Updating Epistemic Uncertainty

—

an essay in the logic of information change

Master's thesis

ILLC, University of Amsterdam

Ben Rodenhäuser

August 23, 2001

Contents

Acknowledgements

In preparing this thesis, I was lucky to have the support of three people who invested a considerable amount of their time during all stages of writing. I would like to thank all of them.

My supervisor Johan van Benthem read, and commented on, many earlier versions of the present paper; I am grateful for his numerous suggestions and inspirations concerning both the contents of the thesis and its organization; also, I would like to thank him for his patience with my constant changes of interest, and for his help in finding a clear focus in the end.

Yde Venema generously agreed to discuss several earlier versions of the thesis with me at the time Johan was in Stanford. His comments and careful criticism greatly improved the contents and the presentation of the thesis.

My most special thanks, however, go to Alexandru Baltag, for his uninterrupted support, and for the enormous amount of time he spent reading my drafts, and discussing them. His help in technical, conceptual, and personal matters was essential for me; without this help, I might not have finished the thesis at all.

I would also like to thank Peter van Emde Boas for his questions at my defense, and Annette Bleeker, Boudewijn de Bruin, Marie Nilsenová, and Marc Pauly for their help and friendship.

1 Introduction

This thesis is about the formal analysis of information change, a topic that has received steadily increasing attention over the past decade.[1] A key issue that has emerged in this line of research is the problem of *epistemic update*: Given an information state $\mathcal{S}$, representing the knowledge of a group of agents about each other and the facts of the world, and furthermore, given a representation $\mathcal{A}$ of an action that changes this information state in some particular way, how can we compute the information state resulting from applying $\mathcal{A}$ at $\mathcal{S}$, *using $\mathcal{S}$ and $\mathcal{A}$ only*? This calls for an *operation of epistemic update*, taking as input epistemic structures of some kind, and producing outputs of the same kind, that are intuitively plausible. Once we have found a particular solution to this problem, "linguistic" update problems pop up: how do assertions about the new information state formally depend on assertions about the old information state and about the action that lead from one to the other? For instance, what are necessary and sufficient conditions for agent i to know ϕ after action a has been executed? More generally, how does truth of statements before the update relate to statements afterwards; how much is preserved? and how much gets changed? Such questions can obviously only be posed in the context of a logical language devised to talk about information states, and information change, in some way; in particular, we need a *mixed language*, apt to analyze both epistemic and dynamic features of interaction.

A number of modal systems for the analysis of this type of questions has been proposed in recent years.[2] Probably the most general analysis in this area of research has been given by Baltag, Moss, and Solecki.[3] One of the leading ideas pursued by Baltag et al. is that in order to understand epistemic change, one has to study the epistemic properties of actions themselves. The objective is, to "take actions seriously" and consider them "as epistemic objects in their own right, in full generality" ([Bal00b]): actions come endowed with internal structure, most notably they are endowed with epistemic accessibility relations – in this sense the intuitions behind the treatment of actions are symmetric to the ideas underlying the usual modelling of uncertainty about states in a Kripke–style possible worlds setting.

[1]See, for instance, the monographs [G8̈8], [FHMV95], [vB96], [BS97]; for a survey, the reader may consult [MvBV97], which also references the extensive literature on dynamic semantics of natural languages.

[2]See the papers [Pla89], [GG97], [Ger99], [BMS99], [vD00], and the survey article [vB00b].

[3]cf. [BMS99], the revised version [BMS01], and the further developments in [Bal00b] and [Bal00a].

Upon reflection, this symmetry is entirely natural: just as the notion of an action is the dynamic counterpart of the notion of a state, the notion of learning is the dynamic counterpart to the static notion of knowledge – and since knowledge, or more generally, belief, is usually represented via a structured set of possible states, as an *epistemic state*, why not pair this with a notion of *epistemic action* to represent learning, or more generally beliefs about actions? The formal solution to the update problem [BMS99] then present, consists in an operation of conditional multiplication of two Kripke structures, a static one, representing uncertainty about states, and a dynamic one, representing uncertainty about actions. The authors use this operation, called *product update*, to give the semantics of an epistemic modal language with announcement operators of various types.

On a conceptual level, the work of Baltag et al. seems to be akin in spirit to a *two–level analysis* of statics and dynamics as advocated in recent work on logical dynamics, figuring prominently in [vB96] – "declarative statements and dynamic procedures", we read there, "both have strong motivations. There is no need to favor one over the other. Actual reasoning is a mixture of more dynamic sequential short–term processes and more static long–term ones, presumably over different representations. Thus, both kinds of systems must co–exist." Indeed, some of the developments in the quoted monograph indicate that a fruitful trend goes not only towards a mere co–existence of both levels, but in the direction of an *integration* of static and dynamic concerns in a unifying, two–sorted architecture, in which two domains interact via suitable "modes" and "projections". As one good candidate for such an integrated approach, [vB96] discusses *dynamic arrow logic* [4] Dynamic arrow logic combines a language geared towards analyzing arrows, or actions, with a boolean component intended for interpretation at states, resulting in a two–sorted language.

Our investigations in this thesis start from two observations. On the one hand, it is interesting to see that while the intuitions behind the work of Baltag et al. seem to indicate a *symmetry*, or at least close analogy between epistemics of states and actions, their logical implementation of the above sketched ideas is not that symmetrical: their system does not provide us with epistemic operators, paralleling the usual knowledge operators on states, and indeed, their language does not *talk* about actions at all in the sense that it does not have evaluation of formulas at the level of the latter.[5] Our claim

[4] See also the earlier papers [vB94], and [Mar96], which themselves build on earlier work on arrow logic in [Ven91], and [vB91]. [MPM96] is a collection of research papers on the subject.

[5] [BMS99] consider an "auxiliary language" interpreted on actions. However, this lan-

here is not that this would be a shortcoming of the systems considered by Baltag et al.; however, we do think that the initial idea of studying actions as epistemic objects, and the essentially symmetric way in which action uncertainty is conceived in the approach of [BMS99] on a semantic level, strongly suggest the explicit analysis of action epistemics via some suitable language including epistemic operators on actions; it seems thus promising to explore this direction as well.

Dynamic arrow logic, on the other hand, offers an abstract framework in which states and actions live on a par. It seems thus both feasible and promising, to explore the symmetric intuitions indicated above in an extension of the latter setting, and see in how far one can then recover the work of Baltag et al. For this purpose, we will introduce a fine–tuned, epistemic version of dynamic arrow logic, interpreted on general structures which we call update frames. We think that this approach is interesting for a number of reasons. First, the overall semantic setting we shall propose is somewhat different from the "dynamic" product semantics given by [BMS99]. This allows for possibly illuminating comparison, both on the level of semantic structures and on the level of the languages themselves, as we shall see in chapter 2.4, and chapter 5, respectively. Furthermore, the *DAL* framework is not tied to any particular notion of update; it is thus of potential use for other purposes as well, and the greater generality resulting from this may also serve to understand the specific assumptions embodied in the system of [BMS99] better. We also think that our framework enables the investigation of interesting questions concerning the product update operation, that cannot be addressed in this form otherwise. An example of this is the *entailment* problem discussed in the conclusions.

The thesis falls naturally into three parts. Chapters 2 discusses update from a purely semantic point of view. The next three chapters, chapter 3–5, discuss possible languages for the study of update. Chapter 6 studies what we believe to be one of the key problems associated with the notion of epistemic update, both from a purely semantic, and from a logical point of view.

In a bit more detail, we will proceed as follows. Chapter 2 introduces the update problem mentioned above more carefully. It furthermore introduces the ontology we shall use in the remainder of the thesis. Most importantly,

guage lives separate from the main framework presented by the authors. [Bal00b] analyzes actions with a dynamic logic style language; the dynamic modalities, if one wishes to say so, do "talk" about actions. But these modalities are terms, rather than formulas; thus the language of [Bal00b] does not include the classical epistemic operators on the level of actions.

we introduce update frames, and the operation of product update, and compare the two. The operation of product update lies at the heart of the work of [BMS01], which we shall discuss in chapter 3. The notion of an update frame will provide the basis for the epistemic version of dynamic arrow logic we shall introduce in chapter 4. We compare the two languages in chapter 5. In chapter 6, finally, we turn to one of the "logical update problems" mentioned above: which statements are preserved under update, in the sense that their truth before updating guarantees truth afterwards? Our findings are summarized in 7, which also contains some directions for further research.

2 Epistemic update

The plan of this chapter is as follows. We start with some examples that are meant to motivate the study of information change induced by actions. We then formulate the update problem, that naturally arises in this context. In section 2.2 and 2.3, we introduce the main semantic objects of this thesis: *update frames*, and the operation of *product update*. Update frames provide an abstract setting to study information change. They will form the semantic base into which we interpret the epistemic dynamic arrow language $\mathcal{L}_{EDA}$ in chapter 4. The operation of product update is a specific solution to the update problem; it is also the core notion for giving the semantics of the language $\mathcal{L}_{BMS}$, proposed by [BMS99], which we introduce in chapter 3. We discuss the relationship between product update and update frames in some detail in section 2.4. Finally, in preparation of the following chapters, we discuss possible languages to study epistemic update, and conclude with a quick overview of things to come.

2.1 The update problem

To introduce the main issues of this paper, we first discuss two examples highlighting the topic we want to address: Information change. Imagine the following situation[6]:

> *Nature has dealt me an envelope which might either contain an invitation to attend a lecture on epistemic logic, or an invitation to a lecture on probability theory (both of which are going to take place tonight). My friend Bob witnesses this situation. We both know that the envelope might have either of these two contents. Now I am asked to open the envelope, without Bob seeing the contents. I do this. Then I tell Bob what I have learned.*

This is a simple example. Nevertheless, there are some important intuitions about information flow attached to it. First, it is clear that upon opening the envelope I learn where I am going to spend the night, resulting in me knowing this very fact. Bob, on the other hand, does not obtain this information. However, he learns something else: He learns that I learned. As a result, he knows this. Furthermore, I know that he learns that I learned (and so on . . .). Next, after I told Bob he also knows where I am going to spend the night. And I know that he knows. (and so on . . .).

Let us now look at the following situation:

[6] This type of examples was introduced by Johan van Benthem.

Nature has dealt both me and Bob an envelope. Both contain a message to one of us by a fortune–teller. The message is either: "You are going to be happy tonight." Or: "You are going to be depressed tonight." We are both asked to open our respective envelopes without telling the other one what we learn. Now two things might happen:

1. *Both Bob and me open our envelopes.*
2. *I do not believe in fortune–tellers, but Bob does. I am not interested in the contents of my envelope (I won't believe it anyway), however I want to make fun of Bob. So I exchange the envelopes without Bob noticing it,* before *one of us has opened his envelope. Then both of us open an envelope: Bob opens the one he thinks is his. I open the one I know is Bob's.*

Here, the main new feature compared to the first example is the misleading nature of the second scenario. There, my move of exchanging the envelopes remains unnoticed by Bob, and results in Bob being deceived: after opening his envelope he acquires a false belief about the world, namely that the fortune–teller sent him the message that he will be happy tonight, while to the contrary she wanted to tell him that he will be depressed (or the other way round). The basic intuition remains the same: I exchange the envelopes, at the same time Bob thinks that nothing is happening at all, and, as a result, he believes that nothing happened. Bob opens my envelope, thinking that he opens his envelope, and as a result he believes that he knows what is in his envelope (while he actually knows – *de re* what was in mine).

There are two more general observations to be made here: The first is, that in both examples (and one might add countless others, cf. for instance [BMS01] and [vB00b]) there seems to be a *systematic relationship* between the information state *before* some action happening, and the information state *after* this very action. In the first example: not knowing what is in my envelope, opening it and reading the message in it, results in my knowing what is in the envelope. In the second example: Bob's not knowing what is in his envelope, my exchanging the envelopes, and his opening an envelope that he believes to be his and reading the message in it results in him believing that he knows what is in his envelope. The second observation is that in our example scenarios, the specific *epistemic appearance* of an action to an agent is essential for the resulting state. If Bob, for instance, would have suspected that I might be exchanging our envelopes at the moment I was doing just that, then his information state after my exchanging would have been different: namely, he would still consider the actual world (were I cheated on him) as one of the possibilities how things might be.

The first observation, that input state and output state are systematically related, leads to the study of epistemic update of a state with an action. The second observation, that actions can be, and often are, epistemic objects in their own right (cf. [Bal00b]), gives the framework we shall be concerned with its specific flavor. Namely, according to this observation, actions that induce changes should themselves come endowed with some epistemic appearance; they should carry epistemic structure.

What, then, is the update problem? It is just this: given an information state and an action that has itself some epistemic appearance, how should the information state resulting from "applying the action at the state" look like? The study of one particular solution to this problem, the *product update operation*, will form the main issue in this thesis.

We first introduce a general setting in which updates may be studied via what we call *update frames*. Then, we introduce the product update operation.

2.2 Update frames

The main semantic objects to be introduced here are *update frames*. They consist of three components: a state frame, an action frame, and an access map. State frames are formal representations of information states, action frames formally represent actions together with their epistemic appearance; formally, they will be the same kind of objects: Kripke frames. Access maps provide a link between states and actions on a dynamic level.

Epistemic frames We start with the notion of an *epistemic frame*. State frames and action frames will be special instances of epistemic frames. Epistemic frames are nothing else but multi–modal Kripke frames.

Definition 2.1 (Epistemic frames) An *epistemic frame* is a tuple $\mathcal{F} = (F, R_i)$, where F is a non–empty set, and $R_i \subseteq F \times F$ for each agent $i \in Ag$. $\dashv$

The relation R_i in the definition of an epistemic frame $\mathcal{F}$ gives the uncertainty of an agent concerning the set of objects in the domain of $\mathcal{F}$. This is just the standard way of modelling uncertainty in a possible worlds setting:

if the present world is w, then according to agent i any world he sees at w could be the actual world.

State frames and *action frames* are both the same kind of objects: they are epistemic frames; this symmetry reflects the general modelling strategy mentioned in the introduction. However, in order to keep track of the different role actions and states play, we have a special name and a special notation for both. State frames will be written as tuples $\mathcal{S} = (S, \rightarrow_i)$, while action frames will be represented as tuples $\mathcal{A} = (A, \twoheadrightarrow_i)$. In a state frame, for each agent i, the relation $\rightarrow_i$ gives the uncertainty of i about the states in S. In an action frame, for each agent i, the relation $\twoheadrightarrow_i$ tells us, for each action a in the domain, which actions i considers possible while a is happening.

State frames are the formal counterparts of information states, while action frames formalize the notion of an action with epistemic appearance. Note, however, that we do not assume epistemic frames to be pointed. Thus, we should say more accurately, that a state frame, together with a given state, which we then take to be the actual one, represents a particular information state; similarly for the case of actions.

As will have become clear by now, we are interested in the interaction between state frames and action frames. This has two dimensions: we want to know what new state results from applying an action to an old one, and eventually, we are also interested in what new uncertainty pattern results when we combine the epistemics of two inputs in some way. Update frames, as we shall see, do not give a uniform account of the epistemics; however, they fix the dynamic part. This is made precise in the concept of an *access map*.

Access maps The task of an access map is to tell us which actions are executable where, and what the result of applying some action at some state is, on a purely dynamic level.

Definition 2.2 (Access map) Let two non–empty sets S and A be given. An *access map* over S and A is a partial function $. : S \times A \to S$ that satisfies the following requirements:

1. The relation $T = \{(s, s') | s, s' \in S, \exists a \in A : s.a = s'\})$ is acyclic, i.e. for all $s \in S$, $(s, s) \notin T^+$.

2. For all $s, t \in S$: if $s.a = t.b$, then $s = t$ and $a = b$. $\dashv$

As we said, access maps provide the link between a state frame and an action frame. Thus, let us think of the set S as a set of *states*, and of the set A as a set of *actions*. The information conveyed in an access map, then, is the following: for any $s, s' \in S$, and $a \in A$, we read $s.a = s'$ as "state s accesses s' via a".

The conditions imposed in definition 2.2 both derive from similar motivations, related to the idea that we are *recording* both the passing of time, and the states and actions we encounter when travelling through time. Condition (1), acyclicity, basically can be understood as meaning that the "system" has a built-in clock, and that this clock never confuses the future with the past. Condition (2) says in words, that every state has a unique predecessor, and moreover, that for each state there is a unique action that immediately preceded it and lead to it. Taken together, this means that each state records the full sequence of past states and events; the "system" never confuses different pasts; time is backwards linear. Two things are worth noting: these conditions should not be understood epistemically; they express assumptions on our *modelling* of the world, rather than assumptions on the agents' *perception* of the world. And furthermore, our use of temporal vocabulary is meant purely motivational; we shall not encounter in the rest of this thesis any explicit modelling of time.

Update frames With the notions of state frame, action frame – both being instances of epistemic frames – and an access map we have all ingredients to define our "update setting".

Definition 2.3 (Update frame) An update frame is a triple $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$, where $\mathcal{S}$ is a state frame, $\mathcal{A}$ is an action frame, and . is an access map over S and A. $\dashv$

Update frames provide us with an abstract setting to study the notion of an epistemic update. They provide information about epistemics both on the level of states and on the level of actions, and, via the access map, they give a picture of sequential, dynamic interaction over time. However, update frames do not incorporate a specific account of information flow: the interaction between the epistemic relations on the two levels our setting deals with is left entirely unspecified. Giving a specific such account in our setting means restricting the class of update frames to those one considers "well–formed".

But even though the update frame view on update is not a uniform one, we observe that each update frame comes equipped with *its* particular, internal update notion.

Definition 2.4 ($\mathfrak{F}$–update) Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be an update frame. Then $\mathcal{S}.\mathcal{A}$, the $\mathfrak{F}$–update of $\mathcal{S}$ with $\mathcal{A}$ is the smallest subframe of $\mathcal{S}$ containing $\{s.a | s \in S, a \in A, s.a \text{ is defined}\}$. ⊣

Our remark on the lack of a specific account of information flow in the most general setting sketched so far, is reflected in the fact that the notion of $\mathfrak{F}$–update is not a "well–behaved" one in general – indeed, it is not even guaranteed that the epistemics depend on the dynamics only: a natural condition one might wish to impose on $\mathfrak{F}$–update is for instance that $S.A$ actually equals $\{s.a | s \in S, a \in A, s.a \text{ is defined}\}$ – the intuition being that an agent will not consider actions possible that were not executable at any of the states he considered possible previously. However, in general, it is not forbidden that entirely new possible states just "pop up" out of nowhere. Imposing this constraint, and others, means to consider a subclass of the class of all update frames, containing those update frames that cohere with our intuitions about information flow.

In this thesis, we shall be concerned with a particular notion of update, and its embedding in the present abstract setting: product update. We present it now and then turn to a comparison.

2.3 Product update

The notion of product update was first introduced in [BMS99]. It gives a particular answer to the question of how to obtain a new state frame $\mathcal{S}'$ from two given inputs, a state frame $\mathcal{S}$, and an action frame $\mathcal{A}$. In some sense, this might be seen as a way of lifting an access map to the level of epistemics; where the access map "updates" the dynamics only, an epistemic update operation takes care of the epistemics as well. However, there is one, apparently slight, but important difference. Namely, the product update operations is *external* in the sense, that it "creates" a *new* object out of two given ones; this is different from the *internal* operation of $\mathfrak{F}$–update we have seen in the last section, where updating consisted in making steps within the given update frame. Set–theoretic operations, however, and the product update operation is one of them, usually do not act over a pre–given state space. They just take an input (or an input pair, or countable many inputs) – and return something else. Thus, the concept of an access map is not the right one for an external operation of update. What we rather need, is the following:

Definition 2.5 Let S and A be non–empty sets. An *access condition* on S

and A is a relation $\mathcal{C} \subseteq S \times A$. $\dashv$

The way one should think of the sets S and A, of course, is as of the domain of some state and action frame; just as in the case of access maps. The difference between an access condition and an access map is just the one between the internal and the external perspective pointed at above – in update frames, the output of applying an action at a state needs to be specified in the access map, since it is part of the update frame already. As the output of the update operation will be an object formally distinct from the inputs, we do not specify it in advance of the operation itself.

Definition 2.6 (Product update operation) Let $\mathcal{S} = (S, \rightarrow_i)$ be a state frame, let $\mathcal{A} = (A, \twoheadrightarrow_i)$ be an action frame. Let $\mathcal{C}$ be an access condition on S and A. Then *the product update of $\mathcal{S}$ with $\mathcal{A}$ under $\mathcal{C}$* is defined as $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A} = (\mathcal{C}, \rightarrow_i)$, where for all $(s, a), (s', b) \in S \otimes_{\mathcal{C}} A$ we have that $(s, a) \rightarrow_i (s', b)$ iff $s \rightarrow_i s'$ and $a \twoheadrightarrow_i b$. $\dashv$

Observe that the reuse of the notation $\rightarrow_i$ in the output cannot give rise to confusion, since $S \otimes_{\mathcal{C}} A$ is *disjoint* from S.[7]

What is the idea of this particular update notion? Quoting [BMS99], it can be described as follows:

> *We model the update of a state by an action as a partial update operation, given by a restricted product of the two structures: the uncertainties present in the given state and in the given action are multiplied, while the "impossible" combinations of states and actions are eliminated (by testing the actions' preconditions on the state[8]). The underlying intuition is that the agent's uncertainties concerning the state and the ones concerning the action are mutually independent, except for the consistency of the action with the state.*

[7]Our presentation of the product update operation as a purely semantic operation on epistemic frames differs from the one in [BMS01] in that the authors represent what we call an access condition as a domain function from A to the powerset of S, giving for each action the set of states where it can be executed; both views are equivalent.

[8]We have not touched upon the issue of preconditions yet; preconditions, roughly speaking, will help to determine access conditions. For the present context, the passage in the quote may be understood as "while the combinations of states and actions that are not in the access condition are eliminated". We will formally introduce preconditions later, when discussing the semantics of [BMS99] in chapter 3

If one wants to roughly summarize the epistemic effects of product update in a slogan, a candidate might be the conjunction of "if there is confusion about the input states and the input actions, then there will be confusion about the output states" and "if there is confusion about the output states, then there has already been confusion about both the input states and the input actions". Thus, the update operation has two *directions*: the "downwards" direction determines, that (s, a) could be (s', b) if s could have been s' and a could have been b. Thus, if some action does not help you to distinguish between to situations you could not distinguish before, you will remain confused. The "upwards" direction says that if (s, a) could be (s', b), then s could have been a and s' could have been b. Thus, no confusion arises out of nowhere: current ignorance always has a source in previous ignorance.

2.4 Product update frames

How can we characterize the operation of product update in our update frame setting? What we obviously need first, is a "product condition" for update frames.

Definition 2.7 (Product update frame) Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be an update frame. Then $\mathfrak{F}$ is a *product update frame* iff

1. For all $i \in Ag$, for all $s, s' \in S, a, b \in A$ such that $s.a, s'.b$ is defined: If $s \rightarrow_i s'$ and $a \twoheadrightarrow_i b$, then $s.a \rightarrow_i s'.b$.
2. For all $i \in Ag$, for all $s \in S$, for all $a \in A$ such that $s.a$ is defined: If $s.a \rightarrow_i t$, then $t = s'.b$ for some $s' \in S, b \in A$ and $s \rightarrow_i s'$ and $a \twoheadrightarrow_i b$.

$\dashv$

The condition in definition 2.7 translates the relational conditions of the product update operation to the setting of update frames. It is evident that this condition establishes a match between update frames and product update frames on a very global level.

Theorem 2.8 Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be a product update frame. Let $\mathcal{S}.\mathcal{A}$ be the $\mathfrak{F}$–update of $\mathcal{S}$ with $\mathcal{A}$. Let $\mathcal{C} = \{(s, a) | s.a \text{ is defined}\}$. Then the following are equivalent:

1. $\mathfrak{F}$ is a product update frame.
2. The map $F : S.A \rightarrow \mathcal{C}$, defined by $F(t) = (s, a)$ iff $t = s.a$, and $s.a$ is defined, is an isomorphism from $\mathcal{S}.\mathcal{A}$ onto $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$.

Proof: Obvious by definition of product update frames. □

However, this is not all we would like to know. The level of updating considered in the result is extremely global. Intuitively, however, update frames represent *sequences* of updates. This raises the question how product update and update frame relate, once we focus on a more local, sequential, and iterative view on update. The remainder of this section is devoted to this issue. We first introduce a way of obtaining product update frames in a more constructive manner, by iterating the operation of product update starting from a given state frame. Next, we introduce the notion of a *synchronously generated* update frame. We then go on to compare these two notions.

Induced product update frames

We shall now introduce a canonical way of constructing a product update frame from a state frame, an action frame, and a collection of access conditions. This construction will be relevant both in the present section and in chapter 5, where we carry out a comparison between the two languages we will have introduced by then.

The construction proceeds as follows: we assume we have a state frame $(S, \rightarrow_i)$ and an action frame $(A, \twoheadrightarrow_i)$. We construct a domain of states over these, by iterating the cartesian product. This enables us to iterate the product update operation, starting from the given $\mathcal{S}$ and $\mathcal{A}$, using some access conditions $\mathcal{C}_\omega$ we assume as given as well. Collecting all these updated frames, we obtain a state frame $\mathcal{S}_\omega$, constructed according to the prescriptions of the update operation. We then take the union of all our access conditions and turn this union into an access *map*. As a result, we have constructed a product update frame, as we confirm in theorem 2.10 after the construction.

Construction 2.9

1. Let a state frame $\mathcal{S} = (S, \rightarrow_i)$ and an action frame $(A, \twoheadrightarrow_i)$ be given. Then we define, for each $n < \omega$,

$$D_0 := S$$

$$D_{n+1} := D_n \times A$$

$$D_\omega := \bigcup_{n<\omega} D_n$$

2. For $n < \omega$, let $\mathcal{B}_n \subseteq D_n \times A$ be given. Then we define

$$\mathcal{S}_0 := \mathcal{S}$$

$$\mathcal{S}_{n+1} := \mathcal{S}_n \otimes_{\mathcal{C}_n} \mathcal{A}$$

where $\mathcal{C}_n$ is the restriction of $\mathcal{B}_n$ to S_n. We let

$$\mathcal{S}_\omega := \bigcup_{n<\omega} \mathcal{S}_n$$

3. Finally, we put

$$\mathcal{C}_\omega := \bigcup_{n<\omega} \mathcal{C}_n$$

and

$$._\omega := id_{\mathcal{C}_\omega}$$

where $id_{\mathcal{C}_\omega}$ is the identity map on $\mathcal{C}_\omega$. ⊣

Theorem 2.10 Let $\mathcal{S}$ be a state frame, let $\mathcal{A}$ be an action frame. Let $\mathcal{S}_\omega$, and $._\omega$ be defined as in construction 2.9, using a given access condition $\mathcal{C}_\omega$. Then

$$(\mathcal{S}_\omega, \mathcal{A}, ._\omega)$$

is a product update frame, and $S_\omega \subseteq D_\omega$.

Proof: Clear by construction. □

In the remainder of this chapter we shall need a special case of this construction. We make use of the fact that any access map . gives rise to access conditions $\mathcal{C}_n$, for each n, in a natural way.

Construction 2.11 Let $(\mathcal{S}, \mathcal{A}, .)$ be an update frame; assume $G \subseteq S$. For each $n < \omega$, define

$$\mathcal{C}_n := \{((\ldots((s, a_1), a_2), \ldots), a_n) | s \in G, s.a_1.\ldots.a_n \text{ is defined}\}$$

and put

$$\mathcal{C}_\omega := \bigcup_{n<\omega} \mathcal{C}_n$$

⊣

Corollary 2.12 Let $(\mathcal{S}, \mathcal{A}, .)$ be an update frame, and let $G \subseteq S$. Let $\mathcal{S}_\omega$, and $._\omega$ be defined as in construction 2.9, using $(G, \rightarrow_i \restriction G)$, $\mathcal{A}$, and $\mathcal{C}_\omega$, where the latter is obtained as in construction 2.11. Then

$$(\mathcal{S}_\omega, \mathcal{A}, ._\omega)$$

is a product update frame.

Proof: Apply theorem 2.10. □

This motivates the following definition:

Definition 2.13 (Induced product update frame) Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be an update frame, and $G \subseteq S$. We call

$$(\mathcal{S}_\omega, \mathcal{A}, ._\omega)$$

obtained as in corollary 2.12, the *product update frame induced by* $\mathfrak{F}$ *and* G. Given $\mathfrak{F}$ and G, we write $\mathfrak{F}_\otimes^G$ for the product update frame induced by $\mathfrak{F}$ and G. ⊣

The preceding shows, that given an update frame, and some subset of its state space, we can always construct an induced product update frame, using this material alone. The notion of a product update frame induced by some $\mathfrak{F}$ and G, then, is what we were looking for: we know now how we can construct product update frames from local information, namely the given set G, in a uniform way. Clearly, such induced product update frames have some special properties, besides their being product update frames, which we need to account for if we want to match them up with update frames in general. In particular, they have a specific dynamic structure; namely, they have a *beginning*, and everything else comes afterwards. Note that this does not hold for update frames in general: they might be infinite towards the past. The purpose of the next section is to capture this feature more precisely, on the general level of update frames.

Synchronously generated update frames

We need two auxiliary notions. First, we define the notion of *precedence*.

Definition 2.14 Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be an update frame. For two states $s, s' \in S$, we say that s *precedes* s' in $\mathfrak{F}$, if there exists a sequence of actions $a_1 \dots a_n$, where $a_j \in A$ for $1 \leq j \leq n$, such that $s.a_1.\dots.a_n = s'$. We write $s \prec_{\mathfrak{F}} s'$ if s precedes s' in $\mathfrak{F}$. We also write $s \preceq_{\mathfrak{F}} s'$ if $s \prec_{\mathfrak{F}} s'$ or $s = s'$. ⊣

Note that the precedence order $\prec_{\mathfrak{F}}$ is a strict partial order on any update frame $\mathfrak{F}$; however, $\prec_{\mathfrak{F}}$ is not necessarily well–founded.

Definition 2.15 (Synchronicity) Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be an update frame. Let $G \subseteq S$. We say that G is *synchronous* in $\mathfrak{F}$, if for all $s, s' \in G$: $s \not\prec_{\mathfrak{F}} s'$. ⊣

Thus, synchronous sets consist of states that are mutually incomparable w.r.t. the precedence order. If we cut off their past, we can think of them as "being of the same time", thus the name. The following is the key definition.

Definition 2.16 (Synchronously generated update frames) Let $\mathfrak{F}' = (\mathcal{S}', \mathcal{A}, .)$ and $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be update frames. We say that $\mathfrak{F}$ *is dynamically generated by G within* $\mathfrak{F}'$, if $\mathcal{S}$ is a subframe of $\mathcal{S}'$, and S is the smallest set containing G that is closed under precedence in $\mathfrak{F}'$, i.e., for all $s \in S'$: if $s \in S$, and $s \prec_{\mathfrak{F}'} s'$, then $s' \in S$.

We say that $\mathfrak{F}$ is *synchronously generated by G within* $\mathfrak{F}'$, if $\mathfrak{F}$ is dynamically generated by G within $\mathfrak{F}'$, and G is synchronous in $\mathfrak{F}'$. ⊣

Update frames $\mathfrak{F}$ that are dynamically generated within another update frame $\mathfrak{F}'$ can be obtained by starting from some set and closing it off under the precedence order on $\mathfrak{F}'$. Synchronously generated update frames have the further property that their generating set is required to be synchronous; this makes for the following special property: synchronously generated update frames are those for which the order $\prec$ is *well–founded*; no state has an infinite past. We can now assign a *height* to each element of such an update frame. By an *immediate successor* of a state s we mean a state t such that $s.a = t$ for some action a.

Definition 2.17 (Height) Let $\mathfrak{F}$ and $\mathfrak{F}'$ be update frames. Assume $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ is synchronously generated by G within $\mathfrak{F}'$. Then we inductively define the *height* of elements of S. The height of elements of G is 0. The height of immediate successors of elements of height n is $n + 1$. ⊣

Note that this is well–defined, since (1) no two distinct elements of G precede each other, and (2) any element of $\mathfrak{F}$ has a unique immediate predecessor. Furthermore, any element of S is preceded by some $s' \in G$; thus every element of S has a unique height.

We can think of such an update frame $\mathfrak{F}$ as of a (possibly infinite) collection of *trees*, where uncertainty links across the trees may occur ("epistemic forests"). Important special cases of this are update frames $\mathfrak{F}$ that are synchronously generated within themselves, i.e. they are synchronously generated within $\mathfrak{F}' = \mathfrak{F}$. A yet more restricted special case is that of an update frame which is synchronously generated within itself and has a unique minimal element: a tree.[9]

[9]Note, however, that the tree picture is not fully accurate, in that there may be actions which are not executable at any state; "tree–like" would probably be a better word.

Dynamics and epistemics

We first observe that the notion of a synchronously generated update frame captures the *dynamics* of the construction of a product update frame above.

Definition 2.18 Let $\mathfrak{F}$ and $\mathfrak{F}'$ be update frames. Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be synchronously generated within $\mathfrak{F}'$ by G. Let $\mathfrak{F}^G_\otimes = (\mathcal{S}_\omega, \mathcal{A}, ._\omega)$ be the product update frame induced by $\mathfrak{F}$ and G. Then the map

$$F : S \to S_\omega$$

defined by

$$F(s.a_0.\ldots.a_n) = ((\ldots((s, a_0), a_1)\ldots), a_n) \text{ iff } s \in G, s.a_0.\ldots.a_n \text{ is defined}$$

is called *the natural map.* ⊣

Lemma 2.19 Let $\mathfrak{F}$ and $\mathfrak{F}'$ be update frames. Suppose $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$, $\mathfrak{F} = (\mathcal{S}', \mathcal{A}, .)$, and $\mathfrak{F}$ is dynamically generated by G within $\mathfrak{F}'$. Let $\mathfrak{F}^G_\otimes$ be the product update frame induced by $\mathfrak{F}$ and G. Then the following are equivalent:

1. $\mathfrak{F}$ is synchronously generated within $\mathfrak{F}'$ by G.
2. The natural map $F : S \to S_\omega$ is an isomorphism from $(S, A, .)$ onto $(S_\omega, A, ._\omega)$.

Proof: Obvious. □

In which circumstances is the "dynamic isomorphism" supplied by the natural map also an "epistemic isomorphism"? The answer lies in the notion of height. Observe that in an induced product update frame, the epistemic uncertainty relations are ordered according to the stages of the construction: it is never the case, that an agent is confused about states that are constructed at different stages. Intuitively, agents have clocks that never confuse the system time. This is what we need to capture.

Definition 2.20 (Epistemic height) Let $\mathfrak{F}$ and $\mathfrak{F}'$ be update frames. Assume $\mathfrak{F}$ is synchronously generated within $\mathfrak{F}'$ by G. Then $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ is an *update frame with epistemic height* if for all $s, s' \in S$: if $s \to_i s'$, then $h(s) = h(s')$. $\mathfrak{F}$ is an *update frame with initial epistemic height* if for all $s \in G, s' \in S$: if $s \to_i s'$, then $s' \in G$. ⊣

Note that the second part of the definition implies that G is closed under uncertainty, so to speak: any alternative to a state in s for some agent i is an element of G. The key observation for us, is that for synchronously generated *product* update frames, epistemic height and initial epistemic height collapse.

Lemma 2.21 Let $\mathfrak{F}$ and $\mathfrak{F}'$ be update frames. Assume $\mathfrak{F}$ is synchronously generated within $\mathfrak{F}'$ by G, and assume furthermore that $\mathfrak{F}$ is a product update frame. Then the following are equivalent:

1. $\mathfrak{F}$ is an update frame with epistemic height

2. $\mathfrak{F}$ is an update frame with initial epistemic height.

Proof: The non–trivial direction is from right to left. We show by induction on the height of the states in $\mathfrak{F}$ that $s \rightarrow_i s'$ implies $h(s) = h(s')$. For the states of height 0, the claim follows by assumption. The inductive step uses the fact that $\mathfrak{F}$ is a product update frame. Thus $\mathfrak{F}$ is an update frame with epistemic height. $\square$

Now we can prove the following:

Theorem 2.22 Let $\mathfrak{F}$ and $\mathfrak{F}'$ be update frames. Suppose $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$, $\mathfrak{F} = (\mathcal{S}', \mathcal{A}, .)$. Suppose that $\mathfrak{F}$ is dynamically generated by G within $\mathfrak{F}'$. Let $\mathfrak{F}^G_\otimes = (S_\omega, \mathcal{A}, ._\omega)$ be the product update frame induced by $\mathfrak{F}$ and G. Then the following are equivalent.

1. $\mathfrak{F}$ is synchronously generated by G within $\mathfrak{F}'$ by G, and $\mathfrak{F}$ is a product update frame with initial epistemic height.

2. The natural map $F : S \rightarrow S_\omega$ is an isomorphism from $\mathfrak{F}$ onto $\mathfrak{F}^G_\otimes$.

Proof:

($\Longrightarrow$) Suppose $\mathfrak{F}$ is a product update frame with initial epistemic height. F is an isomorphism w.r.t. the structures $(S, A, .)$ and $(S', A, ._\omega)$ by 2.19. So it is enough to show that F is a strong homomorphism w.r.t. the uncertainty relations on the state frames underlying $\mathfrak{F}$ and $\mathfrak{F}^G_\otimes$. By assumption, $\mathfrak{F}$ is an update frame with initial epistemic height, and by lemma 2.21, using that $\mathfrak{F}$ is a product update frame, it is an update frame with epistemic height. Hence for any n, the set of states S_n of height n in $\mathfrak{F}$ is epistemically closed, i.e. $s \in S_n$ and $s \rightarrow_i s'$ in $\mathfrak{F}$ implies $s' \in S_n$; it is now easy to see that for any n, the restriction of F to the states of height n is a strong homomorphism; the claim follows.

($\Longleftarrow$) From the assumption, using lemma 2.19, we infer that $\mathfrak{F}$ is synchronously generated within $\mathfrak{F}'$ by G. It remains to show that $\mathfrak{F}$ is a product update frame with initial epistemic height. Both follow easily from the assumption, since, by construction, $\mathfrak{F}^G_\otimes$ has these properties. $\square$

The theorem encompasses a number of interesting special cases. As a corollary, we have for instance, that for an update frame $\mathfrak{F}$, which is synchronously generated by G within *itself*, (1) and (2) above coincide. An instance of this are update frames that are synchronously generated (within themselves) by a singleton set. Such a tree–like update frame $\mathfrak{F}$ will be isomorphic to the product update frame induced by $\mathfrak{F}$ and its *root*, given that the agents know the root, while being at the root, and given that $\mathfrak{F}$ is a product update frame. This establishes an interesting connection to research pursued in game theory. In this field of research, trees with uncertainty relations on the level of states are used to represent games of imperfect information; the result makes formally explicit in what sense the product update operation differs from game theoretic research in the view on epistemics it provides: since one additionally assumes uncertainty relations on the level of actions, the whole game can be fixed by just specifying an initial information state and an action frame, representing possible actions that may be taken. This is also the view taken in the notion of a *rule–based* game in [Bal00a], where an analysis of games in a product update setting (i.e. without a given tree) is developed.

Allowing for a slightly more general format, we can consider update frames that are synchronously generated by an epistemically point–generated set; these are also characterized by the previous result in their relationship to the product update. Here, one still keeps the requirement that interaction starts at a certain time, but drops the assumption that there is a specific world, which is known to everyone at this time.

Thus, under the assumptions of theorem 2.22, update frames and product update as an operation coincide. One should, however, be aware of the intuitive differences between the two settings. First, product update provides an essentially *local* view on update: one zooms in on a particular epistemic situation, and some informational input, and looks what happens when we bring those two together. Contrary to this, update frames adopt a *global* perspective: intuitively, they provide us with information about whole streams of information and successive changes of information states. Related to this are the more *procedural* intuitions attached to product update. Essentially, we may think of it as an *algorithm* that allows us to compute a new epistemic structure from some given ones. The focus in update frames, on the other hand, is *descriptive*. Since all interactions are already given, we look at them from a bird's eye view. We may then notice *afterwards* that some particular update frame obeys specific laws governing information change.

2.5 Languages

What are possible languages to study updates? And what are possible semantics for them? In principle, any logical language whose semantic objects have the right similarity type for update models or epistemic models comes with its associated "logical update problems".[10] In this thesis, we shall concentrate on *modal* languages for the analysis of update. This suggests itself, given the modal roots of both epistemic and dynamic logic, and given the focus of previous work on this topic. However, there are many modal languages, and correspondingly there are many modal languages that may be useful to study update. We shall encounter three of them.

The simplest non–trivial language to analyze update is $\mathcal{L}_{ML}$, the basic multi–modal language, with an epistemic interpretation attached to it. $\mathcal{L}_{ML}$ is built over a set of proposition letters, using boolean operators, plus a set of knowledge operators, one for each agent i. Given that we add valuations both on state frames and action frames, we can interpret statements like $\mathcal{K}_i\phi$ on both of them – on the level of states, $\mathcal{K}_i\phi$ means that i knows that ϕ; on the level of actions we read it as i learns that ϕ.[11]

What is notable about a language like $\mathcal{L}_{ML}$ in the present context is that operations of update are beyond the reach of the language, which allows statements built from factual and epistemic information only. The language cannot analyze information change explicitly. This obviously calls for more elaborate systems, and this is why in most of the chapters of the thesis we will study *extensions* of the basic multi–modal language, two of them, to be precise. In chapter 6, however, when dealing with preservation questions, we shall consider a simplified setting; there, basic modal logic will be our system of choice.

The language $\mathcal{L}_{BMS}$ of [BMS99], which we discuss in chapter 3, extends $\mathcal{L}_{ML}$ with epistemic actions, used to formalize announcements of various kinds. Shortly, epistemic actions are action frames "plus something else".

[10] An update model is an update frame together with a valuation of a set of basic facts. Similarly, we obtain an epistemic model by adding a valuation to an epistemic frame.

[11] It should be pointed out, however, that learning that ϕ does not imply that ϕ is true in this thesis, neither does knowing that ϕ mean that ϕ is true! We shall not impose any constraints on the uncertainty relations in the sequel, and it is thus not necessary to adopt a more refined terminology here.

We shall see what this something else is. Such a language can, besides epistemics, also talk about dynamics. As we shall see, the use of Kripke structures as syntactic objects will be put to good use semantically. Namely, it opens up the possibility to build the notion of product update right into the semantics proper. Roughly speaking, as we will see, $[\alpha]\phi$ is true at a state in a state model $\mathcal{M}$ iff the product update of the pair $(\mathcal{M}, s)$ with α makes ϕ true. The system resulting from this semantics is very general; in particular, it can be used to analyze various types of *announcements* in communication – public ones, in which the contents of some message becomes common knowledge among all agents; but also more intricate communication acts, like "starting to believe that some secret communication took place while in fact nothing took place". In fact, any kind of information pattern that may be described by a Kripke frame, can be modelled in this system. [BMS99] also show, that previous proposals in this area can be treated as special cases of their logic.

Despite its merits, however, the way uncertainty about actions is treated in $\mathcal{L}_{BMS}$ may also seem somewhat odd. As pointed out in the introduction, one may also think of studying updates in a more symmetric, two–sorted fashion. This two–sortedness is, of course, already implicit in the work of Baltag et al. , however it is does not figure very explicit on the level of the language they use. The idea behind the language $\mathcal{L}_{EDA}$, which we shall introduce in chapter 4, is to fully implement this two–sorted nature of epistemic update. $\mathcal{L}_{EDA}$ will extend a language of dynamic arrow logic with epistemic operators both on the level of states, and on the level of actions. We will interpret it on update models, i.e. update frames with an additional valuation. To the best of our knowledge, such a language has not yet been studied in the literature.[12] In line with our earlier remarks on update frames, for a system of this type, the product update will not be part of the semantics proper; however, as we shall see, "product update behavior" may be enforced via frame constraints.

The further structure of the paper is given by the preceding. The language $\mathcal{L}_{BMS}$ will be introduced briefly in chapter 3, while $\mathcal{L}_{EDA}$ will be discussed in chapter 4. Chapter 5 is devoted to a comparison between the two. In chapter 6, we turn to one of the "logical update problems" mentioned in the introduction: preservation. We study it using the simpler language $\mathcal{L}_{ML}$.

[12] However, the possibility of considering a language similar to the one introduced here is indicated in [vB00a]. In the latter paper, the operation of product update is studied using propositional dynamic logic.

3 The system $\mathcal{L}_{BMS}$

In this chapter, we give a short introduction to the update semantics of [BMS99]. We will try to stick to the terminology developed in chapter 2 as far as this is possible.

3.1 Syntax

We start by defining the vocabulary of the language $\mathcal{L}_{BMS}$.

Definition 3.1 (Vocabulary) The vocabulary of $\mathcal{L}_{BMS}$ consists of a countable set of proposition letters P, a non–empty finite set of agents Ag, and a non–empty set of action labels L. The operators are $\wedge$, $\neg$, and a collection of operators $\mathcal{K}_i$, one for each $i \in Ag$. $\dashv$

The definition of the syntax proceeds by simultaneous induction. The objects to be defined are *formulas*, and *epistemic actions*. The following defines the set of *formulas*, calling definition 3.3 further down.

Definition 3.2 (Formulas) $\mathcal{L}_{BMS}$ is the smallest set satisfying

- If $\phi \in P$, then $\phi \in \mathcal{L}_{BMS}$.
- If ϕ, $\psi \in \mathcal{L}_{BMS}$, and α is an epistemic action over $\mathcal{L}_{BMS}$, then $\neg\phi$, $\phi \wedge \psi$, $[\alpha]\phi$, $\mathcal{K}_i\phi \in \mathcal{L}_{BMS}$. $\dashv$

Thus, besides the boolean operators, the language $\mathcal{L}_{BMS}$ has both epistemic modalities for interpretation at states and action modalities $[\alpha]$. $[\alpha]\phi$ will mean that executing α at the present state will result in a state that makes ϕ true. The distinguishing feature of the semantics is the way it makes formally precise the phrase "executing α at the present state". Before going into this, we need to define epistemic actions.

Definition 3.3 (Epistemic actions) An *action frame* $\mathcal{A}$ *over* $\mathcal{L}_{BMS}$ is an action frame where $A \subseteq L$. An *action structure* (over $\mathcal{L}_{BMS}$) is a pair $(\mathcal{A}, \textsc{pre})$, where $\mathcal{A}$ is an action frame over L, and $\textsc{pre} : A \to \mathcal{L}_{BMS}$. An *epistemic action* (over $\mathcal{L}_{BMS}$) is a triple $\alpha = (\mathcal{A}, \textsc{pre}, a)$, where $(\mathcal{A}, \textsc{pre})$ is an action structure, and $a \in A$. $\dashv$

Note that this definition refers back to 3.2. Indeed, as mentioned above, the idea is that actions and states are defined *simultaneously*: we can form sentences $[\alpha]\phi$ using actions α we have already formed previously; similarly, the precondition of an epistemic action α can only be a sentence formed before. We shall not go into any more detail than this, but refer the reader to the original paper [BMS99]. It is also of fundamental importance here to realize that epistemic actions are part of the syntax. Thus, they will appear in well–formed formulas, although one usually does not go so far as to make this really explicit in the notation. The usual meta–variables for epistemic actions will be α and β. Note that a given set of epistemic actions itself induces an action frame, by putting $\alpha \to_i \beta$ iff $a \to_i b$, and $\textsc{pre}(\alpha) = \textsc{pre}(a)$, for all epistemic actions α and β, where a is the distinguished token of α, and b is the distinguished token of β.

What is the idea behind epistemic actions? This is best understood by looking at the semantic motivations underlying their definition. *Action frames* are known already from chapter 2. They are meant to formally represent epistemic uncertainty concerning actions. *Precondition maps* will turn out to be a syntactic solution to the problem of uniformly generating an access condition for any given state model. The underlying idea is that the availability of an action at a state should depend on features of the state *itself*: thus, an action will be executable at a particular state if the state makes all its preconditions true. This will be the case for all elements of an epistemic actions (i.e. of the underlying action frame). The *designated action* in an epistemic action tells us of which of the actions in the action frame we should think as being the current, or actual one. Thus, epistemic actions model action uncertainty in a more fine–grained way than action frames do: they include information about the *domain of applicability* of each action in the action frame, and about which action in the action frame is the "actual one". This, at least, is the way one should think of epistemic actions, and it is also the way they will get used in the semantics. Since they are just pieces of syntax, they are *really* only symbols.

3.2 Semantics

The language will be interpreted on state models. We define a *state model* to be a pair $(\mathcal{S}, V)$, where $\mathcal{S}$ is a state frame, and $V : P \to \mathcal{P}(S)$, called *valuation*.

As with the syntax, the semantics will be defined by simultaneous induction. The important clause is, of course, the one involving epistemic actions. The clauses for proposition letters, boolean operators, and knowledge operators are standard. We give them now, and then focus on the product update and the interpretation of the action modalities.

Definition 3.4 Let an epistemic model $\mathcal{M} = (\mathcal{S}, V)$ be given.

$$\mathcal{M}, s \models p \text{ iff } s \in V(p)$$
$$\mathcal{M}, s \models \neg\phi \text{ iff } \mathcal{M}, s \not\models \phi$$
$$\mathcal{M}, s \models \phi \wedge \psi \text{ iff } \mathcal{M}, s \models \phi \text{ and } \mathcal{M}, s \models \psi$$
$$\mathcal{M}, s \models \mathcal{K}_i\phi \text{ iff } (\forall t \in S): \text{ if } s \rightarrow_i t \text{ then } \mathcal{M}, t \models \phi$$

The clause for the epistemic action modalities will be interpreted using the operation of product update. Informally, a state s in a state model $\mathcal{M} = (\mathcal{S}, V)$ makes $[\alpha]\phi$ true iff the product update of the state model with the epistemic action α makes ϕ true at the world (s, a), where a is the distinguished action in α, given that (s, a) is an element of the updated model. Note how the definition of truth requires information *beyond* the original model we started with here. To be able to define this formally, we need to know two things: what is the product update assumed to do with the valuation V on $\mathcal{S}$? And how is the access condition defined? As we shall see, the second question cannot be answered without giving the remaining clause of the truth definition at the same time (this is the announced simultaneous induction). The answer to the first question is in the following definition.

Definition 3.5 Let $\mathcal{M} = (\mathcal{S}, V)$ be a state model, and let $\mathcal{A}$ be an action frame. Let $\mathcal{C}$ be an access condition. Then the *product update of $\mathcal{M}$ with $\mathcal{A}$ under $\mathcal{C}$* is defined as $\mathcal{M} \otimes_{\mathcal{C}} \mathcal{A} := (\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}, V')$, where for all $(s, a) \in S \otimes_{\mathcal{C}} A$: $(s, a) \in V'(p)$ iff $s \in V(p)$. $\dashv$

This is obviously the simplest choice possible: the update just leaves the valuation unchanged. The underlying idea is to focus on the *purely epistemic* effects of an epistemic action. This is clearly not appropriate for some situations one might want to model. We note, however, that this type of update is *not* the only choice considered by Baltag, Moss, and Solecki (although they concentrate on this solution to a large extent; see, however, [Bal00a]). We chose to start with this case to keep things simple at this point. In chapter 5.4, we will investigate more complex ways of updating the truth of facts.

How do we obtain access conditions? One choice would be to just stipulate them in some arbitrary fashion. However, while this is arguably not unnatural in the setting of update frames, it would be a bit odd here. The solution of [BMS99], as hinted at above, is to use special access conditions, that are obtained from the *precondition map* given by the syntax, and the state model currently at stake: (s, a) will be in the access condition induced by some model and some epistemic action just in case s is in the domain of the model and makes the precondition of a true, which is an element of the epistemic action. This obviously involves the notion of truth, and this is why access conditions cannot be separated from presenting the last clause of the truth definition.

Definition 3.6 Let $\mathcal{M} = (\mathcal{S}, V)$ be a state model, and let $\alpha = (\mathcal{A}, \textsc{pre}, a)$ be an epistemic action. Then we define

1. $\mathcal{M}, s \models [\alpha]\phi$ iff if $(s,a) \in \mathcal{C}^{\mathcal{M}}_{\alpha}$, then $\mathcal{M} \otimes_{\mathcal{C}^{\mathcal{M}}_{\alpha}} \mathcal{A}, (s,a) \models \phi$
2. $\mathcal{C}^{\mathcal{M}}_{\alpha} = \{(s,a) | s \in S, a \in A, (\mathcal{M}, s) \models \textsc{pre}(a)\}$. ⊣

The first clause assumes that some access condition is given, and defines truth using it. The second clause defines the access condition as explained above, using the first clause.[13]

[BMS99] also consider a larger language, which is obtained by adding common knowledge operators $\mathcal{K}^*_{Gr}$ for groups of agents $Gr \subseteq Ag$ to the above syntax. The interpretation of these operators is given by taking $\mathcal{K}^*_{Gr}\phi$ to abbreviate the infinitary conjunction

$$\bigwedge_{\langle i_1,\ldots,i_n\rangle \in Gr^*} \mathcal{K}_{i_1} \ldots \mathcal{K}_{i_n}\phi$$

where Gr^* is the set of all finite sequences of elements of Gr. Since the empty sequence is such a finite sequence, $\mathcal{K}^*_{Gr}\phi$ logically implies ϕ.

We shall refer to the larger language, which includes the common knowledge operators, as $\mathcal{L}^*_{BMS}$. The authors present a sound and complete axiomatization for both logics, which are also both decidable. They furthermore show, that the language $\mathcal{L}_{BMS}$, which does not contain common knowledge operators, can be reduced to standard modal logic. The language $\mathcal{L}^*_{BMS}$, which does contain common knowledge operators, however, is strictly more expressive than modal logic with common knowledge operators, but without epistemic action modalities. Formally speaking, thus, the simpler system $\mathcal{L}_{BMS}$ is less interesting than its big brother $\mathcal{L}^*_{BMS}$. However, this does of course not diminish the philosophical and conceptual interest of the less expressive system $\mathcal{L}_{BMS}$.

This concludes our short introduction to $\mathcal{L}_{BMS}$ and $\mathcal{L}^*_{BMS}$. We will shall have more to say about it in chapter 5.

[13] This is similar to the mutual recursion of programs and formulas in dynamic logics like *PDL*.

4 The system $\mathcal{L}_{EDA}$

In this chapter, we define a logical language for the analysis of information flow and interpret it on update models. The language takes insights from a number of well–known modal systems, most prominently *dynamic arrow logic* (*DAL*). *DAL* is a two–sorted language; formulas are evaluated at both states and actions. We combine ideas from *DAL* with the modelling of [BMS99]: as in $\mathcal{L}_{BMS}$, we will have uncertainty relations both on states and on actions. However, unlike the latter language, the relations on actions are interpreted using epistemic operators. Furthermore, we consider additional operators taking state formulas to action formulas, enabling us to talk about preconditions and postconditions of actions in ways not available in standard *DAL*; these additional operators are in the spirit of a *logic of sufficiency*, that has been studied in a number of contexts (cf. the comments in section 4.3). Finally, using ideas from *hybrid logic*, we endow our system with a mechanism of *naming actions*.

The following two sections introduce the system formally. We then have some further remarks and observations on the set–up of the language in 4.3.

4.1 Syntax

Definition 4.1 (Vocabulary) The vocabulary of $\mathcal{L}_{EDA}$ consists of a countable set of proposition letters P, a finite, nonempty set of action labels L, and a finite, nonempty set of agents Ag. We call a triple (P, L, Ag) a *signature*. Our state operators are $\wedge$,$\neg$,$\mathcal{K}_i$ (for each agent i), and $[]$. Our action operators are $\wedge$, $\neg$, $\mathcal{K}_i$ (for each agent i), $\uparrow_1$, $\uparrow_2$, $\downarrow_1$, $\downarrow_2$. $\dashv$

We shall use the symbols $\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots$ for action labels. As announced above, these labels will designate semantic actions in a syntactic way. The distinction between the actions (usually denoted with $a, b, c, \ldots$) and the labels that denote them is important formally: while the former are *semantic objects*, the latter are *syntactic symbols*. However, the action labels will be used as *names* for actions: we assume, that each action label is true at exactly one state.[14] Furthermore, we will require that each action *has* a name, that is, some action label is true at each action.

Definition 4.2 (Syntax of $\mathcal{L}_{EDA}$) Let a signature (P, L, Ag) be given. The language $\mathcal{L}_{EDA} := \mathsf{SF} \cup \mathsf{AF}$ is the smallest set satisfying

- $P \subseteq \mathsf{SF}$, and

[14] This is just the usual requirement on valuations known from hybrid logic.

- if $\phi, \psi \in \mathsf{SF}$, $\alpha \in \mathsf{AF}$, and $i \in AG$, then $\neg\phi$, $\phi \wedge \psi$, $\mathcal{K}_i\phi$, $[\alpha]\phi \in \mathsf{SF}$.
- $L \subseteq \mathsf{AF}$, and
- if $\alpha, \beta \in \mathsf{AF}$, $\phi \in \mathsf{SF}$, and $i \in AG$, then $\neg\alpha$, $\alpha \wedge \beta$, $\mathcal{K}_i\alpha$, $\upharpoonright_1\phi$, $\upharpoonright_2\phi$, $\downharpoonleft_1\phi$, $\downharpoonleft_2\phi \in \mathsf{AF}$.

We call elements of SF *state formulas* and elements of AF *action formulas.* $\dashv$

The operators $\mathcal{K}_i$ will, by syntactic ambiguity, be interpreted using the uncertainty relations $\twoheadrightarrow_i$ on actions and the uncertainty relations $\rightarrow_i$ on states familiar from the preceding chapters.

$[]$ will be interpreted using the access map: $[\alpha]\phi$ will be true at a state s given that any action executable at s which makes α true leads to a state where ϕ hold.

The four operators $\upharpoonright_1$, $\upharpoonright_2$, $\downharpoonleft_1$, and $\downharpoonleft_2$, will also be interpreted using the access map. All of them are in some way or other related to the notions of *preconditions* and *postconditions.* $\upharpoonright_1\phi$ will be true at an action a if any state where a can be executed makes ϕ true. Hence $\upharpoonright_1$ talks about *necessary* conditions for an action to be executable. $\upharpoonright_2\phi$, on the other hand, will be true at an action a if a can be executed at any state that makes ϕ true. So $\upharpoonright_2$ can be used to say something about *sufficient* conditions for an action to be executable. $\downharpoonleft_1$ and $\downharpoonleft_2$ form a pair in precisely the same way: $\downharpoonleft_1\phi$ means "any state reachable with the current action makes ϕ true", while $\downharpoonleft_2\phi$ means that "any state that makes ϕ true can be reached with the current action".

Notation 4.3 We denote atomic formulas in P with $p, q, r, \ldots$, and, as announced, action labels in L with $\mathsf{a}, \mathsf{b}, \ldots$. We use $\phi, \psi, \chi, \ldots$ as meta–variables for state assertions, and $\alpha, \beta, \gamma, \ldots$ for action formulas.

We also agree on some abbreviations. $\top$ denotes $p \rightarrow p$, for some fixed proposition letter p; by abuse of notation, $\top$ is also taken to abbreviate $\mathsf{a} \vee \neg\mathsf{a}$, for some action label a. $\bot$ abbreviates $\neg\top$. We define $\overline{\mathcal{K}}_i$ as $\neg\mathcal{K}_i\neg$. For $j \in \{1, 2\}$, we define $\Uparrow_j$ as $\neg\upharpoonright_j\neg$, and $\Downarrow_j$ as $\neg\upharpoonright_j\neg$. Finally, we define $\langle\alpha\rangle$ as $\neg[\alpha]\neg$. $\dashv$

4.2 Semantics

The language will be interpreted on update models. Update models are pairs consisting of an update frame and a valuation. Update frames, in turn, as introduced in chapter 2, are triples $(\mathcal{S}, \mathcal{A}, .)$, where $\mathcal{S}$ is a state frame, $\mathcal{A}$ is an action frame, and . is an access map; we refer the reader to definition 2.3. We will always assume $\mathcal{A}$ to have a finite domain, when we use update frames in connection with the language $\mathcal{L}_{EDA}$.

Definition 4.4 Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be an update frame. A *valuation on* $\mathfrak{F}$ is a function $V = V_S \cup V_A$, where $V_S : P \to \mathcal{P}(S)$ and $V_A : L \to A$. We require V_A to be total and surjective. $\dashv$

An *update model* is defined to be a pair $(\mathfrak{F}, V)$, where $\mathfrak{F}$ is an update frame, and V is a valuation on $\mathfrak{F}$. Note that the surjectivity of V_A ensures that any action has a name.[15] We also write update models as $((\mathcal{S}, V_S), (\mathcal{A}, V_A), .)$, or even $((\mathcal{S}, V), (\mathcal{A}, V), .)$, when this is convenient. No confusion is likely to arise. A pair $(\mathcal{S}, V_S)$ is called a *state model*, as in the previous chapter. A pair $(\mathcal{A}, V_A)$ is called a *labelled action frame*.

To state the semantics succinctly, it is useful to make the operations used to interpret the "links" between the two sorts explicit. Given an update frame $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$, we define for all $s \in S$, and $a \in A$,

1. $D(a) = \{s | s.a \text{ is defined}\}$
2. $R(a) = \{t | \exists s : s.a = t\}$
3. $A(s) = \{a | s.a \text{ is defined}\}$

We call D the *domain function*, and R the *range function*. Note that the function A, for which we do no introduce a special name, is defined by abuse of notation. Observe also that these notions depend on $\mathfrak{F}$.

We can now define truth of a state formula at a state in an update model ($\models \subseteq \mathfrak{M} \times S \times \mathsf{SF}$) and truth of an action formula at an action in an update model ($\models \subseteq \mathfrak{M} \times A \times \mathsf{AF}$). For this purpose, let an update model $\mathfrak{M} = (\mathcal{S}, \mathcal{A}, ., V)$ be given.

[15]Observe also, that the requirement that action frames be finite in the context of $\mathcal{L}_{EDA}$ is forced upon us, since we have only finitely many action labels at our disposal.

Definition 4.5 (Truth)

$$
\begin{aligned}
\mathfrak{M}, s \models p &\text{ iff } s \in V(p)\\
\mathfrak{M}, s \models \neg\phi &\text{ iff } \mathfrak{M}, s \not\models \phi\\
\mathfrak{M}, s \models \phi \wedge \psi &\text{ iff } \mathfrak{M}, s \models \phi \text{ and } \mathfrak{M}, s \models \psi\\
\mathfrak{M}, s \models \mathcal{K}_i\phi &\text{ iff } (\forall t \in S) : \text{if } s \rightarrow_i t, \text{ then } t \models \phi\\
\mathfrak{M}, s \models [\alpha]\phi &\text{ iff } (\forall a \in A) : \text{if } a \in A(s) \text{ and } \mathfrak{M}, a \models \alpha, \text{ then } s.a \models \phi
\end{aligned}
$$

$$
\begin{aligned}
\mathfrak{M}, a \models \mathsf{a} &\text{ iff } V(\mathsf{a}) = a\\
\mathfrak{M}, a \models \neg\alpha &\text{ iff } \mathfrak{M}, a \not\models \alpha\\
\mathfrak{M}, a \models \alpha \wedge \beta &\text{ iff } \mathfrak{M}, a \models \alpha \text{ and } \mathfrak{M}, a \models \beta\\
\mathfrak{M}, a \models \mathcal{K}_i\alpha &\text{ iff } (\forall b \in A): \text{ if } a \twoheadrightarrow_i b, \text{ then } b \models \alpha\\
\mathfrak{M}, a \models \uparrow_1\phi &\text{ iff } (\forall s \in S): \text{ If } s \in D(a) \text{ then } \mathfrak{M}, s \models \phi\\
\mathfrak{M}, a \models \uparrow_2\phi &\text{ iff } (\forall s \in S): \text{ If } \mathfrak{M}, s \models \phi \text{ then } s \in D(a)\\
\mathfrak{M}, a \models \downarrow_1\phi &\text{ iff } (\forall s \in S): \text{ If } s \in R(a) \text{ then } \mathfrak{M}, s \models \phi\\
\mathfrak{M}, a \models \downarrow_2\phi &\text{ iff } (\forall s \in S): \text{ If } \mathfrak{M}, s \models \phi \text{ then } s \in R(a)
\end{aligned}
$$

As usual, we can now also define global truth and validity. For a given update model $\mathfrak{M} = (\mathfrak{F}, V)$, and formulas $\phi \in \mathsf{SF}$, and $\alpha \in \mathsf{AF}$, we say that ϕ *is globally true in* $\mathfrak{M}$ (notation: $\mathfrak{M} \models \phi$) iff for all $s \in S$: $\mathfrak{M}, s \models \phi$. We say that α *is globally true in* $\mathfrak{M}$ (notation: $\mathfrak{M} \models \alpha$) iff for all $a \in A$: $\mathfrak{M}, a \models \alpha$. Furthermore, we say that ϕ *is valid in* $\mathfrak{F}$ (notation: $\mathfrak{F} \models \phi$) iff for *all* valuations V: $(\mathfrak{F}, V) \models \phi$. We say that α *is valid in* $\mathfrak{F}$ (notation: $\mathfrak{F} \models \alpha$) iff for *all* valuations V: $(\mathfrak{F}, V) \models \alpha$.

We shall also consider a stronger language, which we will refer to as $\mathcal{L}^*_{EDA}$, which is obtained by adding common knowledge operators $\mathcal{K}^*_{Gr}$ on states and on actions. Given the symmetry of the language, it is natural for us to consider common knowledge operators not in isolation, for one sort, but uniformly for both. Although on the level of actions it might be more accurate to speak of *common learning* than of common knowledge, we shall not be insistent on this point. Let us now give the interpretation of these infinitary operators. It is completely analogous as in the last chapter.

$$\mathfrak{M}, s \models \mathcal{K}^*_{Gr}\phi \text{ iff } \mathfrak{M}, s \models \bigwedge_{\langle i_1,\ldots,i_n\rangle \in Gr^*} \mathcal{K}_{i_1} \ldots \mathcal{K}_{i_n}\phi$$

where, as in the previous chapter, Gr^* is the set of all finite sequences of elements of $Gr \subseteq Ag$. The interpretation on actions runs exactly parallel.

4.3 Discussion

The main inspiration for the basic formal setting of $\mathcal{L}_{EDA}$ comes from dynamic arrow logic, *DAL*, as introduced in [Mar96, vB94]. *DAL* builds on two major ideas; the first of these is related to the relativization approach to relational algebra (cf. [Mar99]): one assumes a set of abstract objects as given that need not be representable as the full cartesian product over some underlying set of objects. It has been shown that this move leads to a *decrease* in complexity – the undecidability of full relation algebra disappears (see, for instance, [MPM96]). The second main feature of *DAL* is the perspective on actions and states one takes: both kinds of objects are treated essentially on a par and may interact in various ways. This obviously meshes well with our general purpose of analyzing updates, with the setting developed in chapter 2, and also with the *symmetric intuitions* about actions and states we put forward in the introduction to this thesis. Let us now say something about the additional features that distinguish the present language from the basic *DAL* language.

First, a comment on the features we choose to *omit* here. In arrow logic, one usually considers the additional operations of *reversal* and *composition* on the level of actions, which we shall not encounter in the main body of this thesis. This is mainly for reasons of simplicity. More comments can be found in the conclusions of the thesis.

Two–level epistemics As to a first difference: the present language is an *epistemic version* of dynamic arrow logic. Languages which combine dynamic and epistemic features are well–known in the literature. The distinguishing feature here is that we add epistemic operators both on states and actions, or arrows. Here, the influence of the modelling of Baltag et al. shows; we have discussed its motivation in earlier chapters. An equal treatment of states and actions on an epistemic level seems also to be very much in the spirit of dynamic arrow logic itself, as it started from the idea of treating actions and states equally.

Action labels Furthermore, our treatment of action labels is different from standard *DAL*. Here, we use an adaptation of a technique known from hybrid logic, which basically consists in requiring that the denotation of each action label in L is a singleton set, and that furthermore every action is in the denotation of some action label. The idea here is that we think of the *semantic actions* in an action frame as corresponding to generic types of actions. For instance, we can think of some action a as being the action of "taking a card". Now our primary interest is in properties actions have by virtue of their being of a specific type. This was very obvious in the last chapter, when we discussed the system $\mathcal{L}_{BMS}$: in this language, the uncertainty about an action and its preconditions are taken to depend on its type; both components are made syntactically explicit for easy reference. A similar thing is happening here. However, we want less: we do not incorporate epistemic and precondition features of an action into the syntax directly; still, we want to be able to refer to actions by syntactic means. To be able to do this, we need some syntactic mechanism. This mechanism is provided by the action labels, which give names to actions. Note that we do not exclude that an action "falls under two different descriptions": it is possible to single out actions differently, i.e. $a \models \mathsf{a} \wedge \mathsf{b}$ is not contradictory. Further features of actions, including their epistemic appearance, and their preconditions, may be *described* in our language using the additional operators discussed above.

Fact 4.6 Let $\mathfrak{F} = (\mathcal{S}, \mathcal{A}, .)$ be an update frame. Then $\mathfrak{F}, a \models \overline{\mathcal{K}}_i \mathsf{b}$ iff there exists $b \in A$, such that $b \in V(\mathsf{b})$, and $a \twoheadrightarrow_i b$. □

Note that this may be seen as a formal counterpart of natural language expressions like "when he took the knife, I thought it might be that he actually took the fork". Here, "taking the knife" is used as a way to single out some real action; the formal correspondent of such a definite description would be an action label a, used to designate some action $a \in A$; similar remarks apply to b, b, and the description "he took the fork". The difference between $\mathcal{K}_i \mathsf{b}$, and "I thought it might be that he took the fork", is that the former rather corresponds to "it might be that he actually takes the fork", representing an epistemic statement of an agent *while* an action is happening. But this is only what we wanted to have: a way of ascribing epistemic properties to agents while actions are happening.

Pre-/Postconditions As a last addition, we have enriched the dynamic arrow logic repertoire somewhat, by considering further operators taking state formulas to action formulas, namely the operators $\uparrow_2$ and $\downarrow_2$. These accompany the operators $\uparrow_1$ and $\downarrow_1$, which are standard and well–known from the literature: $\uparrow_1$ and $\downarrow_1$ talk about *necessity*; $\uparrow_2$ and $\downarrow_2$, on the other hand, talk about *sufficiency*.[16] The interpretation of the duals $\Uparrow_2$ and $\Downarrow_2$

[16]Sufficiency operators of this kind have been studied in various contexts, in an attempt to overcome the modal bias towards necessity. The state modality usually considered in this case, is the *window operator* $\boxplus$. $\boxplus\phi$ then says that all states where ϕ is true are

runs as follows:

$$\mathfrak{M}, a \models \Uparrow_2\phi \text{ iff } (\exists s \in S) : s \notin D(a) \text{ and } \mathfrak{M}, s \not\models \phi$$

$$\mathfrak{M}, a \models \Downarrow_2\phi \text{ iff } (\exists s \in S) : s \notin R(a) \text{ and } \mathfrak{M}, s \not\models \phi$$

The most important feature about the sufficiency operators is, that we can use them to define *iff* operators:[17]

$$\uparrow\phi := \uparrow_1\phi \wedge \uparrow_2\phi$$

$$\downarrow\phi := \downarrow_1\phi \wedge \downarrow_2\phi$$

We have that $a \models \uparrow\phi$ iff $D(a) = \{s \in S | s \models \phi\}$, and similarly $a \models \downarrow\phi$ iff $R(a) = \{s \in S | s \models \phi\}$. We shall refer to $\uparrow$ as the *precondition operator*, and to $\downarrow$ as the *postcondition operator*. Thus, the notion of precondition and postcondition we adopt here is the following: ϕ is a precondition (respectively, postcondition) for some action a, if truth of ϕ at a state s is necessary and sufficient for s to lie in the domain (respectively, range) of a.

An immediate questions is which models are *precondition–based* in the sense that for each each action a, there exists a formula ϕ such that $\mathfrak{M}, a \models \uparrow\phi$.[18]. A first answer is trivial: namely, for an action label a,

$$\mathsf{a} \rightarrow \uparrow\langle\mathsf{a}\rangle\top$$

is valid in all update models, and since any action has a name, we have that for any action a, a precondition of a is given as $\langle V(a)\rangle\top$, that is, $\uparrow\langle V(a)\rangle\top$ is true at a. This, however, is not particularly interesting, since it says nothing more than that for any action a, $D(a) = \{s \in S | a \in A(s)\}$. What we we would like to see are *interesting preconditions*, that tell us something illuminating about the properties of states where an action is executable. What are interesting preconditions? To give a particular answer, let us call a state formula *pure* if it does not contain occurrences of []. A well–known result in modal logic says that any state s in a finite epistemic model $\mathcal{M}$ can be characterized up to epistemic bisimiliarity by a formula ϕ of the basic modal language with common knowledge operators, in the sense that $\mathcal{N}, t \models \phi$ iff s and t are epistemically bisimilar, for all models $\mathcal{N}$ and states

reachable by a transition from the current state. The reader may consult [GPT87] for details.

[17]We can also derive further interesting operators; for instance, a "cross–sort" version of the existential modality is given by $\Uparrow_2\neg\phi \vee \Uparrow_1\phi$, which is true at an arbitrary action iff there exists a state s such that $s \models \phi$.

[18]An analogous question, of course, can be raised for the case of postconditions. We focus on the case relevant for the following

t in $\mathcal{N}$. Furthermore, epistemic bisimilarity and modal equivalence coincide in the case of finite epistemic models.[19] Now the basic modal language with common knowledge operators is just the set of pure state formulas. To derive a statement about preconditions from this, we need to make the assumption that the availability of an action in an update model is *dependent* on properties of states that are describable by means of pure state formulas. Given an update model $\mathfrak{M}$, we say that the domain of a is closed under epistemic equivalence, if, for all $t \in S$, if (1) $s \in D(a)$, and (2) $s \models \phi$ iff $t \models \phi$ for all pure state formulas ϕ, then $t \in D(a)$. We then have the following:

Fact 4.7 Let $\mathfrak{M} = (\mathcal{S}, \mathcal{A}, ., V)$ be a finite update model, and $a \in A$. Suppose the domain of a is closed under epistemic equivalence. Then there exists a pure state formula $\phi \in \mathcal{L}^*_{EDA}$, such that $\mathfrak{M}, a \models \uparrow\phi$. $\square$

A question which we leave open here is whether this analysis can be extended to the case where we allow arbitrary *label–free* state formulas.

Some validities

We conclude this chapter by discussing some basic validities of the languages $\mathcal{L}_{EDA}$ and $\mathcal{L}^*_{EDA}$. We shall, however, not go into issues of axiomatization proper.

Principles 4.8

1. $\bigvee_{\mathsf{a} \in L} \mathsf{a}$
2. $\bigwedge_{\mathsf{a} \in L}(\mathsf{a} \rightarrow \uparrow\langle \mathsf{a} \rangle \top)$
3. $\bigwedge_{\mathsf{a} \in L}(\langle \mathsf{a} \rangle p \rightarrow [\mathsf{a}]p)$
4. $[\alpha]\phi \wedge [\beta]\phi \rightarrow [\alpha \wedge \beta]\phi$
5. $\langle \uparrow_1 \phi \rangle \top \rightarrow \phi$

The first principle reflects the finiteness of the label set and our "hybdridness" assumption on valuations. The second principle was discussed above. The third principle corresponds to the fact that the access map is a function: given a state and an action, the output is determined, if an output exists. The next two principles are self–explaining. We also get the usual distribution principles for the operators $[]$, $\uparrow_1$, $\downarrow_1$, and $\mathcal{K}_i$.

[19] Details on epistemic bisimilarity may be found in section 5.2, where we shall also re–encounter versions of the mentioned results.

Principles 4.9

1. $[\alpha]\phi \wedge [\alpha]\psi \leftrightarrow [\alpha](\phi \wedge \psi)$

2. $\mathcal{K}_i\phi \wedge \mathcal{K}_i\psi \leftrightarrow \mathcal{K}_i(\phi \wedge \psi)$

3. $\mathcal{K}_i\alpha \wedge \mathcal{K}_i\beta \leftrightarrow \mathcal{K}_i(\alpha \wedge \beta)$

4. $\upharpoonleft_1\phi \wedge \upharpoonleft_1\psi \leftrightarrow \upharpoonleft_1(\phi \wedge \psi)$

5. $\downharpoonleft_1\phi \wedge \downharpoonleft_1\psi \leftrightarrow \downharpoonleft_1(\phi \wedge \psi)$

The sufficiency operators $\upharpoonleft_2$ and $\downharpoonleft_2$ are not normal modal operators. However, they validate, a special principle, which is, for the case of $\upharpoonleft_2$:

Principle 4.10 $\upharpoonleft_2\phi \wedge \upharpoonleft_2(\neg\phi \wedge \psi) \rightarrow \upharpoonleft_2\psi$

The analogous principle holds for $\downharpoonleft_2$. The two operators also validate a rule which is similar in its structure to the modal rule of necessitation: if ϕ is valid in an update frame, then we can conclude that $\upharpoonleft_2\neg\phi$ is also valid: since ϕ is valid, trivially any action can be executed at all $\neg\phi$–states, since there are none.

Considering the larger language $\mathcal{L}^*_{EDA}$, we also get the usual principles for the common knowledge operators. We consider only the case of common knowledge on states, since matters are completely analogous on actions.

Principles 4.11

1. $\mathcal{K}^*_{Gr}\phi \rightarrow \phi \wedge \bigwedge_{i \in Gr} \mathcal{K}_i\mathcal{K}^*_{Gr}\phi$

2. $\mathcal{K}^*_{Gr}\phi \wedge \mathcal{K}^*_{Gr}\psi \leftrightarrow \mathcal{K}^*_{Gr}(\phi \wedge \psi)$

Note that we have not made any specific assumptions on the epistemic frames that form part of update frames. Thus, it comes as no surprise, that, apart from the distribution laws, none of the above principles mentions knowledge operators. Indeed, all but the first principle in 4.8 describe the behavior of the *access map.*

5 Comparison

In chapter 2, an ontology for the study of update was introduced; we have seen two ways of approaching the subject, the first given by the notion of an update frame, the second given by the operation of product update. Chapter 2.4 gave an analysis of the relationship between these two approaches. The present chapter parallels section 2.4, in that we compare here the two languages, $\mathcal{L}_{BMS}$, and $\mathcal{L}_{EDA}$; each of them was designed to fit one of the two approaches, and thus a more detailed comparison on a linguistic level suggests itself.

We carry out our comparison along three dimensions. In section 5.1 right below, we show that we can give a syntactic condition on update frames, that is enough to force them to be *product* update frames. This leads to some reflection on the way the language mirrors the account of information flow the update operation gives. Section 5.2 compares the relevant notions of *bisimulation* for our languages. Section 5.3 shows how we can *simulate* the stronger language $\mathcal{L}^*_{BMS}$ with the stronger language $\mathcal{L}^*_{EDA}$. Section 5.4 discusses a topic we have neglected so far: fact change. It is primarily meant as a preparation for the investigations in the final chapter on closure and preservation.

5.1 Characterizing product update frames

The question of this subsection is how we can characterize the class of product update frames within the class of update frames. The interest of this question derives, first of all, from the fact that we want to know if the system $\mathcal{L}_{EDA}$ is expressive enough to capture the condition characteristic for product update. Second, our characterization provides a further insight in the algorithmic picture on information flow given by the product update operation. It is also interesting to relate our principle to other characterizations of the product update. It is closely related to the *action knowledge axiom* used in [BMS99] in their axiomatization of the system introduced in chapter 3; the importance of this particular axiom is also stressed in [Bal00a]. We also compare our result with the work in [vB00a].

Below, the semantic condition characteristic for product update frames is repeated. We shall refer to it as the *product update principle.*

1. For all $i \in Ag$, for all s, s', a, b such that $s.a, s'.b$ is defined: If $s \rightarrow_i s'$ and $a \twoheadrightarrow_i b$, then $s.a \twoheadrightarrow_i s'.b$.

2. For all $i \in Ag$, for all $s \in S$, for all $a \in A$ such that $s.a$ is defined: If $s.a \rightarrow_i t$, then $t = s'.b$ for some $s' \in S, b \in A$ and $s \rightarrow_i s'$ and $a \twoheadrightarrow_i b$.

Using our system, we can define the class of update frames satisfying this condition, the class of product update frames, with the following principle:

Principle 5.1 $[\mathsf{a}]\mathcal{K}_i\phi \leftrightarrow \bigwedge_{\mathsf{b}\in L}(\langle \mathsf{a} \wedge \overline{\mathcal{K}}_i\mathsf{b}\rangle\top \to \mathcal{K}_i[\mathsf{b}]\phi)$

It may be useful to state this in words: if after the current action, agent i will know ϕ, then for any alternative to the current action that i considers possible, i knows for this alternative that ϕ will hold after its execution. Conversely, if for any action that is an alternative for agent i to the current action, i knows that after this alternative ϕ will hold, then after the current action i knows ϕ.

We remark at this point, that this very principle is the reason why we need to assume our label set to be finite: we need to quantify over the actions accessible for some agent from a given current one. Note that Baltag et al. require their epistemic actions to be finite as well – however, they have potentially countably many of them.

Proposition 5.2 Let $\mathfrak{F}$ be an update frame. Then the following are equivalent:

1. $\mathfrak{F}$ is a product update frame.

2. $\mathfrak{F}$ validates principle 5.1.

Proof:

($\Longrightarrow$) Assume $\mathfrak{F}$ is a product update frame. Suppose, for some $s \in S$, $s \models [\mathsf{a}]\mathcal{K}_i\phi$. We need to show that $s \models \langle \mathsf{a} \wedge \overline{\mathcal{K}}_i\mathsf{b}\rangle\top \to \mathcal{K}_i[\mathsf{b}]\phi$ for any label b. By assumption, we know that for all $a \in A$, and for all $t \in S$, if $a \in V(\mathsf{a})$, $s.a$ is defined, and $s.a \to_i t$, then $t \models \phi$. Consider now the right side. Assume that $s \models \langle \mathsf{a} \wedge \overline{\mathcal{K}}_i\mathsf{b}\rangle\top$ for some b. Thus there exist $a \in A$, $b \in A$, such that $a \in V(\mathsf{a})$, $b \in V(\mathsf{b})$, $s.a$ is defined, and $a \twoheadrightarrow_i b$. We need to show that $s \models \mathcal{K}_i[\mathsf{b}]\phi$, i.e., we need to prove that for all $s' \in S$, and for all $b \in B$, if $s'.b$ is defined, and $b \in V(\mathsf{b})$, then $s'.b \models \phi$. Consider some such s'. By the assumption, we have a unique $b \in A$ such that $b \in V(\mathsf{b})$, and we know that $a \twoheadrightarrow_i b$. Since also $s \to_i s'$, by the assumption on $\mathfrak{F}$, we see that $s.a \to_i s'.b$. But then by the initial assumption, $s'.b \models \phi$.

Now suppose, for some $s \in S$, $s \models \bigwedge_{\mathsf{b}\in L}\langle \mathsf{a} \wedge \overline{\mathcal{K}}_i\mathsf{b}\rangle\top \to \mathcal{K}_i[\mathsf{b}]\phi$. We need to show that $s \models [\mathsf{a}]\mathcal{K}_i\phi$. We know from the assumption that for all $\mathsf{b} \in L$, if $s.a$ is defined, $a \in V(\mathsf{a})$, and there exists $b \in B$ such that $b \in V(\mathsf{b})$, $a \twoheadrightarrow_i b$, then we can conclude that for all $s' \in S$ such that $s \to_i s'$, if $s'.b$ is defined, then $s'.b \models \phi$. We need to show that if $s.a$ is defined, and $a \in V(\mathsf{a})$, then for all $t \in S$: if $s \rightarrow_i t$, then $t \models \phi$. Suppose that $s.a$ is defined, and $a \in V(\mathsf{a})$. Take some t such that $s \rightarrow_i t$. By the definition of product update, there exist s', and b, such that $t = s'.b$, $s \rightarrow_i s'$, and $a \twoheadrightarrow_i b$. We have $b \in V(\mathsf{b})$ for some $\mathsf{b} \in L$, since V is surjective. We can now apply the assumption and conclude that $s'.b \models \phi$.

($\Longleftarrow$) Suppose $\mathfrak{F}$ is not a product update frame. So $\mathfrak{F}$ violates one of the two conditions for product update frames. As for the first case, suppose that there are states s, s' and actions a, b such that $s.a, s'.b$ are defined, $s \rightarrow_i s'$ and $a \twoheadrightarrow_i b$, whereas not $s.a \rightarrow_i s'.b$. Take such states and actions s, s', a, b. Let V be any valuation such that $V(p) = S \backslash \{s'.b\}$, $a \in V(\mathsf{a})$, and $b \in V(\mathsf{b})$. Then for all t such that $s.a \rightarrow_i t$: $t \models p$. So $s \models [\mathsf{a}]\mathcal{K}_i p$. Furthermore, $s \models \langle \mathsf{a} \wedge \overline{\mathcal{K}}_i \mathsf{b} \rangle \top$, since $a \twoheadrightarrow_i b$, and by the choice of V. However, $s \not\models \mathcal{K}_i[\mathsf{b}]p$, since $s \rightarrow_i s'$, and $s'.b \not\models p$, again by the choice of V.

As for the other case, suppose there is a state s and an action a such that $s.a$ is defined, and $s.a \rightarrow_i t$ for some $t \in S$, but there do not exist s' and b such that $t = s'.b$ and $s \rightarrow_i s'$, $a \twoheadrightarrow_i b$. Let s and t and a be such states and such an action. Take any valuation V such that $V(p) = S \backslash \{t\}$, and $a \in V(\mathsf{a})$. Then $s \not\models [\mathsf{a}]\mathcal{K}_i p$. However, we claim that the right side of the principle is true at s; if we establish this, we are done, since then the principle cannot be true at s. So take any $\mathsf{b} \in L$ such that $s \models \langle \mathsf{a} \wedge \overline{\mathcal{K}}_i \mathsf{b} \rangle \top$. Then there exists $b \in A$, such that $a \twoheadrightarrow_i b$ and $b \in V(\mathsf{b})$. Let s' be any state in S such that $s \rightarrow_i s'$. We need to show that $s' \models [\mathsf{b}]p$. Suppose $s'.b$ is defined (otherwise, we are done). We know that $s'.b \neq t$ by assumption. So by the choice of V, $s'.b \models p$. Thus, $s' \models [\mathsf{b}]p$. Since s' was chosen arbitrarily, $s \models \mathcal{K}_i[\mathsf{b}]p$. Since b was chosen arbitrarily as well, s makes the right side of the principle true. $\Box$

Observe that principle 5.1 is in *Sahlqvist* form, so one may also compute the frame condition it defines using the well–known *Sahlqvist* algorithm (cf. [BdRV01]).

Note also, that the principle only captures the "frame part" of the product update operation. If we wish to add the requirement that the valuation is kept constant on update frames, we need to add another principle.

Principle 5.3 $\langle \top \rangle \top \rightarrow (p \leftrightarrow [\top]p)$

It is fairly obvious, that this captures the way valuations were treated in chapter 3; contrary to principle 5.1 above, however, it is a constraint on models; we call update models that are based on product update frames, and which make principle 5.3 globally true, *product update models.*

The proposition on product update frames above shows that the principle linguistically captures the specific account of information flow given by the product update: the assertions that hold about an individual agent's knowledge at some output state is fully determined by assertions true about the input state it came from; which also involve the epistemic properties of the action leading from input to output. The following is a restatement of proposition 5.2, that makes this explicit.

Corollary 5.4 Let $\mathfrak{M} = (\mathcal{S}, \mathcal{A}, ., V)$ be a product update model. Suppose $s \in S$, $a \in A$, and $s.a$ is defined. Assume that $a \models \mathsf{a}$. Then we have that

$$s \models \bigwedge\nolimits_{\mathsf{b} \in L} \langle \mathsf{a} \wedge \overline{\mathcal{K}}_i \mathsf{b} \rangle \top \rightarrow \mathcal{K}_i [\mathsf{b}] \phi$$

iff

$$s.a \models \mathcal{K}_i \phi$$

Proof: Follows from proposition 5.2. □

It is interesting to compare principle 5.1 with the *action knowledge axiom* used by [BMS99]:

$$[\alpha]\mathcal{K}_i\phi \leftrightarrow (\textsc{pre}(\alpha) \rightarrow \bigwedge_{\alpha \rightarrow_i \beta} \mathcal{K}_i[\beta]\phi)$$

The "precondition check" $\textsc{pre}(\alpha)$ in this axiom corresponds to the "availability check" $\langle \mathsf{a} \rangle$ in principle 5.1, while the role of the quantification over the alternative epistemic actions corresponds to our quantification over action *labels*, together with the use of the epistemic operators on actions.

Another analysis of the product update may be found in [vB00a]. It is cast in the context of an analysis of games using propositional dynamic logic. We make a brief comparison. The discussion of [vB00a] revolves around the product update principle applied to "game frames" where all uncertainty relations are assumed to be equivalences relations. Such game frames are based on trees, and represent game–like interaction; in particular, one assumes that players' interaction is driven by a turn taking schedule, telling at each state in the frame which player is to move. This is reflected in special atomic symbols $turn_i$, one for each $i \in Ag$, with the meaning that player i is to move. At each state, one and only one of these symbols can be true. The principles involved are the following formulas:

1. $[a]\mathcal{K}_i\phi \rightarrow \mathcal{K}_i[a]\phi$

2. $\neg turn_i \to (\mathcal{K}_i[A]\phi \to [A]\mathcal{K}_i\phi)$

3. $turn_i \to (\mathcal{K}_i[a]\phi \to [a]\mathcal{K}_i\phi)$

4. $E(\langle a^{-1}\rangle\top \wedge \overline{\mathcal{K}}_i\neg\langle b^{-1}\rangle) \to U(\langle a^{-1}\rangle\overline{\mathcal{K}}_i\phi \to \mathcal{K}_i[b^{-1}]\phi)$

Here A is the union of all atomic programs (which are assumed to form a finite set), and a is an arbitrary atomic program. a^{-1} denotes the converse execution of the atomic program a, and E is the existential modality, its dual being U.[20]

The first principle expresses, one could say, that knowledge about the future entails knowledge in the future: if, at s, agent i knows that action a can only lead to a state where ϕ holds, then after executing a agent i will know that ϕ is the case. The analogue of this principle $[\mathsf{a}]\mathcal{K}_i\phi \to \mathcal{K}_i[\mathsf{a}]\phi$ in the language $\mathcal{L}_{EDA}$ is valid on all product update frames where the actions have reflexive uncertainty relations, as one may easily check.[21] Note that it is not valid anymore, if we drop reflexivity, i.e if we want to account for the possibility of misleading actions. The second and the third principle correspond to *perfect recall* type assumptions one usually makes in game theory: if it is i's turn to move, then i can distinguish all actions available, i.e. , formulated for our system, we have that $a \to_i b$ iff $a = b$, given that i is to move at s, and $s.a$ is defined. The same is not assumed to hold for the moves of other players, which might be non–transparent for i – thus the weaker principle (3). One may check, that the $\mathcal{L}_{EDA}$ principle $\mathcal{K}_i[\top]\phi \to [\top]\mathcal{K}_i\phi$, corresponding to this third principle, is valid on all product update frames. The second principle will, as predicted by the [vB00a] analysis, be valid at specific states in an update frame – those where some player can distinguish between all available actions. The fourth condition expresses a "uniformity" condition: if, at some state s, a is executable, and my be by some agent i with b, then i will always confuse a with b. That this principle is needed depends on the somewhat different (relational) setting in *PDL*. In our system, this principle holds trivially, and need not be enforced. However, one may also point out that the *PDL* analysis shows that it is not unconceivable that action uncertainty should actually depend on specific *instances* of actions – as will happen if one drops principle (4). This reiterates a point we have made earlier: the update setting we are studying here does not have much to say about specific *instances* of actions.

[20]For details on these additional expressions, the reader may consult [BdRV01].

[21]It would lead us too far afield to make formally precise what we mean by "analogue" here.

The analysis in [vB00a], then, should probably be seen as a restricted version of ours for a different system. Our brief comparison also shows that PDL would have been a less suited basic language than the DAL framework is: PDL seems less suited to analyze the kind of two–level epistemics we are interested in here, for the simple reason that it does not have a notion of uncertainty about actions; it does not seem straightforward to extend it in that direction, either. This shows also in the fact that it seems hard, if possible at all, to extend the PDL analysis to the general case of all frames validating the product update principle. Note furthermore, that already the above principles use rather heavy machinery: the global modality, and, additionally, converse execution of programs. Finally, it is not quite in the spirit of the product update operation, to make its workings dependent on turn patterns; it is actually not hard to find cases, where players cannot distinguish moves they take themselves; imagine, for instance, choosing a card blind–folded – this is an action that will typically not be transparent to you, even though you execute it yourself.[22] The same point is made by [vB00a]; it is noted there, that one may give a similar analysis, which does not involve epistemics made dependent on turn patterns.

5.2 Bisimulations

We shall now introduce, and discuss, bisimulations for $\mathcal{L}_{BMS}$ and $\mathcal{L}_{EDA}$. This is instructive by way of comparison; some important differences between the two systems come out more clearly in this perspective. Generally speaking, analyzing suitable notions of equivalence for some language is useful if one wants to get clear on what features of semantic objects one wishes to formalize using this language. It is also preparatory for the work of the next section. To avoid confusion, in the following we shall write $\models_{\mathcal{L}_{BMS}}$ for the truth predicate of the language $\mathcal{L}_{BMS}$ of chapter 3, and $\models_{\mathcal{L}_{EDA}}$ for the truth predicate of the language $\mathcal{L}_{EDA}$ of chapter 4.

The key notion we need is that of an *epistemic bisimulation.*

Definition 5.5 (Epistemic bisimulation) Let $\mathcal{M} = (\mathcal{F}, V)$, and $\mathcal{M}' = (\mathcal{F}', V')$, where $\mathcal{F}$ and $\mathcal{F}'$ are epistemic frames, and $V : At \to \mathcal{P}(F)$ and $V' : At \to \mathcal{P}(F')$ are valuations on some given set At of sentences. An *epistemic bisimulation between* $\mathcal{M}$ *and* $\mathcal{M}'$ is a relation X between states in $\mathcal{M}$ and states in $\mathcal{M}'$ such that

1. for all atomic formulas $p \in At$, $w \in V(p)$ iff $w' \in V'(p)$ (*atomic harmony*),

[22] Note that this may still be modelled within the boundaries of $S5$.

2. if wXw' and $w \rightarrow_i v$, then there exists a state v' such that $w' \rightarrow_i v'$ and vXv' (*forth*),

3. if wXw' and $w' \rightarrow_i v'$, then there exists a state v such that $w \rightarrow_i v$ and vXv' (*back*).

If X is an epistemic bisimulation between $\mathcal{M}$ and $\mathcal{M}'$ such that wXw', we write $X : \mathcal{M}, w \sim \mathcal{M}', w'$. We also write $\mathcal{M}, w \sim \mathcal{M}', w'$, if there is an epistemic bisimulation X such that $X : \mathcal{M}, w \sim \mathcal{M}', w'$. If the models $\mathcal{M}$ and $\mathcal{M}'$ we are comparing are clearly given from the context, we sometimes write $w \sim w'$, if $\mathcal{M}, w \sim \mathcal{M}', w'$. $\dashv$

This definition naturally applies to both state models and labelled action frames; in the first case, we assume *At* to be P, the set of proposition letters, in the second case we take *At* to be L, the set of action labels. Indeed, the definition also applies to *epistemic actions*: one takes the valuation function to be the precondition map PRE, and the set *At* to be $\mathcal{L}_{BMS}$.[23] Since epistemic actions are "model–world pairs", for them we will also require that an epistemic bisimulation between two epistemic actions $\alpha = (\mathcal{A}, \text{PRE}, a)$ and $\beta = (\mathcal{B}, \text{PRE}, b)$ links the two distinguished actions a and b.

Quite obviously, this definition has something interesting to say both about the language $\mathcal{L}_{EDA}$ of chapter 4 and about the language $\mathcal{L}_{BMS}$ of chapter 3. For the latter language, this definition is indeed all we need to consider, if we want to prove invariance. For the following, fix a set of epistemic actions.

Proposition 5.6 ([BMS99]) Let $\mathcal{M}$ and $\mathcal{M}'$ be state models. If $X : \mathcal{M}, s \sim \mathcal{M}', s'$, $\phi \in \mathcal{L}_{BMS}$, and $\alpha = (\mathcal{A}, \text{PRE}, a)$ is an epistemic action, then

1. $\mathcal{M}, s \models_{\mathcal{L}_{BMS}} \phi$ iff $\mathcal{M}', s' \models_{\mathcal{L}_{BMS}} \phi$.

2. $\mathcal{M} \otimes_{\mathcal{C}_\alpha^{\mathcal{M}}} \mathcal{S}, (s, a) \sim \mathcal{M}' \otimes_{\mathcal{C}_\alpha^{\mathcal{M}}} \mathcal{S}, (s', a)$.

Proof: We show (1) and (2) simultaneously. All cases but the one for the action modalities follow directly. Claim (1) uses (2) in the case of a formula $[\alpha]\phi$: namely, by (2), we can assume that the models obtained by the product update agree on ϕ. Claim (2) uses the fact that (1) holds for the preconditions of α. Note that given $X : \mathcal{M}, s \sim \mathcal{M}', s'$, and using (1), we

[23] Since we think that there is no possibility of confusion, we choose to ignore here the difference between a function mapping a given sentence to a singleton set, and a function mapping a given sentence to a single object.

have for an epistemic action $\alpha = (\mathcal{A}, \textsc{pre}, a)$ that $s \models \textsc{pre}(a)$ iff $s' \models \textsc{pre}(a)$, and thus $(s, a) \in \mathcal{C}_\alpha$ iff $(s', a) \in \mathcal{C}_\alpha$. Furthermore, we relate (s, a) and (s', a) in the models obtained via product update, just in case sXs', which gives us an epistemic bisimulation. $\square$

To get an analogue of the previous proposition for update models and the language $\mathcal{L}_{EDA}$, we need to add in more structure: first, we need to account for the two–sortedness of $\mathcal{L}_{EDA}$. Thus, the clauses (1) and (2) in definition 5.7 below require that there be an epistemic bisimulation both on the level of state models and on the level of labelled action frames. And second, we need to ensure coherence between the two sorts to account for formulas of the form $\uparrow\phi$, and $\langle\alpha\rangle\phi$. This is accomplished by the items (3)–(5) in the below definition.

Definition 5.7 Let $\mathfrak{M} = (\mathcal{S}, \mathcal{A}, ., V)$ and $\mathfrak{M}' = (\mathcal{S}', \mathcal{A}', .', V')$ be update models. A *bisimulation* between $\mathfrak{M}$ and $\mathfrak{M}'$ is a relation $X \subseteq (S \times S') \cup (A \times A')$ such that

1. if sXs', then $X : (\mathcal{S}, V), s \sim (\mathcal{S}', V'), s'$,
2. if aXb, then $X : (\mathcal{A}, V), a \sim (\mathcal{A}', V'), b$,
3. if sXs', and $s.a = t$, then there exists b such that $s'.'b = t'$, $X : (\mathcal{A}, V), a \sim (\mathcal{A}', V'), b$, and $X : (\mathcal{S}, V), t \sim (\mathcal{S}', V'), t'$, and vice versa,
4. if aXb, and $s \in D(a)$, then there exists s', such that $X : (\mathcal{S}, V), s \sim (\mathcal{S}', V'), s'$, and $s' \in D(b)$, and vice versa,
5. if aXb, and $s \notin D(a)$, then there exists s', such that $X : (\mathcal{S}, V), s \sim (\mathcal{S}', V'), s'$, and $s' \notin D(b)$, and vice versa,
6. if aXb, and $s \in R(a)$, then there exists s', such that $X : (\mathcal{S}, V), s \sim (\mathcal{S}', V'), s'$, and $s' \in R(b)$, and vice versa,
7. if aXb, and $s \notin R(a)$, then there exists s', such that $X : (\mathcal{S}, V), s \sim (\mathcal{S}', V'), s'$, and $s' \notin R(b)$, and vice versa.

If X is a bisimulation between $\mathfrak{M}$ and $\mathfrak{M}'$ such that xXx', we write $X : \mathfrak{M}, x \equiv \mathfrak{M}', x'$. We also write $\mathfrak{M}, x \equiv \mathfrak{M}', x'$ if there is a bisimulation X such that $X : \mathfrak{M}, x \equiv \mathfrak{M}', x'$. If $\mathfrak{M}$ and $\mathfrak{M}'$ are clearly given by the context, we write $x \equiv x'$, if $\mathfrak{M}, x \equiv \mathfrak{M}', x'$. $\dashv$

With this definition, we have the following proposition:

Proposition 5.8 Let $\mathfrak{M}$ and $\mathfrak{M}'$ be update models. Let X be a bisimulation between $\mathfrak{M}$ and $\mathfrak{M}'$. Then the following hold, for formulas $\phi \in \mathsf{SF}$, and $\alpha \in \mathsf{AF}$:

1. If $X : \mathfrak{M}, s \equiv \mathfrak{M}', s'$, then $\mathfrak{M}, s \models_{\mathcal{L}_{EDA}} \phi$ iff $\mathfrak{M}, s' \models_{\mathcal{L}_{EDA}} \phi$.

2. If $X : \mathfrak{M}, a \equiv \mathfrak{M}', b$, then $\mathfrak{M}, a \models_{\mathcal{L}_{EDA}} \alpha$ iff $\mathfrak{M}, s' \models_{\mathcal{L}_{EDA}} \alpha$.

Proof: We show the claim by simultaneous induction on the structure of both state and action formulas. All boolean cases are standard, as are the cases for epistemic operators. For a state formula of the form $\langle\alpha\rangle$, we can use the induction hypothesis that (2) holds for α. For the last case, suppose $a \models \Uparrow\phi$. We have two cases. Suppose there exists s, such that $s.a$ is defined, and $s \models \phi$. By definition of a bisimulation, there exists s', such that $s \sim s'$, and $s'.'b$ is defined. By the induction hypothesis concerning (1), $s' \models \phi$, so $b \models \Uparrow\phi$. For the other case, suppose there exists s such that $s.a$ is not defined, and $s \models \neg\phi$. By definition of a bisimulation, there exists s' such that $s \sim s'$, and $s'.'b$ is not defined. By the induction hypothesis concerning (1), $s' \models \neg\phi$, so $b \models \Uparrow\phi$. $\square$

A question that arises immediately is if the notions of bisimulation we have introduced are the right ones for the languages $\mathcal{L}_{BMS}$ and $\mathcal{L}_{EDA}$. One could imagine that they impose more structure than is needed to prove modal equivalence. However, for both types of bisimulation, and the corresponding language, it may be established that the class of finite models is a *Hennessy–Milner class*, that is: on finite models, bisimulation and modal equivalence coincide; in particular, this gives us converses for proposition 5.6 and 5.8 for the finite case, thereby showing the adequacy of our notions of bisimulation. We demonstrate this for the case of $\mathcal{L}_{EDA}$.

Proposition 5.9 Let $\mathfrak{M}$ and $\mathfrak{M}'$ be finite update models, let s and s' be states in $\mathfrak{M}$ and $\mathfrak{M}'$, respectively. Suppose that s and s' make the same formulas true, i.e. for all $\phi \in \mathcal{L}_{EDA}$: $s \models \phi$ iff $s' \models \phi$. Then s and s' are bisimilar, i.e. $\mathfrak{M}, s \equiv \mathfrak{M}', s'$. Similarly, if two actions a and b in $\mathfrak{M}$ and $\mathfrak{M}'$, respectively, make the same formulas true, then a and b are bisimilar: $\mathfrak{M}, a \equiv \mathfrak{M}', b$.

Proof: The claim is shown by proving that the relation of modal equivalence *is* itself a bisimulation. The proof is the standard one, for which we refer the reader to [BdRV01]. Let s and s' be modally equivalent states in $\mathfrak{M}$ and $\mathfrak{M}'$, and suppose that s and s' are not bisimilar. We check only that the assumption that no relation between s and s' satisfies condition (3) in the definition of a bisimulation leads to a contradiction, the other cases being similar. Suppose that there exists an action a, such that $s.a = t$; we will derive a contradiction from the assumption that there do not exist an action b, and a state t', such that a and b, t and t' satisfy the same formulas, respectively. There are two cases, that are essentially analogous; we consider one: assume there is no such action b. Let A' be the set of actions such that $s'.'b$ is defined. Since $\mathfrak{M}'$ is finite, we can list this set as $\{a_1, \ldots, a_n\}$. By assumption, for any a_i, there exists a formula α_i, such that $a \models \alpha_i$, whereas $a_i \not\models \alpha_i$. Take the conjunction $\vartheta = \bigwedge_{1 \leq i \leq n} \alpha_i$. Then $s \models \langle\vartheta\rangle\top$, whereas $s' \not\models \langle\vartheta\rangle\top$, since any $a_i \in A'$ makes one of the conjuncts of ϑ false. We have a contradiction.

Let now a and b be modally equivalent actions in $\mathfrak{M}$ and $\mathfrak{M}'$, and suppose that a and b are not bisimilar. Again, we check one case: suppose that no relation between a and b satisfies condition (5) in the definition of a bisimulation leads to a contradiction; the other cases are again similar. Suppose $s \notin D(a)$ and assume there is no $s' \notin D(b)$ such that s and s' make the same formulas true. Let $T = S' \backslash D(b)$. By assumption we have that for all $t \in T$, there exists ϕ_t such that $t \models \phi$, and $s \not\models \phi$. Since T is finite, $\bigvee_{t \in T} \phi_t$ is a formula, and we see that $a \models \Uparrow_2 \bigvee_{t \in T} \phi_t$, while $b \not\models \Uparrow_2 \bigvee_{t \in T} \phi_t$ (the reader may want to compare the truth conditions for $\Uparrow_2$ which are given in chapter 4.3), which yields a contradiction. □

The analogous proposition may be shown for the case of $\mathcal{L}_{BMS}$ and epistemic bisimulation, showing that the relationship between structural and linguistic equvalence is the right one.

Let us turn to the maybe most appalling difference between bisimulation and epistemic bisimulation: for the system $\mathcal{L}_{BMS}$, we get by with considerably less requirements. Where does this come from? After all, the two systems are not *that* different.

Remark 5.10

1. One reason lies in the fact that the epistemic actions of $\mathcal{L}_{BMS}$ are not subject to the definition of bisimulation. They are part of the syntax, and thus they are fixed up to identity. Thus we need not consider a notion of bisimilarity of epistemic actions separately (as in condition (2) in definition 5.7 for $\mathcal{L}_{EDA}$).

2. Furthermore, the fact that the access conditions are defined in terms of the precondition map implies that the domain of an action is closed under epistemic bisimulation: if two elements of a state model s and s'

are epistemically bisimilar, then $(s, a) \in \mathcal{C}_\alpha$ iff $(s', a) \in \mathcal{C}_\alpha$ (note that $\mathcal{C}_\alpha$ depends on the state model at hand. We shall suppress this in our notation). This explains the lack of conditions which would correspond to the last two items, (4) and (5), in definition 5.7 for $\mathcal{L}_{EDA}$, which are designed to work for the precondition operator of $\mathcal{L}_{EDA}$. In the system of [BMS99], preconditions are hardwired in the semantics, and there is no need to mention any conditions governing them.

3. Finally, as pointed out in the proof of proposition 5.6, applying an epistemic action to bisimilar states yields bisimilar outputs: product update respects bisimulation. Thus we need not specify that those outputs are bisimilar (as we did in condition (3) of definition 5.7 for $\mathcal{L}_{EDA}$), since it follows from the general features of the product update. Taken together, these conditions ensure that if two states in two state models are epistemically bisimilar, then all the other structural features needed to prove invariance between the two models match as well. $\dashv$

We can indirectly verify these remarks with a simple observation about $\mathcal{L}_{BMS}$, incorporating the points we mentioned. For two update models $\mathfrak{M} = (\mathcal{M}, (\mathcal{A}, V_A), .)$ and $\mathfrak{M}' = (\mathcal{M}', (\mathcal{A}, V_A), .')$, and an action $a \in A$, we say that *the domain of $a \in A$ is closed under epistemic bisimulation* if for all $s \in S, s' \in S'$: $\mathcal{M}, s \equiv \mathcal{M}', s'$ and $s.a$ is defined implies $s'.a$ is defined.[24]

Proposition 5.11 Let $\mathfrak{M} = (\mathcal{M}, (\mathcal{A}, V_A), .)$ and $\mathfrak{M}' = (\mathcal{M}', (\mathcal{A}, V_A), .')$ be product update frames. Suppose for all $a \in A$, the domain of a is closed under epistemic bisimulation. Then the following are equivalent, for all $s \in S$, and $s' \in S'$:

1. $\mathcal{M}, s \sim \mathcal{M}', s'$

2. $\mathfrak{M}, s \equiv \mathfrak{M}', s'$

Proof: The non–trivial direction is from left to right. Take two update models $\mathfrak{M} = (\mathcal{S}, \mathcal{A}, ., V)$ and $\mathfrak{M}' = (\mathcal{S}', \mathcal{A}, .', V')$, and suppose $\mathcal{M}, s \sim \mathcal{M}', s'$. We need to check the third condition in definition 5.7. So suppose $s.a = t$ is defined for some a. By the assumption, $s'.a = t'$ for some t'. Clearly, a and a are epistemically bisimilar. That s and s' are epistemically bisimilar is our assumption; so we can conclude, adapting the observation in the proof of

[24] Note that this condition is *not* implied by our definition of a bisimulation in 5.7, since this definition does not tell us that the same action should have similar structural features in any way in two update models that are linked by a bisimulation.

proposition 5.6, using the fact that $\mathfrak{M}$ and $\mathfrak{M}'$ are product update models, that u an u' are epistemically bisimilar as well. $\square$

In what sense does this proposition capture the informal remarks above? First, we have made the action frames the same in both update models (remark (1) above). Second, we have simply stipulated that the domain of an action should be closed under epistemic bisimulation (remark (2) above). And third, we have used product update frames. (remark (3) above). The conjunction of these assumptions leads to a "collapse" of the notions of epistemic bisimulation and full bisimulation on states.

Another point about $\mathcal{L}_{BMS}$ relates to bisimilarity of epistemic actions. The idea is the following. If we look at the syntax of $\mathcal{L}_{BMS}$, then we see that epistemic action modalities involve some information that we are not really interested in: actually, we are only interested in the actions' preconditions, and in their structural pattern. This suggests to distinguish epistemic actions only up to bisimilarity. In [BMS99], the following is shown:

Proposition 5.12 ([BMS99]) Let $\alpha = (\mathcal{A}, \text{PRE}, a)$ and $\beta = (\mathcal{B}, \text{PRE}', b)$ be epistemic actions, and suppose that $\alpha \sim \beta$. Then for any state models $\mathcal{M}$ and $\mathcal{M}'$, if $\mathcal{M}, s \sim \mathcal{M}', s'$, then $\mathcal{M} \otimes_{\mathcal{C}^{\mathcal{M}}_{\alpha}} \mathcal{A}, (s, a) \sim \mathcal{M}' \otimes_{\mathcal{C}^{\mathcal{M}'}{}_{\beta}} \mathcal{B}, (s', b)$, and thus $\mathcal{M}, s \models_{\mathcal{L}_{BMS}} [\alpha]\phi$ iff $\mathcal{N}, s' \models_{\mathcal{L}_{BMS}} [\beta]\phi$. $\square$

The main point of the argument is, again, that product update respects bisimulation: bisimilar epistemic actions applied to bisimilar states result in bisimilar outputs.

5.3 Simulating $\mathcal{L}^*_{BMS}$ with $\mathcal{L}^*_{EDA}$

This section is about embedding the language $\mathcal{L}^*_{BMS}$ in the language $\mathcal{L}^*_{EDA}$. Thus, we consider the *extended* languages that we have briefly introduced in chapter 3 and 4: the extension of $\mathcal{L}_{BMS}$ with common knowledge operators, and the extension of $\mathcal{L}_{EDA}$ with common knowledge operators both on states and actions. The use of the word "embedding" should not be taken too literally, since both languages are interpreted in different types of structures. For this reason, we prefer to speak about a *simulation* of $\mathcal{L}^*_{BMS}$ using $\mathcal{L}^*_{EDA}$. For such a simulation to be possible, we will need a way to make both languages comparable. For this purpose, we will use a construction similar to the one we have seen in section 2.4; in fact, the construction in this section will be a rather straightforward extension of construction 2.9.

We first introduce this construction and the result that comes along with it. The shortcoming of this result is that it does not make any essential use of action formulas, besides using action labels to refer to individual actions. However, we designed $\mathcal{L}_{EDA}$ and $\mathcal{L}^*_{EDA}$ to give us expressivity in describing epistemic action structure and the domain of applicability of actions. We put this expressive power to use in the second half of the section, where we present a translation of $\mathcal{L}^*_{BMS}$ into $\mathcal{L}^*_{EDA}$, which works *modulo* the construction presented in the first half. This translation will map formulas of $\mathcal{L}^*_{BMS}$ to state formulas of $\mathcal{L}^*_{EDA}$, and epistemic actions over $\mathcal{L}^*_{BMS}$ to action formulas of $\mathcal{L}^*_{EDA}$. Formulas of $\mathcal{L}^*_{EDA}$ will be equivalent to their translation (along the construction), while epistemic actions will be characterized by their translation up to epistemic bisimilarity, in a sense to be made precise in due course.

While the first half of the section does not depend on the fact that we are considering the extended languages in an essential way, the second part will make crucial use of the common knowledge operators $\mathcal{L}^*_{EDA}$ has on actions.

For the remainder of this section, fix a *finite* set of epistemic actions Act over $\mathcal{L}^*_{EDA}$. Without loss of generality, we assume these actions to be mutually *disjoint*.

Construction 5.13 Let a state model $\mathcal{M} = (\mathcal{S}, V)$ be given. In preparation of the following, for any given state model $\mathcal{N}$, let $\mathcal{C}^{\mathcal{N}}_{\mathsf{Act}} := \bigcup_{\alpha \in \mathsf{Act}} \mathcal{C}^{\mathcal{N}}_{\alpha}$. Let furthermore $\mathcal{A}_{\mathsf{Act}}$ be the union of all action frames underlying epistemic actions in Act. Then we define:

$$\mathcal{M}_0 := \mathcal{M}$$

$$\mathcal{M}_{n+1} := \mathcal{M}_n \otimes_{\mathcal{C}^{\mathcal{M}_n}_{\mathsf{Act}}} \mathcal{A}_{\mathsf{Act}}$$

$$\mathcal{M}_\omega := \bigcup_{n<\omega} \mathcal{M}_n$$

$$\mathcal{C}_\omega := \bigcup_{n<\omega} \mathcal{C}^{\mathcal{M}_n}_{\mathsf{Act}}$$

$$\cdot_\omega := id_{\mathcal{C}_\omega}$$

In the last clause, $id_{\mathcal{C}_\omega}$ denotes the identity map on $\mathcal{C}_\omega$. To obtain a labelling of the actions in $\mathcal{A}$ with actions labels, we identify the set of labels L of $\mathcal{L}^*_{EDA}$ with the epistemic actions in Act, and for any such epistemic action $\alpha = (\mathcal{A}, \text{PRE}, a)$, we put $V(\alpha) = a$. $\dashv$

Note that construction 5.13 is an extended version of construction 2.9 in chapter 2.4. We inductively construct an update model using the product update mechanism given by the semantics of $\mathcal{L}^*_{BMS}$: the action frame is just given by taking the union of all action frames underlying the epistemic actions. The state model is given by repeated applications of the product update. We then collect the access conditions that were generated during the construction, and turn them into an access map. We finally label each action in the action frame of the resulting update model with an *epistemic action* in Act; namely, $V(a) = \alpha$ iff $\alpha = (\mathcal{A}, \text{PRE}, a)$. It is easy to see that this is a total, surjective map. Note that we have chosen our language $\mathcal{L}^*_{EDA}$ depending on the set of epistemic actions Act. As a result, we have $\mathcal{L}^*_{BMS} \subseteq \mathcal{L}^*_{EDA}$.

The construction induces a product update model $(\mathcal{M}_\omega, (\mathcal{A}_{\mathsf{Act}}, V), \cdot_\omega)$, which we call, for a given state model $\mathcal{M}$, *the product update model corresponding to* $\mathcal{M}$, referring to it with $\mathfrak{M}^{\mathcal{M}}_{\otimes}$. We now have the following:

Theorem 5.14 Let $\mathcal{M}$ be a state model. Let $\mathfrak{M}^{\mathcal{M}}_{\otimes} = (\mathcal{M}_\omega, (\mathcal{A}_{\mathsf{Act}}, V), \cdot_\omega)$ be the product update model corresponding to $\mathcal{M}$. Then

$$\mathcal{M}, s \models_{\mathcal{L}^*_{BMS}} \phi \text{ iff } \mathfrak{M}^{\mathcal{M}}_{\otimes}, s \models_{\mathcal{L}^*_{EDA}} \phi$$

for all $\phi \in \mathcal{L}^*_{BMS}$.

Proof: By construction. □

As pointed out above, this result does not use the expressive power of $\mathcal{L}^*_{EDA}$ on actions in any sense. In effect, we have just turned part of the syntax of $\mathcal{L}^*_{BMS}$ into semantic objects for $\mathcal{L}^*_{EDA}$, and adapted our set of labels of $\mathcal{L}^*_{EDA}$ to the set of epistemic actions at hand.

We now want to *describe* the epistemic actions, using a translation of formulas of $\mathcal{L}^*_{BMS}$ into $\mathcal{L}^*_{EDA}$. We define, by double recursion, a function Tr, which maps each formula $\phi \in \mathcal{L}^*_{BMS}$ to a state formula $Tr(\phi) \in \mathcal{L}^*_{EDA}$, which will be a state formula, and which furthermore maps each epistemic action α over $\mathcal{L}^*_{EDA}$, to a translation $Tr(\alpha)$, which will be an action formula. For the formulas, we define:

1. $Tr(p) = p$
2. $Tr(\neg\phi) = \neg Tr(\phi)$
3. $Tr(\phi \wedge \psi) = Tr(\phi) \wedge Tr(\psi)$
4. $Tr(\mathcal{K}_i\phi) = \mathcal{K}_i Tr(\phi)$

5. $Tr(\mathcal{K}^*_{Gr}\phi) = \mathcal{K}^*_{Gr} Tr(\phi)$

6. $Tr([\alpha]\phi) = [Tr(\alpha)]Tr(\phi)$

We now need to define $Tr(\alpha)$, for an epistemic action $\alpha = (\mathcal{A}, \text{PRE}, a)$. A *model world* pair is a pair consisting of an epistemic model $\mathcal{M}$, and a distinguished world in $\mathcal{M}$. We can now code the preconditions as proposition letters. For this purpose, let a total injective map I from the range of PRE to P, the set of proposition letters, be given. Define for all $a \in A$, $V(a) = I(\text{PRE}(a))$. This gives us a model world pair $(\mathcal{A}, V, a)$. A well–known fact in modal logic tells us that any finite model world pair can be characterized by a formula of the standard modal language with common knowledge operators. Let δ be the modal definition of $(\mathcal{A}, V, a)$. This is a formula, built up from proposition letters p, boolean operators, knowledge and common knowledge operators. Since δ contains proposition letters, it is not the formula we want. However, we obtain the desired translation as follows: consider any proposition letter p occurring in δ. By injectivity of I, we find a unique formula $\phi \in \mathcal{L}^*_{BMS}$, such that $I(\phi) = p$. Replace p uniformly in δ with $\uparrow Tr(\phi)$, where we have $Tr(\phi$ by the first part of the definition of Tr above. Do this for all proposition letters occurring in δ, and call the resulting formula δ'. This is the translation we want, i.e. we put

$$Tr(\alpha) = \delta'$$

for each epistemic action α. Note that this is an action formula of the language $\mathcal{L}^*_{EDA}$.

To state the theorem accompanying the translation Tr, we need a further piece of notation. Let $\mathfrak{M} = ((\mathcal{S}, V), (\mathcal{C}, V_C), .)$ and $\mathfrak{N} = ((\mathcal{S}, V), (\mathcal{D}, V_D), .')$ be update models. Let $c \in C$ and $d \in D$. We write $c \approx d$, if there exists a relation $X \subseteq C \times D$, such that cXd, where X satisfies the back and forth clauses of the definition of an epistemic bisimulation, and mXn implies $D(m) = D(n)$. So $c \approx d$ expresses that c and d share the same epistemic pattern up to bisimilarity, and that furthermore c and d have the same domain.

Theorem 5.15
Let a state model $\mathcal{M}$ be given, and let $\mathfrak{M}^{\mathcal{M}}_{\otimes} = (\mathcal{M}_\omega, (\mathcal{A}_{\mathsf{Act}}, V), \cdot_\omega)$ be the update model corresponding to $\mathcal{M}$. Let $\mathfrak{N} = (\mathcal{M}_\omega, (\mathcal{B}, V'), .)$ be an update model.

1. Let $s \in S$ and $\phi \in \mathcal{L}^*_{BMS}$. Then

$$\mathcal{M}, s \models_{\mathcal{L}^*_{BMS}} \phi \text{ iff } \mathfrak{M}^{\mathcal{M}}_{\otimes}, s \models_{\mathcal{L}^*_{EDA}} Tr(\phi)$$

2. Let $\alpha = (\mathcal{A}, \textsc{pre}, a)$, and $b \in B$. Then

$$a \approx b \text{ iff } \mathfrak{N}, b \models_{\mathcal{L}^*_{EDA}} Tr(\alpha)$$

Proof: By double induction. □

Thus we see that, along the construction of associated product update models, $\mathcal{L}^*_{EDA}$ is a conservative extension of $\mathcal{L}^*_{BMS}$. The key to the preceding theorem lies the possibility of describing both epistemic uncertainty patterns and preconditions of actions by means of operators of $\mathcal{L}^*_{EDA}$. It can thus be seen as a verification of the claim that in $\mathcal{L}^*_{EDA}$, mechanisms of describing actions replace the syntactic use of epistemic actions.

5.4 Fact change

If we have valuations conveying information about specific facts that hold somewhere, the question arises what happens to these facts when we update. To study this, we need a notion of *sentential update* to deal with the phenomenon of *fact change.* In the previous chapters we have considered only the simplest case of *purely epistemic* sentential update, where valuations are simply copied from input to output state across some action happening. For this reason, we did not need to consider valuations of facts at actions. In this section, we shall suggest some more possibilities in some detail. The present section is meant as a preparation for the investigation of preservation questions in the next chapter. Also, we want to at least briefly discuss a topic that we have neglected up until now.

Definition 5.16 A *sentential update operation* is a function $\mathsf{Sen} : \mathcal{P}(P) \times \mathcal{P}(P) \to \mathcal{P}(P)$, where P is the set of proposition letters. ⊣

Thus, an operation of sentential update is supposed to take two sets of proposition letters, and produce another complete description from them. The way we interpret the inputs, is, of course, as those proposition letters true at a *state* (the first input), respectively at an *action* (the second input), while the output of the sentential update operation should give us the set of proposition letters true at the output state resulting from applying the action at the state. To build this notion into our setting, we obviously first need a notion of proposition letters *holding at actions.*

Definition 5.17 An *action model* is a pair $(\mathcal{A}, V)$, where $\mathcal{A}$ is an action frame, and $V : P \to \mathcal{P}(A)$ is a function from the set of proposition letters P to the powerset of the domain of $\mathcal{A}$. ⊣

Note that we have not used the term "action model" before, so this does not give rise to ambiguity. It is obviously also feasible to extend the notions of a *labelled action frame* and an *epistemic action* in order to incorporate the idea of valuations of proposition letters at actions. We shall write, by abuse of notation, $V(a) = \Phi$, if Φ is the set of those p such that $a \in V(p)$.

Definition 5.18 An *epistemic action model* is a pair (α, V), where $\alpha = (\mathcal{A}, \text{PRE}, a)$ is an epistemic action, and $(\mathcal{A}, V)$ is an action model. A *labelled action model* is a pair $(\mathcal{A}, V)$, where $V = V_1 \cup V_2$, and $V_1 : A \to L$ is a valuation on the action labels, and $(\mathcal{A}, V_2)$ is an action model. $\dashv$

For the language $\mathcal{L}_{EDA}$, the changes needed to accommodate this new notion of a labelled action model are the obvious ones; we extend the language by allowing proposition letters as atomic action formulas, and adapt the semantics correspondingly. We trust that the reader can fill in the details. As we shall see further down, we can then characterize specific sentential update operations by means of frame constraints. For the language $\mathcal{L}_{BMS}$, we need to say how the product update operation incorporates the new notion of sentential update.

Definition 5.19 (Sentential Product update)
Let Sen be an operation of sentential update. Let $\mathcal{M} = (\mathcal{S}, V)$ be a state model, and $(\mathcal{A}, V')$ be an action model. Let $\mathcal{C}$ be an access condition on S and A. Then the Sen–update of $\mathcal{M}$ with $(\mathcal{A}, V')$ is defined as $\mathcal{M} \otimes_{\mathcal{C}}^{\mathsf{Sen}} (\mathcal{A}, V') = (\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}, V'')$, where for each $(s, a) \in S \otimes_{\mathcal{C}} A$: $V''(s, a) = \mathsf{Sen}(V(s), V'(a))$. $\dashv$

Thus, each of the sentential update notions to be discussed will lead to a different semantics for $\mathcal{L}_{BMS}$: two sentential update notions will lead to different valuations on the states in the state model obtained via the product update.

These preliminaries out of the way – what should happen to the facts? Let us look at some proposals.

1. $\mathcal{L}(\Phi, \Psi) = \Phi$
2. $\mathcal{R}(\Phi, \Psi) = \Psi$
3. $\triangle(\Phi, \Psi) = (\Phi \backslash \Psi) \cup (\Psi \backslash \Phi)$
4. $\bigtriangledown(\Phi, \Psi) = \overline{\triangle(\Phi, \Psi)}$

The first proposal is just a way of recovering purely epistemic update as an explicit notion of sentential update in circumstances where we regard it as convenient to just ignore the valuation on the action.

The second proposal is based on the intuition that actions *bring about* facts (including negative ones). Thus, the action gives us a complete picture of anything that will be true or false due to its execution. This is arguably a bit unintuitive if we are dealing with an *infinite* set of proposition letters, since we then need to get clear, for each and every fact p, on what we assume some action a to do with this p. However, this does not seem to be a principled reason against it, since in modelling real–world interaction, one will no doubt zoom in on a finite set of *relevant* facts.

The third proposal focusses on the *change* induced by actions. The operation $\triangle(\Phi, \Psi)$ is known as the *symmetric difference* of the two sets Φ and Ψ. It includes precisely those elements that were in one of Φ and Ψ but not in the other. The intuition is that this operation *flips* truth–values. For instance, given that $p \in V(s)$, and $p \in V(a)$, we get that $p \notin V(s.a)$, since we take $p \in V(a)$ to mean that action a flips p. However, if $p \in V(s)$, and $p \notin V(a)$, then $p \in V(s.a)$, since a did not flip p. This proposal is studied in [Bal00a].

The fourth proposal, defined in terms of the third, is similar, but *presents* the idea of change somewhat different: one makes those proposition letters true at an action of which one assumes that they are *not* flipped by the action. One talks about change by saying what is not changed, so to speak. Consequently, if $p \in V(s)$, and $p \in V(a)$, then $p \in V(s.a)$, while if $p \in V(s)$, and $p \notin V(a)$, then $p \notin V(s.a)$, since in the first case we assume that $p \in V(a)$ means that a does not change p.

Let us look at some examples.

$\mathsf{R}(\Phi, \Psi)$**:** Suppose a is "set the counter to zero". Assume p means "the counter is zero". Take two states s and s', suppose at s, the counter is zero, while at s' it is not zero. Then p will "hold" at a according to the confirmation interpretation, and at both $s.a$ and $s'.a$, p will hold (since the counter at both states will be zero).

$\triangle(\Phi, \Psi)$**:** Suppose there are two cards lying on the table, a blue one, and a red one. Suppose p means "the left card is the blue one". Assume p is true at s (the blue card is left), and not true at s' (the blue card is not left). Suppose a is "take the left card and put it to the right side of the right card". Then p will "hold" at a according to the change interpretation, and at $s.a$, p will not be true (now the blue card is not left anymore), while at $s'.a$, p will hold (now the blue card is left).

$\mathsf{L}(\Phi, \Psi)$	$\langle\top\rangle\top \rightarrow (p \leftrightarrow [\top]p)$
$\mathsf{R}(\Phi, \Psi)$	$[p]p$
$\triangle(\Phi, \Psi)$	$\langle\top\rangle\top \rightarrow (p \leftrightarrow [p]\neg p) \wedge (\neg p \leftrightarrow [p]p)$
$\triangledown(\Phi, \Psi)$	$\langle\top\rangle\top \rightarrow (p \leftrightarrow [p]p) \wedge (\neg p \leftrightarrow [p]\neg p)$

Table 1: Characterizing sentential update operations

$\triangledown(\Phi, \Psi)$: Any example for $\triangle$ can be made into a "dual" example for $\triangledown$: consider the same example as in the previous item. Then the only difference is, that we will evaluate p false at a (because it is changed by a); the result is the same: at $s.a$, p will not hold; p will hold at $s'.a$ (where now we computed the new valuations using $\triangledown$.

To round this discussion up, we indicate how the relevant notions of sentential update may be captured in the language $\mathcal{L}_{EDA}$. This is summarized in table 1. Note that the initial tests $\langle\top\rangle\top$ we use to check if a state has available actions is needed in the third and fourth row – otherwise, the formulas would be inconsistent at states that do not have successors.

6 Closure and preservation

One way to study the relationship between the languages we have been considering, and the update patterns they describe more closely is via an investigation of *preservation questions*. The problem of preservation is one of the "logical update problems" mentioned in the introduction to this thesis. The possibility of studying such questions, we think, could be seen as one of the main contributions model–theory, or logic, can make to the study of information change. Preservation theorems are ways of characterizing an operation via the formulas the truth value (or validity) of which it leaves unaffected. Thus an investigation of this topic promises to yield further insights in the features of the product update operation we have been studying in the previous chapters. The fact that a formula is preserved has a very direct interpretation in the context of information change: if ϕ is preserved, then the property expressed by ϕ is "resistent" to change (in some way to be made precise below). We can then say, that the update operation *respects* the property expressed by ϕ: ϕ does not become false just because of the particular way the product update operation processes some given inputs.

There is a multitude of questions revolving around the issues of *preservation under product update*. In particular, there is a semantic version of the preservation problem, which we will also discuss: *closure*. The introduction to this chapter charts part of this territory, while later sections treat some of the raised issues in a bit more detail.

But let us first give the setting in which we are going to discuss these problems. As announced already in chapter 2, we focus in this chapter on a simple language. We are going to work with the *basic multi–modal language*. The basic multi–modal language forms precisely the intersection of the *state formulas* and the *action formulas* of $\mathcal{L}_{EDA}$, extended with proposition letters on actions, as discussed in the last section of the previous chapter. $\mathcal{L}_{ML}$ is also a sublanguage of $\mathcal{L}_{BMS}$; thus, the findings of this chapter will be relevant to the earlier introduced systems as well. The semantic setting for the language $\mathcal{L}_{ML}$ is given by state models, action models, access conditions, and various operations of sentential product update (cf. section 5.4 above). We will not always be insistent on the distinction between state and action models, since the language is interpreted on both in exactly the same way; and obviously so, since they are the same kind of objects – epistemic models.

6.1 Introduction

Suppose a given state frame $\mathcal{S}$ has a specific relational property, say, reflexivity (for simplicity, assume for all agents i). In epistemic terms, reflexivity of the uncertainty relations corresponds to the fact that, at each state s, the agents consider s itself as one possibility how things might be. Thus they might be uncertain about the present state of the world, and about other agents, but they are certainly not mislead: If, at state s, agent i knows ϕ, then ϕ is actually true at s. Now suppose $\mathcal{S}$ is updated with some action frame $\mathcal{A}$. Suppose furthermore, that $\mathcal{A}$ itself is also reflexive, i.e. there is nothing misleading about $\mathcal{A}$. Is it guaranteed that the updated state frame will still be reflexive? Note that, if this would not be the case, some doubts about the behavior of the product update operation would arise: for how can one be mislead if one was not mislead and has not been mislead? Fortunately, however, updating a reflexive state frame with an reflexive action frame always results in a new state frame which is itself reflexive. This statement is actually true for all the familiar $S5$ properties.

Example 6.1 Let $\mathcal{S}$ be a state frame, let $\mathcal{A}$ be an action frame. Let $\mathcal{C}$ be an access condition on S and A. Then we have the following:

1. If $\mathcal{S}$ and $\mathcal{A}$ are *reflexive*, then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$ is reflexive.
2. If $\mathcal{S}$ and $\mathcal{A}$ are *symmetric*, then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$ is symmetric.
3. If $\mathcal{S}$ and $\mathcal{A}$ are *transitive*, then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$ is transitive.

For reflexivity, suppose $\rightarrow_i$ and $\twoheadrightarrow_i$ are reflexive relations on S and A. Let (s, a) be any element of $\mathcal{C}$. Then $s \rightarrow_i s$ and $a \twoheadrightarrow a$ by assumption, and so $(s, a) \rightarrow_i (s, a)$ by definition of $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$. For symmetry, assume $\mathcal{S}$ and $\mathcal{A}$ are symmetric models, and take elements (s, a) and (s', b) from $\mathcal{C}$ and suppose $(s, a) \rightarrow_i (s', b)$. By definition of $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$, $s \rightarrow_i s'$ and $a \twoheadrightarrow_i b$. By assumption, $s' \rightarrow_i s$ and $b \twoheadrightarrow_i a$. Again by definition of $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$, we have $(s', b) \rightarrow_i (s, a)$. Transitivity works similar. Thus, as also remarked in [Bal00a], "updating $S5$ states with $S5$ actions yields $S5$ states". $\dashv$

The observation in this example may be formulated in terms of a closure property, as follows: the class of reflexive epistemic frames is *closed under (product) update*, as are the class of transitive and symmetric epistemic frames: If $\mathfrak{C}$ is one of these classes, and $\mathcal{S}$ and $\mathcal{A}$ are elements of $\mathfrak{C}$, then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$ is also an element of $\mathfrak{C}$, for any $\mathcal{C}$.

Definition 6.2 Let $\mathfrak{C}$ be a class of epistemic frames. We say that $\mathfrak{C}$ is *closed under update*, if for all epistemic frames $\mathcal{S}$, $\mathcal{A}$, if $\mathcal{S} \in \mathfrak{C}$ and $\mathcal{A} \in \mathfrak{C}$, then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A} \in \mathfrak{C}$, for all access conditions $\mathcal{C}$ on S and A. $\dashv$

The question then arises how we can capture *all and only* those classes of frames that are closed under update. This is the *closure problem.*

One can also ask in what cases it is "enough" that only one of the input frames has a specific property. Note that for the cases of symmetry, reflexivity, and transitivity from the above example, this is not the case: if you are not mislead, and something misleading happens (i.e. the current state frame is updated with a non–reflexive action frame), you may be mislead afterwards – so this makes a difference. Two cases where one input *is* enough are given in the following example.

Example 6.3 Let $\mathcal{S}$ be a state frame. Let $\mathcal{A}$ be an action frame, and let $\mathcal{C}$ be an access condition on S and A. Let $i \in Ag$. Then we have the following:

1. Suppose $\mathcal{S}$ is *well–founded*, i.e. there is no infinite sequence $s_0, s_1, s_2, \ldots$ of elements of S such that $\ldots s_n \rightarrow_{i_n} s_{n_1} \rightarrow_{i_{n-1}} \ldots s_2 \rightarrow_i s_1 \rightarrow_{i_1} s_0$ (where $i_j \in Ag$). Then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$ is well–founded as well.

2. Suppose all relational sequences in $\mathcal{S}$ have at most length n, i.e. any sequence $(s_1, \ldots, s_m)$ such that $s_k \rightarrow_{j_k} s_{k+1}$, $j_k \in Ag$, $1 \leq k \leq m-1$, is shorter than some fixed n. Then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$ has this property as well.

For the case of well–foundedness, suppose $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$ is not well–founded. Then there is an infinite sequence $v_0, v_1, v_2, \ldots$ of elements of $\mathcal{C}$ such that $\ldots v_n \rightarrow_{i_n} v_{n-1} \rightarrow_{i_{n-1}} \ldots v_2 \rightarrow_{i_2} v_1 \rightarrow_{i_1} v_0$ (where $i_j \in Ag$). Any element of V is of the form (s, a) for some $s \in S$ and $a \in A$. By definition of product update, we have that for any elements v_1, v_2 of V, if $v_1 = (s_1, a_1)$, $v_2 = (s_2, a_2)$, and $v_1 \rightarrow_i v_2$, then $s_1 \rightarrow_i s_2$. So the above infinite sequence of elements of V gives rise to an infinite sequence of elements of S as well, contradicting the assumption that $\mathcal{S}$ is well–founded. We can conclude that $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$ is well–founded. For the second property, one can argue in a similar fashion. $\dashv$

Consider now the following: suppose $\mathcal{S}$ has an *irreflexive point*, i.e. there is some $s \in S$ such that not $s \rightarrow_i s$ for some agent i. Then for any $a \in A$, such that $(s, a) \in \mathcal{C}$, (s, a) will be irreflexive as well. This shows that once you are mislead, that is, once you do not access the actual world anymore, you will never access the actual world in the future. However, it is not the case that the class of epistemic frames having an irreflexive point is closed under update. To see this, just take, in the above example, an epistemic frame $\mathcal{A}$, and an access map such that for all $a \in A$: $(s, a) \notin \mathcal{C}$. Assuming that s was the only irreflexive point in S, we have a counter–example.

The above observations can again be made precise as a – stronger – notion of closure.

Definition 6.4 Let $\mathfrak{C}$ be a class of epistemic frames. We say that $\mathfrak{C}$ is *strongly closed under update* if for all epistemic frames $\mathcal{S}$ and $\mathcal{A}$, if $\mathcal{S} \in \mathfrak{C}$, then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A} \in \mathfrak{C}$, for all access conditions $\mathcal{C}$. $\dashv$

Again, the question then arises how we can capture *all and only* those classes of frames that are strongly closed under update. This is the *strong closure problem.*

Turning to epistemic *models*, we see that, essentially, the same notions of closure reappear: we can wonder what happens if both inputs to the update operation have a specific property, and also in which circumstances one input is "enough" for the output to have it. However, we need to account for another *parameter*: the way in which the valuations are changed by the product update operation. Thus, we need to define our notions of closure in terms of *sentential product update.* Furthermore, since some of the notions of sentential product update we want to consider do not treat states and actions symmetrical, we need to distinguish two notions of strong closure: *action closure* and *state closure.*

Definition 6.5 Let $\mathfrak{C}$ be a class of epistemic models. We say that $\mathfrak{C}$ is *closed under* Sen*–update,* if for all epistemic models $\mathcal{M}$ and $\mathcal{M}'$, if $\mathcal{M}, \mathcal{M}' \in \mathfrak{C}$, then $\mathcal{M} \otimes_{\mathcal{C}}^{\mathsf{Sen}} \mathcal{M}' \in \mathfrak{C}$, for all access maps $\mathcal{C}$. We say that $\mathfrak{C}$ is *state closed under* Sen*–update,* if for all epistemic models $\mathcal{M}$ and $\mathcal{M}'$, if $\mathcal{M} \in \mathfrak{C}$, then $\mathcal{M} \otimes_{\mathcal{C}}^{\mathsf{Sen}} \mathcal{M}' \in \mathfrak{C}$, for all access maps $\mathcal{C}$. We say that $\mathfrak{C}$ is *action closed under* Sen*–update* if for all epistemic models $\mathcal{M}$ and $\mathcal{M}'$, if $\mathcal{M}' \in \mathfrak{C}$, then $\mathcal{M} \otimes_{\mathcal{C}}^{\mathsf{Sen}} \mathcal{M}' \in \mathfrak{C}$, for all access maps $\mathcal{C}$. $\dashv$

Note that we have defined a family of closure notions in each case: one for each operation Sen.

The logical siblings of closure problems are preservation problems – the task is to determine, for some given logical language, the set of those formulas the truth–value, or validity, of which is not affected by performing the operation of product update. Given our earlier investigations, it is natural for us to investigate such questions for modal languages, and indeed for the simplest non–trivial fragment of both the languages $\mathcal{L}_{BMS}$ and $\mathcal{L}_{EDA}$: the basic multi–modal language, denoted, as the reader may recall, with $\mathcal{L}_{ML}$. We call formulas $\phi \in \mathcal{L}_{ML}$ *modal formulas.*

By way of example, the discussion of the previous section has shown that the validity of the reflexivity axiom $\Box\phi \to \phi$ is preserved under product update, in the sense that validity at both inputs entails validity at both outputs (and analogously for the case of symmetry and transitivity). Similarly, the well–known Gödel–Löb axiom $\Box(\Box p \to p) \to \Box p$, which characterizes well–foundedness, was shown to be strongly preserved, in the sense that its validity at one input frame guarantees validity at the output of applying the product update operation.

Definition 6.6 We say that ϕ is *preserved under update on frames*, if for all epistemic frames $\mathcal{S}$, and $\mathcal{A}$, if ϕ is valid in $\mathcal{S}$ and $\mathcal{A}$, then ϕ is valid in $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$, for all access conditions $\mathcal{C}$. We say that ϕ is *strongly preserved under update on frames*, if for all epistemic frames $\mathcal{S}$ and $\mathcal{A}$, if ϕ is valid in $\mathcal{S}$, then ϕ is valid in $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A}$, for all access conditions $\mathcal{C}$. $\dashv$

Turning to models, for a modal analysis, it is most natural to consider a more local preservation problem, relative to a given state at which evaluation takes place. Note that, as in the case of closure on models above, we carry along another parameter: the type of sentential update we are performing.

Definition 6.7 We say that ϕ is *preserved under* Sen*–update on models*, if for all epistemic models $\mathcal{M} = (\mathcal{S}, V)$ and $\mathcal{M}' = (\mathcal{A}, V')$, if $\mathcal{M}, s \models \phi$ and $\mathcal{M}', s \models \phi$, then $\mathcal{M} \otimes^{\mathsf{Sen}}_{\mathcal{C}} \mathcal{M}', (s, a) \models \phi$, if defined, for all $s \in S$, $a \in A$, and for all access conditions $\mathcal{C}$. We say that ϕ is *state preserved under update on models*, if for all epistemic models $\mathcal{M}$, and $\mathcal{M}'$, given as above, if $\mathcal{M}, s \models \phi$, then $\mathcal{M} \otimes^{\mathsf{Sen}}_{\mathcal{C}} \mathcal{M}', (s, a) \models \phi$, if defined, for all $s \in S$, $a \in A$, and for all access conditions $\mathcal{C}$. We say that ϕ is *action preserved under update on models*, if for all epistemic models $\mathcal{M}$, and $\mathcal{M}'$, given as above, if $\mathcal{M}', s \models \phi$, then $\mathcal{M} \otimes^{\mathsf{Sen}}_{\mathcal{C}} \mathcal{M}', (s, a) \models \phi$, for all $s \in S$, $a \in A$, and access conditions $\mathcal{C}$ such that $(s, a) \in \mathcal{C}$. $\dashv$

Given the discussion in section 5.4, it will be natural for us to consider the notions of L–update, R–update, $\triangle$–update, and $\bigtriangledown$–update here.

Finally, a remark is in order on the type of preservation and closure problems considered here. Consider the case of preservation under update on models. We defined a formula ϕ to be preserved under update if truth of ϕ at an input state s and an action a guarantees its truth at (s, a), *for any* *access condition* $\mathcal{C}$ containing the latter element (s, a). If we think of the way access conditions were defined in the framework of $\mathcal{L}_{BMS}$, ultimately one will also be interested in a more fine–grained preservation problem, where the access conditions considered are tied to *preconditions*. Thus, we will associate each action a with a modal sentence $\text{PRE}(a)$, analogous to what we have seen in chapter 3, and consider access conditions of the form

$$\{(s, a) | s \in S, a \in A, s \models \text{PRE}(a)\}$$

In our simplified setting, we do not consider this more sophisticated type of preservation.

The structure of the remainder of this chapter will parallel the introduction. We start by discussing closure, turning to preservation afterwards. In each case, we discuss frames and models, in this order.

6.2 Closure

Frames

Closure under update on frames can be characterized using the familiar operations of taking subframes and direct products. Note that in the following, we assume that $\mathfrak{C}$ is closed under isomorphism.

Theorem 6.8 Let $\mathfrak{C}$ be a class of epistemic frames. Then $\mathfrak{C}$ is closed under update iff $\mathfrak{C}$ is closed under taking finite direct products and subframes.

Proof:

($\Longrightarrow$) Let $\mathfrak{C}$ be a class of epistemic frames, and suppose that $\mathfrak{C}$ is closed under update. First, suppose that $\mathcal{S} \in \mathfrak{C}$, and $\mathcal{S}'$ is a subframe of $\mathcal{S}$. We have to show that $\mathcal{S}' \in \mathfrak{C}$. We update $\mathcal{S}$ with itself. To define the access condition, we put for all $s \in S$: $(s, s') \in \mathcal{C}$ iff $s = s'$ and $s \in S'$. We see that $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{S}$ is isomorphic to $\mathcal{S}'$. And since the former is in $\mathfrak{C}$, also $\mathcal{S}' \in \mathfrak{C}$. Second, suppose that $\mathcal{S}, \mathcal{A} \in \mathfrak{C}$. We have to show that $\mathcal{S} \times \mathcal{A} \in \mathfrak{C}$. Define an access condition by the requirement that $(s, a) \in \mathcal{C}$ iff $s \in S, a \in A$. Then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A} = \mathcal{S} \times \mathcal{A}$. So $\mathcal{S} \times \mathcal{A} \in \mathfrak{C}$.

($\Longleftarrow$) Let $\mathfrak{C}$ be a class of epistemic frames, and suppose $\mathfrak{C}$ is closed under taking finite direct products and subframes. Suppose $\mathcal{S}$ and $\mathcal{A}$ are elements of $\mathfrak{C}$, and let $\mathcal{C}$ be an access condition on S and A. We have to show that $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{A} \in \mathfrak{C}$. First, consider $\mathcal{S} \times \mathcal{A}$. We take the subframe $\mathcal{S}'$ of $\mathcal{S} \times \mathcal{A}$ defined by the requirement that $(s, a) \in S'$ iff $(s, a) \in \mathcal{C}$. Then we have that $\mathcal{S}' = \mathcal{S} \otimes_{\mathcal{C}} \mathcal{A} \in \mathfrak{C}$, which completes the proof. □

Turning to strong closure under update, we observe the following.

Fact 6.9 Let $\mathfrak{C}$ be a class of epistemic frames. If $\mathfrak{C}$ is strongly closed under update, then $\mathfrak{C}$ is closed under update. $\square$

Thus the closure conditions we need to characterize strong closure under update have to be at least as strong as the ones used in the above result. More specifically, we will use the following notions.

Let $\mathcal{S}$ and $\mathcal{S}'$ be epistemic frames. We say that $\mathcal{S}'$ is a *generalized subframe* of $\mathcal{S}$, if $\mathcal{S}'$ can be obtained from $\mathcal{S}$ by taking a subframe of $\mathcal{S}$ and possibly taking a subset of the relations on this subframe.

We say that $\mathcal{S}$ is a *homomorphic pre–image* of $\mathcal{S}'$, if there is a function $F : S \to S'$, such that F is a total, surjective strong homomorphism from $\mathcal{S}$ onto $\mathcal{S}'$.

Equipped with these definitions, we can characterize strong closure under update on frames.

Theorem 6.10 Let $\mathfrak{C}$ be a class of frames. Then the following are equivalent.

1. $\mathfrak{C}$ is strongly closed under update.
2. $\mathfrak{C}$ is closed under taking generalized subframes and homomorphic pre–images.

Proof:

($\Longrightarrow$) Suppose $\mathfrak{C}$ is strongly closed under update. Suppose $\mathcal{S} \in \mathfrak{C}$ and $\mathcal{S}'$ is a generalized subframe of $\mathcal{S}$. We update $\mathcal{S}$ with $\mathcal{S}'$. We define an access condition by the requirement that $(s, s') \in \mathcal{C}$ iff $s = s'$ and $s \in S'$. As a result, $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{S}' = \mathcal{S}'$, which shows that $\mathcal{S}' \in \mathfrak{C}$. Assume now that $\mathcal{S} \in \mathfrak{C}$ and $\mathcal{S}'$ is a strong homomorphic pre–image of $\mathcal{S}$. We have to show that $\mathcal{S}' \in \mathfrak{C}$. Let F be a total, surjective strong homomorphism from $\mathcal{S}'$ onto $\mathcal{S}$. We shall update $\mathcal{S}$ with $\mathcal{S}'$. Define an access condition by the requirement that $(s, s') \in \mathfrak{C}$ iff $s \in S$, $s' \in S'$, and $F(s') = s$. We claim that $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{S}'$ is isomorphic to $\mathcal{S}'$, which shows that $\mathcal{S}' \in \mathfrak{C}$. To show the claim, consider the map $I : \mathcal{C} \to \mathcal{S}'$, defined by $I(s, s') = s'$, for all $(s, s') \in \mathcal{C}$. This is clearly a bijection. Suppose now for some $(s, s'), (t, t') \in \mathcal{C}$, we have that $(s, s') \to_i (t, t')$. Then by definition of update, $s' \to_i^{S'} t'$. Conversely, suppose $s' \to_i t'$ for some $s', t' \in S'$. We know that there exist unique s, and t, such that $(s, s'), (t, t') \in \mathcal{C}$. By definition of $\mathcal{C}$, $F(s') = s$, and $F(t') = t$, and since F is a strong homomorphism, we see that $s \to_i t$. So by definition of update, $(s, s') \to_i (t, t')$. Hence $\mathcal{S}'$ is isomorphic to $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{S}'$, and thus $\mathcal{S}' \in \mathfrak{C}$.

($\Longleftarrow$) Assume that $\mathfrak{C}$ is closed under taking generalized subframes and homomorphic pre–images. Let $\mathcal{S}, \mathcal{S}'$ be epistemic frames, let $\mathcal{C}$ be an access condition, and suppose that $\mathcal{S} \in \mathfrak{C}$. We have to show that $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{S}' \in \mathfrak{C}$. Consider the frame $\mathcal{T}$, which has as its domain $\mathcal{C} \cup \{(s,0) | s \in S, \neg\exists s' \in S' : (s,s') \in \mathcal{C}\}$, and the relations of which are defined by setting $(s,s') \rightarrow_i (t,t')$ iff $s \rightarrow_i s'$. Clearly, $\mathcal{T}$ is a strong homomorphic pre–image of $\mathcal{S}$, along the map F defined by $F(s,s') = s$: since we glued in dummy points, F is obviously surjective, it is also total, and the homomorphic condition is clear by definition. So $\mathcal{T} \in \mathfrak{C}$. We now restrict the domain of $\mathcal{T}$ to $\mathcal{C}$, throwing away the dummies, which gives us a new frame $\mathcal{T}'$, which by assumption, is in $\mathfrak{C}$. Note that its domain is the domain of the frame we want to arrive at. Finally, we restrict the relations by removing those pairs $(s,s') \rightarrow_i (t,t')$, such that $s' \not\rightarrow_i t'$. The resulting frame is in $\mathfrak{C}$ by assumption, and it is just $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{S}'$, which completes the proof. $\Box$

We think that these two result show that the product update fits in rather nicely with other well–known set–theoretic operations.

Models

In the case of models, we concentrate on two cases, that fit in neatly with the preceding. Corollary 6.11 below uses the notion of $\cap$–update, thereby simply referring to the sentential update operation of *intersecting* the two input sets of proposition letters; that is: given Φ and Ψ, $\cap(\Phi, \Psi) = \Phi \cap \Psi$.

Corollary 6.11 Let $\mathfrak{C}$ be a class of epistemic models. Then $\mathfrak{C}$ is closed under $\cap$–update iff $\mathfrak{C}$ is closed under taking submodels and finite direct products.

Proof: Adapt theorem 6.8. Since $\cap$–update behaves exactly like a direct product, the changes are trivial. $\Box$

$\cap$–update has the somewhat odd feature that it allows for negative fact change only: facts may become false, but never true. We mention it here because it naturally extends theorem 6.8 above to the case of models.

Corollary 6.12 Let $\mathfrak{C}$ be a class of epistemic models. Then $\mathfrak{C}$ is state closed under L–update iff $\mathfrak{C}$ is closed under taking generalized submodels and strongly homomorphic pre–images.

Proof: Adapt theorem 6.10. Since L–update copies the valuation from state input to output, the changes are again trivial. $\Box$

The preceding corollary deserves some attention, since on the purely epistemic level, the comparison between product update and other notions of update suggest itself. One such notion is *public update*, which consists essentially in taking a submodel of the present state model. Another one might be dubbed *private update*: here one considers the removal of relational pairs from the present model; our very first example was of this kind: when I open the envelope, the worlds were I am invited to the lecture on probability theory and where I am invited to a lecture on epistemic logic, get disconnected for me, but not for Bob. The preceding corollary now tells us in which way product update, on the level of purely epistemic update, generalizes the two by combining them, and by furthermore allowing us to "boost" models via homomorphic pre–images. Note, however, that this remark should be understood modulo the subtlety that public update does not consist in taking arbitrary submodels, but in taking submodels consisting of those worlds that make some formula ϕ true – the formula ϕ we are updating with. Thus, the update is "self–defined" by the model in which we perform it (cf. [vB00b]). This is analogous to the case of access conditions which are defined using preconditions; the reader may compare the above comment on this issue.

6.3 Preservation

Paralleling the structure of the previous section, we first study preservation on frames, and then turn to preservation on models.

Frames

As we will see in this section, first–order logic allows for a rather neat characterization of the effects of product update.

Definition 6.13 Let a nonempty set of agents Ag be given. Let $\Lambda(R)$ be the first order language with equality which is built over a signature containing a countable set of variables $Var = x_0, x_1, \ldots$, a binary relation symbol R_i for each agent $i \in Ag$, and the symbols $\top$ and $\bot$. $\dashv$

For this language, we need to define a separate notion of preservation:

Definition 6.14 A first–order sentence $\phi \in \Lambda(R)$ is *preserved under update* iff for all epistemic frames $\mathcal{S}$ and $\mathcal{S}'$, and for all access conditions $\mathcal{C}$: if $\mathcal{S}$ and $\mathcal{S}'$ are models of ϕ, then $\mathcal{S} \otimes_{\mathcal{C}} \mathcal{S}'$ is a model of ϕ. $\dashv$

The relevant class of formulas we need in order to characterize preservation under update is given by the following definition. Here, a *strict formula* is a formula which does not contain occurrences of $\bot$.

Definition 6.15 Let a first order language Λ be given. The set of *universal Horn formulas* of Λ is the least set X such that

1. If ϕ_0 is an atomic formula, then $\phi_0 \in X$,
2. If $n \in \mathbb{N}$, $\phi_0, \ldots, \phi_n$ are strict atomic formulas, and ϕ is an atomic formula, then $\phi_0 \wedge \ldots \wedge \phi_n \rightarrow \phi \in X$,
3. If $\phi, \psi \in X$, then $\phi \wedge \psi \in X$,
4. If $\phi \in X$, then $\forall x \phi \in X$.

The set of universal Horn *sentences* is the set of universal Horn formulas with no free variables. $\dashv$

Universal Horn formulas form an important class for a number of reasons (for a survey, the reader may consult [Hod92]). For us, the following characterization is useful:

Theorem 6.16 (Mal'cev) Let ϕ be a first–order sentence over some signature. Then ϕ is preserved under taking finite direct products and substructures iff ϕ is equivalent to a universal Horn sentences. $\square$

With theorem 6.8 from the previous section, we get

Corollary 6.17 Let ϕ be a first–order sentence in $\Lambda(R)$. Then ϕ is preserved under update iff ϕ is equivalent to a universal Horn sentence. $\square$

Note that the examples we discussed as 6.1 all follow from this corollary, via modal frame correspondences. However, what one would like to have in the end, of course, is a characterization like this for a *modal* language.

Open Problem 6.18 Determine the class of basic modal formulas, that are preserved under update on frames. $\dashv$

Note that this will involve also second order axioms – as, for instance, the Gödel-Löb axiom discussed in the introduction. It seems thus unlikely that a modal characterization of preservation for the frame case could draw on existing model–theoretic characterizations like Mal'cev's theorem above.

	Preservation	Action preservation	State preservation
$\triangle$	$-$	$-$	$-$
$\triangledown$	$+$	$-$	$-$
R	$+$	$+$	$-$
L	$+$	$-$	$+$

Table 2: Atomic preservation on models

We note also that we are unaware of a characterization of the first–order formulas that are preserved under homomorphic pre–images and generalized subframes – which would give us a (first order) characterization of strong preservation.

Models

On the level of models, we will do two things. We start with a fine structure analysis of preservation patterns for various notions of sentential update. We then consider characterizations of (1) preservation under $\cap$–update, and (2) state preservation under L–update.

As to the first point, let us start with preservation of *atomic* formulas. Here, it is interesting to consider what the different notions of sentential update discussed in section 5.4 amount to. Our findings are summarized in table 2. The + signs mean that, using the sentential update operation of the corresponding row, atomic formulas are preserved in the sense of the corresponding column. Upon inspection, we see, of course, that each of the notions of sentential update we discussed, in some sense, is *designed* for the notions of preservation it supports. $\triangle$ does not care about preservation at all, since it tells us something about *change*. R basically says that the facts true at an action should be preserved to its output states. Similar remarks may be made for the remaining operations. Given the atomic case as a basis, we can investigate the closure properties of the set of preserved formulas, i.e. which logical connectives "preserve preservation"? Here, our results are collected in table 3. In the table, an occurrence of a logical operator $\heartsuit$ indicates that, given that ϕ and ψ are preserved in the sense of the corresponding column, then $\phi\heartsuit\psi$ is also preserved, respectively not necessarily preserved (depending on the row), i.e. , the set of preserved formulas is closed, respectively not closed under $\heartsuit$. The case of state and action preservation are entirely analogous here, for the obvious reason that the semantics works analogously in both cases. Differences between state and

	Preservation	Action/state preservation
Closed	$\wedge, \mathcal{K}_i$	$\wedge, \vee, \mathcal{K}_i$
Not closed	$\neg, \overline{\mathcal{K}}_i, \vee$	$\neg, \overline{\mathcal{K}}_i$

Table 3: Closure properties of formulas on models

action preservation arise only in connection with specific sentential update operations, that assign a different role to propositional facts on states and on actions in the process of updating.

Let us now turn to the two specific cases. As for the case of preservation under $\cap$–update, there is a nice characterization in terms of modal universal Horn sentences, as introduced in [Stu00].

Definition 6.19 The set of *basic modal Horn formulas* is the least set B such that $\neg p \in B$, if $p \in P$, $\perp \in B$, and B is closed under disjunction, conjunction, and applying $\mathcal{K}_i$. The set of *modal universal Horn sentences* is the least set X such that

1. $B \cup P \subseteq X$.
2. if $\phi, \psi \in X$ then $\phi \wedge \psi \in X$ and $\mathcal{K}_i \phi \in X$.
3. if $\phi, \psi \in X$, and $\phi \in B$ or $\psi \in B$, then $\phi \vee \psi \in X$. $\dashv$

The following result is due to Sturm:

Theorem 6.20 ([Stu00]) Let ϕ be a modal formula. Then ϕ is preserved under taking submodels and finite direct products iff ϕ is equivalent to a modal universal Horn sentence. $\square$

Sturm's theorem also works more generally for arbitrary modal similarity types, with a generalization of definition 6.19. As we have seen, for a class of frames, $\cap$–closure under update is equivalent to closure under taking submodels and finite direct products. Using Sturm's result, we obtain the following:

Corollary 6.21 Let ϕ be a modal formula. Then ϕ is $\cap$–preserved under update iff ϕ is equivalent to a modal universal Horn sentence.

Proof: By theorem 6.20 and corollary 6.11. $\square$

Let us now consider the case of state preservation under L–update. In the previous section, we have seen that state closure under L–update is equivalent to closure under strong homomorphic pre–images and generalized submodels. There are two observations to be made here. First, *any* modal formula is preserved under strong homomorphic pre–images (cf. [BdRV01]). Second, the modal language cannot make a difference between generalized submodels and submodels. In the following, by a relational restriction of a model $\mathcal{M}$ we mean a model obtained by removing relation pairs in $\mathcal{M}$.

Proposition 6.22 let ϕ be a modal formula. ϕ is preserved under taking relational restrictions iff ϕ is preserved under taking submodels.

Proof:

($\Longrightarrow$) Assume ϕ is preserved under relational restrictions. Suppose $\mathcal{M}, s \models \phi$ and $\mathcal{M}'$ is a submodel of ϕ containing s. Let $\mathcal{M}''$ be the model which has the domain and the valuation of $\mathcal{M}$, and the relations of $\mathcal{M}'$. Then $\mathcal{M}''$ is a relational restriction of $\mathcal{M}$, so $\mathcal{M}'', s \models \phi$. However, $\mathcal{M}', s \sim \mathcal{M}'', s$, so also $\mathcal{M}', s \models \phi$.

($\Longleftarrow$) Suppose ϕ is preserved under submodels. Suppose $\mathcal{M}, s \models \phi$ and $\mathcal{M}'$ is a relational restriction of $\mathcal{M}$. Suppose $\mathcal{M} = (S, \rightarrow_i^S, V_S)$. Consider the *unravelling of $\mathcal{M}$ around* s, i.e. the model $\mathcal{N}$, where $\mathcal{N} = (U, \rightarrow_i^U, V_U)$ is defined as follows: the domain, U, consists of all finite sequences $(s, u_1, \ldots, u_n)$, where $n \geq 0$ and $s \rightarrow_{i_1} u_1 \ldots u_{n-1} \rightarrow_{i_n} u_n$ for some $i_1, \ldots, i_n \in Ag$. The relations are defined by the requirement that for all $i \in Ag$, $(s, u_1, \ldots, u_n) \rightarrow_i^U (s, v_1, \ldots, v_m)$ iff $m = n + 1$ and $u_j = v_j$ for $j = 1, \ldots, n$, and $u_n \rightarrow_i^S v_m$. Finally, $(s, u_1, \ldots, u_n) \in V_U(p)$ iff $u_n \in V_S(p)$. Now by a standard modal argument, we have that $\mathcal{N}, s \sim \mathcal{M}, s$, so $\mathcal{N}, s \models \phi$. Consider now the set U' of those $(s, u_1, \ldots, u_n) \in U$ such that there is a path from s to u_n in $\mathcal{M}'$. U' determines a submodel $\mathcal{N}'$ of $\mathcal{N}$, and since ϕ is preserved under taking submodels, $\mathcal{N}', s \models \phi$. However, also $\mathcal{N}', s \sim \mathcal{M}', s$, so $\mathcal{M}', s \models \phi$. □

This gives us the following:

Corollary 6.23 Let ϕ be a modal formula. Then the following are equivalent:

1. ϕ is state preserved under L–update.

2. ϕ is preserved under taking submodels.

Proof: Apply corollary 6.12, and the preceding observations. □

We can now make use of the following:

Theorem 6.24 (van Benthem) Let ϕ be a modal formula. ϕ is preserved under taking submodels iff ϕ is equivalent to a *universal modal formula*, i.e. , built up from proposition letters, and negated proposition letters, using $\vee$, $\wedge$, and knowledge operators $\mathcal{K}_i$.

Proof: A proof may be found in [AvBN96]. $\square$

Using this, we conclude with the following corollary to 6.23.

Corollary 6.25 Let ϕ be a modal formula. Then ϕ is state preserved under L–update iff ϕ is equivalent to a universal modal formula. $\square$

7 Conclusions

By way of conclusion, we summarize the findings of this thesis and mention some further directions.

7.1 Summary

The results of this thesis may be summarized as follows. In chapter 2, we have introduced and motivated a general framework for analyzing informational update, given by the concept of an *update frame*; the main topic of this thesis was the comparison of this setting with what seems to be the most general proposal for solving the update problem: the operation of product update, introduced in the work of Baltag, Solecki, and Moss. We stressed the differences between the two settings: update frames are not tied to a specific rule of updating, while the product update provides just that. But even when restricting the class of update frames to those that obey such a rule, the basic perspective is different: update frames provide a bird's eye view on streams of epistemic interaction, while the product operation locally *performs* updates in a specific situation. Section 2.4 compared the two settings, product update, and update frames, focussing on the interplay between the local and the global level. The main result was a characterization of the relationship between product update and update frames that applies to several interesting classes of update frames. On the way, several features of the product update operation became clear: first and foremost, it encodes a notion of – what we called – epistemic height, that we recovered in the setting of update frames. Also, it became clear how the product update operation generates a stream of information flow in an algorithmic manner.

The next chapter introduced the language $\mathcal{L}_{BMS}$ of [BMS99], and the extension $\mathcal{L}^*_{BMS}$. The main feature of this semantics we stressed is that it builds the product update operation right into the interpretation of the language. Chapter 4 introduced a new language, $\mathcal{L}_{EDA}$, drawing upon the insights of Baltag et al. , but also using tools known from the literature on dynamic arrow logic and hybrid logic. Its main novelty is the use of epistemic operators both on the level of states and on the level of actions. Additional operators enable the study of preconditions and postconditions.

Chapter 5 was devoted to comparing the two approaches. We showed how to characterize the operation of product update using the system $\mathcal{L}_{EDA}$. A comparison of the structural notions of bisimulations for the two languages shed light on the circumstances in which dynamic and epistemic bisimulation "collapse", in that they mutually imply each other. We then showed how to simulate $\mathcal{L}^*_{BMS}$, the extension of $\mathcal{L}_{BMS}$ with common knowledge operators, using $\mathcal{L}^*_{EDA}$, which itself is obtained by extending $\mathcal{L}_{EDA}$ with common knowledge/learning operators both on states and actions.

Chapter 6 dealt with the closure and preservation problems raised by the product update operation. We analyzed them using the basic multi–modal language $\mathcal{L}_{ML}$, which is a sublanguage of both systems discussed in the preceding chapters of this thesis. We established some results concerning the purely semantic side of the enterprize, closure under update. Linguistically, we presented a preservation result for first–order logic, and a fine structure analysis of preservation questions for modal logic on models.

7.2 Further directions

Process like epistemic structure

In chapter 2.4, we have emphasized the interest in the interplay between the local workings of the update operation and the patterns emerging from it on a more global level. This line of thinking was not taken up later on in our logical analysis. However, it is also interesting to explore more process like structures of update by logical means. In this direction, [Bal00a] introduces a *dynamic* logic for product update improving on the work in [BMS99]; this language is obtained by combining an epistemic logic of belief and common knowledge with a programming language of epistemic action terms. One then has formulas of the form $[\tau]\phi$, where τ is an inductively built action term, the denotation of which is an epistemic action, and ϕ is a formula. The quoted paper also mentions a complete equational system for the resulting algebra of epistemic actions. The possibility of adding further process like structure to $\mathcal{L}_{EDA}$ is certainly open to us. Sequential composition, for instance, is an obvious candidate; indeed, this operation forms a part of the standard arrow logic repertoire we have neglected here (cf. the remarks in section 4.3). One can then wonder what properties this operation of composition should have in the context of epistemic update, in particular in what way should the epistemic features of actions lift to actions that are themselves composed of smaller actions? Another operator the behavior of which has been explored in dynamic arrow logic is the $*$–operator known from *PDL* ([vB94],[Mar96]). It might be useful to have it in a language like $\mathcal{L}_{EDA}$ as well, also with an eye towards possible applications (for instance to games, cf. the remarks further down).

For such an extended language, interesting questions concerning the overall structure of information flow may be raised. The focus of a dynamic logic version of $\mathcal{L}_{EDA}$ would be different from the dynamic language of [Bal00a];

the latter uses action terms to refer to actions, while with the former we describe actions using formulas. This makes for different questions; it might also be interesting to compare two such systems. For instance, one may wonder what the classes of propositions are, that can be "learned" interchangeably – i.e. one may try to find interesting classes of formulas α and β such that $\langle \alpha \circ \beta \rangle \leftrightarrow \langle \beta \circ \alpha \rangle$ is valid. An suggestive instance of a formula that one would be expect to have this interchange property is the following:

$$\langle \mathcal{K}_i {\uparrow} p \circ \mathcal{K}_i {\uparrow} q \rangle \leftrightarrow \langle \mathcal{K}_i {\uparrow} q \circ \mathcal{K}_i {\uparrow} p \rangle$$

This might depend on several additional assumptions.

Another question concerns the long–term behavior of product update. It is easy to see, for instance, that product update is not terminating, not even on finite models (just iterate the direct product of two epistemic frames; as we have seen, direct products are a special case of product update). Public update, on the other hand, is terminating on finite models (see [vB00b]). For particular types of actions, this may still hold using product update. Furthermore, notions of *confluence* seem worth studying. Having executed two different actions, do the resulting states have a "common future" (up to bisimilarity, say)?

Complexity analysis

Using the product update operation as a core part of the semantics fixes the epistemic mechanism driving the system $\mathcal{L}_{BMS}$ right from the beginning. This allows for much more compact, local representations: a given initial state model will typically be conveniently small–sized, while update models for $\mathcal{L}_{EDA}$ can easily become huge, and, one might suspect, computationally less efficient to work with (compare [Bal00a] for a similar point). We feel that it would be interesting to actually compare the devised systems from this angle: are there complexity differences resulting from the somewhat different way of modelling? In terms of *sizes of representations*, this seems very likely to be the case. But is *model checking* a given state model less complex than carrying out the same task for a product update model in general? This may be suggestive to assume as well; however, it is also worth keeping in mind, that to determine truth of a $\mathcal{L}_{BMS}$–formula in a state model one may have to construct and maintain loads of representations of *other* updated state models. It might be worthwhile to confirm, or disconfirm, these conflicting intuitions. Similar questions may of course be asked for *satisfiability* – even though a satisfying model for a $\mathcal{L}_{EDA}$ formula may be larger in general than a model for a $\mathcal{L}_{BMS}$ formula – is it less costly to find one in the latter case than in the former? Although it may seem far–fetched to raise such questions before we even have established the *decidability* of $\mathcal{L}_{EDA}$, we hope that the spirit of this questions is clear.

Entailment

Maybe our analysis in chapter 6 was to traditional in its focus on the invariances of update – while update, one might argue, is rather about *change*. On the other hand, as invariance is a special case of change, one could argue that our investigation dealt with a special case of the following question, which asks for a more fine–grained characterization of the logical effects of product update. Let us restrict attention to purely epistemic product update (using the sentential update operation L). Let ϕ and ψ be state formulas, and let ϑ be an action formula, built up possibly using proposition letters from P. Then we say that ϕ *entails* ψ *along* ϑ ($\phi \vdash_\vartheta \psi$), if for every product update model $\mathfrak{M} = (\mathcal{S}, \mathcal{A}, ., V)$ that validates Sen, and for every state $s \in S$, and action $a \in A$ such that $s.a$ is defined, if $s \models \phi$, and $a \models \vartheta$, then $s.a \models \psi$.

Question 7.1 What are the triples of formulas ϕ, ψ, and ϑ, such that $\phi \vdash_\vartheta \psi$? $\dashv$

Such questions have been studied in general model theory (cf., for instance, [BvB99]). Many interesting problems are sub–instances of this more general question. Some of them we have already encountered in this thesis. For instance, for *preserved* formulas ϕ, we have that

$$\phi \vdash_\phi \phi$$

Principle 5.1, which we used to characterize the class of product update frames in section 5.1 provides us also with a specific entailment pattern, as we have seen in corollary 5.4:

$$\bigwedge\nolimits_{\mathsf{b} \in L} \langle \mathsf{a} \wedge \overline{\mathcal{K}}_i \mathsf{b} \rangle \rightarrow \mathcal{K}_i [\mathsf{b}] \phi \vdash_{\mathsf{a}} \mathcal{K}_i \phi$$

One can also study restricted versions of this problem. For instance, is there an interesting entailment calculus of public announcements?

Games

Games of imperfect information have been studied in economic theory for some decades now, mostly with an emphasis on strategic aspects and solution concepts ([OR94, BB98, BB97]). They are very specific epistemic processes, highlighting the interaction between knowledge and action in a particular way. In particular, games involve a lot of additional structural aspects that relate to the epistemic side of interaction in interesting ways. While in a general analysis, actions are modelled as largely "neutral" objects that are "happening" in some way or other, in games they are assigned to specific *players* that *take* these actions. This yields new concerns; for instance, do we want to make sure that a player who is to move knows what actions are available to him? Moreover, game behavior is usually *rule–based*. With the concept of a game, one associates a set of rules governing which moves are appropriate at which states. But besides the appropriate moves, there are also those that are not appropriate[25] Again, this interacts with epistemic questions: how much do I know about my opponent's behavior? Did he take a non–appropriate move? Is there a chance for me to find out? Strategies, finally, are complete *plans* for playing a game; as such, they are ontologically more complex than actions. The epistemic questions associated with them are probably the most complex ones: how can we express things like "player 1 knows that he is currently playing according to strategy σ"?

[25] This distinction is usually "modelled away" in game theory by just representing the "good moves", not the "bad ones".

References

[AvBN96] Hajnal Andreka, Johan van Benthem, and Istvan Nemeti. Modal languages and bounded fragments of predicate logic. *Manuscript*, 1996.

[Bal00a] Alexandru Baltag. A logic for suspicious players: Epistemic actions and belief updates in games. *CWI Technical Report*, 2000. Final version to appear in Bulletin of Economic Research.

[Bal00b] Alexandru Baltag. A logic of epistemic actions (extended version). *CWI Technical Report*, 2000.

[BB97] Pierpaolo Battigalli and Giacomo Bonanno. Synchronic information, knowledge and common knowledge in extensive games. In P. Mongin M. Bacharach, L.A. Gerard-Varet and H. Shin, editors, *Epistemic logic and the theory of games and decisions*, pages 235–263. Kluwer Academic, 1997.

[BB98] Pierpaolo Battigalli and Giacomo Bonanno. Recent results on belief, knowledge and the epistemic foundations of game theory. *Manuscript*, 1998.

[BdRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal logic*. Cambridge University Press, 2001.

[BMS99] Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. The logic of public announcements, common knowledge, and private suspicions. *CWI Technical Report SEN-R9922, Preliminary version*, 1999.

[BMS01] Alexandru Baltag, Lawrence S. Moss, and Slawomir Solecki. The logic of public announcements, common knowledge, and private suspicions. *Unpublished Manuscript, Preliminary version*, 2001.

[BS97] Jon Barwise and Jerry Seligman. *Information flow: the logic of distributed systems*. Cambridge University Press, 1997.

[BvB99] Jon Barwise and Johan van Benthem. Preservation, interpolation, and pebble games. *Journal of Symbolic Logic*, 64:884–903, 1999.

[FHMV95] Ronald Fagin, Joseph Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[G$\ddot{8}$8] Peter Gärdenfors. *Knowledge in flux.* MIT Press, 1988.

[Ger99] Jelle Gerbrandy. *Bisimulations on planet Kripke.* PhD thesis, University of Amsterdam, 1999.

[GG97] Jelle Gerbrandy and Willem Groeneveld. Reasoning about information change. *Journal of logic, language and information*, 6:147–169, 1997.

[GPT87] George Gargov, Solomon Passy, and Tinko Tinchev. Modal environment for boolean speculations. In D. Skordev, editor, *Mathematical logic and its applications*, pages 253 – 263. Plenum Press, 1987.

[Hod92] Wilfried Hodges. Logical features of horn clauses. In *Handbook of logic in artificial intelligence and logic programming.* Oxford University Press, 1992.

[Mar96] Maarten Marx. Dynamic arrow logic. In Maarten Marx, Laszlo Polos, and Michael Masuch, editors, *Arrow logic and multidimensional logic*, pages 109–123. CSLI Publications, 1996.

[Mar99] Maarten Marx. Relativized relation algebras. *Manuscript*, 1999.

[MPM96] Maarten Marx, Laszlo Polos, and Michael Masuch, editors. *Arrow Logic and Multi-Modal Logic.* CSLI Publications, 1996.

[MvBV97] Reinhard Muskens, Johan van Benthem, and Albert Visser. Dynamics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of logic and language*, pages 587–648. Elsevier Science, 1997.

[OR94] Martin Osborne and Ariel Rubinstein. *A course in game theory.* MIT Press, 1994.

[Pla89] Jan Plaza. Logics of public communications. In *Proceedins 4th International Symposium on Methodologies for Intelligent Systems*, 1989.

[Stu00] Holger Sturm. Modal horn classes. *Studia Logica*, 64:301–313, 2000.

[vB91] Johan van Benthem. *Language in action. Categories, lambdas, and dynamic logic.* North Holland, 1991.

[vB94] Johan van Benthem. A note on dynamic arrow logic. In Jan van Eijck and Albert Visser, editors, *Logic and information flow*, pages 15–29. MIT Press, 1994.

[vB96] Johan van Benthem. *Exploring Logical Dynamics*. CSLI Publications, 1996.

[vB00a] Johan van Benthem. Games in dynamic–epistemic logic. *Manuscript*, 2000.

[vB00b] Johan van Benthem. Update delights. *Manuscript*, 2000.

[vD00] Hans van Dittmarsch. *Knowledge games*. PhD thesis, University of Groningen, 2000.

[Ven91] Yde Venema. *Many-dimensional modal logic*. PhD thesis, University of Amsterdam, 1991.