if all preference relations that occur in these models are functional (with respect to the interpretation function π of the model). If a model is function, it means that the atomic propositions encode all properties that agents use in their preferences. Thus, in functional models the atomic propositions are all that agents care about. The right model in figure 6.3 is functional, but the left model is not.

The following four formulas characterize the backward induction solution concept, at least for functional models. This means that all functional models on which all instances of the following four formulas hold, have the backward induction solution concept as their solution. One can thus say that the concept of backward induction is described by the logical formulas.

In fact the first four properties hold on any solved model, because of the constraints we have put on the solution relation. Therefore, one could say that the fourth property characterizes backward induction. The formula B_5 expresses what kind of reasoning can be used to motivate the backward induction solution concept, and can thus be used to explain this concept. This way of using logic to characterize solutions concepts gives more insight in the ideas or assumptions behind these concepts [30].

$B_1 = \langle \Sigma \rangle \top \to \diamondsuit_{\text{sol}} \top$

    If one can do a move, one can do a recommended move

$B_2 = \diamondsuit_{\text{sol}} \phi \to \langle \Sigma \rangle \phi$

    If moving to a ϕ state is recommended, it is possible

$B_3 = \diamondsuit^*_{\text{sol}} \phi \leftrightarrow (([\Sigma]\bot \wedge \phi) \vee \diamondsuit_{\text{sol}} \diamondsuit^*_{\text{sol}} \phi)$

    A recommended outcome can be reached in zero or more steps

$B_4 = \diamondsuit_{\text{sol}} \phi \to \square_{\text{sol}} \phi$

    Only one move is recommended

$B_5 = (\diamondsuit_{\text{sol}} \square^*_{\text{sol}} \phi \wedge \langle X \rangle \square^*_{\text{sol}} \psi) \to (\phi \langle Pref \rangle_X \psi)$

    X should not move against its preferences

6.5.4. THEOREM. *If a pointed solved model M, w has the backward induction relation as its solution then it satisfies the formulas B_1 to B_5.*

PROOF. Suppose that M, w is a solved model, and that the relation R_{sol} is indeed the backward induction relation BI. For each instance of formula B_i we show that it holds in this model.

- Suppose that $M, w \models \langle \Sigma \rangle \top$. This means that w is an internal node. From $\| rz(W, BI, w) \| = 1$ it follows that there must be a next BI move, and therefore $M, w \models \Diamond_{\text{sol}} \top$.

- Suppose that $M, w \models \Diamond_{\text{sol}} \phi$, which means there is a state w' such that $(w, w') \in BI$ and $M, w' \models \phi$. Since $BI \subseteq R$, it holds that $(w, w') \in R$ and thus $M, w \models \langle \Sigma \rangle \phi$.

  From $\| rz(W, BI, v) \| = 1$ for all nonterminal nodes v, it follows that there is only one state w' with $(w, w') \in BI$. Therefore, for any state w' with $(w, w') \in BI$ it is the case that $M, w' \models \phi$ and thus $M, w \models \Box_{\text{sol}} \phi$.

- Suppose that $M, w \models \Diamond_{\text{sol}}^* \phi$, which means there is an outcome $w' \in rz(W, BI, w)$ so that $M, w' \models \phi$. If w is an outcome itself, then $w' = w$ and $M, w \models ([\Sigma] \bot \wedge \phi)$. Otherwise there is a w'' with $(w, w'') \in BI$ and $M, w'' \models \Diamond_{\text{sol}}^* \phi$ and thus $M, w \models \Diamond_{\text{sol}} \Diamond_{\text{sol}}^* \phi$.

- Suppose that $M, w \models \Diamond_{\text{sol}} \phi$, which means that there is a state w' such that $(w, w') \in BI$ and $M, w' \models \phi$. In the definition of a solved model it is stated that this world w' is unique, and hence there are no other worlds w'' such that $(w, w'') \in BI$. Therefore, $M, w \models \Box_{\text{sol}} \phi$.

- Finally, assume $M, w \models (\Diamond_{\text{sol}} \Box_{\text{sol}}^* \phi \wedge \langle X \rangle \Box_{\text{sol}}^* \psi)$. Thus, there is a state w_1 with $(w, w_1) \in BI$ and $M, w_1 \models \Box_{\text{sol}}^* \phi$. There is also a state w_2 such that $(w, w_2) \in R_X$ with $M, w_2 \models \Box_{\text{sol}}^* \psi$. Hence $\exists v_1 \in rz(W, BI, w')$ with $M, v_1 \models \phi$ and $\exists v_2 \in rz(W, BI, w_2)$ with $M, v_2 \models \psi$. The definition of the BI relation now tells us that v_1 is possibly at least as good as v_2, and thus $v_1 \succeq_X v_2$, and therefore $M, w \models \phi \langle Pref \rangle_X \psi$.

$\square$

6.5.5. THEOREM. *If a pointed solved model M, w with a functional interpretation satisfies the formula B_4, then it has the backward induction relation as its solution.*

PROOF. Suppose that a solved model M, w has preferences that are functional: Outcomes w, w' with $\pi(w) = \pi(w')$ are equally preferred. Assume that all instances of B_4 hold on M, w.
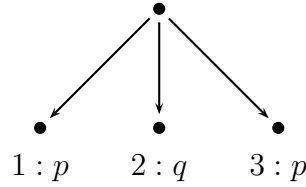
Figure 6.5: A non-functional tree model M

We have to show that the relation R_{sol} in M is a backward induction relation. Not surprisingly, this has to be shown inductively, starting with the final nodes. The base case is formed by nodes w that are outcomes, and for these nodes nothing has to be proven: The R_{sol} relation of an outcome is always the empty relation, because there are no moves to recommend.

Assume now that w is a nonterminal node such that $(w, w_1) \in R_{sol}$, and let $(w, w_2) \in R_X$, and that the recommended move (w, w_1) is in violation of the backward induction properties. This assumption is made to derive a contradiction. Thus, we assume that all outcomes in $rz(W, BI, w_1)$ are strictly worse than all outcomes in $rz(W, BI, w_2)$, according to agent X. Since the preferences are functional, it holds that $\pi(o_1) \neq \pi(o_2)$ for all $o_1 \in rz(W, BI, w_1)$ and $o_2 \in rz(W, BI, w_2)$. Take a formula ϕ_1 that describes an outcome o_1 exactly, and a formula ϕ_2 that describes an outcome o_2 exactly. Naturally $M, o_1 \models \neg\phi_2$ and vice versa. Since the preferences are functional, all ϕ_1 states are strictly less preferred than all ϕ_2 states, and thus $M, w \models \neg(\phi_1 \langle Pref \rangle_X \phi_2)$. Since $W, w \models (\diamondsuit_{\mathrm{sol}} \square^*_{\mathrm{sol}} \phi_1 \wedge \langle X \rangle \square^*_{\mathrm{sol}} \phi_2)$ and B_4 is supposed to hold, this is a contradiction. Thus, it is not possible that the move (w, w_1) does not satisfy the backward induction properties. Using induction we can now conclude that all moves recommended by R_{sol} are moves recommended by backward induction, and hence that $R_{sol} = BI$. □

It is necessary to restrict the relations in the model to be functional. There are non-functional solved models that satisfy the given formulas, but whose solution relation is not a backward induction solution. An example of such a non-functional model M is displayed in figure 6.5. Assume that M is a solved model where outcome 3 is preferred over the other two outcomes, and that the solution relation in M recommends outcome 1. This solution is not the backward induction relation, since outcome 3 is clearly better. However it satisfies B_1 to B_5.

In order to show that the logical approach of this chapter is more expressive than the use of the logic EFL, consider again the first solution in figure 4.1 on
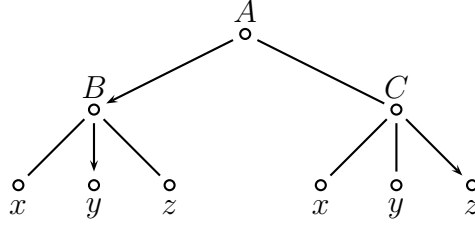
Figure 6.6: Solved model M_2

page 58 to the example voting problem of chapter 4. On page 115 we defined a preference tree model by combining the protocol of figure 4.1 with a given set of preferences. We extend this model to a solved model M_2 by adding suggested moves. The solution relation added is that A chooses w_B, B chooses y and C chooses z. This solution relation is indicated in figure 6.6.

Call the root node w_A, B's decision node w_B and C's decision node w_C. Assume that the following preference formulas hold.

$$M, w_A \models x[Pref]_A y \wedge y[Pref]_A z$$
$$M, w_A \models y[Pref]_B (x \vee z)$$
$$M, w_A \models z[Pref]_C (x \vee y)$$

If the solution of this solved model is the backward induction relation, then it follows from $y[Pref]_B(x \vee z)$ that $M, w_B \models \Diamond_{\text{sol}} y$. Similarly, $M, w_C \models \Diamond_{\text{sol}} z$. Since one can derive from the first line that $M, w_A \models \neg(z\langle Pref\rangle_A y)$, it follows from B_5 that not $M, w_A \models \Diamond_{\text{sol}} \square^*_{\text{sol}} z$ and hence $M, w_A \models \square^*_{\text{sol}} y$. Therefore, the indicated solution relation is indeed the backward induction solution.

After considering more different preference assumptions, one can show that agents B and C have the best 'chance'(assuming all different preference relations are equally likely) to get the options they prefer most. Agent A on the other hand is least likely to get the option it prefers least. Thus, using preference logic one can give advice to agents which role they should take in this voting problem, which was not possible based on EFL. Agents that strongly prefer one option should take the role of agent B or C, whereas agents that strongly dislike one of the options are better of as agent A.

6.6 Conclusion

In this chapter, the focus has not been on model checking but on proof theory. A language $\mathcal{L}_P$ for reasoning has been presented and a complete proof system for this language is given. We have also developed a more modern language for

preferences $\mathcal{L}_P^2$, and shown that this language is more expressive than $\mathcal{L}_P$, while having a proof system with a simpler completeness proof.

In section 6.4, we have used preference logic in combination with a logic for reasoning about finite trees. This allows one to reason about extensive games. Again a proof system for this logic has been defined, using techniques that are standard in modal logic. The logic can be used for characterising the backward induction solution concept. An analysis of the example voting problem of chapter 4 shows that this logic is more expressive than EFL.

The most important conclusion that can be drawn from this chapter is that modal logic, because so much is known about it, is a suitable tool for modeling game-theoretic reasoning. Possible future work would be to analyse other solution concepts as well, such as iteration of dominant strategies, and the (not subgame-perfect) Nash equilibrium. A more challenging project would be to analyse imperfect information games. One such attempt already exists in the form of ATEL [102], also discussed in section 7.5. Recent ATEL research has shown that reasoning about imperfect information games is a challenging problem [3, 54, 55, 109]. It would be interesting to compare the ATEL approach, with the more direct modal logic approach used in this chapter.

Chapter 7
Knowledge Condition Games

7.1 Introduction

In the previous chapters, we have looked at protocols that can be modelled as perfect information game forms. In such protocols all agents are aware of all previous events, and therefore no aspects of the current situation are unknown. In this chapter the focus is on protocols that can be modelled as imperfect information game forms. Such protocols are interesting for at least two reasons:

- The imperfect information of agents has consequences of what strategies they can use. Finding optimal strategies for imperfect information games is therefore a more complex problem than for perfect information games.

- The knowledge that agents do and do not have of the current situation can be used in the definition of the game. Having certain knowledge can be the goal of an agent or a coalition of agents.

In order to study these two aspects of multi-agent protocols, we define a new class of games, called *knowledge condition games*. In a knowledge condition game, two coalitions of agents enact a protocol. One coalition strives to reach a certain knowledge situation, and the other coalition tries to prevent the first coalition from reaching its goal. In other words, one coalition "wins" if it is able to force a certain condition to hold in the world, where this condition relates to the knowledge (and absence of knowledge) of the agents in the game. Formally, we specify the goal situation (i.e., the condition that the agents strive to achieve) using epistemic logic, and protocols are modeled as interpreted game forms with imperfect information.

After defining these games and illustrate using various examples, we focus on the computational complexity of determining who wins a knowledge condition game under various assumptions. Specifically the following questions are answered.

- Whether the presence of opponents make it harder to determine the existence of a winning strategy in a knowledge condition game

- Whether winner determination is harder if one assumes that strategies are known to agents

- Whether one can identify variants of knowledge condition games in which winner determination is tractable

The complexity results also allow one to see whether reasoning about knowledge in strategic situations is indeed a complex problem. The fact that we have collected in this chapter many different complexity results shows that this is indeed the case.

The structure of this chapter is as follows. In the next section, section 7.2 we have collected all necessary definitions. Section 7.3 provides four examples of knowledge condition games. The first example shows how knowledge properties are important in a voting protocol. The second example involves a more playful quiz problem. It shows how signaling can enter into reasoning about knowledge. The third example, the Russian Cards problem, is larger than the previous two and hence the corresponding knowledge condition game is not easily solved by hand. In the last example the use of a multi-step strategy for a coalition of three agents is demonstrated. Section 7.4 presents four results relating to the complexity of knowledge condition games. We prove the complexity of deciding a knowledge condition game in which strategies are known, first for the restricted case without opponents, then with opponents. We then do the same for knowledge condition games in which strategies are unknown. Section 7.5 discusses some related work, and section 7.6 presents some conclusions.

7.2   Defining Knowledge Condition Games

In this section we define how one can create a knowledge condition game G from an interpreted game form F. This is done in two definitions at the end of the section. Before these definitions, we define epistemic logic, game forms, strategies and updates, which are all needed in order to define knowledge condition games.

The notion of an interpreted game form has been introduced in chapter 3. In that chapter, only interpreted game forms with perfect information have been defined. Definitions for imperfect information game forms are given below.

7.2.1. DEFINITION. An *interpreted game form* F is a tuple

$$F = (\Sigma, H, turn, \sim, P, \pi),$$

where:
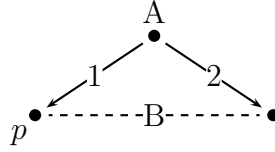
Figure 7.1: Interpreted game form F_0

- Σ is a finite set of agents;

- H is a non-empty, prefix-closed set of finite sequences;

- *turn* is a function $turn : H \setminus Z(H) \to \Sigma$;

- for each $X \in \Sigma$ the relation $\sim_X \subseteq H \times H$ is an equivalence relation between sequences;

- P is a finite set of atomic propositions; and

- $\pi : Z(H) \to 2^P$ returns the true atomic propositions of any terminal history.

These components must satisfy the following condition:

if $turn(h) = X$ and $h' \sim_X h$ then also $turn(h') = X$ and $A(H, h) = A(H, h')$.

(This definition is adapted from Osborne and Rubinstein [79, p.200]). We have extended their notion of information sets such that agents also have information when they are not in charge, which is a not uncommon for logical purposes [15, 98].

Atomic propositions can be used to refer to certain terminal histories, for instance to histories where an agent achieves a certain goal. The idea of annotating end states or terminal histories with logical propositions has been used before by Harrenstein et al [45] and the author [113]. Approaches based on temporal logic [101, 102] often annotate all nodes of the model with propositions, so that formulas can be interpreted anywhere in the model.

An example interpreted game form F_0 is depicted in figure 7.1. In this example, agent A can make a choice from two alternatives (numbered 1 and 2), one of which satisfies p. After this choice, A can distinguish these situations, but B cannot.

For every interpreted game form F we can calculate an epistemic model $M = m(F)$ representing the knowledge in the end states of F. We do this by taking all the terminal histories of F as the set of states of M. The states of the model M are all outcomes of the interpreted game form F, and two outcomes are related in M iff they are related in F.

7.2.2. DEFINITION. Let $F = (\Sigma, H, turn, \sim, P, \pi)$ be an interpreted game form. The *end situation model $m(F)$* is defined as $m(F) = (\Sigma, Z(H), \sim', P, \pi)$ where for each agent X, $\sim'_X$ is the restriction of $\sim_X$ to $Z(H) \times Z(H)$.

The transformation m is used to express when an interpreted game form F makes a formula ϕ true. The function m only uses the epistemic relation between end states. The relations between other states are however used in the definition of uniform strategies.

7.2.1   Strategies

Strategies are an important part of every game. Informally a strategy σ_Γ is a function that tells all agents in coalition Γ what to do next in the histories they control. We use nondeterministic strategies for our agents. These strategies have been defined in definition 3.3.6 on page 43. Such a strategy does not return a unique option that the agent should take, but it returns a set of options, with the intention that the agent should randomly select an element of this set. Our strategies are thus akin to the randomized or 'mixed' strategies, or more correctly the *behavioural strategies*, of game theory [79, p.212], except that we do not consider probabilities of making particular choices. Since we deal with imperfect information games, only *uniform strategies*, as defined in definition 3.3.13, are considered.

For the example interpreted game form F_0 there are three different strategies $\sigma_{\{A\}}$ for agent A. The strategy can either tell the agent to take the first option, or it can prescribe the second option, or the strategy can express that the agent should randomly choose between both options. Formally, these possibilities are defined by respectively $\sigma^1_{\{A\}}(\epsilon) = \{1\}$, $\sigma^2_{\{A\}}(\epsilon) = \{2\}$ and $\sigma^3_{\{A\}}(\epsilon) = \{1, 2\}$.

For any strategy σ_Γ for an interpreted game form F we can consider a restricted interpreted game form F' in which the agents $X \in \Gamma$ only choose options that are part of the strategy. The agents $Y \notin \Gamma$ can still do whatever they want in F'. Such a restricted interpreted game form models the situation in which coalition Γ is committed to the given strategy. The restricted model F' is computed by an *update function* $F' = u(F, \sigma_\Gamma)$.

7.2.3. DEFINITION. Let $F = (\Sigma, H, turn, \sim, P, \pi)$ be an interpreted game form. The update function u is defined by

$$u(F, \sigma_\Gamma) = (\Sigma, H', turn', \sim', P, \pi'),$$

where:

- H' is the smallest subset of H such that $\epsilon \in H'$ and for each $h \in H'$ and $a \in A(H, h)$: if $turn(h) \notin \Gamma$ or $a \in \sigma_\Gamma(h)$ then $ha \in H'$;

- $\sim'$ is such that for all X: $\sim'_X = \sim_X \cap (H' \times H')$; and

- $turn'$ and π' are the same as $turn$ and π, but with their domain restricted to H'.

An update of the example F_0 with strategy $\sigma^3_{\{A\}}$ does not change anything: $u(F_0, \sigma^3_{\{A\}}) = F_0$. An update with $\sigma^1_{\{A\}}$ returns a model F_1 with only two histories: ϵ and 1. This means that the epistemic model of F_1 only has one state in which p holds. Thus, using the interpretation of epistemic logic (defined on page 16), it holds that $m(u(F_0, \sigma^1_{\{A\}})), 1 \models K_B p$.

7.2.2 Strategic Games

The function $G = kcg(F, \Gamma, \Xi, \phi)$ defines a knowledge condition game in which Γ wishes to achieve ϕ, while Ξ hopes to prevent it. The game G is not an extensive game, but a game in normal or strategic form. It is not possible to consider G as an extensive game, because whether the knowledge condition holds is not a local property of each end state.

We are only interested in two-player, constant-sum, win-loss games, and in these games only two payoff vectors are possible: $(1, 0)$ which is best for the first player, and $(0, 1)$ which is best for the second player. In these games one can say that an agent can win if it has a strategy that guarantees that the agents gets utility 1. If the first player can win we write $w(G) = 1$.

7.2.4. DEFINITION. Let $G = (\{A, B\}, \{S_A, S_B\}, \mathfrak{U})$ be a two player constant-sum win-loss game. The winner function w is defined by

$$w(G) = 1 \Leftrightarrow \exists \sigma_A \in S_A \; \forall \sigma_B \in S_B : \mathfrak{U}(\sigma_A, \sigma_B) = (1, 0)$$

7.2.3 Knowledge Condition Games

A *knowledge condition game* is a two-player, constant-sum, win-loss strategic game. It is played between two coalitions Γ and Ξ of agents. These sets must be disjoint, but not every agent has to be in one of those sets. If an agent $X \in \Sigma$ is not in $\Gamma \cup \Xi$ then this agent is said to be *neutral*. The agents in Γ are called *proponents*, and the agents in Ξ *opponents*. To define a knowledge condition game, we must give an interpreted game form F and an epistemic logic formula ϕ: The proponents try to make this formula true on F, and the opponents try to make it false on F. Formally:

7.2.5. DEFINITION. Let $F = (\Sigma, H, \mathit{turn}, \sim, P, \pi)$ be an interpreted game form, $\Gamma, \Xi \subseteq \Sigma$ disjoint sets of agents and $\phi \in \mathcal{L}_K$ a knowledge formula. Define $kcg(F, \Gamma, \Xi, \phi) = (\{\Gamma, \Xi\}, \{S^\Gamma_n, S^\Xi_n\}, \mathfrak{U})$ where S^Γ_n, S^Ξ_n contain all nondeterministic strategies of Γ and Ξ in F respectively, and

$$\mathfrak{U}(\sigma_\Gamma, \sigma_\Xi) = \begin{cases} (1, 0) & \text{iff } \forall w \in W : (\Sigma, W, \sim, P, \pi'), w \models \phi \\ (0, 1) & \text{otherwise} \end{cases}$$

where $(\Sigma, W, \sim, P, \pi') = m(u(u(F, \sigma_\Gamma), \sigma_\Xi))$.

Take the example game form F_0 and take $\phi_0 = K_B p$. For the game $G_0 = kcg(F_0, \{A\}, \emptyset, \phi_0)$ we can compute a payoff matrix. As calculated before, $\{A\}$ has three strategies. The empty coalition only has the unique empty function $f_\emptyset$ as a strategy.

	$\sigma^1_{\{A\}}$	$\sigma^2_{\{A\}}$	$\sigma^3_{\{A\}}$
$f_\emptyset$	(1,0)	(0,1)	(0,1)

We see that for this game, $\{A\}$ has a winning strategy (namely $\sigma^1_{\{A\}}$). Therefore, $w(kcg(F_0, \{A\}, \emptyset, \phi_0)) = 1$. In the above definition, we use the updated model $m(u(u(F, \sigma_\Gamma), \sigma_\Xi))$ as a model for what all agents know. We have thus implicitly assumed that everybody commonly knows which strategies are used by Γ and Ξ, r if one assumes that strategies are somehow visible to other agents. As we have argued on page 81 in the case of perfect information games, this is a reasonable assumption. It makes sense if one considers strategies as well-known conventions. Also if a game is played by computer programs that are open for inspection, this is a reasonable assumption. Finally, one can argue that assuming that no details can be kept secret is a very conservative and thus sound assumption if one tries to prove the correctness of security protocols. In some circumstances, however, one might not want to make this assumption. Therefore, we present below a variant kcg' of knowledge condition games in which the knowledge formulas ϕ is evaluated the original model $m(F)$. The strategies are used to determine the reachable states w and the proponents win if in all these states w, it holds that $m(F), w \models \phi$.

7.2.6. Definition. Let $F = (\Sigma, H, turn, \sim, P, \pi)$ be an interpreted game form, $\Gamma, \Xi \subseteq \Sigma$ disjoint sets of agents and $\phi \in \mathcal{L}_K$ a knowledge formula. Define $kcg'(F, \Gamma, \Xi, \phi) = (\{\Gamma, \Xi\}, \{S_\Gamma, S_\Xi\}, \mathfrak{U}')$ where S_Γ and S_Ξ contain all strategies of Γ, Ξ in F respectively, and

$$\mathfrak{U}'(\sigma_\Gamma, \sigma_\Xi) = \begin{cases} (1,0) & \text{iff } \forall w \in W : \ m(F), w \models \phi \\ (0,1) & \text{otherwise} \end{cases}$$

where W is defined by $(\Sigma, W, \sim, P, \pi) = m(u(u(F, \sigma_\Gamma), \sigma_\Xi))$.

The difference between kcg and kcg' lies in their respective utility function. The function $\mathfrak{U}$ evaluates the formula ϕ in the model $m(u(u(F, \sigma_\Gamma), \sigma_\Xi))$, in all states. The function $\mathfrak{U}'$ evaluates the formula ϕ in the model $m(F)$, thus in the model before updates. This difference reflects the idea that in kcg, strategies are commonly known, whereas in kcg' they are not known. The function $\mathfrak{U}'$ only evaluates the formula ϕ in states w that occur in $m(u(u(F, \sigma_\Gamma), \sigma_\Xi))$. The idea here is that the truth of ϕ only matters in states that are actually reached, and which states are reachable depends on the strategies chosen.

7.3 Examples

7.3.1 Anonymous Voting

A voting protocol can be used when a group of agents has to make a joint decision on a certain issue. A common protocol is *majority voting*: Each agent can vote for an option and the option that gets the most votes is the outcome of the protocol. In the example interpreted game form $F_V = (\Sigma, H, turn, \sim, P, \pi)$, three agents A, B and C use majority voting to decide whether a plan $\mathcal{P}$ should be accepted or not. Thus $\Sigma = \{A, B, c\}$ and $P = \{a, b, c, p\}$. Each agent has to choose from two actions: support the plan (s), or reject it (r). They vote in alphabetical order, so first A chooses between action s and r, then B (without knowing A's choice) chooses either s or r and finally C does the same, unaware of what A and B did. This protocol thus has eight terminal histories. The proposition p indicates whether $\mathcal{P}$ is accepted and p holds if at least two agents choose s. Furthermore the proposition a holds if A chooses s, b if B chooses s and the same for C with c. The interpretation function is thus $\pi(sss) = \{a, b, c, p\}, \pi(ssr) = \{a, b, p\} \dots \pi(rrr) = \emptyset$. We assume that $s \not\sim_X s'$ if s and s' differ in the evaluation of the outcome p, or if the vote of X differs in s from that in s'.

The following game results hold.

$$w(kcg(F_V, \{A, B\}, \{C\}, p)) = 1$$
$$w(kcg(F_V, \{A, B\}, \{C\}, K_B c \vee K_B \neg c)) = 1$$
$$w(kcg(F_V, \{B\}, \{C\}, K_B c \vee K_B \neg c)) = 0$$

A and B together can ensure that p is true, by both voting s. They can also vote differently, so that a and $\neg b$ result. In this case the outcome will solely depend on C's choice. They thus learn what C voted. Agent B cannot learn what C did on its own.

One example, described by Schneier [89, p. 133], is a voting protocol where B would have the option of copying A's (encrypted) vote. In that case one might get

$$w(kcg(F_V', \{B\}, \{A, C\}, K_B a \vee K_B \neg a)) = 1$$

This is an unwanted property and thus a 'bug' in the protocol. It is necessary to reason about knowledge to express this bug, so a standard game-theoretic analysis might not have revealed this shortcoming.

7.3.2 Fifty-Fifty Problem

Consider the following scenario:

> In a TV quiz show the quiz master asks a candidate the following question: Which day of the week comes directly after Tuesday? Is it a) Monday, b) Wednesday, c) Friday or d) Saturday. The candidate
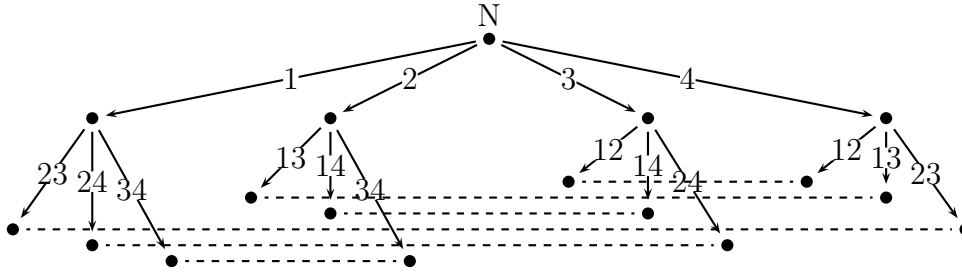
Figure 7.2: The fifty-fifty problem F_Q

has no clue whatsoever about the days of the week, and replies: 'I am not sure. Can I do fifty-fifty?'. The quiz master has to remove two options that are not the answer, so he says: 'The answer is not Monday and neither Friday'. Does the candidate now know the answer?

This situation frequently occurs on television in several European countries in the 'Millionaire show'. One can also consider this situation to be a metaphor for a multi-agent information exchange situation. One can model this in an interpreted game form $F_Q = (\Sigma, H, turn, \sim, P, \pi)$. The set of agents is $\Sigma = \{N, Q, C\}$, involving an agent N (Nature) that determines what the right answer is, a quiz master Q that eliminates two answers, and a candidate C. This interpreted game form is depicted in figure 7.2. First Nature selects one of the answers to be the right answer: It can choose from the actions $1, 2, 3$ and 4. The quiz master, who knows the right answer, can then select an action ij that indicates that the two options i and j are eliminated; i and j must be different from the right answer. The terminal histories are thus all histories (k, ij). For such histories, $(k, ij) \sim_C (k', i'j')$ if the same options are eliminated: $ij = i'j'$. The set of atomic propositions is $P = \{a_i \mid 1 \leq i \leq 4\} \cup \{e_i \mid 1 \leq i \leq 4\}$, and each terminal history is interpreted in the following way: $\pi((k, ij)) = \{a_k, e_i, e_j\}$. The question is whether the candidate knows the answer at the end of the protocol. This is expressed by $\psi = K_C a_1 \vee K_C a_2 \vee K_C a_3 \vee K_C a_4$. The following table lists several properties of this situation.

Nature may favour the candidate:	$w(kcg(F_Q, \{N\}, \emptyset, \psi)) = 1$
Nature may not favour the candidate:	$w(kcg(F_Q, \{N\}, \emptyset, \neg\psi)) = 1$
The quiz master can help the candidate:	$w(kcg(F_Q, \{Q\}, \emptyset, \psi)) = 1$

We thus see that whether the candidate knows the answer depends on Nature and on the quiz master Q. If Nature uses a deterministic strategy, in which for instance a_1 always holds, then the candidate knows that this is the right answer. However, if Nature uses the nondeterministic strategy in which each answer could be the right answer, the candidate will not know the answer.
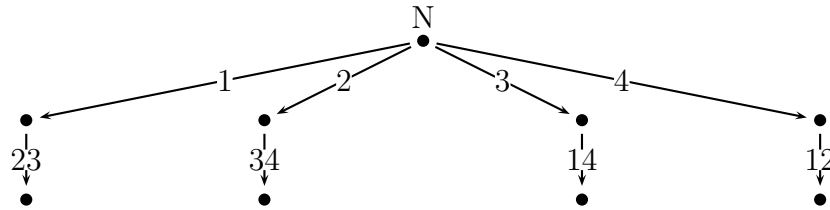
Figure 7.3: The updated interpreted game form $u(F_Q, \sigma_{\{Q\}})$

The situation becomes more interesting if the quiz master gets involved. In this game the quiz master has the ability to signal the right answer to the candidate. Consider, for example, strategy $\sigma_{\{Q\}}$, defined as follows.

$$\sigma_{\{Q\}}(1) = \{23\}$$
$$\sigma_{\{Q\}}(2) = \{34\}$$
$$\sigma_{\{Q\}}(3) = \{14\}$$
$$\sigma_{\{Q\}}(4) = \{12\}$$

This strategy tells the candidate exactly what the right answer is: The answer directly before the two eliminated options (assuming 4 comes before 1). The updated model $u(F_Q, \sigma_{\{Q\}})$ is given in figure 7.3. This strategy acts as a code between the candidate and the quiz master. It is the strategy that proves that $w(kcg(F_Q, \{Q\}, \emptyset, q)) = 1$. A practical conclusion one can draw is that one should not bet on this quiz if one does not know what the interests of the quiz master are.

This example also demonstrates why we prefer to assume that strategies are commonly known. If one would have used the alternative definition kcg', in which agents do not know what strategies are used, then one can obtain the following results.

Nature cannot favour the candidate: $\qquad w(kcg(F_Q, \{N\}, \emptyset, \psi)) = 0$
The quiz master cannot help the candidate: $\quad w(kcg(F_Q, \{Q\}, \emptyset, \psi)) = 0$

These results are counter-intuitive, since signaling in games is a phenomenon that does occur in practice. When proving the security of a protocol, it is a good principle to make the weakest assumptions possible. At first sight, it seems that assuming that strategies are not known is the weakest possible assumption. However, in the case of proving ignorance, first sight can be misleading. It is harder to prove that the candidate does not know the answer when he or she knows all strategies that are used, than it is to prove ignorance when he or she does not know the strategies. Therefore, the weakest and safest assumption is to assume that he does know the strategies. This shows that it is best to use the definition of kcg rather than the alternative kcg' for these ignorance proofs. This motivates the choice to make kcg the default and call kcg' the alternative.

7.3.3   Russian Cards Problem

The Russian cards problem first appeared in the Russian mathematics olympiad in 2000 [106]. It has subsequently been picked up by logicians as an example of a information based security. An informal description, taken from [106], is the following.

> From a pack of seven known cards two players each draw three cards and a third player gets the remaining card. How can the players with three cards openly (publicly) inform each other about their cards, without the third player learning from any of their cards who holds it?

Following the analysis by Van Ditmarsch [106] we call the agents A, B and C and the cards $0, 1, 2, 3, 4, 5$ and 6. The interesting thing about this problem is that certain solutions to this problem appear sound, but are not sound. A solution to this problem is a joint strategy for A and B that prescribes what they should communicate to each other. We are mostly interested in direct exchanges: statements by A such that B directly learns all of A's cards. Agent B can respond by telling A which card C has. Assume for the moment that the actual deal of cards is that A holds $0, 1$ and 2, that B holds $3, 4$ and 5 and that C holds 6. Instead of reasoning about complete strategies for A and B,we settle for identifying which statements by A for this situation van can be part of a strategy. Here are some solution attempts. Imagine that the next public statements are made by agent A so that all agents can hear it.

I have 012 or 345: This statement is true and when taken literally it does not tell C anything about a single card. Unfortunately A can imagine that C holds card 5. in which case this statement would reveal A's cards to C. So A cannot make this statement safely.

I have 012 or I have none of these cards: A knows that C cannot pinpoint any card after learning this statement. Unfortunately C can reason like this: Suppose A has 345. In that case she cannot exclude that I hold card 2. If I had card 2 I would know that B holds card 0 and 1. Alice would never allow me to learn that. Contradiction. By this line of reasoning she can eliminate all possibilities except 012.

I have 012, 034, 056, 135, 146 or 236 This is an example of a statement A can make.

In this section we define a knowledge condition game corresponding to the Russian Cards problem. This approach can be compared to previous attempts using epistemic model checking and dynamic epistemic logic.

We introduce a set of agents $\Sigma = \{N, A, B, C\}$ and a set of cards $D = \{0, 1, 2, 3, 4, 5, 6\}$. A nice way to use propositions for this problem is to use a_i to indicate that agent A holds card i, and similarly for b_i and c_i. The set of all deals of cards allowed in this problem is

$$\Delta = \{abc|def|g \,|\{a, b, c, d, e, f, g\} = \{0, 1, 2, 3, 4, 5, 6\} \wedge a < b < c \wedge d < e < f\}$$

For a deal $abc|def|g$, the cards of A are a, b, c, agent B owns d, e, f and C holds card f. We can thus say that in a situation with deal $012|345|6$, the following formula holds: $a_0 \wedge a_1 \wedge a_2 \wedge b_3 \wedge b_4 \wedge b_5 \wedge c_6$.

An interpreted game form

$$F_{RC} = (\Sigma, H, \mathit{turn}, \sim, P, \pi)$$

can now be defined as follows. As indicated above, we take $\Sigma = \{N, A, B, C\}$. The set H is described by

$$H = \{\epsilon, \delta, (\delta, x)|\delta \in \Delta, x \in \{0, 1, 2, 3, 4, 5\}\}$$

The variable x is used to indicate a symbol that A communicates publicly to B and C. In the examples above, this symbol was a sentence such as "I have 012 or 345". In a knowledge condition game, it is sufficient to use abstract symbols. The agents B and C know when A uses a signal x, and thus the meaning of x. We have chosen to allow only six different signals. It is not obvious beforehand whether six signals is enough, but we will answer this question later.

The function turn is defined such that $\mathit{turn}(\epsilon) = N$ and $\mathit{turn}(\delta) = A$ for all $\delta \in \Delta$. The equivalence relations $\sim$ are defined such that agents know their on cards, and the action selected by A. Thus the following definition applies, where X can stand for any agent, h for any element of H, and x for any signal.

$h \not\sim_X h'$	if $\|h\| \neq \|h'\|$
$abc|def|g \not\sim_A a'b'c'|d'e'f'|g'$	iff $(a, b, c) \neq (a', b', c')$
$abc|def|g \not\sim_B a'b'c'|d'e'f'|g'$	iff $(d, e, f) \neq (d', e', f')$
$abc|def|g \not\sim_A a'b'c'|d'e'f'|g'$	iff $g \neq g'$
$((abc|def|g), x) \not\sim_A ((a'b'c'|d'e'f'|g'), x')$	iff $(a, b, c, x) \neq (a', b', c', x')$
$((abc|def|g), x) \not\sim_B ((a'b'c'|d'e'f'|g'), x')$	iff $(d, e, f, x) \neq (d', e', f', x')$
$((abc|def|g), x) \not\sim_C ((a'b'c'|d'e'f'|g'), x')$	iff $(g, x) \neq (g', x')$
$h \sim_X h'$	otherwise

For the set P of propositions, we take $P = \{a_i, b_i, c_i | i \in \{0, 1, 2, 3, 4, 5, 6\}\}$. The function π is defined as $\pi(((tuv|wxy|z), s)) = \{a_t, a_u, a_v, b_w, b_x, b_y, c_z\}$. Thus in the situation $(012|345|6, 1)$ the propositions $a_0, a_1, a_2, b_3, b_4, b_5, c_6$ hold.

The knowledge goal of this problem can then be expressed by conjunction bknows $\wedge$ cig. The positive part bknows expresses that at B knows the deal of cards.

$$\text{bknows} = \bigwedge_{i \in D} (b_i \to K_B bi)$$

The second part cig expresses that C does not know for any card who holds it. We call this a negative knowledge requirement, or an ignorance requirement.

$$\text{cig} = \bigwedge_{i \in D} (\neg K_C a_i \wedge \neg K_C b_i)$$

We can now solve the Russian cards problem by finding a strategy that fulfills, for a suitable model F, the following question.

$$w(kcg(F, \{A, B\}, \emptyset, \text{bknows} \wedge \text{cig})) = 1$$

We can verify that there are strategies $\sigma = \sigma_{\{A\}}$ such that

$$w(kcg(F, \{A, B\}, \emptyset, \text{bknows} \wedge \text{cig})) = 1$$

One such strategy is the following

$$
\begin{aligned}
0 \in \sigma(abc|def|g) \text{ iff} && abc \in \{012, 034, 056, 135, 146, 236, 245\} \\
1 \in \sigma(abc|def|g) \text{ iff} && abc \in \{013, 026, 045, 125, 146, 234, 356\} \\
2 \in \sigma(abc|def|g) \text{ iff} && abc \in \{014, 025, 036, 123, 156, 246, 345\} \\
3 \in \sigma(abc|def|g) \text{ iff} && abc \in \{015, 024, 036, 126, 134, 235, 456\} \\
4 \in \sigma(abc|def|g) \text{ iff} && abc \in \{016, 023, 045, 124, 135, 256, 346\} \\
5 \in \sigma(abc|def|g) \text{ iff} && abc \in \{012, 035, 046, 136, 145, 234, 256\}
\end{aligned}
$$

It follows that "I have 012,034,056,135,146,236 or 245" is a safe statement for A to make when she has one of these sets of cards. Another conclusion one can draw is that in the Russian Cards problem, a vocabulary of six different signals is sufficient for A to communicate its hand safely to B.

Note that the strategy described above is nondeterministic. For certain hands, agent A has a choice of two actions. For instance $\sigma(012|def|g) = \{0, 5\}$ and $\sigma(146|def|g) = \{0, 1\}$. It is necessary for A to make truly random choices, and not simplify its strategy by always going for one action in these cases. To see this, suppose that agent A would decide that it does not use action 0 if another action is available, and suppose that the card deal is 056|124|3. In that case agent A would use action 0, and C would know that A can only have card deals 034, 056, 135, 236, or 245. Since C has card 3 itself, it can deduce that the only card deals that A can have are 056 and 245. Therefore C knows that A must have card

5. When using this strategy, A should therefore make a genuine random choice between the two available actions, for instance using a coin flip.

Other strategies for A are deterministic. One deterministic strategy, that requires at least seven different signals, is the strategy σ_2, described by the following formula.

$$\sigma_2(abc|def|g) = \{(a + b + c) \bmod 7\}$$

The idea is that agent A announces the sum of its cards modula 7. For instance if A has cards 024 it should use action 6. When C hears that A has chosen action 6, it can deduce that A has cards 024, 015, 123, 256 or 346. In case that C has card 6, it would know that A has either cards 024, 015 or 123, but does not know for any specific card that A has it. Thus we know that there are deterministic as well as nondeterministic strategies for agent A.

In the existing literature it was already proven that the statement "I have 012,034,056,135,146,236 or 245" can be made. In Van Ditmarsch's paper [106] all statements that can be used when A holds 012 are given. However Van Ditmarsch does not present a complete strategy, and indeed it is not trivial to come up with a set of six statements that cover all possible card combinations of agent A. Thus knowledge condition games is a suitable framework for searching detailed strategies for situations such as the Russian Cards problem.

7.3.4 Communication Example

In this example four agents are communicating to each other. The agents take turns in sending out a message, a single bit in this example. Not all agents can see all messages: agent A can only see what signal D sends, B what D sends, C can see what signal A sends, and D can see what B and C send. The problem is that B would like to know what message C sends.

In order to model a situation with four communicating agents, we define an interpreted game form $F_C = (\Sigma, H, turn, \sim, P, \pi)$ with the following components.

- $\Sigma = \{A, B, C, D\}$

- $H = \{\epsilon, a, ab, abc, abcd | a, b, c, d \in \{0, 1\}\}$

- $P = \{p\}$

- The function $turn$ is defined by

$$turn(\epsilon) = A$$
$$turn(a) = B$$
$$turn(ab) = C$$
$$turn(abc) = D$$

where $a, b, c \in \{0, 1\}$.

- The equivalence relations $\sim$ are defined by the following equations. In these equations, X can be any agent, and a, b, c, d and their primed counterprats can be either 0 or 1.

$$h \not\sim_X h' \text{ if} \qquad\qquad \|h\| \neq \|h'\|$$
$$abcd \sim_A a'b'c'd' \text{ iff} \qquad\qquad ad = a'd'$$
$$abcd \sim_B a'b'c'd' \text{ iff} \qquad\qquad bd = b'd'$$
$$abcd \sim_C a'b'c'd' \text{ iff} \qquad\qquad ac = a'c'$$
$$abcd \sim_D a'b'c'd' \text{ iff} \qquad\qquad bcd = b'c'd'$$
$$abc \sim_X a'b'c' \text{ iff} \qquad\qquad abc0 \sim_X a'b'c'0$$
$$ab \sim_X a'b' \text{ iff} \qquad\qquad ab00 \sim_X a'b'00$$
$$a \sim_X a' \text{ iff} \qquad\qquad a000 \sim_X a'000$$

- $\pi(ab1d) = \{p\}$ and $\pi(ab0d) = \emptyset$

¿From the definition of the equivalence relations one sees that A can see what D does, B can see what D does, C can see what A does, and D can see what B and C does. Also, agents can remember their own action, and they can see all messages that are sent.

In order to come up with a knowledge condition game, we assume that the three agents B, C and D act as a team. The goal is to make B know what action C selects. It is not hard to see that the three agents can do this: D can copy the message of A. Agent B can see this copied signal, and therefore knows what C has done. In order to make this example more interesting, we add another requirement: We assume that the coalition of agents does not want A to know what message C sends. Since agent A can also see what agent D does, the copying strategy sketched in this paragraph no longer works. The three agents must use a more complicated coalition strategy.

The knowledge goal described above consists of a positive and a negative part: agent B must gain some knowledge about C's action, and agent A must not gain knowledge about C. This complex goal is described by the following formula ϕ.

$$\phi = (K_B p \vee K_B \neg p) \wedge \neg(K_C p \vee K_C \neg p)$$

Note that the proposition p corresponds to C's signal.

The question is whether the three-agent coalition has a strategy that makes this goal true. In terms of knowledge condition games, we thus would like to have an answer to the following question.

$$w(kcg(F_C, \{B, C, D\}, \{A\}, \phi)) = 1$$

The answer to this question is "yes". The following strategy σ_{BCD} is a successful strategy in this knowledge condition game.

$$\sigma_{BCD}(a) = \{0, 1\}$$
$$\sigma_{BCD}(ab) = \{0, 1\}$$
$$\sigma_{BCD}(abc) = \{0|b = c\} \cup \{1|b \neq c\}$$

This strategy is nondeterministic in the first two steps, but deterministic in the last step. Agent B, who knows its own action, can deduce from D's action whether the proposition p holds. Agent A cannot do this, since it does not know what B has done. It is necessary for B and C that they make a nondeterministic choice, otherwise A could deduce whether p holds from knowing the strategy that is used.

7.4 Computational Complexity

Looking at computational complexity is interesting for two reasons. First of all it tells whether a certain problem is 'tractable', i.e. whether the problem can be solved in practice. Secondly, it can tell you more about the problem. It can tell you for instance whether something is a very general problem (i.e., whether the problem format can be used to formulate questions about many different situations, such as logic), or what features makes a problem difficult. In this section we look at the complexity of the *kcg decision problem*, which is the problem of deciding for a game $kcg(F, \Gamma, \Xi, \phi)$ whether the first coalition Γ has a winning strategy. We look at this problem under various assumptions, and report four theorems, as follows:

- The first theorem is concerned with the problem of deciding whether a coalition Γ can win a knowledge condition game with an empty set of opponents. This is called the *no-opponents* knowledge condition game decision problem. It turns out that this problem is already NP-complete, and thus not tractable.

- The second theorem states that the general *kcg* decision problem is even harder: with opponents the problem is Σ_2P-complete.

For the other theorems we use the alternative version of knowledge condition games kcg'.

- In the third theorem we claim that the no-opponents kcg' decision problem is as hard as the general problem.

- Both problems are NP-complete, which is the fourth theorem.

In this chapter, we encode interpreted game forms in an explicit way, by listing all histories. In practice protocols are often specified in an implicit way (for instance in some form of source code) and such representations can be exponentially more efficient.

7.4.1. THEOREM. *The problem to decide for given coalitions F, Γ and formula ϕ whether $w(kcg(F, \Gamma, \emptyset, \phi)) = 1$ is NP-complete.*

PROOF. Assume that F, Γ, ϕ are given. The empty coalition has only one strategy $\sigma_\emptyset$. This strategy is such that $u(F, \sigma_\emptyset) = F$. Therefore

$$w(kcg(F, \Gamma, \emptyset, \phi)) = 1 \iff \exists \sigma_\Gamma \ m(u(F, \sigma_\Gamma)) \models \phi$$

A nondeterministic polynomial algorithm for this problem exists. Find or guess nondeterministically a strategy σ_Γ. Since a strategy encodes a subset of actions available in F, the size of σ_Γ is smaller than the size of F and thus polynomial in the input size. Now calculate $M = m(u(F, \sigma_\Gamma))$, and verify for each state w of M that $M, w \models \phi$. The number of states in M is at most the number of terminal histories of F, so $\|M\| \leq \|F\|$. All of this can be done in polynomial time. Therefore, this problem can be solved using a nondeterministic polynomial algorithm and this problem is in NP.

In order to show that the restricted *kcg* problem of the theorem is as hard as any NP problem, we show that any instance of the 3SAT problem described on page 30 can be transformed into an equivalent restricted *kcg* instance. Let $\phi^3 = \bigwedge_i (a_i \vee b_i \vee c_i)$ be a propositional logic formula in conjunctive normal form with three literals per clause. The literal formulas a_i, b_i, c_i must be either atomic propositions or negated atomic propositions. We can construct an interpreted game form F with a single agent $\Sigma = \{A\}$ and a formula ϕ such that $w(kcg(F, \{A\}, \emptyset, \phi)) = 1$ if and only if $\exists S : S \models \phi^3$.

The model $F = (\{A\}, H, turn, \sim, P, \pi)$ is constructed in the following way. Let P^3 be the set of atomic propositions occurring in ϕ^3. The new set of atomic propositions P contains two propositions for any old proposition: $P = \{x^+ | x \in P^3\} \cup \{x^- | x \in P^3\}$. For each new proposition a history is created: $H = \{\epsilon\} \cup \{e_p | p \in P\}$. The interpretation function is such that only the corresponding atomic proposition is true: $\pi(e_p) = \{p\}$. Furthermore $turn(\epsilon) = A$. Agent A cannot distinguish any end state: $e_p \sim_A e_q$ for all terminal histories e_p and e_q.

The formula $\phi = \phi_1 \wedge \phi_2$ is a conjunction of two parts. The part ϕ_1 expresses that for each original atomic proposition $p \in P^3$, either the positive proposition p^+ is considered possible or the negative p^-, but not both:

$$\phi_1 = \bigwedge_{p \in P^3} (M_A p^+ \vee M_A p^-) \wedge \neg(M_A p^+ \wedge M_A p^-)$$

The idea is that the strategy that A uses is actually an assignment of values to all atomic propositions in P^3. The condition ϕ_1 expresses that such assignment must assign either the truth value true (p^+) or false (p^-) to each proposition p.
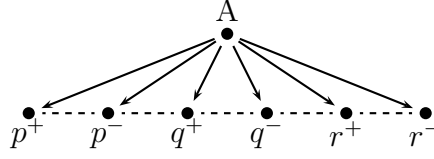
Figure 7.4: The model of 3SAT formula ψ

The ϕ_2 part encodes the original formula $\phi^3 = \bigwedge_i (a_i \vee b_i \vee c_i)$. In the next definition we use a helper function f defined such that $f(\neg p) = p^-$ and $f(p) = p^+$. Using this function we define B as follows.

$$\phi_2 = \bigwedge_i M_A(f(a_i) \vee f(b_i) \vee f(c_i))$$

It is not hard to see that any strategy $\sigma_{\{A\}}$ such that $m(u(F, \sigma_{\{A\}})) \models \phi_1 \wedge \phi_2$ corresponds to an assignment S such that $p \in S$ if and only if $p^+ \in \sigma_{\{X\}}(\epsilon)$, and that this assignment satisfies $S \models \phi^3$. Since the formula and model constructed have sizes that are linear with respect to the size of ϕ^3, this is a polynomial reduction. Therefore, the restricted *kcg* problem is NP-hard. Since we have also shown that the problem is in NP, we conclude that the restricted *kcg* problem is NP-complete. $\qquad\square$

As an example, consider the satisfiability of the 3SAT formula $\psi = (p \vee \neg q \vee r) \wedge (\neg q \vee \neg p \vee r)$. This formula contains three propositions, so the corresponding interpreted game form, depicted in figure 7.4, contains six terminal histories. The corresponding knowledge formula is ψ_K.

$$
\begin{aligned}
\psi_K = &(M_A p^+ \vee M_A p^-) \wedge \neg(M_A p^+ \wedge M_A p^-) \wedge \\
&(M_A q^+ \vee M_A q^-) \wedge \neg(M_A q^+ \wedge M_A q^-) \wedge \\
&(M_A r^+ \vee M_A r^-) \wedge \neg(M_A r^+ \wedge M_A r^-) \wedge \\
&M_A(p^+ \vee q^- \vee r^+) \wedge M_A(q^- \vee p^- \vee r^+)
\end{aligned}
$$

A typical NP-complete problem is to determine whether a propositional logic formula is satisfiable. Suppose ϕ is a formula with atomic propositions $x_1, x_2, \ldots x_n$. We can thus write $\phi = \phi(\vec{x})$ where the vector $\vec{x}$ consists of all the x_i. The satisfaction problem can now be phrased as deciding whether $\exists \vec{x} : \phi(\vec{x})$. In the same way we can formulate more difficult problems, by allowing more quantifiers: $\exists y \forall x : \phi(\vec{x}, \vec{y})$ is the problem where one has to decide whether there is an $\vec{x}$ such that $\phi(\vec{x}, \vec{y})$ is true for all $\vec{y}$. This problem, called SAT_2, is a typical $\Sigma_2 P$ complete problem [81, ch. 17]. It is widely believed, but not proven, that these problems are strictly more difficult than NP-complete problems.
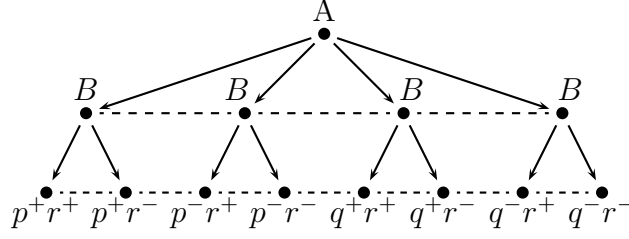
Figure 7.5: The construction of the $\Sigma_2 P$ proof

7.4.2. THEOREM. *Deciding for given F, Γ, Ξ and ϕ whether $w(kcg(F, \Gamma, \Xi, \phi)) = 1$, is $\Sigma_2 P$-complete problem.*

PROOF. First we have to prove that this problem is indeed in $\Sigma_2 P$. In order to do this, consider the winning condition of a knowledge condition in more detail.

$$w(kcg(F, \Gamma, \Xi, \phi)) = 1 \;\Leftrightarrow\; \exists \sigma_\Gamma \forall \sigma_\Xi \; m(u(u(F, \sigma_\Gamma), \sigma_\Xi)) \models \phi$$

Suppose that F, Γ, Ξ and ϕ are given. It is possible to encode strategies of Ξ as assignments to a vector of propositional variables $\vec{y}$, and the strategy of Γ as assignments to $\vec{x}$. One can then find a formula $\psi(\vec{x}, \vec{y})$ that is true if $m(u(u(F, \sigma_\Gamma), \sigma_\Xi)) \models \phi$. The size of this formula is polynomial in $|F| + |\phi|$. The *kcg* decision problem is equivalent to a SAT_2 problem:

$$w(kcg(F, \Gamma, \Xi, \phi)) = 1 \;\Leftrightarrow\; \exists \vec{x} \forall \vec{y} : \psi(\vec{x}, \vec{y})$$

Deciding whether $\exists \vec{x} \forall \vec{y} : \psi(\vec{x}, \vec{y})$, is a SAT_2 problem, and is thus in $\Sigma_2 P$.

The second part of the proof is to show that the *kcg* decision problem is indeed complete for this class, and this can be done by reducing SAT_2 to a knowledge condition game. The proof is similar to the previous NP-completeness proof, but now involves two agents. Assume that a SAT_2 problem $\exists \vec{y} \forall \vec{x} : \psi(\vec{x}, \vec{y})$ is given. We can assume that ψ is in 3SAT form: $\psi = \bigwedge_i (a_i \vee b_i \vee c_i)$. First we define an interpreted game form $F = (\Sigma, H, turn, \sim, P, \pi)$. Let $\Sigma = \{A, B\}$, and $Z(H) = \{(a, b) | \exists i, j : a = x_i^+ \text{ or } a = x_i^-, b = y_j^+ \text{ or } b = y_j^-\}$. The set H contains all histories of $Z(H)$ and all prefixes of these histories. The function *turn* is defined such that A moves first, and then B moves: $turn(\epsilon) = A$ and $turn((x_i^\pm)) = B$. The relations $\sim_A$ and $\sim_B$ are equal, and defined such that each agent only knows the length of each history: $s \sim_A s' \Leftrightarrow |s| = |s'|$. The set of propositions P of the *kcg* problem is $\{z^+ | z \in (\vec{x} \cup \vec{y})\} \cup \{z^- | z \in (\vec{x} \cup \vec{y})\}$. The function π is defined by $\pi(a, b) = \{a, b\}$. This completes the definition of the interpreted game form F. The number of terminal histories of F is $2|\vec{x}| \cdot 2|\vec{y}|$, and thus the size of F is polynomial in the size of the input problem.

We define $\Gamma = \{A\}$ and $\Xi = \{B\}$. Next, we define an epistemic logic formula ϕ such that Γ can win the game $kcg(F, \Gamma, \Xi, \phi)$ iff $\exists \vec{x} \forall \vec{y} : \psi(\vec{x}, \vec{y})$. Let $\phi =$

$\neg\phi^B \vee (\phi^A \wedge f(\psi(\vec{x}, \vec{y})))$. The part ϕ^B expresses that the strategy of B corresponds to an assignment to $\vec{y}$. The part ϕ^A expresses that the strategy of A corresponds to a strategy for $\vec{x}$. Finally, $f(\psi(\vec{x}, \vec{y}))$ is a translation of the input formula $\psi(\vec{x}, \vec{y})$.

$$\phi^B = \bigwedge_j ((M_B y_j^+ \vee M_B y_j^-) \wedge \neg(M_B(y_j^+ \wedge y_j^-)))$$

$$\phi^A = \bigwedge_i ((M_A x_i^+ \vee M_A x_i^-) \wedge \neg(M_A(x_i^+ \wedge x_i^-)))$$

$$f(\psi(\vec{x}, \vec{y})) = f(\bigwedge_i (a_i \vee b_i \vee c_i)) = \bigwedge_i (f(a_i) \vee f(b_i) \vee f(c_i))$$

The function f is defined such that $f(\neg p) = p^-$ and $f(p) = p^+$. The size of ϕ is linear in the size of ψ. Therefore, this is a polynomial reduction. This completes the proof that the knowledge condition game decision problem is Σ_2P-hard. Since it is also in Σ_2P, we conclude that the problem is Σ_2P-complete. $\square$

The construction of a model F is illustrated in figure 7.5. This is the model that you would get in the reduction of $\psi(\vec{x}, \vec{y})$ where $\vec{x}$ contains p and q and $\vec{y}$ consists of r. The model is again relatively small: only two actions happen in each play of this interpreted game form. The first one is decided by agent A, the second one by B.

In the two previous proofs, it is essential that the agents are aware of the strategies they choose. Both constructions would not work with the alternative definition kcg'. One can hope that the computational complexity of the kcg' decision problem would be lower. Indeed one can prove that in this case it does not matter whether there are opponents.

7.4.3. THEOREM. *Assume that F, Γ, Ξ and ϕ are given. $w(kcg'(F, \Gamma, \Xi, \phi)) = 1$ iff $w(kcg'(F, \Gamma, \emptyset, \phi)) = 1$.*

PROOF. Let $G = kcg'(F, \Gamma, \Xi, \phi)$ be a kcg' decision problem. Notice that the goal of coalition Ξ is to choose a strategy σ_Ξ such that $\mathfrak{U}'(\sigma_\Gamma, \sigma_\Xi) = (0, 1)$, where $\mathfrak{U}'$ is the utility function of the game G. Since $\mathfrak{U}'$ is defined using universal quantification over the set of terminal histories of $u(u(G, \sigma_\Gamma), \sigma_\Xi)$, the best thing to do for coalition Ξ is to make sure that this set is as large as possible. In order to achieve this, σ_Ξ should choose the neutral strategy that allows any action: The strategy σ with $\sigma(h) = A(H, h)$. Since we have assumed that neutral agents can do any action, we might as well assume that the agents $X \in \Xi$ are neutral, and determine the value of the game $w(kcg'(F, \Gamma, \emptyset, \phi)) = 1$. $\square$

We see thus that the presence of opponents is not relevant, and indeed in ATEL no distinction between opponents and neutral agents is made. The question is

now whether solving the kcg' decision problem is still as hard as the original no-opponents kcg problem. The answer is yes. The no-opponents kcg' problem is also NP-complete. However, the proof is different in an interesting way.

7.4.4. THEOREM. *Deciding for given F, Γ and ϕ whether $w(kcg'(F, \Gamma, \emptyset, \phi)) = 1$, is an NP-complete problem.*

PROOF. We can prove that this problem is in NP by a similar argument as given for theorem 7.4.1. For the hardness result we again show a reduction from 3SAT. Assume that $\phi^3 = \bigwedge_{i=1}^{n}(a_i \vee b_i \vee c_i)$ is a propositional formula in conjunctive normal form with three literals per clause. Let P^3 be the set of atomic propositions occurring in ϕ^3. We define an interpreted game form between two agents: an agent Q that asks questions, and an agent A that answers them. The proponent coalition is $\Gamma = \{A\}$ and Q is assumed to be neutral. Every terminal history is of the form (p, b, i, x), where $p \in P^3$, $b \in \{0, 1\}$, $i \in \{1, 2, \dots, n\}$ and $x \in \{a_i, b_i, c_i\}$. The first action p is chosen by agent Q and must be one atomic proposition of ϕ^3. The agent A must then reply by giving a boolean value b. This indicates what truth value A has in mind for p. Then agent Q choses one triplet $(a_i \vee b_i \vee c_i)$ that appears in ϕ^3. Agent A then has to choose which of these three parts it thinks should be true: either a_i or b_i or c_i. The trick however is that $\sim_A$ is defined in such a way that for all histories h and h', agent A only knows the length of the histories: $h \sim_A h'$ iff $|h| = |h'|$.

A does not know, when making its final decision, which answer it has given on its first turn. The agent thus risks giving inconsistent information. For instance in the history $(p, 1, (\neg p \vee q \vee r), \neg p)$ agent A first says that p is true, and then says that it thinks that $\neg p$ holds. The goal of agent A in the game is to avoid these inconsistent histories. We let $P = \{e\}$ consist of one proposition and define for all $p \in P^3$ the interpretation function such that $\pi((p, 1, i, \neg p)) = \{e\}$, $\pi((p_j, 0, i, p_j)) = \{e\}$ and $\pi((p, b, i, x)) = \emptyset$ otherwise. One can now consider the knowledge condition game $G = kcg'(F, \{A\}, \emptyset, \neg e)$. Agent A can win the game G iff there is a satisfying assignment for ϕ^3. $\square$

The proof given above is very similar to a proof given by Schobbens [91] for the NP-completeness of ATL with imperfect information. This corroborates our claim that this variant of knowledge condition games is closely related to ATL and thus to ATEL. The proof exploits the fact that in games where coalitions do not have perfect recall, it is very difficult for agents and coalitions to coordinate their own actions.

7.4.1 Tractable Variants

In the previous section we proved that, in general, the kcg decision problem is not tractable. In this section we identify some easier cases.

7.4.5. THEOREM. *Let F be an interpreted game form with perfect recall, Γ any coalition of agents and ϕ an epistemic formula. Deciding whether $w(kcg'(F,\Gamma,\emptyset,\phi)) = 1$ can be done in polynomial time.*

PROOF. Let $M = (\Sigma, W, R, P, \pi) = m(F)$ be the end state model of F. We can compute the set $S = \{w \in W \mid M, w \models \phi\}$ in polynomial time. Define a utility function $\mathfrak{U}$ such that $\mathfrak{U}(w) = 1$ iff $w \in S$ and $\mathfrak{U}(w) = 0$ otherwise. The pair $F, \mathfrak{U}$ is now an extensive game with perfect recall. The optimal solution σ for this game can be computed in polynomial time [58]. If the expected payoff of σ is exactly one, then $w(kcg'(F,\Gamma,\emptyset,\phi)) = 1$, otherwise $w(kcg'(F,\Gamma,\emptyset,\phi)) \neq 1$. $\qquad\square$

For perfect recall frameworks and the variant kcg' the decision problem is thus tractable. One might wonder whether the same claim can be made for kcg. The answer is no, because one can modify the NP-completeness proof for kcg in such a way that it uses a perfect recall interpreted game form. The modification is that one has two agents, A and A', so that A is the agent that chooses a strategy, and A' is the agent that cannot distinguish end states and occurs in the knowledge condition. In general one can always find a perfect recall interpreted game form that is equivalent for the kcg decision problem by choosing a fresh agent for each decision node, and use fresh agents in the knowledge condition.

Instead of asking whether there are interpreted game forms F that make decision problems easy, one can also ask whether there are easy formulas ϕ. The answer to this question is yes. To see how this works, we first formulate the notion of *positive formulas* and *negative formulas* formulas.

7.4.6. DEFINITION. For any $p \in P$, the formula p is both positive and negative. Falsum $\bot$ is also both positive and negative. If ϕ is positive and ψ is negative, then $\phi \to \psi$ is negative. Vice versa, if ψ is positive and ϕ is negative, then $\phi \to \psi$ is positive. If ϕ is positive then $K_X \phi$ is positive.

Positive and negative formulas are both called *monotone* formulas, because one can prove that they preserve truth in the following way. Suppose that $M = (\Sigma, W, \sim, P, \pi)$ and $M' = (\Sigma, W', \sim', P, \pi')$ are models such that $W' \subseteq W$ and $\sim'$ and π' are the restrictions of $\sim$ and π to W'. In this case we say that M' is a submodel of M. Suppose ϕ^+ is a positive formula, and ϕ^- is a negative formula. Then the following statements can be proven.

$$\begin{aligned} M, w \models \phi^+ \quad &\text{implies} \quad M', w \models \phi^+ \\ M', w \models \phi^- \quad &\text{implies} \quad M, w \models \phi^- \end{aligned}$$

The proof of these statements is done by induction over the formula structure. The interesting step involves the knowledge operator. Suppose that $M, w \models K_X \phi^+$. By definition this means that $\forall v \in W : w \sim_X v \implies M, v \models \phi^+$. Since W' is a subset of W, this means that $\forall v \in W' : w \sim'_X v \implies M, v \models \phi^+$. Using

the induction hypothesis we obtain $\forall v \in W' : w \sim'_X v \implies M', v \models \phi^+$ and thus $M', w \models K_X \phi^+$.

Knowledge condition games with monotone formulas are easier to solve than general knowledge condition games.

7.4.7. Theorem. *The problem to decide for given F, Γ, Ξ and a monotone formula ϕ whether $w(kcg(F, \Gamma, \Xi, \phi)) = 1$ can be solved in polynomial time.*

Proof. We prove the case where ϕ is a positive formula. The argument for negative formulas is similar. Recall that by definition, $w(kcg(F, \Gamma, \Xi, \phi)) = 1$ iff $\exists \sigma_\Gamma \forall \sigma_\Xi \forall w \in W$ it holds that $m(u(u(F, \sigma_\Gamma), \sigma_\Xi)), w \models \phi$ where W is the set of worlds in the model $m(u(u(F, \sigma_\Gamma), \sigma_\Xi))$. Since ϕ is a positive formula, we know that $m(u(F, \sigma_\Gamma)), w \models \phi$ implies $m(u(u(F, \sigma_\Gamma), \sigma_\Xi)), w \models \phi$. The best thing for coalition Ξ to do is to use a strategy that does not eliminate any action. They should use a neutral strategy σ^0 such that $u(F, \sigma^0) = F$. This strategy is described by $\sigma^0(h) = A(H, h)$.

For coalition Γ things are exactly opposite. Suppose that σ^1 and σ^2 are strategies so that σ^1 is more specific than σ^2. Formally, this means that $\forall h : \sigma^1(h) \subseteq \sigma^2(h)$. The monotonicity of ϕ implies that $m(u(F, \sigma^2)), w \models \phi$ implies $m(u(F, \sigma^1)), w \models \phi$. Coalition Γ thus does best be choosing the more specific strategy σ^1. For coalition Γ we thus only have to consider the most specific strategies. These most specific strategies are what one can call *pure*, because they select exactly one action at each decision point. A backward induction argument can be used to show that there are as many pure strategies for F as there are terminal histories in F. We can try all pure strategies σ^p to see if one satisfies $\forall w : m(u(F, \sigma^p)), w \models \phi$. This gives an algorithm that needs time $\mathcal{O}(\|F\|^2 \cdot \|\phi\|)$. The first $\|F\|$ is caused by the fact that we need to consider all pure strategies. The remaining term $\|F\| \cdot \|\phi\|$ is the time needed to determine whether for all w it is the case that $u(M, \sigma^p), w \models \phi$. The decision problem can thus be done in polynomial time.

For negative formulas the roles of Γ and Ξ are interchanged. For a negative formula ϕ, coalition Γ can use the neutral strategy σ^0. The opponent coalition Ξ should now try all pure strategies σ^p. $\square$

7.5   Related Work

Knowledge conditions games are games based on epistemic logic, that can be used for modelling games about knowledge. This makes them very similar to the logic ATEL [102, 103]. This logic is an extension of epistemic logic with operators to talk about group abilities and time. It is based upon ATL, discussed in chapter 2.

The language of ATEL contains temporal operators similar to CTL and knowledge operators. The temporal operators are always preceded by an agent operator.

7.5.1. DEFINITION. Let Σ be a set of agents, and P a set of atomic propositions. The logic ATEL contains formulas ϕ generated by the following rule. In this rule, p is a typical element of P, $X \in \Sigma$ and $\Gamma \subseteq \Sigma$

$$\phi ::= p \mid \phi \rightarrow \phi \mid \bot \mid \langle\langle\Gamma\rangle\rangle\psi \mid K_X\phi$$
$$\psi ::= \Box\phi \mid \phi\mathcal{U}\phi$$

This logic is interpreted over alternating epistemic transition systems. These are defined as tuples $(P, \Sigma, Q, \sim, \pi, \delta)$. As usual P is a set of atomic propositions and Σ a set of agents. The set Q is a set of states the system can be in, and $\pi : Q \rightarrow P$ adds propositions to these states. For any agent X the relation $\sim_X \subseteq Q \times Q$ is an equivalence relation, and $\delta : Q \times \Sigma \rightarrow 2^{2^Q}$ assigns to each agent in each state a set of sets of states. Each agent can choose one set of states, and the next state of the system will be from that set.

An example would be a system where $Q = \{0, 1, 2, 3, 4\}$. Suppose that $\delta(0, X) = \{\{1, 2\}, \{3, 4\}\}$ and $\delta(0, Y) = \{\{1, 3\}, \{2, 4\}\}$. Agent X can now choose $\{1, 2\}$ and Y can choose $\{2, 4\}$. They make these choices simultaneously. The next state of the system will be 2, because that is the only common state in their chosen sets. It is necessary to put some constraints on δ so that a next state can always be chosen.

The interpretation of this logic uses the notion of strategy to interpret the coalition operator $\langle\langle\Gamma\rangle\rangle$. A strategy for Γ is any function that makes a choice $\sigma_\Gamma(X, q) \in \delta(q, X)$ for any agent $X \in \Gamma$ in any state $q \in Q$. Based on a strategy σ_Γ, one can define the set of possible walks $\mathcal{W}(\sigma_\Gamma)$ through Q so that all choices for agents $X \in \Gamma$ are made as recommended by the strategy. This set of walks is used in the following interpretation of ATEL.

$$
\begin{array}{ll}
M, q \models \bot & \text{never} \\
M, q \models p \text{ where } p \in P & \text{iff } p \in \pi(v) \\
M, q \models \phi \rightarrow \psi & \text{iff } M, q \models \phi \text{ implies } M, q \models \psi \\
M, q \models K_X\phi & \text{iff } \forall(q, q') \in \sim_X: M, q' \models \phi \\
M, q \models \langle\langle\Gamma\rangle\rangle\phi & \text{iff } \exists\sigma_\Gamma : \forall w = v... \in \mathcal{W}(\sigma_\Gamma) : M, w \models \phi \\
\\
M, w \models \Box\phi & \text{iff } \forall n > 0 : Q, w(n) \models \phi \\
M, w \models \phi\mathcal{U}\psi & \text{iff } \exists m > 0 : M, w(m) \models \psi \text{ and} \\
& \quad \forall m > k > 0 : M, w(k) \models \phi
\end{array}
$$

A main advantage of ATEL over *kcg* is that ATEL extends temporal logic, and can thus be used to express different kinds of goals such as eventually achieving something, or avoiding some state forever. When this logic was presented it was reported that the logic has a low model checking complexity [102]. Unfortunately this only holds if one allows strategies that are not *uniform* (see definition 3.3.13). If one demands uniform strategies, model checking becomes NP-complete, even without using the knowledge operator [91]. Another point of discussion for this logic is the fact that the existence of a strategy, used in the interpretation of $\langle\langle\Gamma\rangle\rangle\phi$, is a very weak condition. One can come up with situations were $\langle\langle X\rangle\rangle\phi$ holds but one would not expect X to achieve ϕ [54, 55, 109]. Thus, it seems that the interpretation of this logic still needs some sorting out, and indeed ATEL currently receives a lot of research attention [3, 87].

Knowledge condition games is a less versatile verification framework than ATEL, because *kcg* does not allow complicated temporal reasoning. Only the special case of knowledge at the outcome stage of the protocol is studied. Knowledge condition games also do not allow for concurrent moves. This has the advantage that knowledge condition games are easier to understand, and that the complications that arise in the interpretation of ATEL do not arise in the context of knowledge condition games. An interesting difference between ATEL and *kcg* is that in *kcg* nondeterministic (and hence arguably "richer") strategies are used, whereas ATEL uses deterministic strategies.

One can also compare knowledge condition games to variants of dynamic epistemic logic, described on page 54, since dynamic epistemic logic allows reasoning about the effect of actions on the knowledge of agents. Indeed the quiz master problem is inspired by Van Ditmarsch' analysis of the Russian Cards problem [106].

7.6   Conclusion

By combining protocols and knowledge conditions into games, one can express properties of multi-agent protocols relating to security and secrecy. In a knowledge condition game, one can make fine distinctions between for instance neutral and opponent agents, and one can give examples where this distinction is significant. Therefore, these games are a promising direction for future research into the interaction between knowledge and strategies.

The complexity results reported in this chapter draw an interesting picture. There seems to be a computational cost for assuming that agents know strategies. The single agent decision problem is already intractable. The presence of opponents makes it even harder to compute whether a coalition can guarantee a property. If we drop this assumption and reformulate the notion of winning a knowledge condition game, then the extra complexity of adding opponents disappears. However, the problem without opponents is still NP-complete and thus

intractable, but for different reasons. The complexity proof is no longer based upon formulating a difficult knowledge formula, but on the hardness of coordinating in an interpreted game form without perfect recall.

Future research could focus on comparing decision problems for knowledge condition games to other game-theoretic decision problems, in order to establish what exactly the complexity cost is of considering knowledge goals. It would also be interesting to find out under which assumptions knowledge condition games can be solved in polynomial time. Other directions include looking at knowledge condition games from a logical viewpoint by searching for axioms, and to consider the mechanism design problem to find an interpreted game form with given properties.

Chapter 8
Entropy and Privacy

8.1 Introduction

Information is valuable, and thus agents do not always want to give it away. Both organisations and individuals often want to keep certain information private. At the same time they might want to act upon it. Does this reveal the information? In this chapter we study how agents should act if they want to maximize their utility, while at the same time not giving away too much information. Unlike the previous chapter, in this chapter we do this based on explicit probabilities. We define two classes of games in which the utility for each agent does not only depend on the payoff of the chosen action, but also on the information properties of the strategy used. These games are called *minimal information games* and *most normal games* and might be applied to the following situations.

- Supermarkets and e-commerce shops register what is bought by each of their customers. Customers know this and even assist in this process by using so-called 'bonus cards' (Albert Heyn) or 'club cards'(Tesco). Nevertheless, many customers are worried about their privacy. They would prefer it if the shop knew less about them. Customers can do something to minimize the knowledge of the shop. First of all they can make their shopping less regular (i.e. randomly buy items so that the shop is not sure which products the customer actually uses). Secondly, they can sign up for more than one card(account) or swap cards between each other. On the Internet, deleting cookies at random intervals and using a different IP number can have the same effect.

- In a second price auction it is optimal to bid exactly as much as you think the item is worth [63]. However, you might have spent a lot of time to estimate the value of the item, so you do not want to reveal your estimate. Since your bid has to be public, it seems that you might do better by bidding slightly random. By modeling this as a minimal information game, one can

compute how one should randomise. A similar argument applies when you send out an artificial agent to do your shopping. If the agent is sent over an insecure network, everyone can inspect the source code and thus the bidding strategy of the agent. You might not want to send an agent that is exactly optimal for your preferences, in order to hide your preferences.

- Many public places are now monitored by closed circuit television systems. If you come to one such place regularly, the camera attendants learn a lot about your habits and thus about you. You feel that this is a breach of your personal privacy, and decide to hide your habits by changing your behaviour often, for instance by going to different shops in a different order every time. This situation can also be modeled as a minimal information game. Again one can translate this example to the domain of artificial agents and the Internet.

- Consider now the case of a criminal who wants to steal from a shop guarded by a closed circuit television system. He wants to look like a regular shopper, but has different goals. He thus wants to behave so that he can steal the most, while at the same time appear to be a normal shopper. This can be modeled as a most normal game.

As the similar setting of the last two examples suggest, minimal information games and most normal games are related to each other. From these examples it should also be clear that we assume that the strategies that agents use are publicly known. This assumption makes our results stronger (if you have privacy while your strategy is public, you will have even more privacy when you can keep your strategy secret).

Privacy has received a lot of attention from economists and in legal settings. Some key sources have been collected on a website [1]. The work in this chapter differs from these economic papers for two reasons. First of all we only deal with personal information privacy, whereas the word 'privacy' also has other meanings. The second difference is that these papers try to explain the need for personal privacy in terms of economic utility. Odlyzko for instance relates privacy and price discrimination [77]. It is assumed here that privacy is a fundamental value, that is not instrumental to any gain. Privacy itself is a good cause that can be enjoyed directly.

Distributed constraint optimization techniques can be used by agents to solve coordination problems such as scheduling a meeting at the most suitable time and place. In these applications agents have to reveal information on their preferences for the meeting, but this information is also privacy-sensitive [33, 67]. In this application domain there is also a trade-off between solution quality and privacy, and this can also be modeled using entropy [33]. Thus, privacy-related research certainly has practical applications and it would be interesting to study these further. Therefore, we agree with Maheswaran and others' [67] 'call to arms

to improve privacy protection algorithms and further research on privacy'. The results of this chapter can be seen as a response to this call, since the strategies that are developed in this chapter can be used within privacy protection algorithms. For example the agent in the examples in section 8.7 use a randomised strategy, computed using the results of this chapter, in order to make it as hard as possible for observers to learn their preferences.

The games defined in this chapter use a soft (probabilistic, quantitative) approach towards information. They deal with probabilities explicitly, and can make subtle distinctions between possible, likely and almost certain events. This soft approach can be contrasted to the hard approach (discrete, qualitative) of logic and model checking. When taking a hard approach in protocol analysis, one is only interested in what is possible and what not, with a complete disregard for the relative likelihoods of different outcomes. Both the soft and the hard approach have been used for multi-agent systems. The use of epistemic logic to understand the game of Cluedo [106] is an example of the hard approach, as well as other logical approaches to reasoning about knowledge and knowledge change [6, 9, 32, 102, 115]. Recent work on privacy preserving auctions [18] and work on the Dining Cryptographer problem [19] or the Russian Cards problem [106, 112] can also be classified as 'hard'. At the same time there is some work on reasoning about uncertainty [43, 60] that combines logic and a soft approach to information. The soft approach is more detailed than the hard approach, because it gives exact probabilities. In certain circumstances this is an advantage. The hard approach can tell us that agents do best by randomising their strategy, but does not indicate the exact probabilities of an optimal strategy. On the other hand the higher level of abstraction of the hard approach makes it easier to interpret the results.

A quantitative approach, based on information theory, can also be used to look at natural language pragmatics. See for instance the ongoing work by Van Rooij [116]. Another way to use entropy in a game-related setting is in a searching game such as Mastermind [59].

In this chapter, the focus is on strategic games, whereas in most previous chapters of this dissertation we use extensive games. The reason is that it is quite complicated, starting with the notation, to do a similar exercise for extensive games. It is also not necessary: An extensive game is a more detailed description of a strategic game, so the results of this chapter can be applied to extensive games.

The layout of this chapter is as follows. Section 8.2 describes a detailed example problem. The next section, section 8.3, introduces basic information theory notions such as entropy. In section 8.4 we define minimal information games, and calculate the best strategies in these games. In section 8.5 we do the same for most normal games. Section 8.6 shows that these concepts can be used for defining new solution concepts. As application is discussed in section 8.7. Finally, the conclusions are presented in section 8.8.

8.2   Example

The following problem serves as an example. Alice (agent 1) needs to buy one box of breakfast cereals every week. Every week she is faced with the following choice: whether to buy Allgrain (A), Barley (B) or Cornflakes (C). Alice is not indifferent to which brand she eats. In fact she likes A better than B and B better than C, as is indicated by the following matrix of utilities.

action	A	B	C
utility	3.0	2.0	1.0

If Alice is solely interested in maximising her expected utility, she should buy A every week. However, Alice knows that the shop is watching her shopping behaviour closely, and she is concerned about her privacy. She decides that the decision that she makes should be private, and she can achieve this by flipping a coin and letting her decision depend on this coin flip. This way the shop cannot predict her decision.

Alice first attempts to use the following random strategy.

action	A	B	C
probability	0.98	0.01	0.01

If Alice uses this strategy, then the shop does not know anything about her decision: All three actions may occur with positive probability. At the same time her expected payoff is still very high, because the suboptimal actions occur with a very low probability. Problem solved, so it seems. But this is not the whole story. Even though the shop does not gain any knowledge, it does gain information from this strategy. If the shop learns, from repeated observation, that Alice uses this strategy, then it is quite certain that she will buy A. The shop has gained quite a lot of information. Therefore, the indicated strategy is not the right strategy if one analyses the situation using information theory.

One can argue that no new types of games are needed, because one can capture Alice's wish for privacy in the utility function of some modified pure or mixed strategy game. This is not the case because in these games the utility of strategies is determined solely by the utility of single actions: The utility function must be of the form $U = \Sigma_a p(a)u(a)$, where $p(a)$ is the probability of action a, and $u(a)$ the payoff of this individual action. Privacy and uncertainty are not reducible to certain individual actions, and therefore no suitable pure or mixed strategy game can be found.

A more sophisticated idea is to model privacy by adding an extra player G that tries to guess Alice's actions. In such a game, Alice would gain a high payoff by randomising her actions, and thus optimal strategies for this game would also be privacy-preserving strategies. The following payoff matrix gives such a game. The parameters $\epsilon_1, \epsilon_2, \epsilon_3$, and $\delta_1, \delta_2, \delta_3$ are all positive.

$G\backslash$ Alice	A	B	C
A	$\epsilon_1, 3.0 - \delta_1$	$0, 2.0$	$0, 1.0$
B	$0, 3.0$	$\epsilon_2, 2.0 - \delta_2$	$0, 1.0$
C	$0, 3.0$	$0, 2.0$	$\epsilon_3, 1.0 - \delta_3$

This strategic game, in which both agent choose their strategy independently at the same time, has been designed such that agent G has incentives ϵ_i to choose the same action as Alice, while Alice receives penalties δ_i if G has 'guessed' her next action correctly. This game is thus arguably a good model for a situation in which A wants privacy. It is however not clear how one should estimate all the variables that one needs for this larger game. These considerations have convinced us that it is easier to treat privacy as an independent aspect of an agent's utility.

8.3   Information Theory

Information theory is the field of science that deals with the measurement of information [28]. It has applications in signal processing, communication networks, cryptography and error correction codes. In this chapter we use information theory, and its central notion *entropy*, to estimate the amount of information in strategies. Strategies will be modeled as stochastic variables ranging over a finite set of actions, so we define entropy over stochastic variables. The entropy of a stochastic variable is the amount of randomness in, the disorder of, or uncertainty about the value that the variable will take. The concept of entropy was introduced by Shannon [95], and it is widely seen as the most natural measure for information [28]. We define the following function $f(x, y)$, that is helpful for the definition of entropy. Let lg be the base 2 logarithm.

$$f(x, y) = \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0 \\ \infty & \text{if } x > 0 \text{ and } y = 0 \\ -x \lg y & \text{if } x \geq 0 \text{ and } y > 0 \end{cases}$$

For a discrete random variable X we define the entropy $E(X)$, which is measured in *bits*, in the following way.

$$E(X) = \sum_k f(p(X = k), p(X = k))$$

This definition of entropy does not work for continuous random variables. A different definition for continuous variables also exists [95, p. 35], based on integration rather than summation. Since this is slightly more complicated and continuous random variables are not used in this chapter, the details are not discussed here.

A random variable X with values in the domain $\{1, 2, \ldots, m\}$ can be specified by giving a vector of length m with the probabilities of each value: ($p(X =$
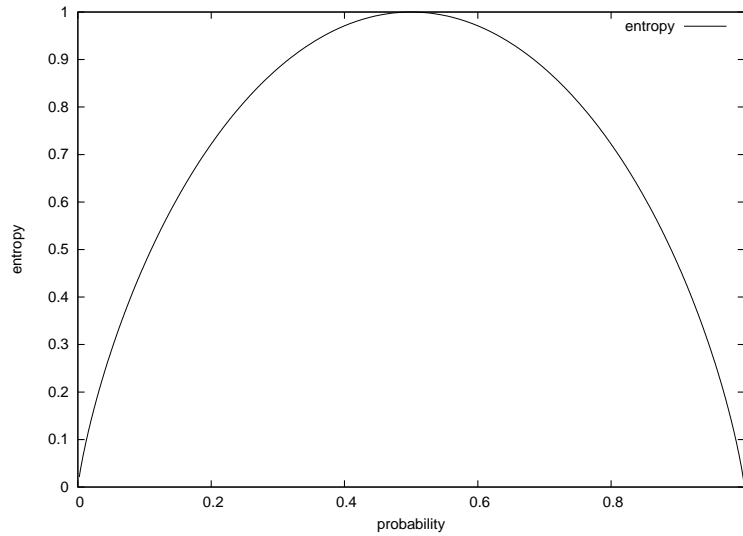
Figure 8.1: The function $E((x, 1 - x))$

$1), p(X = 2), \ldots, p(X = m))$. For a mixed strategy, the numbers $\{1, 2, \ldots, m\}$ represent the available actions. A requirement for probability measures on stochastic variables is that the probabilities should add up to 1. We can thus only use vectors x that indeed add up to 1, so it is convenient to define the set of all these vectors. The set definition of the set $\mathbf{P}^m$ from page 40 is repeated here, and we also define $\mathbf{Q}^m$ as the set of nonzero vectors.

$$\mathbf{P}^m = \{x \in [0, 1]^m \mid \sum_i x_i = 1\}$$

$$\mathbf{Q}^m = \{x \in (0, 1]^m \mid \sum_i x_i = 1\}$$

The set $\mathbf{P}^m$ contains all vectors of length m that add up to 1, and $\mathbf{Q}^m$ contains all vectors that add up to 1 and do not take the value 0. The set $\mathbf{Q}^m$ is important in some of the proofs, but often we work with the more general set $\mathbf{P}^m$. We can apply the notion of entropy to probability vectors $x \in \mathbf{P}^m$.

$$E(x) = \sum_k f(x_k, x_k)$$

In figure 8.1 the function $E((x, 1 - x))$ is displayed (here we apply the function E to a probabilty vector $(x, 1 - x)$ that depends on a variable $x \in [0, 1]$). Thus the figure shows the entropy of a two-valued random variable $(y_1, y_2) = (x, 1 - x)$, where x is the probability of the first action, and $1 - x$ the probability of the second action. As you can see the entropy in the two pure strategies, namely $(1, 0)$ and $0, 1$ is zero. The entropy is maximal if both actions are equally likely,

at $(0.5, 0.5)$. In the context of strategies, a strategy with a higher entropy leaves observers with more uncertainty, and thus gives the agent that uses that strategy more privacy. Below we give five examples of entropy. The example strategy vectors can all be seen as strategies over three basic actions. A strategy (a, b, c) contains the probability a of selection the first action, b for the second action and c for the third.

$$E((1/3, 1/3, 1/3)) = 1.585 \text{ bits}$$
$$E((0.5, 0.25, 0.25)) = 1.5 \text{ bits}$$
$$E((0.5, 0.5, 0)) = 1 \text{ bit}$$
$$E((0.98, 0.01, 0.01)) = 0.161 \text{ bits}$$
$$E((1.0, 0, 0)) = 0 \text{ bits}$$

Pure strategies, in which only one action gets a positive probability, have an entropy of zero bits. The entropy function is bounded. It cannot be negative, and a vector x of length m can have at most an entropy of $\lg m$. It has this entropy if all the entries x_i are equal to $1/m$, thus if the vector represents a stochastic variable with a uniform distribution.

The second idea that we use from information theory is *relative entropy* [28]. The function $E^{rel}(x, y)$ can be used to compare two probability vectors $x, y \in \mathbf{P}^n$. The underlying idea is that $E^{rel}(x, y)$ measures how much difference one would notice if probability vector x is used instead of y for selecting actions. In order to compute this difference, we add up the differences for each action k. The probability x_k corresponds to the probability that action k is chosen, given that strategy x is used: $x_k = P(k|x)$. Similarly $y_k = P(k|y)$. Using Bayes' law one can calculate the relative likelihood of strategy x instead of strategy y when observing that action k is chosen: $P(x|k)/P(y|k)$. Assuming that the a priori probabilities $P(x)$ and $P(y)$ are equal, one can derive that this is x_k/y_k.

$$\frac{P(x|k)}{P(y|k)} = \frac{P(x \cap k)P(k)}{P(y \cap k)P(k)} = \frac{P(k|x)P(x)}{P(k|y)P(y)} = \frac{P(k|x)}{P(k|y)}$$

This observation is the motive behind the following definition.

$$E^{rel}(x, y) = \sum_k f(x_k, y_k/x_k)$$

The function E^{rel} almost behaves as a distance function or metric. It is never negative and only zero if $x = y$. It also satisfies the triangle inequality. It is infinite if for some k it is the case that $x_k > 0$ and $y_k = 0$. The only difference between this function and a distance function or metric is that E^{rel} is not symmetric. In

many cases $E^{rel}(x, y) \neq E^{rel}(y, x)$.

$$E^{rel}((0.5, 0.5), (0.75, 0.25)) = 0.2075 \text{ bits}$$
$$E^{rel}((0.75, 0.25), (0.5, 0.5)) = 0.1887 \text{ bits}$$
$$E^{rel}((0.9, 0.1), (0.75, 0.25)) = 0.1045 \text{ bits}$$
$$E^{rel}((0.75, 0.25), (0.9, 0.1)) = 0.1332 \text{ bits}$$

If x has a higher entropy than x', then on average for a random vector y it is the case that $E^{rel}(y, x) < E^{rel}(y, x')$. It is harder to notice a difference between y and a high entropy vector x than to notice a difference between y and a low entropy vector x'.

8.4   Minimal Information Games

The next definition of a minimal information game aims to capture the following situation. Agents choose a mixed strategy with two goals in mind. First of all, they want a high payoff. Secondly, they want privacy. They feel that they have more privacy if others are more uncertain about the action they will choose, and thus they prefer strategies with a high entropy. These games thus model the situation where agents have a fundamental desire for privacy.

We have to specify how the agent would like to trade privacy against payoff. This is governed by a parameter $\alpha > 0$ that indicates the value of privacy. It expresses how much expected payoff the agent is willing to trade against a bit of privacy. The higher α, the more the agent values privacy.

8.4.1. DEFINITION. Let A be a $m_1 \times m_2 \ldots \times m_n$ multi-matrix and $\alpha > 0$. The *minimal information game* $\mathrm{Mi}^\alpha(A)$ is a tuple $(\Sigma, \{S\}_\sigma, \mathfrak{U})$ where $\Sigma = \{1, 2, \ldots, n\}$, the strategy sets are $S_X = \mathbf{P}^{m_X}$ and $\mathfrak{U}^X(\vec{s}) = \sum_i s_i^X A_i^X(\vec{s}) + \alpha E(s^X)$

The parameter α regulates how much all the agents value the fact that there is uncertainty over their next action. If we would allow $\alpha = 0$, then the game becomes a mixed strategy game: $\mathrm{Mi}^0(A) = \mathrm{Mx}(A)$. As α approaches infinity, the actual payoff becomes less and less important. It would have been possible to choose α differently for each agent, but this would have made the definition less clear.

As an example, we consider the shopping game from the introduction. This game has only one agent, that has three options A, B, C with respective payoffs $3, 2, 1$. The optimal strategies for the minimal information game with different values of α is given in the next table. It also lists the utility of s that the agent would get in the mixed strategy game $\mathrm{Mx}(A)$ for the given strategy s and the utility that the agent would get in the minimal information game $\mathrm{Mi}^\alpha(A)$.

α	p_1	p_2	p_3	$\mathrm{Mx}(A)$	$\mathrm{Mi}^\alpha(A)$
0.1	0.999	$4 \cdot 10^{-5}$	$2 \cdot 10^{-9}$	3.0	3.0
0.5	0.876	0.117	0.015	2.852	3.168
1.0	0.665	0.244	0.090	2.575	3.775

The best payoff that the agent can get is 3.0 by only choosing the first action. However this would result in no privacy, because if everybody knows that the agent uses this strategy, then any observer knows beforehand what the agent will do every week. For a low value of α the utility of s in $\mathrm{Mi}^\alpha(A)$ is very close to this optimal value of 3. For higher values, the average payoff without entropy becomes lower. We could call this the *cost* of privacy. From the table we can see that if the agent values privacy at one unit per bit (α is expressed in units per bit) then the agent does best by paying 0.425 in order to obtain 0.775 bits of privacy.

The question is of course how we can calculate the strategies that maximize the utility in minimal information games. For the linear functions of the mixed strategy games this is a solved problem, but for more complicated functions, such as the utility function of a minimal information game, this can be difficult. In the next theorem the solution for this optimisation problem is shown.

8.4.2. THEOREM. *Let* $\mathrm{Mi}^\alpha(A)$ *be a minimal information game and* $\vec{s}$ *a strategy profile. The set* $b^X(\vec{s})$ *is a singleton* $\{b\}$ *such that*

$$b_i = \frac{2^{\alpha^{-1}A_i^X(\vec{s})}}{\sum_k 2^{\alpha^{-1}A_k^X(\vec{s})}}$$

PROOF. Let $\mathrm{Mi}^\alpha(A) = (\Sigma, \{S_X\}_{X\in\Sigma}, \mathfrak{U})$ be a minimal information game. We have to prove that the set $b^X(\vec{s})$ contains one element, and that that element is described by the given formula. We first show that all points in $b^X(\vec{s})$ are interior points. Then we derive an equation that any best response must satisfy, and show that this equation has a unique solution, namely the one given in the theorem.

Let n be the number of actions that agent X can choose from. Take any vector $\vec{x} \in S_X$ and assume that $\vec{x} \in \mathbf{P}^n \setminus \mathbf{Q}^n$. We are going to show that there is a better vector $\vec{y}$, and thus $\vec{x}$ is not a best response. There is some i such that $x_i = 0$ and some j such that $x_j \neq 0$. We will show that there is some ϵ such that $\vec{y} = [[x_{-i}, \epsilon]_{-j}, x_j - \epsilon]$ is a better vector: $\mathfrak{U}^X([\vec{s}_{-X}, \vec{y}]) > \mathfrak{U}^X([\vec{s}_{-X}, \vec{x}])$. To show this, note that the utility function $\mathfrak{U}^X$ is continuous and differentiable. Note further that $\frac{\delta}{\delta x_i}\mathfrak{U}^X([\vec{s}_{-X}, \vec{x}]) = +\infty$ and $\frac{\delta}{\delta x_j}\mathfrak{U}^X([\vec{s}_{-X}, \vec{x}]) < +\infty$. Therefore, for sufficiently small ϵ, the gain from raising x_i outweighs the potential loss from lowering x_j. Therefore, for sufficiently small ϵ we have that $\mathfrak{U}^X([\vec{s}_{-X}, \vec{y}]) > \mathfrak{U}^X([\vec{s}_{-X}, \vec{x}])$ and thus $\vec{x} \notin b^X(\vec{s})$.

Now suppose that $b \in b^X(\vec{s})$. We know that $b \in \mathbf{Q}^n$. Take $i, j \in \{1, 2, \ldots, m\}$ as two different indices. Since b is optimal, it should not be possible to increase $\mathfrak{U}^X$

by increasing b_i while decreasing b_j, and therefore for any optimal point it holds that $\frac{\delta}{\delta b_i}\mathfrak{U}^X([\vec{s}_{-X}, b]) = \frac{\delta}{\delta b_j}\mathfrak{U}^X([\vec{s}_{-X}, b])$. We can use this as a starting point for the following link of equations. First we compute the derivative $\frac{\delta}{\delta b_i}\mathfrak{U}^X([\vec{s}_{-X}, b])$.

$$\frac{\delta}{\delta b_i}\mathfrak{U}^X([\vec{s}_{-X}, b]) =$$
$$\frac{\delta}{\delta b_i}(\sum_j b_j A_j^X([\vec{s}_{-X}, b]) + \alpha E(\vec{b})) =$$
$$A_i^X(\vec{s}) + \alpha\frac{\delta}{\delta b_i}(E(\vec{b})) =$$
$$A_i^X(\vec{s}) + \alpha(-\lg b_i - \lg e) =$$
$$A_i^X(\vec{s}) - \alpha\lg b_i - \alpha\lg e$$

Using this derivative one can reduce the equation given above in the following way.

$$\frac{\delta}{\delta b_i}\mathfrak{U}^X([\vec{s}_{-X}, b]) = \frac{\delta}{\delta b_j}\mathfrak{U}^X([\vec{s}_{-X}, b]) \qquad \Leftrightarrow$$
$$A_i^X(\vec{s}) - \alpha\lg b_i = A_j^X(\vec{s}) - \alpha\lg b_j \qquad \Leftrightarrow$$
$$A_i^X(\vec{s}) - A_j^X(\vec{s}) = \alpha\lg b_i - \alpha\lg b_j \qquad \Leftrightarrow$$
$$\frac{2^{A_i^X(\vec{s})}}{2^{A_j^X(\vec{s})}} = \frac{b_i^\alpha}{b_j^\alpha}$$

Since $b \in \mathbf{P}^n$ it holds that b sums up to $\sum_i b_i = 1$. For any $b \in b(\vec{s})$ one can find some positive constant c such that $b_i = c \cdot 2^{\alpha^{-1}A_i^X(\vec{s})}$. It now follows from the above equation that for any b_j it is the case that $b_j = c 2^{\alpha^{-1}A_j^X(\vec{s})}$. We can now calculate $\sum_k b_k = 1 = c\sum_k 2^{-\alpha A_k^X(\vec{s})}$ and thus we know that $\frac{1}{c} = \sum_k 2^{\alpha^{-1}A_k^X(\vec{s})}$. Thus, we have proven that there is a unique point $b \in b^X(\vec{s})$ which satisfies the equation in theorem 8.4.2 $\qquad\qquad\qquad \square$

8.4.3. THEOREM. *Every minimal information game* $\mathrm{Mi}^\alpha(A)$ *has a Nash equilibrium.*

PROOF. Let f be the function from $S_1 \times \ldots \times S_n$ to $S_1 \times \ldots \times S_n$ that returns the strategy vector with the best responses for each agent. Thus, f is the function that for each x returns the unique point $f(x)$ such that $f(x) \in b(x)$. The previous theorem shows that this is a continuous function. The set $S_1 \times \ldots S_n$ is topological isomorphic to some closed sphere $\mathbb{B}^m$. We can now use Brouwer's fixed point theorem, which tells us that every continuous function $f : \mathbb{B}^m \to \mathbb{B}^m$ must have

a point x with $f(x) = x$ [4]. We thus obtain a strategy vector x with $f(x) = x$, and thus a point x such that $x \in b(x)$. This point is a Nash equilibrium. □

This proof is related to Nash's original proof that Nash equilibria exist in mixed strategy games by the fact that both theorems can be proven using Brouwer's fixed point theorem. The difference however is that the mixed strategy games have linear payoff functions. Minimal information games do not have linear payoff functions, so in this proof the fixed point theorem is used in a different way.

The two theorems of this section, theorem 8.4.3 differ in their constructiveness. Theorem 8.4.2 gives a concrete way to compute optimal responses in minimal information games. This theorem can therefore be applied immediately. Indeed we have used the result formula of this theorem to compute the optimal strategies in the table on page 160. Thus one can immediately apply this theorem in order to decide how to act, or to predict how others will act, in situations that can be modelled as minimal information games. Indeed in section 8.7 we apply the theorem again to find strategies for agents.

Bach or Stravinsky

Theorem 8.4.3 is not immediately applicable, because it does not tell one how one should find a Nash Equilibrium. It is thus not constructive in a practical sense. However it is important to know that a Nash equilibrium exists, since this can be a strong motivation for finding one. In the next example we use the following bi-matrix A for defining a two-person minimal information game.

$$
\begin{array}{cc}
2,1 & 0,0 \\
0,0 & 1,2
\end{array}
$$

This matrix is often used in a game called *Bach or Stravinsky*[79, p. 16]. The story behind these payoffs is that both agents can decide where they want to go tonight, either to a Bach concert or a Stravinsky concert. Both agents enjoy each others company, and hence they receive zero payoff if they do not go to the same concert. The first agent prefers Bach and thus experiences 2 units of value when both agent choose the first option. The second agent values Bach at 1 and Stravinsky as 2.

Since we are interested in privacy, we assume that both agents value their privacy. Hence we define a minimal information game $\mathrm{Mi}^\alpha(A)$, where $\alpha = 0.5$. As we have seen in theorem 8.4.2 it is optimal for agents to randomize their behaviour somehow. Theorem 8.4.3 tells us there is at least one Nash equilibrium. We have used computer search to find one for the stated value of α.

Agent	prob. action 1	prob action 2
1	0.148	0.851
2	0.042	0.957

One can see that in this Nash equilibrium Stravinsky is the most likely outcome. Both agents choose action 2 most often. However they do not do this with absolute certainty, in order to leave some uncertainty for observers. The exact probabilities are different for both agents since they have slightly different payoffs.

8.5   Most Normal Games

So far we have discussed the situation in which the agents try to protect their privacy against an opponent interested in their next action. In this section we look at another situation, in which agents try to hide their preferences. It is assumed that an average strategy for 'normal' users is given. One agent however has different preferences from the normal users, but does not want to be identified as not normal. Therefore, the agent is searching for a strategy that appears as normal as possible and maximizes its payoff at the same time.

We approach the problem in exactly the same way as we have approached the first problem. We define most normal games $\mathrm{Mn}^{\alpha}(A)$ that depend on a parameter α expressing how important normal behaviour for the agent is.

8.5.1. Definition. Let A be a $m_1 \times m_2 \ldots \times m_n$ multi-matrix, let $\alpha > 0$, and let $\vec{t}$ be a strategy vector for the game $\mathrm{Mx}(A)$. The *most normal game* $\mathrm{Mn}^{\alpha}(A, \vec{t})$ is a tuple $(\Sigma, \{S_X\}, \mathfrak{U})$ where $\Sigma = \{1, 2, \ldots, n\}$, the strategy sets are $S_X = \mathbf{P}^{m_X}$ and $\mathfrak{U}^X(\vec{s}) = \sum_i s_i^X A_i^X(\vec{s}) - \alpha E^{rel}(s^X, t^X)$

The parameter α again determines the trade-off between selecting actions with a high payoff and acting normal.

8.5.2. Theorem. *Let* $\mathrm{Mn}^{\alpha}(A, \vec{t})$ *be a most normal game and* $\vec{s}$ *a strategy profile for this game. The set* $b^X(\vec{s})$ *is a singleton* $\{b\}$ *such that*

$$b_i = \frac{t_i^X 2^{\alpha^{-1} A_i^X(\vec{s})}}{\sum_k t_k^X 2^{\alpha^{-1} A_k^X(\vec{s})}}$$

Proof. Let $\mathrm{Mn}^{\alpha}(A, \vec{t})$ be a most normal game, $\vec{s}$ a strategy profile and $X \in \Sigma$ an agent. Suppose that $b \in b^X(\vec{s})$ is the best response for agent X and let i be one of B's actions. If $t_i = 0$ and $b_i \neq 0$, then the relative entropy becomes infinite, and the utility thus infinitely low. This cannot be optimal, thus if b maximizes the utility, then $t_i = 0$ implies $b_i = 0$. Thus, in this case the optimal point is not an interior point. It follows that if $t_i = 1$, then for any optimal strategy b we must have $b_i = 1$.

Consider now the case where $t_i > 0$. We calculate the derivative of the relative entropy function.

$$\frac{\delta}{\delta b_i} E^{rel}(b, t^X) = \frac{\delta}{\delta b_i} \sum_i -b_i(\lg t_i^X - \lg b_i) = \lg b_i + \lg e - \lg t_i^X$$

We see that if $b_i > 0$ approaches zero, then this derivative becomes negative infinity. If b_i is sufficiently small, then we would lower the utility $\mathfrak{U}^X([\vec{s}_{-X}, b])$ by decreasing b_i further. Therefore, for any optimal value of b, it cannot be the case that $t_i > 0$ and $b_i = 0$.

Since we have shown that $t_i = 0$ implies $b_i = 0$, it remains for us to find the optimal vector in the space $S = \{b \in [0,1]^m \mid \sum_i b_i = 1 \wedge (t_i = 0 \rightarrow b_i = 0)\}$. The previous argument has shown that b is an interior point of this set S. Such points can only be optimal if $\frac{\delta}{\delta b_i}\mathfrak{U}^X([\vec{s}_{-X}, b]) = \frac{\delta}{\delta b_j}\mathfrak{U}^X([\vec{s}_{-X}, b])$ for any pair i, j with $t_i, t_j > 0$. The next computation will show that there is a unique point satisfying this condition. Since any continuous function on a closed domain must have a maximum, this point b will maximize agent X's utility in the normal form game.

First we calculate the derivative.

$$\frac{\delta}{\delta b_i}\mathfrak{U}^X([\vec{s}_{-X}, b]) =$$

$$A_i^X(\vec{s}) - \alpha\frac{\delta}{\delta b_i}E^{rel}(b, t^X) =$$

$$A_i^X(\vec{s}) - \alpha(\lg b_i + \lg e - \lg t_i^X) =$$

$$A_i^X(\vec{s}) - \alpha \lg b_i - \alpha \lg e + \alpha \lg t_i^X$$

Now find the points b where the derivatives $\frac{\delta}{\delta b_i}\mathfrak{U}^X$ and $\frac{\delta}{\delta b_j}\mathfrak{U}^X$ are equal.

$$\frac{\delta}{\delta b_i}\mathfrak{U}^X([\vec{s}_{-X}, b]) = \frac{\delta}{\delta b_j}\mathfrak{U}^X([\vec{s}_{-X}, b]) \qquad \Leftrightarrow$$

$$A_i^X(\vec{s}) - \alpha \lg b_i + \alpha \lg t_i^X = A_j^X(\vec{s}) - \alpha \lg b_j + \alpha \lg t_j^X \qquad \Leftrightarrow$$

$$\alpha \lg(b_i/b_j) - \alpha \lg(t_i^X/t_j^X) = A_i^X(\vec{s}) - A_j^X(\vec{s}) \qquad \Leftrightarrow$$

$$\frac{b_i}{b_j} = \frac{t_i^X 2^{\alpha^{-1}A_i^X(\vec{s})}}{t_j^X 2^{\alpha^{-1}A_j^X(\vec{s})}}$$

Again we can choose c such that $b_i = ct_i^X 2^{\alpha^{-1}A_i^X(\vec{s})}$ and show that $\frac{1}{c} = \sum_k t_k^X 2^{\alpha^{-1}A_k^X(\vec{s})}$. This leads to the next formula.

$$b_i = \frac{t_i^X 2^{\alpha^{-1}A_i^X(\vec{s})}}{\sum_k t_k^X 2^{\alpha^{-1}A_k^X(\vec{s})}}$$

This formula gives us $b_i = 1$ if $t_i = 1$, and $b_i = 0$ if $t_i = 0$. Therefore, this formula gives us the optimal strategy for any normal form game. $\qquad \square$

Discussion

One consequence of the theorem is the following observation. If a certain action i is not considered by normal agents ($t_i^X = 0$) then the non-normal agent should not consider action i either ($b_i = 0$). If one had used a hard, logical approach one could have reached the same conclusion. In the most extreme case one can consider the case where normal agents use a pure strategy. In that case the non-normal agent has to use the same pure strategy. If the non-normal agent values all actions equally, he also does best by copying the normal strategy. In all other cases the best strategy for the non-normal agent is different. Apparently the agent does best by always taking some risk and getting a higher utility.

8.6   Equilibrium Refinements

By introducing minimal information games we have introduced a game with a new kind of utility function. For small values of α the game $\mathrm{Mi}^\alpha(A)$ is very similar to the mixed strategy game $\mathrm{Mx}(A)$. One can, with some imagination, see a Nash equilibrium x of $\mathrm{Mi}^\alpha(A)$ as a solution of $\mathrm{Mx}(A)$. In that case, one has a new solution concept for mixed strategy games $\mathrm{Mx}(A)$. Such a solution x of some game $\mathrm{Mi}^\alpha(A)$ is not a Nash equilibrium of $\mathrm{Mx}(A)$, but an approximation of it. How good this approximation is depends on the parameter α. We can define a Nash equilibrium by letting α approach zero. This way, we can define a 'minimal information' equilibrium.

8.6.1. DEFINITION. The strategy profile x is a minimal information equilibrium of $\mathrm{Mx}(A)$ iff there is a sequence $\alpha_1, \alpha_2, \ldots$ of positive numbers such that $\lim_{i \to \infty} \alpha_i = 0$, a sequence $x_1, x_2, \ldots$ such that x_i is a Nash equilibrium of $\mathrm{Mi}^{\alpha_i}(A)$ and $\lim_{i \to \infty} x_i = x$.

8.6.2. THEOREM. *Every mixed strategy game* $\mathrm{Mx}(A)$ *has a minimal information equilibrium.*

PROOF.   Define the sequence $\beta_1, \beta_2, \ldots$ by $\beta_i = 1/i$. This sequence converges to zero. By theorem 8.4.3 each game $\mathrm{Mi}^{\beta_i}(A)$ has some Nash equilibrium y_i. The strategy space $S_1 \times \ldots \times S_n$ is a closed and bounded subset of $\mathbb{R}^m$ for some m. Therefore, since any closed and bounded subset of $\mathbb{R}^m$ is compact [121] we derive that every sequence in $S_1 \times \ldots \times S_n$ has some converging subsequence. Let $x_1, x_2, \ldots$ be a converging subsequence of $y_1, y_2, \ldots$ and let x be the limit of $\lim_{i \to \infty} x_i$. Let $\alpha_1, \alpha_2, \ldots$ be the corresponding subsequence of $\beta_1, \beta_2, \ldots$, so that x_i is a Nash equilibrium of $\mathrm{Mi}^{\alpha_i}(A)$. When α approaches infinity, the utility function of $\mathrm{Mi}^{\alpha_i}(A)$ converges uniformly to the utility function of $\mathrm{Mx}(A)$. Since x_i is always maximizing each agents utility in $\mathrm{Mi}^{\alpha_i}(A)$, it must be the case that x maximizes the utility of $\mathrm{Mx}(A)$ for each agent. Therefore, x is a Nash equilibrium

of Mx(A). $\square$

Every minimal information equilibrium is a special case of a proper equilibrium as defined by Myerson, and therefore it is also a trembling hand perfect equilibrium [72]. These refinements can thus be motivated (if one wants to) by an appeal to privacy minded agents. Perhaps there are other applications where one needs a response concept that selects interior solution points, for instance to avoid division by zero. In that case the minimal information best responses seem suitable.

8.7 Telecom Network Example

Modern technology allows governments and other large institution to closely observe the movement of individuals. In the introduction we mentioned closed circuit television systems, but it is also possible using mobile telephone networks and in the near future RFID tags. In this section we therefore assume that an observer can monitor the behaviour of agents in a small part of a city. Three different scenarios have been implemented in a visual computer simulation, that allows the user to take the role of the observer. The user can try to identify what group agents belong to based on their behaviour. The agents have been programmed to optimize their behaviour using the optimal strategies of theorems 8.4.2 and 8.5.2. Explicit strategies that are based on these theorems are for instance given in table 8.7 on page 169. Different agents value their privacy differently and thus use a different value for α. It takes more time to spot agents that use a higher value of α, so the use of optimal strategies for privacy protection is effective in making life harder for an observer. However if the observer has no time constraints, it can ultimately identify all agents.

The simulation is available as a Java applet on the world wide web, at the address www.bluering.nl/sieuwert/programs/privacysim/simprivacy.html. One can see several agents walking between their home and several shops. The simulator currently contains three levels. Each level is a new puzzle or challenge to the user. The user can see all agents, and monitor which places they visit. The user also knows what groups of agents exist, and what the preferences of each group are. The goal is to guess the group of each agent.

All agents use optimal strategies for hiding their preferences. One might say that the agents know that they are being watched, and act in order to make it difficult to identify to which group they belong. In other words, the agents act as if they are playing a minimal information game or a most normal game. The agents are however not in competition with each other, but act independently. Below we quickly describe the settings of the first two levels. For the third level a longer description is given, in which the strategies used by the agents are described in
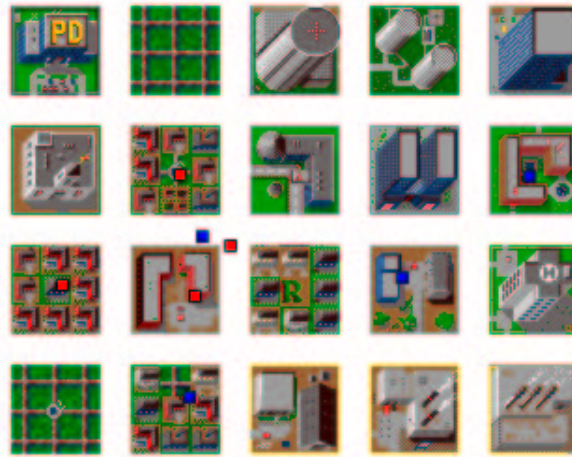
Figure 8.2: Sim Privacy

details.

Level 1: Rich or Poor

The first level shows part of a city with two shops. One shop is a cheap shop, the other one is an expensive shop. Two groups of agents live in this city, namely poor agents and rich agents. The poor agents prefer to go to the cheap shop, and the rich agents prefer the expensive shop. However, all agents do not want anybody to know whether they are rich or poor. Therefore, all agents randomize their shopping behaviour, and visit both shops with some probability. The goal of this level is to determine for each agent whether it is a rich or a poor agent.

The main learning point from studying this level in the simulation is that these puzzles can be solved. Since the agents adapt their strategy towards their payoffs (we have shown that it is optimal for them to do so in theorem 8.4.2), one gains some information from observing the agents behavior. If one is allowed to observe the agents long enough, one will gain enough information to determine the type of each agent with any level of probability. The simulation therefore shows that in the long run it is impossible to protect ones privacy against observers who have this much detailed information about ones daily behaviour. If one believes in the universal human right to privacy, it is therefore necessary to prevent organisations from collecting arbitrary large amounts of data, or to store such data for indefinite amounts of time.

Level 2: Citizens and Criminals

In this level there are again two groups of agents. The citizens shop in any of the four shops, and occasionally have to go to the bank to withdraw money. The

citizens go to each shop with equal probability. The criminals have other sources of income, and thus have no need to visit the bank. However the criminals do not want others to know that they are criminal, so sometimes they do walk to the bank to keep up appearances, but less often than they go to the shop. How often they go to the shop depends on their level of paranoia: Normal agents go less than paranoid agents.

This level demonstrates the influence of the parameter α on the behaviour of agents. The four different types of agents, from *normal* to *paranoid*, have the same preferences but value privacy differently. Anyone who has solved this level has experienced that paranoid agents are harder to identify. One can thus protect ones privacy better by acting more randomly.

Level 3: Crooks and Spooks

In the third level there are four shops and three groups of agents. The four shops are the walmart, drugstore, spy shop and the bank. The three groups are citizens, who are by all considered to be normal, the crooks, who are the unorganised criminals, and the spooks, who are the organised criminals. The utility values of each type of agent is given in the table below.

group	walmart	drugstore	spy shop	bank
citizens	0	1	1	2
spooks	1	1	2	1
crooks	1	1	2	1

The spooks and the crooks have the same preferences. The difference between those two groups is that the spooks know what the citizens do, whereas the crooks have no idea what normal is. Therefore, the crooks use a strategy that is as random as possible, whereas the spooks use a strategy that is as similar as the citizens as possible. The crooks can be said to be playing a most normal game, and the spooks a minimal information game.

In the next table one sees the strategies that the agents use in this simulation. The first column lists the type of an agent. The second lists the value of α that that agent uses. For each agent type, there is a strategy for a not-so paranoid agents ($\alpha = 1$) and for more paranoid agents ($\alpha = 1.5$). The remaining columns list the probability that each agent visits a location.

type	α	walmart	drugstore	spy shop	bank
citizens	1	0.072	0.196	0.196	0.534
citizens	1.5	0.115	0.224	0.224	0.436
spooks	1	0.054	0.146	0.399	0.399
spooks	1.5	0.06	0.165	0.322	0.45
crooks	1	0.174	0.174	0.475	0.174
crooks	1.5	0.202	0.202	0.393	0.202

For computing the strategy of agent types *citizens* and *crooks* we have used theorem 8.4.2 to compute the optimal strategies. For the *spooks* agents we have used theorem 8.5.2, where the strategy of the non-paranoid citizens has been used as the normal strategy. One can see in the table that for all three types of agents, the more paranoid agents choose a strategy with a higher entropy. They act more random. It is also clear that the spooks use a strategy that is more similar to the citizens strategy, and hence they are harder to distinguish from the citizens. For instance the crooks go often to walmart, but the other two types of agents do not. By determining the frequency of walmart visits, an observer can determine whether an agent is a crook or not.

In general, animated simulations such as this one can be used to demonstrate certain phenomena in a more convincing and entertaining way than calculations can. One can simulate much larger systems than one can solve by analytical means, and thus simulations can be of more realistic size than examples can. On the other hand, a proof-by-simulation lacks rigour. One can argue that simulations do not lead to scientific knowledge in a way that proof does.

This simulation has been programmed in Java, a language very suitable for interactive graphical programs. No specific agent systems library has been used. The source code is available on request.

8.8   Conclusion

Two new kinds of games have been defined. First of all, minimal information games, in which agents want to maximize the uncertainty that observers have over their next move. Secondly, most normal games, in which agents want to behave as similar as possible to an existing 'normal' agent, while maximizing their payoff. The definitions use the concepts entropy and relative entropy which are borrowed from information theory. In two theorems it is shown what the optimal best responses are in these games. These turn out to be unique in each situation, and to depend continuously on the payoff matrix and the opponent strategies. From this continuity one can derive that Nash equilibria exist in these games.

Minimal information games can be used to analyse situations with privacy-minded agents. If agents attach some value to privacy, the best strategy always gives them some privacy.

In most normal games, the situation is slightly more complicated. How well the non-normal agent X can do depends very much on the strategy that normal agents use. If the normal agents use a pure strategy, then X has no choice but to adopt the same strategy. The situation however becomes a lot better if the normal agents are privacy-minded. In this case they choose a high-entropy strategy, and this leaves the wanting-to-be-normal agent a lot of room to pursue its own agenda.

One can extend the work in these games in several ways. It would be in-

teresting to look at experimental data, to see whether most-normal or minimal-information strategies are used in the real world. Secondly, one could implement these strategies in order to obtain privacy. The question is then whether the soft approach to privacy is what users want.

On a theoretical side, it seems that these games give approximations to the Nash equilibrium with useful technical properties. Two of these properties are continuity of the best response function and the fact that best responses are always interior.

Chapter 9

Conclusion

For many people, verification of software sounds like watching paint dry: Apparently necessary, but quite dull compared to the creative process that came before it, and the creative uses that come after it. The average user of the verified software hardly learns anything from watching the process: either the program is fine, or a bug is found and fixed, after which the program is also fine.

This dissertation is intented to convince the reader that verification of multi-agent protocols is in fact very interesting. First of all because multi-agent protocols are widely used, sometimes at unexpected places. The debate about the proposed constitution for the European Union which took place in May 2005, is essentially a multi-agent protocol problem: what voting procedure should be used so that every country and person is represented fairly? Often one can capture requirements such as fairness in different ways, and deciding what is the best way is not a mere technical matter.

The second reason why multi-agent protocols are so interesting is that reasoning about multi-agent systems is complicated and can have surprising results. In software verification, the *state-space explosion* problem is often cited as the biggest obstacle: the systems to be verified often have a huge number of different states. Multi-agent protocols can have a small number of states, especially when these protocols have to be explained to and used by humans. On the other hand the requirements for these protocols can be subtle and difficult to interpret: in many cases, properties such as fairness can be hard to define and verify. Different logics based on extensive games have been presented in the previous chapters. Using three examples, a voting problem, the joint decision problem and the independent decision problem, we have shown that more complex logics can be used to identify subtle differences in protocols. These more expressive logics can have less favourable computational properties, making verification intractable. Thus, besides social arguments, there are also technical arguments in favour or against certain approaches.

An important distinction, that has been borrowed from game theory, is the

one between perfect information protocols and imperfect information protocols. In the first class of protocols all actions and all facts about the current state are public. Knowledge about these aspects thus does not play a role in these protocols. In imperfect information protocols knowledge is vitally important. Chapters 4 to 5 are focused on perfect information protocols, the final chapters 7 and 8 on imperfect information protocols.

9.1   Perfect Information Protocols

Modal logic is a very useful tool for studying perfect information protocols. It is easy to define logics that deal specifically with these protocols. The first logic presented, EFL, can be used to reason about which coalitions can enforce what kind of outcome. It can also be used in practice for verification of existing protocols. Unfortunately it is not very expressive: Many protocols that feel different satisfy the same EFL properties.

One can extend the language EFL in order to make more interesting properties. This leads to two meaningful extensions: EFLS and EFLN. The first language can express more complex reasoning involving side effects of adopting certain strategies: "Suppose I know that you want this, can I then do that?". The second logic, EFLN, can be used for expressing nested properties such as "I want to allow you to allow me to do this". For these three logics we have determined the computational complexity of model checking. A fourth language EFLNS that combines features from EFLN and EFLS has also been defined. This language is however is hard to interpret in a conservative way.

In chapter 6 a more explicit logic is used for reasoning about preferences. The language has been extended with operators reasoning about game trees, which makes it possible to use this logic for analysing game-theoretic reasoning in detail. As an example the concept of backward induction has been analysed in this logic.

These different logics illustrate that in order to understand a multi-agent protocol, one has to understand the background assumptions: what do agents know about each other and the situation. One always has a choice how to analyse a protocol. Even protocols with perfect information, that are often seen as the easiest case, can be difficult to compare.

9.2   Imperfect Information Protocols

It is well-known that knowledge and information are very important for agents, and it is also common to use game theory for analysing the interaction between agents with different interests. It is therefore a 'logical' next step to consider games about information . A knowledge condition game is a game between two groups of agents: one group wants to reach a certain knowledge situation, the

other group wants to stop the first group from reaching this situation. This situation that the groups want to reach or avoid is specified using ordinary epistemic logic. The fact that a well known logic is used makes knowledge condition games easier to understand, compared to logical languages with new operators. The fact that knowledge condition games only model the knowledge of the agents in the final situation is also an advantage: no temporal reasoning is necessary. Research in temporal logic has shown that reasoning about time is complex in itself, so it is not wise to make things even more difficult by mixing the aspects of time and strategies.

The complexity results for knowledge condition games indicate that games about knowledge can be intractable. They become tractable when monotone formulas are used. The complexity is thus caused by the fact that in epistemic logic, one can mix knowledge demands (Somebody knows something) and ignorance demands (Somebody does not know something).

It often makes sense to assume that agents are aware of the strategies that are used, for instance of strategies that are so often used that they become conventions, or when dealing with security protocols. One can also assume that agents do not know strategies. This has been defined as kcg'. This alternative definition makes decision problems slightly easier, and is thus a convenient assumption.

In a knowledge condition game where ignorance is demanded, the optimal strategy is often a random one. The coalition of agents that wants somebody to be ignorant should choose their actions in a random, unpredictable way. The fact that making random choices can be optimal has been known to game theorists before [11], but sometimes surprises people: flipping a coin is not often recommended for important decisions. The chapter on knowledge condition games does not tell what kind of coin one should use. It does not tell what exact probabilities one should use to choose between actions, because the logical approach does not work with explicit probabilities.

In order to be able to say something about those probabilities, chapter 8 introduces minimal information games. In these games agents have two goals: getting an optimal payoff by choosing the best actions, and randomizing their behaviour in such a way that an observer is kept ignorant about what the agent might do in the future. In order to measure 'ignorance', information theory is used. I have computed optimal strategies for these games, and these strategies give detailed information how one should randomize. The same is done for the related notion of *most normal games*. In those games, agents want to behave as similar to 'normal' as possible, but also getting the highest payoff. Thus, in this chapter the question about what coin one should use is solved.

Comparing those two approaches, one based on logic and one based on information theory, one can make two observations. On the one hand one can say that the logical approach is more general. Using epistemic logic one can express goals that mix knowledge and ignorance. The games based on information theory only deal with ignorance. In general one needs a logical approach in order to form

complex goals. On the other hand, a logical approach has the disadvantage that it is more abstract: important details, such as the exact probabilities, are often omitted.

9.3   Results

The next table shows the complexity results stated in this dissertation. The problems in the class PSPACE are definitely not tractable: no efficient algorithms for these problems exist. The same is probably true, in practical terms, of the problems in the class $\Sigma_2 P$: Even though $\Sigma_2 P$ problems are theoretically easier to solve, all problems in both classes are too hard to be solved in practice. The class NP contains problems that are also believed to be hard. No efficient algorithms for these problems are known, but sometimes one can find heuristics for those problems. The problems in the final class, P, are called tractable. They can be solved in reasonable time.

number	class	members
1	PSPACE	EFL model checking with linear representation
2		EFLN model checking
3	$\Sigma_2 P$	*kcg* decision problem with opponents
4	NP	*kcg* without opponents
5		*kcg'*
6	P	EFLmodel checking
7		EFLS model checking
8		*kcg* for monotone formulas

It is clear that analysing games is a complex affair: many of these problems are intractable. The intractability has different causes. Sometimes, in cases 3 and 4 for instance, the presence of opponents can make a problem much harder. In other cases, namely 1 and 2, the situation with one agent is already complex. This is a bit surprising.

In the chapter on logic it has been explained that theorem proving and model checking are both important techniques for multi-agent protocol verification. In this dissertation the following complete proof systems are presented.

description	logic	proof system
effectivity logic	EFL	$\mathcal{S}_{EFL}$
preference logic	$\mathcal{L}_P$	$\mathcal{S}_P$
alternative preference logic	$\mathcal{L}_P^2$	$\mathcal{S}_P^2$
finite tree logic	$\mathcal{L}_T$	$\mathcal{S}_T$

The first result, that there is a complete proof system for EFL, supports the hope that logical mechanism design is possible. The completeness proof sketches an

algorithm for constructing protocols. It would be interesting to implement and test this procedure in the future.

Preference logic $\mathcal{L}_P$ is a more natural language to express preferences than propositional logic. One can use preference logic to say things such as "coffee is better than tea", instead of the less informative "coffee is good" or "tea is bad". A proof system for this logic exists. This logic has been used for constructing a logic $\mathcal{L}_{sol}$, that can be used for characterising game-theoretic solution concepts. Interesting future work would be to use this preference logic in an update framework.

Bibliography

[1] A. Acquisti. The economics of privacy. http://www.heinz.cmu.edu/ acquisti/economics-privacy.htm, 2004. Internet page.

[2] D. Adams. *Mostly Harmless*. Harmony Books, NY, 1992.

[3] T. Agotness. A note on syntactic characterization of incomplete information in ATEL. In S. van Otterloo, P. McBurney, W. van der Hoek, and M. Wooldridge, editors, *Proceedings of the first Knowledge and Games Workshop*, pages 34–42. University of Liverpool, 2004.

[4] M. Aigner and G. Ziegler. *Proofs from the Book*. Springer-Verlag: Berlin, Germany, 1999.

[5] R. Alur, L. de Alfaro, T. A. Henzinger, S. C. Krishnan, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Taşiran. MOCHA user manual. University of Berkeley Report, 2000.

[6] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.

[7] R. Aumann and A Brandenburger. Epistemic conditions for a Nash equilibrium. *Econometrica*, 63:1161–1180, 1995.

[8] A. Baltag. A logic for suspicious players: Epistemic actions and belief-updates in games. *Bulletin of Economic Research*, 54:1–45, 2002.

[9] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge and private suspicions. Originally presented at TARK 98, accepted for publication in Annals of Pure and Applied Logic, 2002.

[10] M. Benerecetti and A. Cimatti. Symbolic model checking for multi agent systems. In *Proceedings of the ICLP'01 workshop on Computational Logic in Multi-Agent Systems*, 2001.

[11] K. Binmore. *Fun and Games: A Text on Game Theory*. D. C. Heath and Company: Lexington, MA, 1992.

[12] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press: Cambridge, England, 2001.

[13] P. Blackburn and W. Meyer-Viol. Linguistics, logic, and finite trees. *Logic Journal of the IGPL*, pages 3–29, 1994.

[14] G. Bonanno. Branching time, perfect information games and backward induction. *Games and Economic Behavior*, 36:57–73, 2001.

[15] G. Bonanno. Memory implies von neumann-morgenstern games. *Knowledge Rationality and Action*, to appear, 2004.

[16] G. Boolos. *The Logic of Provability*. Cambridge University Press, 1993.

[17] S. Brahms and D. Taylor. *Fair division: from cake cutting to dispute resolution*. Cambridge University Press, 1996.

[18] F. Brandt and T. Sandholm. (Im)possibility of unconditionally privacy-preserving auctions. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 810–817, New York, 2004.

[19] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[20] R. Chisholm and E. Rosa. On the logic of intrinsically better. *American Philosophical Quarterly*, 3:244–249, 1966.

[21] M. Chwe. *Rational Ritual: Culture, Coordination and Common Knowledge*. Princeton University Press: Princeton, NJ, 2001.

[22] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press: Cambridge, MA, 2000.

[23] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. *Lecture Notes in Computer Science*, 131:52–71, 1981.

[24] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)*, pages 262–273, 2002.

[25] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proceedings of the Uncertainty in Artificial Intelligence Conference (UAI), Edmonton, Canada.*, 2002.

[26] S. Cook. The P versus NP problem. *Manuscript prepared for the Clay Mathematics Institute for the Millennium Prize Problems*, 2000.

[27] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press: Cambridge, MA, 1990.

[28] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.

[29] R. Dash, N. Jennings, and D. Parkes. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, pages 40–47, November 2003.

[30] B. de Bruin. *Explaining games: On the Logic of Game Theoretic Explanations*. PhD thesis, University of Amsterdam, 2004. ILLC Dissertation Series 2004-03.

[31] F. Dechesne. *Game, set, maths: formal investigations into logic with imperfect information*. PhD thesis, Tilburg University and Technische Universiteit Eindhoven, 2005.

[32] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. The MIT Press: Cambridge, MA, 1995.

[33] M. Franzin, F. Rossi, E. Freuder, and R. Wallace. Multi-agent constraint scheduling with preferences: efficiency, solution quality and privacy loss. *Computational Intelligence*, 20:264–286, 2004.

[34] L.T.F. Gamut. *Logic, Language and Meaning: Volume 2*. University of Chicago Press, Chicago, 1991. L.T.F. Gamut is a collective pseudonym for Van Benthem, Groenendijk, De Jongh and Verkuyl; translated from Dutch.

[35] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

[36] A. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41(4), 1973.

[37] H. Gintis. *Game theory evolving*. Princeton University Press: Princeton, NJ, 2000.

[38] J. Glasner. eBay bidders sold on sniping, September 2002. http://www.wired.com/news/business/0,1367,55204,00.html.

[39] J. Goeree, S. Onderstal, E. Maasland, and J. Turner. How (not) to raise money. *Journal of Political Economy*, 2005. forthcoming.

[40] V. Goranko. Coalition games and alternating temporal logics. In *Proceedings of the 8th conference on Theoretical Aspects of Rationality and Knowledge(TARK)*, pages 259–272, 2001.

[41] V. Goranko and G. van Drimmelen. Complete axiomatization and decidability of the alternating-time temporal logic. 2004.

[42] J. Halpern. Defining relative likelyhood in partially-ordered preferential structures. *Journal of Artificial Intelligence Research*, pages 1–24, 1997.

[43] J. Halpern. *Reasoning about uncertainty*. The MIT Press: Cambridge, MA, 2003.

[44] S. Hansson. Preference logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic (Second Edition)*, volume 4, chapter 4, pages 319–393. Kluwer, 2001.

[45] B. Harrenstein, W. van der Hoek, J.-J. Ch. Meyer, and C. Witteveen. A modal characterization of Nash equilibrium. *Fundamenta Informaticae*, 57(2–4):281–321, 2003.

[46] J. Harsanyi. A general theory of rational behavior in game situations. *Econometrica*, 34:613–634, 1966.

[47] J. Harsanyi. Games with incomplete information played by bayesian players. *management science*, 14:159–182,320–334,486–502, 1967-1968.

[48] J. Hintikka. *Knowledge and Belief: an introduction to the logic of the two notions*. Cornell University Press: Ithaca, NY, 1962.

[49] J. Hintikka. *Principles of mathematics revisited*. Cambridge University Press: Cambridge, England, 1996.

[50] W. Hodges. Formal aspects of compositionality. *Journal of Logic, Language and Information*, pages 7–28, 2001.

[51] G. Holzmann. The model checker Spin. *IEEE Trans. on Software Engineering*, 23:279–295, May 1997.

[52] Z. Huang. *Logics for Agents with Bounded Rationality*. PhD thesis, University of Amsterdam, 1994. ILLC Dissertation Series 1994-10.

[53] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press: Cambridge, England, 2000.

[54] W. Jamroga and W. van der Hoek. Some remarks on alternating-time temporal epistemic logic. submitted, 2003.

[55] G. Jonker. Feasible strategies in alternating-time temporal epistemic logic, 2003. Universiteit Utrecht Master Thesis.

[56] M. Kacprzak, A. Lomuscio, and W. Penczek. Verification of multiagent systems via unbounded model checking. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AA-MAS)*, New York, July 2004.

[57] D. Koller and N. Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior 4*, 4:528–552, October 1992.

[58] D. Koller and A.J. Pfeffer. Generating and solving imperfect information games. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1185–1192, Montreal, August 1995.

[59] B. Kooi. *Knowledge, Chance, and Change.* PhD thesis, University of Groningen, Groningen, 2003. ILLC Dissertation Series 2003-01.

[60] B. Kooi. Probabilistic dynamic epistemic logic. *Journal of Logic, Language and Information*, 12:381–408, 2003.

[61] B. Kooi and J. van Benthem. Reduction axioms for epistemic actions. In *Proceedings of Advances in Modal Logic (AiML 2004)*, University of Manchester, September 2004.

[62] D. Kreps and R. Wilson. Sequential equilibria. *Econometrica*, 50:863–894, 1989.

[63] V. Krishna. *Auction theory.* Academic Press, San Diego, 2002.

[64] H. Kuhn. Extensive games and the problem of information. *Contributions to the Theory of Games*, II:193–216, 1953.

[65] D. Lewis. *Convention — A Philosophical Study.* Harvard University Press: Cambridge, MA, 1969.

[66] D. MacKenzie. *Mechanizing proof:computing, risk, and trust.* The MIT Press: Cambridge, MA, 2004.

[67] R. Maheswaran, J. Pearce, P. Varakantham, E. Bowring, and M. Tambe. Valuations of possible states (vps): A quantitative framework for analysis of privacy loss among collaborative personal assistant agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Utrecht, July 2005.

[68] J. Maynard Smith. *Evolution and the theory of games.* Cambridge University Press: Cambridge, England, 1982.

[69] K. McMillan. *Symbolic Model Checking.* PhD thesis, Carnegie Mellon University, 1992.

[70] J.-J. C. Meyer and R. J. Wieringa. Deontic logic: A concise overview. In *Deontic Logic in Computer Science*, volume 15 of *Wiley Professional Computing Series.* John Wiley & Sons, 1993.

[71] J.-J. Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science.* Cambridge University Press: Cambridge, England, 1995.

[72] R. Myerson. Refinements of the Nash equilibrium concept. *International Journal of Game Theory*, 7:73–80, 1978.

[73] J. Nash. Non-cooperative games. *Annals of mathematics*, 54:286–295, 1951.

[74] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour.* Princeton University Press: Princeton, NJ, 1944.

[75] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour.* Princeton University Press: Princeton, NJ, 3rd edition, 1953.

[76] A. Ockenfels and A. Roth. The timing of bids in internet auctions: Market design, bidder behavior, and artificial agents. *Artificial Intelligence Magazine*, 2005.

[77] A. Odlyzko. Privacy, economics, and price discrimination on the Internet. In *Fifth International Conference on Electronic Commerce(ICEC)*, pages 73–80, Pittsburgh, 2003.

[78] O. Ore. *Cardano, the gambling scholar.* Princeton University Press: Princeton, NJ, 1953.

[79] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press: Cambridge, MA, 1994.

[80] Oxford english dictionary (online edition), 2005. http://dictionary.oed.com/.

[81] C. Papadimitriou. *Computational Complexity.* Addison Wesley Longman, Reading, 1994.

[82] R. Parikh. Social software. *Synthese*, 132:187–211, 2002.

[83] D. Parkes. Classic mechanism design. In *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. Ph.D. dissertation, University of Pennsylvania, 2001.

[84] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001. ILLC Dissertation Series 2001-10.

[85] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12:149–166, 2002.

[86] N. Rescher. *Topics in philosophical logic*. D. Reidel publishing company, Dordrecht, 1968.

[87] M. Roberts, W. van der Hoek, and M. Wooldridge. Knowledge and social laws. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Utrecht, July 2005.

[88] T. Schelling. *The strategy of Conflict*. Harvard University Press, 1960.

[89] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[90] P. Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic (AiML'2002), Sep.-Oct. 2002, Toulouse, France.*, 2002.

[91] P.-Y. Schobbens. Alternating-time logic with imperfect recall. In W. van der Hoek, A. Lomuscio, E. de Vink, and M. Wooldrige, editors, *Electronic Notes in Theoretical Computer Science*, volume 85. Elsevier, 2004.

[92] H. Seely. The poetry of D.H. Rumsfeld, 2003. web column, posted April 2, 2003, http://slate.msn.com/id/2081042/.

[93] R. Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *International journal of game theory*, 4:25–55, 1975.

[94] M. Sevenster. A player semantic on IF semantic games. In *Logic, Games and Philosophy: foundational perspectives*, Logic, Epistemology and the unity of sciences, Prague, oct 2004.

[95] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948. http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.

[96] R. Smullyan. *First Order Logic*. Springer-Verlag: Berlin, Germany, Berlin, 1968.

[97] J. van Benthem. Logic and games. continuing electronic lecture notes, 1999–2004.

[98] J. van Benthem. Games in dynamic-epistemic logic. *Bulletin of Economic Research*, 53(4):219–248, 2001.

[99] J. van Benthem. Extensive games as process models. *Journal of Logic, Language and Information*, 11:289–313, 2002.

[100] D. van Dalen. Variants of rescher's semantics for preference logic and some completeness theorems. *Studia Logica*, 33:163–181, 1974.

[101] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In D. Bošnački and S. Leue, editors, *Model Checking Software, Proceedings of SPIN 2002 (LNCS Volume 2318)*, pages 95–111. Springer-Verlag: Berlin, Germany, 2002.

[102] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, pages 1167–1174, Bologna, Italy, 2002.

[103] W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(4):125–157, 2003.

[104] R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. under review, 2004.

[105] H. P. van Ditmarsch. *Knowledge Games*. PhD thesis, University of Groningen, Groningen, 2000.

[106] H. P. van Ditmarsch. The russian cards problem. *Studia Logica*, 75(4):31–62, 2003.

[107] S. van Otterloo. Reasoning about extensive games. *ESSLLI student session*, 2005.

[108] S. van Otterloo. The value of privacy: optimal strategies for privacy minded agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Utrecht, July 2005.

[109] S. van Otterloo and G. Jonker. On epistemic temporal strategic logic. In *Proceedings of the second workshop on logic and communication in Multi Agent Systems (LCMAS)*, Nancy, 2004.

[110] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Knowledge condition games. In S. Parsons and P. Gmytrasiewicz, editors, *Sixth Workshop on Game Theoretic and Decision Theoretic Agents*, New York, 2004.

[111] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Knowledge condition games. In S. van Otterloo, P. McBurney, W. van der Hoek, and M. Wooldridge, editors, *Proceedings of the first Knowledge and Games Workshop*. University of Liverpool, 2004.

[112] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Model checking a knowledge exchange scenario. *Applied Artificial Intelligence*, 18:937–952, 2004.

[113] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Preferences in game logics. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, New York, July 2004.

[114] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Knowledge condition games. *Journal of Logic, Language and Information (to appear)*, 2006.

[115] S. van Otterloo and M. Wooldridge. On the complexity of knowledge condition games. In *Proceedings of the second European workshop on Multi Agent Systems(EUMAS)*, Barcelona, 2004.

[116] R. van Rooij. Utility, informativity and protocols. *Journal of Philosophical Logic*, pages 389–419, 2004.

[117] G. Vergouw. *De Strafschop*. Reed business information, 2003.

[118] J. von Neumann. Zur theorie der gesellschaftspiele. *Mathematische Annalen*, pages 295–320, 1928.

[119] G. von Wright. *The logic of preference*. Edinburgh University Press, Edinburgh, 1963.

[120] J. Weibull. *Evolutionary Game Theory*. The MIT Press: Cambridge, MA, 1995.

[121] E. Weisstein. Compact set. http://mathworld.wolfram.com/CompactSet.html, 2004. from Mathworld - A Wolfram web resource.

[122] E. Weisstein. Trivium. http://mathworld.wolfram.com/Trivium.html, 2005. from Mathworld - A Wolfram web resource.

[123] T. Williamson. *Knowledge and its limits*. Oxford University Press, 2000.

[124] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 2, 1995.

Index

List of Symbols

193

Samenvatting

Multi-agent protocollen zijn collecties van regels die aangeven hoe meerdere partijen met elkaar in contact kunnen treden. Een veiling bijvoorbeeld heeft strikte regels die aangeven hoe er geboden kan worden. Ook de mogelijke zetten van een schaakpartij zijn vastgelegd in een collectie regels, en vormen dus een multi-agent protocol. Tenslotte zijn ook verkiezingen een voorbeeld van een multi-agent protocol. Deze activiteiten hebben gemeen dat ze in het echte leven, zonder ondersteuning van computers gedaan kunnen worden. Men kan zich echter ook voorstellen dat computerprogramma's meedoen aan veilingen en verkiezingen, misschien zelfs met of tegen menselijke spelers. Aangezien computerprogramma's nog niet zo intelligent zijn als wetenschappers soms wensen, is het vaak van belang dat protocollen aan bepaalde veiligsheidseisen voldoen. Men wil dus kunnen nagaan aan welke eigenschappen een protocol voldoet.

Het doel van mijn onderzoek is om methodes te ontwikkelen waarmee men multi-agent protocollen kan vergelijken en analyseren. Om dit te kunnen doen moet men een onderscheid maken tussen verschillende klassen protocollen, en ook verschillende soorten eigenschappen onderscheiden. Protocollen waarin iedere 'speler' geheel op de hoogte is van de huidige toestand (schaak bijvoorbeeld) worden behandeld in het eerste deel van dit proefschrift. Voor deze protocollen geldt dat de eigenschappen die te beschrijven zijn in de logische taal van hoofdstuk 4, efficiënt door een computer te verifiëren zijn. Ook kan men formeel redeneren over deze eigenschappen. Echter, ook voor deze relatief eenvoudige protocollen zijn er eigenschappen, die uitgedrukt kunnen worden in logische talen uit hoofdstuk 5, waarvoor automatische verificatie erg complex is. Over sommige ingewikkeldere eigenschappen kan men echter wel formeel redeneren met het bewijssysteem uit hoofdstuk 6.

Er zijn ook veel protocollen waarin niet alle spelers van alle details van de situatie op de hoogte zijn. Denk bijvoorbeeld aan spelen zoals Stratego of Poker. In deze protocollen is informatie over de huidige situatie, en kennis over wat andere spelers weten een belangrijke factor. In hoofdstuk 7 worden situaties

behandeld waarin het doel van bepaalde spelers is om bepaalde kennis juist wel of juist niet te hebben. De complexiteit van het analyseren van dit soort situaties kan hoog zijn, afhankelijk van welke aannames men maakt.

In al deze eerste hoofdstukken wordt kennis als een kwalitatieve eigenschap behandeld: als iets wat men wel of niet heeft. In het laatste hoodstuk gebruiken we kwantitatieve methoden uit de informatie-theorie, om de hoeveelheid informatie is bepaalde situaties te minimaliseren. Er worden spelen gedefinieerd waarin het de bedoeling van bepaalde spelers is om zo weinig mogelijk informatie bloot te geven, en voor deze spelers worden de optimale strategieën berekend. Een mogelijke toepassing van dit onderzoek ligt in de bescherming van privacy tegen privacy-schendende technologie.

Abstract

The research goal behind this dissertation is to develop ways to compare and analyse multi-agent protocols. In order to do so one has to distinguish different types of protocols, and one has to distinguish different classes of properties. Protocols that can be modelled as imperfect information game forms are therefore discussed in the first part of this dissertation, whereas protocols that can be modeled as imperfect information are the subject of the second part. In both parts we define concepts that help us to analyse and understand protocols, demonstrate these concepts on example protocols, and investigate the computational properties of these concepts.

In chapter 4, a logic for reasoning about what coalitions can achieve in protocols is presented. For this logic, a complete proof system is given, and the model checking complexity is determined.

In chapter 5, logics for reasoning about more complicated properties are presented. Specifically we compare the model checking complexity of logics for reasoning about side-effects and nested abilities.

In chapter 6, protocols are analysed using logics that deal with preferences explicitly. For two different variants of preference logics we give completeness proofs, and as an example, a characterisation of backward induction is given.

Protocols with imperfect information are the topic of the second part of this dissertation. In these protocols the knowledge that agents have plays a leading role. One can look at knowledge in a qualitative way, using epistemic logic, and this is done in chapter 7. In this chapter, it is shown how the computational complexity of protocol verification, depends on the presence of opponents, on whether strategies are known, and on the monotonic nature of the knowledge requirements. In chapter 8, it is shown that one can also model knowledge in a quantitative way. Using this approach, we compute optimal strategies for privacy preservation.

Curriculum Vitae

Sieuwert van Otterloo was born on 20 Maart 1979 in IJsselstein (Utrecht, the Netherlands). He graduated from the Christelijk Gymnasium in Utrecht in 1997, and went on to study at the Universiteit Utrecht. In 2001 he received the title of Doctorandus (Master of Science) in Mathematics Cum Laude, and the year after he also became Doctorandus (Master of Science) in Cognitive Artificial Intelligence.

In 2002 Sieuwert van Otterloo joined the Department of Computer Science at the University of Liverpool, where he worked as a PhD scholar. In the final year he also was a visiting researcher at the Institute for Logic, Language and Computation of the University of Amsterdam. He successfully defended this PhD dissertation in Liverpool in November 2005. At the same time he made a next step in his career, by joining McKinsey&Company in Amsterdam as a business consultant.

Titles in the ILLC Dissertation Series:

ILLC DS-2001-01: **Maria Aloni**
Quantification under Conceptual Covers

ILLC DS-2001-02: **Alexander van den Bosch**
Rationality in Discovery - a study of Logic, Cognition, Computation and Neuropharmacology

ILLC DS-2001-03: **Erik de Haas**
Logics For OO Information Systems: a Semantic Study of Object Orientation from a Categorial Substructural Perspective

ILLC DS-2001-04: **Rosalie Iemhoff**
Provability Logic and Admissible Rules

ILLC DS-2001-05: **Eva Hoogland**
Definability and Interpolation: Model-theoretic investigations

ILLC DS-2001-06: **Ronald de Wolf**
Quantum Computing and Communication Complexity

ILLC DS-2001-07: **Katsumi Sasaki**
Logics and Provability

ILLC DS-2001-08: **Allard Tamminga**
Belief Dynamics. (Epistemo)logical Investigations

ILLC DS-2001-09: **Gwen Kerdiles**
Saying It with Pictures: a Logical Landscape of Conceptual Graphs

ILLC DS-2001-10: **Marc Pauly**
Logic for Social Software

ILLC DS-2002-01: **Nikos Massios**
Decision-Theoretic Robotic Surveillance

ILLC DS-2002-02: **Marco Aiello**
Spatial Reasoning: Theory and Practice

ILLC DS-2002-03: **Yuri Engelhardt**
The Language of Graphics

ILLC DS-2002-04: **Willem Klaas van Dam**
On Quantum Computation Theory

ILLC DS-2002-05: **Rosella Gennari**
Mapping Inferences: Constraint Propagation and Diamond Satisfaction