Epistemic Modelling and Protocol Dynamics

Wang Yanjing

Epistemic Modelling and Protocol Dynamics

Yanjing Wang

Epistemic Modelling and Protocol Dynamics

ILLC Dissertation Series DS-2010-06
IPA Dissertation Series 2010-05

About the cover: It is a modernized illustration of the Chinese phrase "Ju An Qi Mei" (holding the tray up to the eyebrows) which describes a couple having respect and love for each other. In the original story, Meng Guang, a well educated woman who married a poor scholar in the Eastern Han Dynasty, served her husband food with her head bowed and holding the tray up to her eyebrows to show respect for him. As one can see, the cover depicts a twist to the original story. See the introduction of this dissertation for an explanation.

Epistemic Modelling and Protocol Dynamics

A P

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. D.C. van den Boom
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Agnietenkapel
op dinsdag 21 september 2010, te 12.00 uur

door

Yanjing Wang

geboren te Beijing, Volksrepubliek China.

Promotiecommissie

Promotor: Prof. dr. D. J. N. van Eijck

Overige leden:
Prof. dr. K. R. Apt
Prof. dr. J. F. A. K. van Benthem
Dr. H. P. van Ditmarsch
Prof. dr. J. C. van de Pol
Prof. dr. F. J. M. M. Veltman
Prof. dr. Y. Venema
Prof. dr. R. Ramanujam

Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Universiteit van Amsterdam

This research was supported by the *Netherlands Organisation for Scientific Research* (NWO) under project number 612.000.528 (VEMPS). The work reported in this dissertation has been carried out at the *Centrum Wiskunde & Informatica* (CWI) under the auspices of the *Institute voor Programmatuurkunde en Algoritmiek* (IPA) and the *Institute for Logic, Language and Computation* (ILLC).

Cover design by Di Liu & Shaojie Zhou based on a traditional Chinese paper cutting.
Printed and bound by Ponsen & Looijen B.V., Wageningen.

ISBN: 978–90–5776–211–6
ILLC Dissertation Series DS-2010-06
IPA Dissertation Series 2010-05

献给我最亲爱的爸爸妈妈

Contents

Acknowledgments

Writing this dissertation is like approximating a fixed point whose existence is not guaranteed at the beginning. At the end of this long journey, the formulas and proofs in this book turn out to carry more meaning for me than their apparent mathematical denotations[1]. Like a glass of wine may let a winegrower recall those sunny or rainy days in the year when the wine was made, the content of this book reminds me of those good or bad days of my Amsterdam life, and most importantly it reminds me how lucky I was for having so many people around who were willing to help and enlighten me. Without them, I would not be able to put my gusts of thought together into a whole picture as it is presented in this book.

I am in debt to my promoter Jan van Eijck who gave me the PhD job, inspired me with many great ideas, and granted me the freedom to do what I wanted. More importantly, he showed me how to live a meaningful and happy life, and let me gradually understand that the pursuit of excellence (no matter what it means) should not be the goal of life in general, since it will lead to a zero-sum game where the unhappiness is guaranteed for most of the players. Thanks to the presence of Jan, Heleen and their lovely twins Gaia and Rosa, I felt so at home that I can mumble my naive thoughts about life whether in Amsterdam or in a no-name lake at a nowhere place in France.

I thank Johan van Benthem for being himself and consequently inspiring, pushing, and helping me in the past years whenever he got a chance. I enjoyed all of our interactions in which Johan constantly produced beautiful melodies of logic which balance technicality, philosophical thinking, and the jokes that let me feel privileged while laughing (in a way that Concertgebouw together with Boom Chicago can never achieve).

Ram Ramanujam kindly saved me from turning into a (dynamic) epistemic fundamentalist and guided me to a broader way of thinking which combines computer science, logic, and philosophy together. Various parts of Ram and/or Rohit Parikh's work were like eye openers for me which also impacted the philosophy of this dissertation a lot.

[1] To really understand what I mean here, please turn to Chapter 1: *Introduction* of this dissertation.

As a former member of the no-longer-existing group of SEN2 at CWI, I thank Jaco van de Pol and Wan Fokkink for their constant encouragements and efforts to interest this stubborn student in process algebra. In particular, Jaco acted as a very helpful daily supervisor until he left CWI in 2007, which did not stop him from sending very detailed comments on my work every now and then. I also would like to thank Erik de Vink, Sjouke Mauw, and Wiebe van der Hoek for their valuable advices on my PhD project.

Francien Dechesne has been a constant pleasant company in my PhD project for 3 years. She helped me to a great extent in organizing our thoughts and in putting things down before it was too late. I should apologize for being too dynamic at various last moments, which Francien generously tolerated. I also thank Francien and her husband Tijn for the fun and help they gave to me after work.

Dick de Jongh and Khalil Sima'an fit me in Amsterdam when I started my master at ILLC, where Frank Veltman and Maricarmen Martínez supervised my master thesis later on. The friendship with Floris Roelofsen, Ana Aguilar, Fenrong Liu, Chiaki Ohkura, Ji Ruan, Ka-Wo Chan, Edgar Andrade, and Höskuldur Hlynsson made my master's life much more colorful than it would have been. Thank you all for kindly encouraging me and preparing me for an academic life.

I also would like to thank my co-authors: Taolue Chen, Mohammad Torabi Dashti, Francien Dechesne, Hans van Ditmarsch, Jan van Eijck, Lakshmanan Kuppusamy, Simona Orzan, Jaco van de Pol, Floor Sietsma and Wouter Teepe for their genuine contributions. I am grateful to the anonymous referees at various venues who rejected or accepted my papers with good reasons.

I thank the respectful members of my reading committee: Krzysztof Apt, Johan van Benthem, Hans van Ditmarsch, Jaco van de Pol, Frank Veltman, Yde Venema, and Ram Ramanujam who took time to read my manuscript carefully and provide valuable comments.

Throughout my PhD, many people have given very important advice on my work (though some of them might not realize it): Alexandru Baltag, Denis Bonnay, Hans van Ditmarsch, Valentin Goranko, Jun Pang, Rohit Parikh, and Rineke Verbrugge. Thanks for the useful discussions.

As Johan van Benthem once mentioned: your best ideas sometimes came from others ("remember that boring talk a year ago..."). Hereby I would like to thank the soil where my ideas were fertilized, particularly because of the in-between nature of my project and my affiliation.

At SEN2 of CWI, Anton, Bert, Mohammad, Jens, Taolue, and Yarick made great fun company. In particular I would like to thank Taolue Chen for setting a perfect example of a successful PhD life and teaching me many things I did not know as a graduate from a philosophy department. I also thank Mohammad Torabi Dashti and Yarosav Usenko for sharing some artistic ideas that a logician may not want to have at work.

After the "successful termination" of SEN2, I moved to SEN1, where Arnold, Bas, Bert, Floor, Jeroen, Jurgen, Mark, Paul, and Tijs helped me to integrate myself in the

group to some extent, which I appreciated so much. In particular I thank Paul Klint for being such a tolerant boss and Mark Hills for kindly proofreading my dissertation. As my office mate and co-author in the past year, Floor Sietsma witnessed my most crazy and productive time at CWI with her lightning speed reasoning (and typing), a variety of candies/nuts, and her cheerful smile, which helped me a lot.

Alexandra, Helle, Huiye, José, Lăcrămioara, Meng, MohammadMahdi, Young-Joo, and Stephanie from SEN3 came to CWI around the same time as I did. Together with other colleagues from SEN3, they formed very happy (sometimes a bit loud) neighbours who often attracted me to the very international dinner parties after work. In particular, I would like to thank Alexandra Silva for her nice Portuguese food, the working-late goodbyes, and important LaTeX advice. Also, many thanks go to Bikkie, Karin, Irma, Mike, and Susanne for their help in practical issues at CWI.

I thank my ILLC-originated/related colleagues: Andreas, Barteld, Cédric, Daisuke, Davide, Edgar, Fernando, Fenrong, Floris, Gaëlle, Guillaume, Ivano, Lena, Michael, Nina, Jacob, Jakub, Jonathan, Olivier, Raul, Reut, Sara, Sonja, Stefan, Sunil, Sujata, Tikitu and many others, for creating such an excellent environment for studying (Dynamic Epistemic) Logic in Amsterdam (thanks for letting me photograph you on many occasions). In particular I am grateful to Eric Pacuit and Davide Grossi for reading my earlier drafts of this dissertation (usually before or after cake sessions).

Before moving to the other part of the acknowledgements, I would like to thank both IPA and ILLC for the support and training during my PhD and also for including this dissertation in both dissertation series.

Despite the mathematical trappings, this dissertation may serve as a disguised photo album for me, which depicts a large part of my (academic) life for the past 6 years. However, there is also a significant part of my life that involves the Chinese community here in Amsterdam, which I have not mentioned yet. I probably need a separate appendix to list my good Chinese friends who helped me in one way or the other. For the sake of relevance and space, I can only mention a few of them who had direct impact on my Amsterdam life and this dissertation.

I would like to thank my friend Zhiwei Zhan for witnessing my Amsterdam life from the very beginning to the very end (and being witnessed by me at the same time). We survived ups and downs and now it is time to head for new phases of life. I thank Mengxiao Wu for the sincere advice in the past 4 years and for being the Chinese representative in my paranimfs. We could have been in the same project but now is even better. I thank Fengkui Ju for being the-best-roommate-ever. Fengkui skilfully turned every dinner into a wonderful exchange of food and philosophical thoughts on logic and life. I thank Zhisheng Huang for his decision theoretical advises at each important moment of my life and efforts in helping me to get a PhD position 4 years ago. I thank Huiye Ma, Bo Gao, Jian Shi, Yiqin Yang and particularly Xiaolong Liu for saving me from being homeless at the end of my PhD period. The motivating examples in the introduction of this dissertation were inspired by a conversation with Guowen Dai, who also inspired me in many other ways. Di Liu and Shaojie Zhou helped in designing the cover of this dissertation, which I like very much. Thank

you all.

I thank Fenrong Liu, Yue Zhang, and in particular Hai Hu for "tricking" me into ACSSNL-AMS[2], which turned out to be a surprising source of happiness in the first two years of my PhD life. On the other hand, I thank Jiayang Li and Fangbin Liu for taking over the Chair when I had to concentrate on my research (and other distractions). I thank my colleagues from ACSSNL for their cooperation and creative ideas. I hope you all had fun when working with me.

I thank Zhihong Song, Duanyang Zhang, Qi Zhao, Kaikai Jing and Longyuan Chen for keeping my artistic mind active every now and then after days soaked in logic.

I also want to thank my excellent Chinese colleagues at Science Park: Bin Chen, Taolue Chen, Bo Gao, Jiyin He, Chao Li, Xirong Li, Fangbin Liu, Lei Liu, Huiye Ma, Guangzhong Qiu, Meng Sun, Nan Tang, Jun Wu, Mengxiao Wu, Si Yin, Ying Zhang, Zhen Zhang, Xu Zhang, and Zhiming Zhao for joyful conversations and endless dinner and Karaoke parties.

I sometimes find myself dry in words (but wet from sweating - an iconic heritage from my father) and this is one of those moments: I am risking not mentioning all the people that I should mention here. Nevertheless, I want to take this chance to express my sincere love to all my close Chinese friends here in the Netherlands, and without loss of generality I suppose you feel the same.

This almost concludes my acknowledgements and my Amsterdam life. Finally, I reserve my deepest gratitude to my parents for their unlimited love, support, and understanding. I shall be around soon!

Yanjing Wang
Amsterdam, July 2010.

[2] Association of Chinese Students and Scholars in the Netherlands, Amsterdam Branch

Chapter 1

Introduction

The ice cream company Häagen-Dazs has an advertising slogan: "Love her, love Häagen-Dazs". In China, the company uses a slightly revised Chinese version of this slogan, which is literally translated in English as "Love her, take her to Häagen-Dazs!". Compared to the original slogan, the Chinese version refines the conditional "protocol" with an action which is much more explicit than love itself. It seems the subtle revision really can make a difference: this supermarket ice cream in U.S. and Europe has become one of the most popular must-buys among Chinese young couples (ask a Shanghai girl about it!). It seems that although it may take more than one life to really understand what love is, you can simply show your love by taking your girlfriend to a nearby Häagen-Dazs shop (and of course buying something there). Actually, it does not matter what love is, what really matters is that what you do is *commonly known* to be a proof of your (undefined) love. How come an ice cream is associated with love? For that you must know that the company follows a super high-end marketing strategy in China. It is the huge price difference with the regular ice creams that contributes to the protocol "**If** you love her **then** take her to Häagen-Dazs!", for this is what allows you to show (off) your love. The protocol is clearly not about truth, but it makes information flow.

Here by "protocol" we refer to the general notion of procedural rules that govern the actions of humans or machines. Besides giving meaning to actions as in the Häagen-Dazs story, protocols also let us know what to do or what not to do. In many cases they are the reasons for us to act in a certain way. When you are driving a car you are also driving with various traffic protocols. In case an accident happens legal protocols are called into play. While you are sending emails or sms to a friend to complain about the bad luck, communication protocols on computers are running to make sure the messages are delivered. Your friend may reply to you with a remark against the current local government who initiated a construction project which led to the traffic chaos in the city centre and claim he will vote for another party a few days later in the election according to the political protocol. Because of the existence of such protocols which restrict the potential behaviour of humans and machines, we save our civilization from a chaotic state. Without doubt, protocols rule the world.

Due to the importance of protocols, it is crucial to know the protocols. The

French greet each other by cheek kissing for (usually) two times while the Dutch generally do it thrice. The first cheek kissing between someone from France and someone Dutch may leave the proper termination of their greeting protocols in question. However, for someone Chinese used to the greeting protocol of shaking hands, the number of kisses is not the (only) question to execute the first such greeting successfully: from which side should I start? how much noise should I make? why alternating left and right? ... A protocol announcement could solve this in advance. If no information is provided, people can always rely on a default protocol such as wait-and-see or copy-cat. The difference in protocols is the reason behind many conflicts and misunderstandings, so keeping your protocol knowledge updated is also important.

In many cases, protocols are used to reach certain goals, e.g., the exchange protocol *cash and carry* is to guarantee a fair exchange in a (hostile) open market. However, knowing the protocol may also prevent the protocol from achieving its goal. For example, if a girl *knows* that the guy who takes her out on a first date acts out the protocol "ask her about herself, to make her think you are really interested in her feelings", she is maybe less impressed with how the date goes than if she ignores this. As a more intricate example, consider the following story in the historical novel *Romance of the Three Kingdoms*, one of the greatest classics in Chinese literature: After suffering his defeat at the battle of Red Cliffs, the warlord Cao Cao made his escape to a crossroads where the main path was wide and flat but longer than the other treacherous path which led to Huarong. The scouts reported to Cao Cao that smoke was seen rising from the Huarong trail suggesting an ambush. Cao Cao laughed: "I know Zhuge Kongming (the opponent strategist) so well. Everything he did was intended to deceive me. Thus the apparent truth must be a deception. The smoke seems to be signalling an ambush but it must be the enemy's decoy to lure me to the main road." He then ordered his men into the Huarong Trail, only to be trapped there by the ambush. In fact, knowing Cao Cao so well, Zhuge Kongming had guessed how Cao Cao would reason, and taking this into account, he still outsmarted him, and the smoke lured Cao Cao into the ambush. Ironically, just like what Cao Cao said, everything Kongming did was intended to deceive him, and what seemed to be the truth to Cao Cao was indeed a deception.

As we have seen from the above stories, protocol and knowledge have an intricate and dynamic relation with each other, which deserves careful study, and is the starting point of this dissertation:

"Epistemic Modelling and Protocol Dynamics"

1.1 Background

Epistemic Protocols Protocols that involve reasoning about knowledge have been studied, under the name of *knowledge-based programs*, since the pioneering work of [HF89] and [FHMV97] in the setting of *Interpreted Systems (IS)* [FHMV95](or, equiv-

alently *Epistemic Temporal Logic (*ETL*)* [PR85]). Research within this framework has revealed that protocols with knowledge tests (e.g., **if** $K_i p$ **then do** a) are essentially more complex than standard programs [HF89, San91, Hal00]. Given an initial setting, a knowledge-based program may be represented by none or more than one interpreted system while a standard program induces a unique interpreted system. [MDH86, Hal87] showed that knowledge may help to develop efficient algorithms but the verification problem is quite involved. In the ETL framework, [PR03] gave a semantics of actions based on protocols which fleshes out the intuition that protocols let actions carry information as we remarked at the beginning of this chapter (see also [BS97] for a more general treatment).

Knowledge-based programs can be generalized to *epistemic protocols* which not only allow knowledge tests but also actions with epistemic effects, e.g. public announcements. Such actions are studied as objects in their own right in *Dynamic Epistemic Logic (*DEL*)* where actions and their epistemic effects are handled by epistemic event models and the built-in update mechanism [Pla89, GG97, BMS98]. During the last decade DEL has been successfully applied to a variety of scenarios from knowledge puzzles to social norm changes [vDvdHK07], due to its flexibility in modelling various epistemic interactions among agents.

Despite some informal protocols featured in the studies of epistemic puzzles (see, e.g., [vD03, AvDR09, VO07, vD08, DvEW10]), the epistemic protocols have not been formally studied as a central issue in the DEL framework until recently. Aiming at merging the temporal aspect of ETL and the dynamic epistemic aspect of DEL, a series of work has been done with extra protocol information provided to the epistemic models [HY09, vBGHP09, Hos09, HP10b] (see also [Hos10] for a survey). A DEL-*protocol* defined in this line of work is a set of sequences of DEL events (pointed event models [BMS98]) closed under finite prefix, similar to the definition of the protocol of [PR03] in an ETL setting. Moreover, a notion of *state-dependent protocols* is introduced in [vBGHP09], which allows different states in a given epistemic Kripke model to have different DEL protocols. In such a set up, the protocol at the real world may not be common knowledge. Given a DEL protocol and an initial model, we can generate a *unique* ETL-like model capturing both the epistemic dynamics and the *protocol information* as [Hos10] puts it.

However, as remarked in [PR03], an *explicit* set of sequences of events, as in the case of the DEL protocols mentioned above, is an *extensional* notion of the *common sense* protocols which are usually specified by a few rules governing the communications. To formally study epistemic protocols, in particular to address their verification problems, an epistemic protocol specification is preferably high-level, finitely representable and independent from the models. Note that the verification of an epistemic protocol can be tricky. Take the following classic example used in DEL literature: *the Russian Cards Problem* (RCP) (introduced to DEL by van Ditmarsch [vD03]):

1.1.1. E . **(Russian Cards Problem (RCP$_{(n,n,k)}$))** *$2n + k$ cards are distributed randomly to three agents $\{A, B, E\}$ such that agent A has n cards, B has n cards, and E has k cards. Now A and B want to inform each other their hands by public announcements, without*

revealing his cards to E. Is it possible? Ω

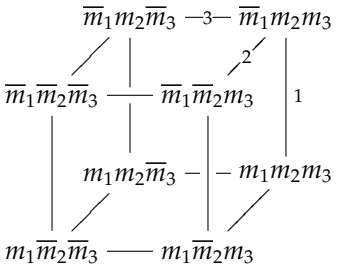
Now let us consider a simple case: $RCP_{2.2.1}$ where the cards are denoted as numbers $(0-4)$. A "promising protocol" for A to let B know A's cards without letting E know any card of A is that: A announces the disjunction of his actual hand (say 01) with all the different combinations of the remaining cards, so he would announce "I have 01 or 23 or 24 or 34." Since B has one more card than E he can eliminate all of 23, 24 and 34, while E can only eliminate two of 23, 24 and 34. However, it does not work like this any more if E knows that the protocol is meant to reveal A's hand to B. Assume that E has 3. Then after the announcement by A, E will know that A has either 01 or 24. Now E can perform the following reasoning: suppose that A has 24 and B has 01. Then B could not have learnt A's hand from A's announcement. So E can infer that A has 01. Another way to see that the would-be protocol is wrong is as follows. Suppose the protocol is commonly known, e.g., the procedure to generate the announcements is known to both A and E. Note that in the above case this procedure is a function from hands of two cards x, y to announcements $f(xy) =$" I have xy or z_1z_2 or z_2z_3 or z_1z_3.", where z_1, z_2, z_3 are the remaining 3 cards other than x, y. This function is injective, so the announcement reveals the hand immediately.

As demonstrated by the above example and many others mentioned in [vD03, vDvdHK07], a notable feature of epistemic protocols, compared to usual communication protocols, is that the correctness of the epistemic protocols heavily relies on the assumptions of the agents' *meta-knowledge* about the protocol itself. It is reasonable to assume that the protocol and its goals are commonly known by all the agents including possible adversaries, if we want to apply the protocol repeatedly in real life cases. To check the correctness of protocols under the assumption that the protocol is commonly known, formalization of protocols is clearly imperative.

Dynamic Epistemic Modelling As in the formal verification of communication protocols, we would like to apply *model checking* to the verification of epistemic protocols, based on a logical language which can specify both the protocol and its goal. However, as observed in [FHMV97], a protocol involving knowledge preconditions should be verified w.r.t. the assumptions about the initial situation, e.g., to verify a protocol for $RCP_{2.2.1}$ on a model with only two agents A, B does not make sense. However, two natural questions arise: how do we specify the assumptions and based on these assumptions, how do we generate a correct model to be checked? Let us now look at another classic puzzle in `DEL` and `ETL` (see e.g., [FHMV95, vDvdHK07]):

1.1.2. E . **(n-Muddy Children)** *Out of n children, $k \geq 1$ got mud on their foreheads while playing. They can see whether other kids are dirty, but there is no mirror for them to discover whether they are dirty themselves. Then father walks in and says: "At least one of you is dirty!" Then he requests "If you know you are dirty, step forward now." If nobody steps forward, he repeats his request: "If you now know you are dirty, step forward now." After exactly k requests to step forward, the k dirty children suddenly do so.* Ω

The changes of the children's knowledge in this classic scenario are "perfectly" modelled by the update mechanism of public announcements on an initial Kripke model, usually in the following shape:



where $\{1, 2, 3\}$ is the set of 3 children, m_i denotes the proposition that i is muddy and $\overline{m}_i$ denotes its negation. The labelled equivalence relations model children's epistemic accessibility relations ($s \longleftrightarrow_i t$ means at state s, i thinks t is possible).

As remarked in [vB09]: *There is no algorithm for producing it, but most people would agree that it fits the situation.* We assume that people, even non-logicians, would be able to "*read off*" the information from the graph representation e.g. "In any case, one agent does not know whether he is dirty or not, but he is sure about the other two." In epistemic logic, it amounts to a conjunction $\phi_1 \wedge \phi_2 \wedge \phi_3$ where $\phi_1 = (K_1m_2 \vee K_1\overline{m}_2) \wedge (K_1m_3 \vee K_1\overline{m}_3) \wedge \neg(K_1m_1 \vee K_1\overline{m}_1)$ and similar for ϕ_2 and ϕ_3. This suggests that we may translate an informal initial setting into a set of logical formulas and try to generate a correct model from this set of formulas.

In the context of dynamic epistemic modelling, [vDvdHK03a] demonstrates that there are intuitive epistemic formulas (*descriptions*) that characterize the initial models in the case of the card games. However, in general, a set of formulas translated from an informal description of the scenario may not have a unique model. In many cases, the informal assumptions in our mind can not be made fully precise. Even if the initial specification induces a model, we still need a method to generate it.

Here we may seek insights from computer science. A useful approach to represent models is the so-called *operational semantics* used for process algebra (e.g. CSP of [Hoa85]), where the model of a process term is generated by the operational rules on its subterms. A similar idea, *Tableau* [Pra80, SE89], appeared in logic as a method for solving the satisfiability problem of logics. Another inspiration is from the `ETL` framework where models are generated by composing local states of each agent. In `DEL` [vD02] made an early attempt to program epistemic actions while [vDvdHK03a] imported the idea of interpreted system in the specific context of card games.

Model Checking During the last three decades, (temporal logic) *model checking* has become a prominent application of logic in computer science (see [CGP99] for an extensive survey). We would like to apply model checking for epistemic protocols as attempted in [vDRV05, vE07, vD03]. However, when dynamic epistemic modelling is applied to complex situations, very large (even infinite) epistemic

models or event models are inevitable (see, for example [DW07]. The verification of certain properties may require the (exhaustive) exploration of such large models. In computer science, this problem is known as the *state space explosion* problem. Various methods have been proposed to handle this problem in temporal logic model checking. A very successful one in practice is the *symbolic model checking* technique initiated by McMillan et.al [BCM+92], which boosted the capability of model checking on large system enormously (see, for example [BCM+92], where more than 10^{20} states are handled in some case studies). Despite the success of BDD-based symbolic model checking and the more recent development of bounded model checking using SAT-solvers (see, e.g., [CBRZ01, McM02]), the state explosion problem still remains a major hurdle to model checking real life complex systems. To reduce the state space, many approaches have been developed, for example, symmetry reduction [CEFJ96, ES96, ID96, SG04], partial order reduction [GPS96, Pel93], abstract interpretation [CC77], and abstraction-refinement methods [CGL94, CGJ+03, GHJ01, SG08]. Among such approaches, the abstraction-refinement method is considered to be the most general and flexible one; also it is fully automated [CGJ+03]. However, such techniques have not been introduced to the epistemic setting until recently [DOW08, CDLR09, CLDQ09].

1.2 Overview of the Dissertation

The general storyline of the dissertation is as follows: In Part I, we introduce logics to specify epistemic protocols including their goals and their dynamics. The verification problem can then be formalized as a model checking problem within a unified logical framework. To perform model checking we need to develop methods for finding/generating epistemic models, and this problem is addressed in Part II. Part III introduces abstraction techniques that are particularly useful on making the model checking more efficient in the epistemic setting. In Part IV we survey the application of epistemic analysis on protocols in a setting of security protocol verification.

The contributions of each chapter are briefly summarized as follows:

In Chapter 2: *Preliminaries*, we list the basic definitions used throughout this dissertation.

Part I

Chapter 3: *Meta-knowledge Matters* departs from the existing discussions about protocols in DEL by introducing a logic which can specify both the epistemic protocols (by regular expressions) and their goals *inside the language*. By formally defining the epistemic protocol specification and their verification problems under the assumption of the meta-knowledge about the intended goal, we flesh out the remarks about the subtleties of epistemic protocol verification. Based on this framework, we discuss

how to find and verify deterministic epistemic protocols for the classic Russian Card Problem $RCP_{3.3.1}$.

In Chapter 4: *Logics of Knowledge and Protocol Change*, we address the question: "how people get to know a protocol?" by developing three logics which are convenient for reasoning about knowledge and protocol changes with different perspectives. With various *protocol announcement modalities*, we can handle the dynamics of protocols and formalize how the protocols let the actions carry new meanings. We show that all the three logics we introduced can be translated back to PDL on standard Kripke models, thus the techniques of modelling and model checking we developed in the other parts of the dissertation can be applied to these logics.

Part II

We then turn to the issues of modelling in Chapter 5: *Composing Models*. We propose new composition operations on static and event models with arbitrary vocabularies, aiming at a compositional method for generating initial epistemic models. We prove some decomposition theorems w.r.t. our new operator and demonstrate the use of our methods by various examples. Algebraic properties linking the new operator to standard product update are also addressed.

In Chapter 6: *Counting Models*, we report some results on counting the number of different models given a finite set of initial assumptions. Restricted to image-finite models, we show that if a modal μ-calculus formula has an infinite model modulo bisimulation then it has $2^{\aleph_0}$ (cardinality of the continuum) different models modulo bisimulation. On the other hand, if it does not have any infinite models then all its models can be represented in a normal form.

Part III

A 3-valued semantics for public announcement logic is defined and studied in Chapter 7: *Making Models Smaller* to facilitate abstractions of models for logic with dynamic modalities. We define a relation with vocabulary and agent mappings between concrete models and their abstractions, thus making it possible to also abstract the signatures of models. It is particularly applicable in an epistemic setting where agents are usually similar to each other. We then give a logical characterization of the abstraction relation thus showing it is safe to check properties on the abstract model instead of the original concrete model.

Chapter 8: *Accelerating the Transitions* studies the PDL on so-called *accelerated Kripke models* where the transitions in the models are labelled by regular expressions in order to obtain informative abstractions. By making use of a technique of regular expression rewriting, we analyse the complexity of the model checking and satisfiability problems of this logic and give a complete axiomatization.

Part IV

Chapter 9: *Epistemic Approaches to Security Protocol Verification* surveys the epistemic approaches to security protocol analysis. We summarize the most important techniques in the ETL and DEL approaches to security protocol verification, and compare

these two approaches in term of convenience. We argue that some security properties can only be faithfully formalized by temporal logic with knowledge operators, but are not expressible by standard temporal logic. However, we need to pay some cost in model checking complexity, in exchange to the expressiveness we gain by using ETL.

1.3 Origins of the Material

The material that forms the main body of this dissertation is based on collaborations with various people: Chapter 3 extends a joint paper with Lakshmanan Kuppusamy and Jan van Eijck [WKvE09]; Chapter 4 is based on an unpublished manuscript; Chapter 5 is an elaborated version of joint work with Jan van Eijck and Floor Sietsma [vEWS10]; Chapter 6 is an extension of a discussion note with Floor Sietsma; Chapter 7 reports joint work with Francien Dechesne and Simona Orzan [DOW08]; Chapter 8 is an updated version of a paper with Taolue Chen and Jaco van de Pol [CvdPW08]; and Chapter 9 is based on a joint paper with Francien Dechesne [DW10].

Some papers related to the general topic of this dissertation are not included in the above chapters. I mention them here as pointers for further reading. With Francien Dechesne, I explored the possibility of using DEL for security protocol verification, as reported in [DW07]. This work also motivated the writing of the material constituting Chapter 9 where the essential ideas of [DW07] are summarized and compared to other approaches. Note that in this dissertation, we focus on knowledge but not belief while in joint work [vEW08] with Jan van Eijck we study a PDL-style DEL as a belief revision logic, which in the end leads to the use of PDL as a protocol logic in Chapter 3 and Chapter 4. Together with Floor Sietsma and Jan van Eijck, I designed a flexible logical framework for reasoning about communications over networks [WSvE10], which combines the dynamics of protocols as in Chapter 4 and the modelling advantages of ETL and DEL respectively. A game theoretical perspective of protocol execution is missing in the current dissertation. However, interested readers may have a look at joint work [TDW08] with Mohammad Dashti which presents a game theoretical analysis of exchange protocols with untrusted third parties. In the end, if the reader prefers a more entertaining introduction to (security) protocols than Chapter 9, she/ he may want to look at [DvETW09, DETW09] written by Francien Dechesne, Jan van Eijck, Wouter Teepe, and me.

Chapter 2

Preliminaries

This chapter introduces a few very basic concepts and notations which are frequently used throughout the thesis. In the follow-up chapters, we will refer to the definitions in this chapter when needed.

2.1 Finite Automata and Regular Expressions

2.1.1. D . **(Finite Automata on Finite Words)** A (non-deterministic) finite automaton is a tuple $\mathsf{A} = (Q, \mathbf{\Sigma}, q_0, \rightarrowtail, F)$ where:

- Q is a finite non-empty set of states, with $q_0 \in Q$ being the *start state*;
- $\mathbf{\Sigma}$ is an alphabet;
- $\rightarrowtail \subseteq Q \times \mathbf{\Sigma} \times Q$ is the set of labelled transitions over Q;
- $F \subseteq Q$ is the set of *accept states*.

$\mathfrak{m}$

Notation For any $a \in \mathbf{\Sigma}$, we write $\overset{a}{\rightarrowtail}$ for $\{(q, q') \mid (q, a, q') \in \rightarrowtail\}$. Let $\mathbf{\Sigma}^*$ be the set of finite (possibly empty) strings of labels in $\mathbf{\Sigma}$, for any $w = (a_0, a_1, \ldots, a_n) \in \mathbf{\Sigma}^*$, we write $q \overset{w}{\rightarrowtail} q'$ if there is a path $q \overset{a_0}{\rightarrowtail} q_1 \overset{a_1}{\rightarrowtail} \cdots \overset{a_n}{\rightarrowtail} q'$ in A. Given an unspecified finite automaton A we use Q_A, $\mathbf{\Sigma}_\mathsf{A}$, q_A, $\rightarrowtail_\mathsf{A}$ and F_A for the corresponding components in the definition of the automaton.

Given $\rightarrowtail$, we let the induced transition function $\delta : Q \times \mathbf{\Sigma} \mapsto 2^Q$ be defined as $\delta(q, a) = \{q' \mid q \overset{a}{\rightarrowtail} q'\}$. Note that $\delta(q, a)$ may be $\emptyset$ for some q and a. A finite automaton on finite words A is said to be *deterministic*, if for any $q \in Q_\mathsf{A}$ and $a \in \mathbf{\Sigma}$: $\delta(q, a)$ is a singleton. We can extend the transition function δ to $\delta^* : Q \times \mathbf{\Sigma}^* \to 2^Q$ such that $\delta^*(q, w) = \{q' \mid q \overset{w}{\rightarrowtail} q'\}$. It is clear that deterministic finite automata (DFA) have the property that for any word $w \in \mathbf{\Sigma}^*$, $\delta^*(q, w)$ is a singleton.

Given a finite automaton $\mathsf{A} = (Q, \mathbf{\Sigma}, q_0, \rightarrowtail, F)$ and a word $w = (a_1, \ldots, a_n) \in \mathbf{\Sigma}^*$, we call a sequence $r = (q_0, q_1, \cdots, q_n)$ a *run* of A over w if for $0 \leq i \leq n : q_i \overset{a_{i+1}}{\rightarrowtail} q_{i+1}$.

A run $r = (q_0, \cdots, q_n)$ is said to be *accepting* if $q_n \in F$. We say A *accepts* w if there exists an accepting run of A over w. The *language* of a finite automaton A is the set $\mathcal{L}(\mathsf{A}) = \{w \in \mathbf{\Sigma}^* \mid \mathsf{A} \text{ accepts } w\}$. We say A and A' are *language equivalent* ($\mathsf{A} =_{\mathcal{L}} \mathsf{A}'$) if $\mathcal{L}(\mathsf{A}) = \mathcal{L}(\mathsf{A}')$.

Given an alphabet $\mathbf{\Sigma}$, regular expressions over $\mathbf{\Sigma}$ are of the form:

$$\pi ::= \mathbf{0} \mid \mathbf{1} \mid a \mid \pi + \pi \mid \pi \cdot \pi \mid \pi^*$$

where $a \in \mathbf{\Sigma}$ and $\mathbf{0}, \mathbf{1}$ are constants for the *empty language* and the *empty string* respectively. We let $Reg_{\mathbf{\Sigma}}$ be the set of all the regular expressions over $\mathbf{\Sigma}$.

Given $L, L' \subseteq \mathbf{\Sigma}^*$, we define $L \circ L'$ to be the set $\{wv \mid w \in L, v \in L'\}$. For $n \geq 0$ we define $L^0 = \{\epsilon\}$ and $L^{n+1} = L \circ L^n$ where ϵ is the empty string. We write $L^\star$ for $\bigcup_{n \geq 0} L^n$.

2.1.2. D . **(Language of Regular Expressions)** The language of a regular expression π (denoted as $\mathcal{L}(\pi)$) is a set of finite strings over $\mathbf{\Sigma}$ defined as follows:

$$\begin{array}{lll} \mathcal{L}(\mathbf{0}) = \emptyset & \mathcal{L}(\mathbf{1}) = \{\epsilon\} & \mathcal{L}(a) = \{a\} \\ \mathcal{L}(\pi \cdot \pi') = \mathcal{L}(\pi) \circ \mathcal{L}(\pi') & & \\ \mathcal{L}(\pi + \pi') = \mathcal{L}(\pi) \cup \mathcal{L}(\pi') & & \\ \mathcal{L}(\pi^*) = (\mathcal{L}(\pi))^\star & & \end{array}$$

$\mathfrak{m}$

The following result is well-known:

2.1.3. T (***Kleene's Theorem***). *For any regular expression π, there exists a finite (deterministic) automaton A such that $\mathcal{L}(\mathsf{A}) = \mathcal{L}(\pi)$. For any finite (deterministic) automaton A there is a regular expression π such that $\mathcal{L}(\pi) = \mathcal{L}(\mathsf{A})$.*

2.2 Kripke Models and Bisimulation

2.2.1. D . **(Kripke Model)** A *Kripke model (KM)* is a tuple:

$$\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V)$$

where:

- S is a non-empty set of states (or *possible worlds*);
- $\mathbf{P}$ is a set of proposition letters;
- $\mathbf{\Sigma}$ is a non-empty set of labels;
- $\rightarrow \subseteq S \times \mathbf{\Sigma} \times S$ is the set of labelled relations over S;
- $V : S \rightarrow 2^{\mathbf{P}}$ is the valuation function.

We call $\mathbf{P}$ the *vocabulary* of $\mathcal{M}$ and $\mathbf{\Sigma}$ the *set of labels* of $\mathcal{M}$. $(\mathbf{P}, \mathbf{\Sigma})$ is called the *signature* of $\mathcal{M}$. A pointed Kripke model $(\mathcal{M}, s)$ is a KM with a designated point in the set of states. Following the tradition in modal logic, we shall call $\mathcal{F} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow)$ a *Kripke frame.* $\mathfrak{m}$

As $\overset{w}{\rightarrowtail}$ in the case of finite automata, we adapt the notion $s \overset{w}{\rightarrow} t$ for $a \in \mathbf{\Sigma}$ and $w \in \mathbf{\Sigma}^*$ in the context of Kripke models, similarly for $S_{\mathcal{M}}$, $\mathbf{P}_{\mathcal{M}}$, $\mathbf{\Sigma}_{\mathcal{M}}$, $\rightarrow_{\mathcal{M}}$ and $V_{\mathcal{M}}$.

A Kripke model $\mathcal{M}$ is said to be *finite*, if $S_{\mathcal{M}}$, $\mathbf{\Sigma}_{\mathcal{M}}$ and $\mathbf{P}_{\mathcal{M}}$ are all finite. A Kripke model is *image-finite* or *finitely branching* if for every state and every label $a \in \mathbf{\Sigma}$, there are only at most finitely many a-successors; it is $\omega-$*branching* if for every state and every label $a \in \mathbf{\Sigma}$, there are only at most countably many a-successors.

An $S5$ *Kripke model* $\mathcal{M}$ is a KM whose labelled relations are *equivalence* relations, i.e., for all $a \in \mathbf{\Sigma}_{\mathcal{M}}$ $:\overset{a}{\rightarrow}$ is *reflexive* $(\forall s : s \overset{a}{\rightarrow} s)$, *symmetric* $(\forall s, t : s \overset{a}{\rightarrow} t \iff t \overset{a}{\rightarrow} s)$, and *transitive* $(\forall s, t, r : (s \overset{a}{\rightarrow} t \wedge t \overset{a}{\rightarrow} r) \implies s \overset{a}{\rightarrow} r)$. Therefore, in the case of $S5$ models, we also use $\sim$ to denote the set of relations. $S5$ models are standard models for *epistemic logic* where the set of labels are interpreted as the set of agents. In such a context we may use $\mathbf{I}$ instead of $\mathbf{\Sigma}$ when defining an $S5$ model and use $\sim_i$ instead of $\overset{i}{\sim}$ for $i \in \mathbf{I}$, following the standard notations in epistemic logic.

Note that in computer science a Kripke frame is usually called a *Labelled Transition System (LTS)* and Kripke models are sometimes called *Kripke Labelled Transition Systems (KLTS)*.

2.2.2. D . **(Bisimulation)** A binary relation R between the domains of two KMs $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V)$ and $\mathcal{N} = (T, \mathbf{P}, \mathbf{\Sigma}, \rightarrow', V')$ is called a *bisimulation* iff $(s, t) \in R$ implies that the following conditions hold:

Invariance For any $p \in \mathbf{P} : p \in V(s) \iff p \in V'(t)$.

Zig if $s \overset{a}{\rightarrow} s'$ in $\mathcal{M}$ then there exists a t' in $\mathcal{N}$ such that $t \overset{a}{\rightarrow}' t'$ and $s' R t'$.

Zag if $t \overset{a}{\rightarrow}' t'$ in $\mathcal{N}$ then there exists an s' in $\mathcal{M}$ such that $s \overset{a}{\rightarrow} s'$ and $s' R t'$.

Two pointed Kripke models $(\mathcal{M}, s)$ and $(\mathcal{N}, t)$ are said to be *bisimilar* $(\mathcal{M}, s \leftrightarrows \mathcal{N}, t)$ if there is a bisimulation R between them such that $(s, t) \in R$. We say a bisimulation R is *total*, if every world in one model is linked by R to some world in the other model. We write $\mathcal{M} \leftrightarrows \mathcal{N}$ if there is a total bisimulation between $\mathcal{M}$ and $\mathcal{N}$. $\mathfrak{m}$

Note that the above standard bisimulation is defined between models with the same signature. In this thesis we will also work with models with different vocabularies. We say two pointed models $(\mathcal{M}, s)$ and $(\mathcal{N}, t)$ are *restricted bisimilar* w.r.t $\mathbf{P}' \subseteq \mathbf{P}_{\mathcal{M}} \cap \mathbf{P}_{\mathcal{N}}$ (notation: $\mathcal{M}, s \leftrightarrows_{\mathbf{P}'} \mathcal{N}, t$), if $\mathcal{M}, s$ and $\mathcal{N}, t$ are bisimilar with the original invariance condition replaced by a *restricted* invariance condition:

[$\mathbf{P}'$-Invariance] for any $p \in \mathbf{P}' : p \in V_{\mathcal{M}}(s) \iff p \in V_{\mathcal{N}}(t)$.

Similarly we can define total restricted bisimulation w.r.t $\mathbf{P}'$ ($\mathcal{M} \leftrightarrows_{\mathbf{P}'} \mathcal{N}$) in the straightforward way.

Note that an *autobisimulation* of a model is an equivalence relation on the state space of a model. Thus we can have a quotient model w.r.t to the maximal autobisimulation on a model.

2.2.3. D . **(Bisimulation Contraction)** Given a Kripke model $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V)$, let $\equiv_b \subseteq S \times S$ be the *autobisimulation*: $\{(s,t) \mid \mathcal{M}, s \leftrightarrows \mathcal{M}, t\}$. The *bisimulation contraction* of $\mathcal{M}$ is the quotient model

$$\mathcal{M}_{/\equiv_b} = (S', \mathbf{P}, \mathbf{\Sigma}, \rightarrow', V')$$

where:

- $S' = \{[s] \mid s \in S\}$ where $[s]$ is the equivalence class containing s w.r.t $\equiv_b$;
- $([s], a, [t]) \in \rightarrow'$ iff $(s, a, t) \in \rightarrow$;
- $V'([s]) = V(s)$.

$\mathfrak{m}$

We can adapt the definition of bisimulation for finite automata by replacing the invariance condition with the following:

$$[\textbf{Accept Invariance}] : s \in F \iff t \in F'$$

where F and F' are the sets of accept states in two automata. We say automata A and B are bisimilar if there is a bisimulation R between Q_{A} and Q_{B} with the accept invariance condition such that $(q_{\mathsf{A}}, q_{\mathsf{B}}) \in R$. It is easy to see that $\mathsf{A} \leftrightarrows \mathsf{A}' \implies \mathsf{A} =_{\mathcal{L}} \mathsf{A}'$, but the converse does not hold.

2.2.4. D . **(n-round Bisimulation Game)** An n-round bisimulation game $\mathcal{G}_n((\mathcal{M}, s), (\mathcal{N}, t))$ between two pointed KMs $(\mathcal{M}, s)$ and $(\mathcal{N}, t)$ with the same signature is a two player game based on the configurations in $S_{\mathcal{M}} \times S_{\mathcal{N}}$. The initial configuration is (s, t) and the players, Spoiler and Verifier, play in rounds. Each round consists of two moves: first by Spoiler and then by Verifier. At each configuration (s', t'), there are two options:

- Spoiler selects an $a \in \mathbf{\Sigma}$ and a state s'' in $\mathcal{M}$ such that $s' \xrightarrow{a}_{\mathcal{M}} s''$ and then Verifier needs to come up with a state in $\mathcal{N}$ such that $t' \xrightarrow{a}_{\mathcal{N}} t''$ and $V(s'') = V(t'')$. The configuration is then changed to (s'', t'').
- Spoiler selects an $a \in \mathbf{\Sigma}$ and a state t'' in $\mathcal{N}$ such that $t' \xrightarrow{a}_{\mathcal{N}} t''$ and then Verifier needs to respond with a state in $\mathcal{M}$ such that $s' \xrightarrow{a}_{\mathcal{M}} s''$ and $V(s'') = V(t'')$. The configuration is then changed to (s'', t'').

Spoiler wins the game if within $n-1$ rounds some configuration (s', t') is reached such that Spoiler can make a legal move but Verifier does not have a legal move to respond. Verifier wins the game otherwise. ℜ

We say $(\mathcal{M}, s)$ and $(\mathcal{N}, t)$ are *modally equivalent* ($\mathcal{M}, s \equiv_{\mathtt{ML}} \mathcal{N}, t$) if $\mathcal{M}, s$ and $\mathcal{N}, t$ satisfy exactly the same basic modal logic (ML) formulas[1]. The following facts are well known (cf., e.g., [BdRV02]).

2.2.5. F . For image-finite pointed Kripke models $(\mathcal{M}, s)$ and $(\mathcal{N}, t)$, the following are equivalent:

- $\mathcal{M}, s \leftrightarrows \mathcal{N}, t$.
- $\mathcal{M}, s \equiv_{\mathtt{ML}} \mathcal{N}, t$.
- for all $n \in \mathbb{N}$: Verifier has a winning strategy in the game $\mathcal{G}_n((\mathcal{M}, s), (\mathcal{N}, t))$.

ℵ

2.3 Three Logics

2.3.1 Propositional Dynamic Logic

*Propositional Dynamic Logic (*PDL*)*, introduced by Fischer and Ladner [FL79] (following the idea of [Pra76]), is a branching-time logic of programs (represented by regular expressions):

$$\phi ::= \top \mid p \mid \phi \wedge \phi \mid \neg\phi \mid \langle\pi\rangle\phi$$

where p ranges over a set of propositions $\mathbf{P}$ and π is a regular expression over some alphabet $\mathbf{\Sigma}$ with *tests* in terms of PDL formulas:

$$\pi ::= \mathbf{0} \mid \mathbf{1} \mid a \mid ?\phi \mid \pi + \pi \mid \pi \cdot \pi \mid \pi^*$$

where $a \in \mathbf{\Sigma}$. When $\mathbf{\Sigma}$ is not fixed, we use $\mathtt{PDL}_{\mathbf{\Sigma}}$ to denote the PDL language based on $\mathbf{\Sigma}$. As usual, we define $\bot$, $\phi \vee \psi$, $\phi \to \psi$ and $[\pi]\phi$ as the abbreviations of $\neg\top$, $\neg(\neg\phi \wedge \neg\psi)$, $\neg\phi \vee \psi$ and $\neg\langle\pi\rangle\neg\phi$ respectively.

Intuitively, $\langle\pi\rangle\phi$ says that there is an execution of program π such that after the execution ϕ holds.

We define the satisfaction relation $\vDash$ between a pointed model $(\mathcal{M}, s)$ with the signature $(\mathbf{P}, \mathbf{\Sigma})$ and a $\mathtt{PDL}_{\mathbf{\Sigma}}$ formula ϕ as follows:

$$\begin{array}{rcl}
\mathcal{M}, s \vDash p & \Leftrightarrow & p \in V_{\mathcal{M}}(s) \\
\mathcal{M}, s \vDash \neg\phi & \Leftrightarrow & \mathcal{M}, s \nvDash \phi \\
\mathcal{M}, s \vDash \phi \wedge \psi & \Leftrightarrow & \mathcal{M}, s \vDash \phi \text{ and } \mathcal{M}, s \vDash \psi \\
\mathcal{M}, s \vDash \langle\pi\rangle\phi & \Leftrightarrow & \exists s' : s[\![\pi]\!]s' \text{ and } \mathcal{M}, s' \vDash \phi
\end{array}$$

where $[\![\pi]\!]$ is defined as:

[1] ML extends propositional logic with modal formulas $\Box\phi$ and their Boolean combinations.

$$
\begin{array}{rcl}
s[\![\mathbf{1}]\!]s' & \Leftrightarrow & s = s' \\
s[\![\mathbf{0}]\!]s' & \Leftrightarrow & \text{never} \\
s[\![a]\!]s' & \Leftrightarrow & s \xrightarrow{a} s' \\
s[\![?\psi]\!]s' & \Leftrightarrow & s = s' \text{ and } \mathcal{M}, s' \vDash \psi \\
s[\![\pi_1 \cdot \pi_2]\!]s' & \Leftrightarrow & s[\![\pi_1]\!] \circ [\![\pi_2]\!]s' \\
s[\![\pi_1 + \pi_2]\!]s' & \Leftrightarrow & s[\![\pi_1]\!] \cup [\![\pi_2]\!]s' \\
s[\![(\pi_1)^*]\!]s' & \Leftrightarrow & s[\![\pi_1]\!]^\star s'
\end{array}
$$

where $\circ, \cup$ and * are the usual composition, union and reflexive transitive closure on relations respectively.

We can view π as a regular expression over $\mathbf{\Sigma} \cup \{?\phi \mid ?\phi \text{ appears in } \pi\}$, then:

$$
\begin{array}{rcl}
\mathcal{M}, s \vDash \langle\pi\rangle\phi & \Leftrightarrow & \text{there exists a path } s = s_0[\![a_1]\!]s_1[\![a_2]\!]\cdots[\![a_n]\!]s_n \\
 & & \text{in } \mathcal{M} \text{ such that } \mathcal{M}, s_n \vDash \phi \text{ and } a_0 a_1 \ldots a_n \in \mathcal{L}(\pi)
\end{array}
$$

PDL can be axiomatized by the following axioms and inference rules [Seg82, Par78][2]:

TAUTOLOGY	all the tautologies
K	$[\pi](\phi \to \phi') \to ([\pi]\phi \to [\pi]\phi')$
0	$[\mathbf{0}]\phi \leftrightarrow \top$
1	$[\mathbf{1}]\phi \leftrightarrow \phi$
TEST	$[?\psi]\phi \leftrightarrow (\psi \to \phi)$
SEQ	$\langle\pi_1 \cdot \pi_2\rangle\phi \leftrightarrow \langle\pi_1\rangle\langle\pi_2\rangle\phi$
OR	$\langle\pi_1 + \pi_2\rangle\phi \leftrightarrow (\langle\pi_1\rangle\phi \vee \langle\pi_2\rangle\phi)$
Star1	$\langle\pi^*\rangle\phi \leftrightarrow (\phi \vee \langle\pi\rangle\langle\pi^*\rangle\phi)$
Star2	$[\pi^*](\phi \to [\pi]\phi) \to (\phi \to [\pi^*]\phi)$
Rules	
$\Box$	$\dfrac{\phi}{[\pi]\phi}$
MP	$\dfrac{\phi, \phi \to \psi}{\psi}$

Note that with the presence of tests $?\phi$ we can eliminate basic programs **0** and **1** by defining them as $?\bot$ and $?\top$ respectively. Sometimes we are interested in the *test-free* fragment of PDL in which we do not have ? as one of the program constructors but we do have **0** and **1**.

We write $K_i\phi$ (i knows ϕ) and $\widehat{K}_i\phi$ (i thinks ϕ is possible) for $[i]\phi$ and $\langle i\rangle\phi$ respectively, when interpreting $\mathtt{PDL}_\mathbf{I}$ on $S5$ models in an epistemic setting. We write $C_G\phi$ (ϕ is common knowledge among the agents in G) as $[(i_1 + \cdots i_n)^*]\phi$ if $G = i_1, \ldots, i_n \subseteq \mathbf{I}$.

[2] The PDL formulas which are valid (i.e. hold on all the pointed models) are precisely the ones that can be derived from the following proof system.

2.3.2 Epistemic Temporal Logic

Developed independently by [PR85] and [HM90], and later made popular by the seminal book [FHMV95], the *Interpreted Systems* (IS) (or *Epistemic Temporal Logic* (ETL)) framework nicely combines the temporal developments of a system (in runs) with epistemic ones in a distributed setting. Following the exposition in [FHMV95], we give the definition of interpreted systems as follows:

2.3.1. D . **(Interpreted System)** Given a set of agents $\mathbf{I} = \{i_1, \ldots, i_n\}$, given $n+1$ non-empty sets $L_\varepsilon, L_1, \ldots, L_n$ of *local states of* (one for the environment ε, and one for each agent in $\mathbf{I}$), the set of *global states* for an interpreted system is a set $S \subseteq L_\varepsilon \times L_1 \times \cdots \times L_n$. An interpreted system $\mathcal{I}$ is a triple $(R, \mathbf{P}, V)$ where R is a set of *runs*, i.e. functions $r : \mathbb{N} \to S$, and $V : S \mapsto 2^{\mathbf{P}}$ is a valuation function. We denote the finite history (m-prefix) of a run r as (r, m). (r, m) and (r', m') are indistinguishable for agent i (notation: $(r, m) \sim_i (r', m')$) if global states $r(m)$ and $r'(m')$ have the same local state for i. A *pointed IS* is an IS with a designated finite history, e.g., $\mathcal{I}, r, n$. ℜ

Each interpreted system can be viewed as an infinite Kripke model with the set of labels $\mathbf{I} \cup \{\tau\}$ where for each $i \in \mathbf{I} : \sim_i$ is an equivalence relation, and $\xrightarrow{\tau}$ represents the temporal development of the system. In the setting of ETL [PR85], the temporal transitions are labelled with explicit actions e in a set $\mathbf{\Sigma}$. Various Epistemic Temporal languages can be defined on such models, for example, the simplest language is:

$$\phi ::= \top \mid p \mid \phi \wedge \phi \mid \neg\phi \mid K_i\phi \mid \langle e \rangle \phi$$

with the following semantics on IS:

$$\begin{array}{rcl}
\mathcal{I}, r, n \vDash p & \Leftrightarrow & p \in V_{\mathcal{I}}(r(n)) \\
\mathcal{I}, r, n \vDash \neg\phi & \Leftrightarrow & \mathcal{I}, r, n \nvDash \phi \\
\mathcal{I}, r, n \vDash \phi \wedge \psi & \Leftrightarrow & \mathcal{I}, r, n \vDash \phi \text{ and } \mathcal{I}, r, n \vDash \psi \\
\mathcal{I}, r, n \vDash K_i\phi & \Leftrightarrow & \text{for all } (r', m) \text{ such that } (r, n) \sim_i (r', m) : \mathcal{I}, r', m \vDash \phi \\
\mathcal{I}, r, n \vDash \langle e \rangle \phi & \Leftrightarrow & (r, n) \xrightarrow{e} (r, n+1) \text{ and } \mathcal{I}, r, n+1 \vDash \phi
\end{array}$$

$\langle e \rangle$ in the above simple language can be replaced by any temporal operator thus obtaining more expressive epistemic temporal logics.

2.3.3 Dynamic Epistemic Logic

A different perspective on the dynamics of multi-agent system is provided by the development of so-called Dynamic Epistemic Logic (DEL) [Pla89, GG97, BMS98]. The focus of DEL is not on the temporal structure of the system but rather on the epistemic impact of the events as the agents perceive them. The following PDL-style DEL language is based on the exposition in [vBvEK06]:

$$\phi \quad ::= \quad \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle \mathcal{A}, e \rangle \phi \mid \langle \pi \rangle \phi$$

where $\mathcal{A}$ is an *event model* defined below with e as its designated event.

2.3.2. D . **(Event Model)** An event model $\mathcal{A}$ is a tuple:

$$\mathcal{A} = (E, \mathbf{\Sigma}, \rightarrowtail, Pre)$$

where:

- E is a finite non-empty set of events.
- $\mathbf{\Sigma}$ is a set of labels.
- $\rightarrowtail \subseteq E \times \mathbf{\Sigma} \times E$ is the set of labelled transitions.
- $Pre : E \mapsto Form(\texttt{DEL})$ where $Form(\texttt{DEL})$ is the set of DEL formulas. Intuitively, Pre assigns to each action a precondition in the form of a DEL formula that can be constructed in an earlier stage of the inductive definition of the language.

$\mathfrak{m}$

Notation In the epistemic setting, the relations $\overset{i}{\rightarrowtail}$ in the action model are assumed to be equivalence relations, thus we may use $\leftrightarrow_i$ to denote them. $\leftrightarrow_i$ models agent i's observational power on events in E (e.g. $e_1 \leftrightarrow_i e_2$ means agent i can not distinguish event e_1 and e_2).

The semantics for PDL formulas is as usual and for $\langle \mathcal{A}, e \rangle \phi$:

$$\boxed{\mathcal{M}, s \vDash \langle \mathcal{A}, e \rangle \phi \iff \mathcal{M}, s \vDash Pre(e) \text{ and } \mathcal{M} \otimes \mathcal{A}, (s, e) \vDash \phi}$$

where $\otimes$ is defined as follows:

2.3.3. D . **(Product Update $\otimes$)** Given a Kripke model $\mathcal{M} = (S, \mathbf{\Sigma}, \rightarrow, V)$ and an event model $\mathcal{A} = (E, \mathbf{\Sigma}, \rightarrowtail, Pre)$, the product model $(\mathcal{M} \otimes \mathcal{A})$ is a Kripke model $(\mathcal{M} \otimes \mathcal{A}) = (S', \mathbf{\Sigma}, \rightarrow', V')$ where:

$$\begin{array}{rcl} S' & = & \{(s, e) \mid \mathcal{M}, s \vDash Pre(e)\} \\ \overset{a}{\rightarrow}' & = & \{((s, e), (s', e')) \mid s \overset{a}{\rightarrow} s' \text{ and } e \overset{a}{\rightarrowtail} e'\} \\ V'((s, e)) & = & V(s) \end{array}$$

$\mathfrak{m}$

The simplest event model is perhaps the one modelling a *public announcement* of ϕ (notation: $!\phi$) depicted as the following event model $\mathcal{A}_{!\phi}$:

where ϕ is the precondition of this singleton model, and $\overset{\mathbf{I}}{\rightarrow}$ denotes the reflexive relations for each $i \in \mathbf{I}$. Let $\mathcal{M}|_\phi$ be the Kripke model $(S, \mathbf{P}, \mathbf{I}, \sim, V)$ where:

- $S = \{s \in S_\mathcal{M} \mid \mathcal{M}, s \vDash \phi\}$;
- $\sim = \sim_\mathcal{M} \cap (S \times \mathbf{\Sigma} \times S)$;

- $V = V_{\mathcal{M}}|_S$ (i.e. the restriction of $V_{\mathcal{M}}$ on the domain S).

It is easy to see that updating $\mathcal{A}_{!\phi}$ on a static model $\mathcal{M}$ amounts to restricting the $\mathcal{M}$ by the states which satisfy ϕ: $\mathcal{M} \otimes \mathcal{A}_{!\phi} \leftrightarrows \mathcal{M}|_\phi$.

As a simple yet important fragment of DEL, the *Public Announcement Logic (*PAL*)* [Pla89, GG97] is usually presented as follows:

$$\phi \quad ::= \quad \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid K_i\phi \mid [!\psi]\phi$$

where $K_i\phi$ and $[!\psi]\phi$ are equivalent to $[i]\phi$ and $[\mathsf{A}_{!\psi}]\phi$ in DEL respectively.

As for the expressivity of DEL, [vBvEK06] showed that adding product updates to PDL does not increase the expressive power of PDL:

2.3.4. F . ([vBvEK06]) For any DEL formula ϕ there is a PDL formula ϕ' such that for all pointed Kripke models $\mathcal{M}, s : \mathcal{M}, s \vDash_{\mathtt{DEL}} \phi \iff \mathcal{M}, s \vDash_{\mathtt{PDL}} \phi'$. ⁂

Part I

Logics of Epistemic Protocols

Chapter 3

Meta-knowledge Matters

3.1 Introduction

Public announcements are the simplest and best studied communication methods in epistemic logic [vDvdHK07]. In this chapter, we focus on the epistemic protocols that use public announcements as the only communication methods. As we mentioned in the introduction, existing work in the framework of DEL represents epistemic protocols as explicit sets of (finite) sequences of epistemic events (see, e.g., [HY09, vBGHP09, Hos09]). However, as remarked in [PR03], an *explicit* set of sequences of events is an *extensional* notion of *common sense* protocols which are usually specified by a few rules governing the communications. Therefore a high-level, finitely representable and model-independent specification is preferable for epistemic protocols. This motivates us to represent epistemic protocols as (syntactic) programs and thus focus on a subclass of "regular protocols" in this chapter. This restriction allows us to define a dynamic epistemic logic where protocols and their consequences are both expressible in the language. Thus the formal specification and verification are unified in a logical framework.

As we motivated in Chapter 1, complications arise in the verification of epistemic protocols compared to the verification of normal protocols. The correctness of the epistemic protocols heavily relies on the assumptions of the agents' *meta-knowledge* about the protocol itself. In particular, if the intended goal of an epistemic protocol is to establish or prevent knowledge, then knowing that the protocol would fulfil the goal may affect the verification of the protocol. It is reasonable to assume that the protocol, its goal, and the underlying initial assumptions are commonly known by all the agents including possible adversaries. Based on the logic we propose, we can formally address the above subtlety and verify epistemic protocols under the assumption that the intended goal is common knowledge.

Moreover, the formal specification of epistemic protocols calls for a more careful study of the classic problems. For example, recall the Russian Cards Problem (RCP) in Example 1.1.1, where A and B want to safely inform each other of their own cards by using public communications only, with the presence of an adversary E. A

satisfactory protocol to realize this safe information exchange should specify what A and B should do under *any* initial card deals. However, in the previous studies of the Russian Cards Problem (e.g. [vD03, vDvdHvdMR06]), the focus was usually on a particular deal of the cards[1]. As we shall see, the framework developed in this chapter can help us design and verify protocols that work on arbitrary initial distributions. The formal discussion also reveals some further subtleties. For example, in the case of $RCP_{3.3.1}$ we can show that correct deterministic protocols that are executable on arbitrary initial distributions of cards do exist, but that they are necessarily biased with respect to single card occurrence in the announcement..

Related work The closest work to ours is [vD02], where, instead of using action models as in [BM04], the author specifies the epistemic events by programs involving atomic epistemic actions such as learning and testing (see also [vDvdHK03c] and [vDvdHK07, Chapter 5] for extensions). Compared to this approach, our focus is on the verification of epistemic protocols, i.e. sequences of epistemic events, which sit at a higher level than the events themselves. This difference is also reflected in the design of the languages. For example, iteration over epistemic events is crucial in our work, while it may not fit in a description of complex epistemic events.

Structure of the chapter An epistemic logic of protocol specification and verification is introduced in Section 3.2, whose model checking problem is shown to be decidable. Section 3.3 formally addresses the specification and verification of epistemic protocols. We show that if the meta-knowledge of the protocol is assumed, then the verification problem should be formalized as model checking a fixed point formula involving iterated announcements. We also define a notion of *universal verification* of epistemic protocols with respect to a model, in which case checking the common knowledge of the correctness of the protocol suffices. To demonstrate the use of our framework, we study the deterministic protocols for the Russian Cards Problem in Section 3.4.

3.2 Preliminaries

In this section we define an *Announcement Protocol Language* L_{AP} for specifying and verifying epistemic protocols with announcements only. Our language is based on test-free PDL but with public announcements as atomic programs. The choice of test-free PDL is based on the observation that each announcement $!\phi$ has an intrinsic guard $?\phi$ which is assumed to be common knowledge in this chapter. For announcements with non-intrinsic tests, e.g, $?K_i(p \wedge q) \cdot !K_i p$, see discussions in Section 3.5 and the next

[1] For example, [vDvdHvdMR06] focuses on announcements for the specific situation of the card deal (012.345.6). The authors mentioned that the model checking task of a protocol that provides an announcement for an arbitrary initial state takes much more time, but the protocol itself is not discussed in the paper.

chapter. The formulas of L_{AP} are defined as:

$$\begin{array}{lll} \phi & ::= & \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid [\pi]\phi \mid C_G\phi \\ \pi & ::= & !\phi \mid \pi_1 \cdot \pi_2 \mid \pi_1 + \pi_2 \mid \pi^* \end{array}$$

where p ranges over a set of basic propositions $\mathbf{P}$ and G is a subset of the set of agents $\mathbf{I}$.

We write $K_i\phi$ for $C_{\{i\}}\phi$ and $C\phi$ for $C_{\mathbf{I}}\phi$. Intuitively, $[\pi]\phi$ expresses "after any possible run of the protocol π, ϕ holds". We write $\langle\!|\pi|\!\rangle\phi$ for $[\pi]\phi \wedge \langle\pi\rangle\phi$. Thus $\langle\!|\pi|\!\rangle\phi$ says "protocol π is executable and after every possible run ϕ holds". Moreover, we use $!_i\phi$ as the abbreviation for the announcement $!K_i\phi$. Intuitively, $!_i\phi$ is a public announcement of ϕ by agent i while $!\phi$ models an *external* announcement from the environment (e.g., the role of *Father* in the Muddy Children example). π is called an *n-step* protocol if $\pi = \pi_1 \cdot \pi_2 \cdots \pi_n$ and for $i \leq n : \pi_i$ does not contain operators * and $\cdot$.

Given an S5 model $\mathcal{M} = (S, \mathbf{P}, \mathbf{I}, \sim, V)$, the truth value of a L_{AP} formula ϕ at a state s in $\mathcal{M}$ is defined as:

$$\begin{array}{rcl} \mathcal{M}, s \vDash p & \Leftrightarrow & p \in V(s) \\ \mathcal{M}, s \vDash \neg\phi & \Leftrightarrow & \mathcal{M}, s \nvDash \phi \\ \mathcal{M}, s \vDash \phi \wedge \psi & \Leftrightarrow & \mathcal{M}, s \vDash \phi \text{ and } \mathcal{M}, s \vDash \psi \\ \mathcal{M}, s \vDash C_G\psi & \Leftrightarrow & \forall t : s \sim_G t \implies \mathcal{M}, t \vDash \psi \\ \mathcal{M}, s \vDash [\pi]\phi & \Leftrightarrow & \text{for all } \mathcal{M}', s : (\mathcal{M}, s)[\![\pi]\!](\mathcal{M}', s) \text{ implies } \mathcal{M}', s \vDash \phi \end{array}$$

where $\sim_G = (\bigcup_{i\in G} \sim_i)^\star$ and π are the epistemic programs functioning as *model changers*:

$$\begin{array}{rcl} (\mathcal{M}, s)[\![!\psi]\!](\mathcal{M}', s) & \Leftrightarrow & (\mathcal{M}', s) = (\mathcal{M}|_\psi, s) \\ (\mathcal{M}, s)[\![\pi_1 \cdot \pi_2]\!](\mathcal{M}', s) & \Leftrightarrow & (\mathcal{M}, s)[\![\pi_1]\!] \circ [\![\pi_2]\!](\mathcal{M}', s) \\ (\mathcal{M}, s)[\![\pi_1 + \pi_2]\!](\mathcal{M}', s) & \Leftrightarrow & (\mathcal{M}, s)[\![\pi_1]\!] \cup [\![\pi_2]\!](\mathcal{M}', s) \\ (\mathcal{M}, s)[\![\pi^*]\!](\mathcal{M}', s) & \Leftrightarrow & (\mathcal{M}, s)[\![\pi]\!]^\star(\mathcal{M}', s) \end{array}$$

where $\mathcal{M}|_\psi$ is the restriction of $\mathcal{M}$ to the states where ψ holds (see Section 2.3.3); $\circ, \cup$ and * express the usual composition, union and reflexive transitive closure on relations respectively. Viewing π as a regular expression over $\{!\phi \mid \phi \text{ is an } L_{AP} \text{ formula}\}$, we have:

$$(\mathcal{M}, s)[\![\pi]\!](\mathcal{M}', s) \iff \text{there is a sequence } w \in \mathcal{L}(\pi) \text{ such that } (\mathcal{M}, s)[\![w]\!](\mathcal{M}', s)$$

A *run* of π on a model $(\mathcal{M}, s)$ is a sequence w of announcements such that $w \in \mathcal{L}(\pi)$ and $(\mathcal{M}, s)[\![w]\!](\mathcal{M}', s)$ for some $(\mathcal{M}', s)$. For each run w on $(\mathcal{M}, s)$ there is a unique path of pointed models $(\mathcal{M}, s), (\mathcal{M}_1, s), \ldots, (\mathcal{M}_n, s)$ such that $\mathcal{M}_n = \mathcal{M}'$, which realizes w.

Iteration is important in specifying epistemic protocols with **while-do** loops (see for example, [vDvdHK07, pp.13] and [DW07]). We will also show, in the next section, that having the Kleene star in the specification language is crucial for verifying epistemic protocols. However, [MM05] showed that the satisfiability problem of a logic containing both iterated announcement ($(!\phi)^*$) and common knowledge operators is undecidable, even on finite models. Fortunately, the model checking problem of L_{AP} on finite models is decidable:

3.2.1. P . *Model checking L_{AP} on finite Kripke models is decidable.*

P The idea of the proof is based on the observation that the basic epistemic programs $!\phi$ of L_{AP} are *eliminative* in nature, which means that the transformed model is only getting smaller after a run of an epistemic program. The aim is then to reduce model checking ϕ on $(\mathcal{M}, s)$ to the standard PDL-style model checking of ϕ on a larger finite model $\mathcal{N}$, where programs π in ϕ are taken as labels of relations in $\mathcal{N}$. The state space of $\mathcal{N}$ is the set of all the possible pointed sub-models $(\mathcal{M}', s')$ of $\mathcal{M}$. We define $(\mathcal{M}', s') \sim_i^{\mathcal{N}} (\mathcal{M}'', s'') \iff (\mathcal{M}' = \mathcal{M}''$ and $s' \sim_i s''$ in $\mathcal{M}')$. We let the valuation $V_{\mathcal{N}}((\mathcal{M}', s')) = V_{\mathcal{M}'}(s')$. Now we are ready to compute all the corresponding relations of π in $\mathcal{N}$ by usual treatments in PDL model checking algorithms for operators $\cdot, +$ and * and the following operation to deal with $!\phi'$: $\mathcal{M}', s' \to_{!\phi'} \mathcal{M}'', s''$ iff $\mathcal{M}'' = \mathcal{M}'|_{\phi'}$ and $s' = s''$. To compute $\mathcal{M}'|_{\phi'}$ we need to call the model checking algorithm again but since ϕ' (a subformula of ϕ) is strictly simpler than ϕ, we will finally arrive at a situation that can be handled by the PDL model checking algorithm. ⋈

3.3 Announcement Protocol and Verification

In this section, we specify the announcement protocols and address their verification problem formally.

To verify an epistemic protocol, it is important to specify the assumptions about the initial setting in which the protocol is to be executed. For example, a protocol for $RCP_{2.2.1}$ (see Example 1.1.1) is expected to be run in the situation where five cards are given to three agents according to the distribution (2.2.1). It does not make sense to run the protocol in a situation with less agents or more cards. As observed in [FHMV97], it is crucial to make the distinction between the protocol (as the rules governing the actions) and the setting in which it is to be executed. In this chapter, we take epistemic programs as protocols and use a set of logical formulas to specify the *initial assumptions* explicitly. The verification of a protocol w.r.t. the intended goal should then be performed against initial models satisfying such assumptions. Based on the above consideration, we let the protocol specification include not only the protocol and its intended goal but also the initial assumptions:

3.3.1. D . **(Protocol Specification)** A *protocol specification* Prot is a triple $\langle \pi, \phi, T \rangle$ where the protocol π is an epistemic program in L_{AP}, ϕ is a program-free epistemic formula of L_{AP} serving as the intended *goal* of the protocol, and T is a set of program-free epistemic formulas of L_{AP} defining the initial assumptions of the protocol. ♏

A protocol specification Prot is *deterministic* if on any pointed model that satisfies $T_{\texttt{Prot}}$, there is at most one run of $\pi_{\texttt{Prot}}$. It is called *non-deterministic* if it is not deterministic. We also say $\pi_{\texttt{Prot}}$ is a *deterministic protocol* if Prot is deterministic.

Note that our definition of *determinism* is not based on the syntactic form of π: we allow nondeterministic choices + in a deterministic specification. Let $has_j x$ be the basic proposition meaning: agent j has card x. A simple example of a deterministic protocol specification in a card game setting is:

$$\langle (!_A has_A c + !_A has_A d), K_B has_A c \vee K_B has_A d, \{(has_A c \wedge \neg has_A d) \vee (\neg has_A c \wedge has_A d)\}\rangle$$

where agent A is to announce his card according to the protocol, in order to let B know his card, under the setting in which A can only have one of the two cards c and d. Note that $has_A c$ and $has_A d$ are not logically exclusive, thus can be both true on *some* model. However, such model is excluded by the initial assumption: $(has_A c \wedge \neg has_A d) \vee (\neg has_A c \wedge has_A d)$. Therefore the above specification is deterministic according to our definition.

Intuitively, a *(complete) verification* of a protocol specification `Prot` should check whether the goal ϕ holds after *any* execution of the protocol $\pi_{\texttt{Prot}}$ on *any* model that satisfies the initial assumptions $T_{\texttt{Prot}}$. Formally, we need to check:

$$T_{\texttt{Prot}} \models [\pi_{\texttt{Prot}}]\phi_{\texttt{Prot}}$$

If $T_{\texttt{Prot}}$ is a finite set then the above complete verification problem amounts to checking the satisfiability of the L_{AP}-formula $\bigwedge T_{\texttt{Prot}} \rightarrow [\pi_{\texttt{Prot}}]\phi_{\texttt{Prot}}$. However, as we mentioned in the previous section, the satisfiability problem for L_{AP} is undecidable even on finite models. On the other hand, whether complete verification is necessary is actually in question. In practice, we often focus on particular initial models that satisfy T, since T may not be a complete characterization of the intended informal initial requirements we have in mind, namely, there can be unintended models that also satisfy T. The ideal case is that the set T has a unique model, and this model can be generated by a certain method. Such issues related to modelling will be addressed in Part II.

In this chapter, we focus on the verification problem of a protocol specification `Prot` against a given pointed model $(\mathcal{M}, s)$ that satisfies $T_{\texttt{Prot}}$. The verification problem then amounts to the following model checking problem:

$$\mathcal{M}, s \models [\pi_{\texttt{Prot}}]\phi_{\texttt{Prot}}$$

In some scenarios (e.g., Russian Cards), we are interested in verifying a protocol *universally* in a certain model $\mathcal{M}$, where each state of $\mathcal{M}$ represents a particular initial distribution of information (e.g., a random card deal). In such cases, the protocol is also required to be executable under arbitrary initial distribution of the information. This *universal verification* of `Prot` against $\mathcal{M}$ can be formalized as the model checking problem:

$$\mathcal{M} \models \langle\!\langle \pi_{\texttt{Prot}} \rangle\!\rangle \phi_{\texttt{Prot}}$$

namely, for all $s \in S_{\mathcal{M}}$: model checking $\mathcal{M}, s \models \langle \pi_{\texttt{Prot}} \rangle \phi_{\texttt{Prot}} \wedge [\pi_{\texttt{Prot}}]\phi_{\texttt{Prot}}$.

As we motivated in Chapter 1, the verification of an epistemic protocol is more subtle than it seems to be: the meta-knowledge of the protocol itself may affect the verification of the protocol. We shall see that the above formalizations of the verification problem are not appropriate any more, if we assume the protocol specification is commonly known.

By commonly knowing a *protocol specification* $\texttt{Prot}$ we mean the following:

1. The protocol itself ($\pi_{\texttt{Prot}}$) is commonly known;
2. The intended goal of the protocol ($\phi_{\texttt{Prot}}$) is commonly known.
3. The set of initial assumptions ($T_{\texttt{Prot}}$) is commonly known.

If an epistemic protocol is publicly available and is to be used repeatedly, then it is natural to assume the above. Therefore a rigid verification of an epistemic protocol should be undertaken under these meta-knowledge assumptions. Note that the third assumption can be fulfilled by letting the formulas in T be of the form $C\psi$. We will address the issues about the second assumption in Chapter 4. In this chapter we focus on the first assumption and address the verification problem under this assumption.

Let us start from an observation made in [vD03] that checking $\mathcal{M}, s \vDash [!\pi]\phi$ is sometimes not sufficient, even for a single step protocol $\pi = !\psi$ aiming at establishing ϕ. As we saw in Example 1.1.1, if the agents know the intended goal of the protocol then they will assume that others do not perform actions which can not lead to the goal. The knowledge assumption about the intended goal lets the agents be able to reason more, which may destroy the correctness of the protocol established without the assumption of agents' knowing the goal.

To incorporate this knowledge assumption in the current framework, a straightforward idea is to just announce the intended goal of the protocol thus make it commonly known. In [vD03], the author proposed that, in a Russian Cards setting, the verification of a protocol with the intended goal ϕ should be undertaken while an announcement $!\psi$ is interpreted as more than just announcing ψ [2]:

$$\mathcal{M}, w \vDash [!(\psi \wedge [!\psi]\phi)]\phi$$

The idea is that by announcing ψ as well as the intended effect of the announcement ψ, we may verify the correctness of the protocol under the assumption that agents know the goal. However, if the correctness of $[!(\psi \wedge [!\psi]\phi)]\phi$ is now assumed and known by agents, we still need to make sure that knowing *this* again does not affect the correctness of the protocol. We can iterate such reasoning *ad infinitum*.

Now let us consider an arbitrary protocol π in L_{AP} and a corresponding goal ϕ. We define:

[2] In the original setting of [vD03], it is suggested that the announcement should be formalized by a *Gricean reading*: the announcement of $!\psi$ by agent a aiming at establishing ψ is formalized as $!(K_a\psi \wedge [!K_a\psi]K_a\phi)$ (the so-called "safe communication"). We omit the details in [vD03] that are relevant to the context of Russian Cards problem.

- $\eta_0 = [\pi]\phi$
- $\eta_{i+1} = [!(\eta_0 \wedge \cdots \wedge \eta_i)][\pi]\phi$

We can simplify η_{i+1} by making use of the following valid formula

$$[!(\psi \wedge [!\psi]\phi)]\chi \leftrightarrow [!\psi][!\phi]\chi \quad (\star)$$

3.3.2. P . $\eta_{i+1} = \underbrace{[!([\pi]\phi) \cdot !([\pi]\phi) \cdots !([\pi]\phi)}_{i}][\pi]\phi$

P Since

$$\eta_{i+1} = [!(\eta_0 \wedge \cdots \wedge \eta_i)]\phi = [!(\eta_0 \wedge \cdots \wedge \eta_{i-1} \wedge [!(\eta_0 \wedge \cdots \wedge \eta_{i-1})][\pi]\phi)][\pi]\phi.$$

By $(\star)$, it is not hard to see that

$$\eta_{i+1} = [!(\eta_0 \wedge \cdots \wedge \eta_{i-1})][![\pi]\phi][\pi]\phi.$$

Repeatedly applying the transformation, we have

$$\eta_{i+1} = \underbrace{[![\pi]\phi] \cdots [![\pi]\phi]}_{i}[\pi]\phi = \underbrace{[!([\pi]\phi) \cdots !([\pi]\phi)}_{i}][\pi]\phi.$$

⋈

Intuitively, we need to check η_i for each i, since there are cases where all the η_i are logically different. To see this, recall the Muddy Children scenario 1.1.2, where we showed that if there are n muddy children and it is commonly known that at least one of them is muddy, then the muddy children will only get to know that they are muddy after announcing "we don't know" for $n-1$ times. Now let π_d be the dummy announcement $!\top$ and let ϕ_n be the formula that expresses "all the n children do not know whether they are muddy or not". It is clear that $[\pi_d]\phi_n \leftrightarrow \phi_n$ is valid. The above analysis of the Muddy Children scenario tells us that for any $i > 0$, there is always a pointed model $(\mathcal{M}_i, s)$ such that:

$$\mathcal{M}_i, s \vDash \underbrace{[!([\pi_d]\phi_{i+2}) \cdots !([\pi_d]\phi_{i+2})}_{i}][\pi_d]\phi_{i+2},$$

but

$$\mathcal{M}_i, s \nvDash \underbrace{[!([\pi_d]\phi_{i+2}) \cdots !([\pi_d]\phi_{i+2})}_{i+1}][\pi_d]\phi_{i+2}$$

It is not hard to see that $[(![\pi]\phi)^*][\pi]\phi$ expresses exactly the infinite conjunction of all the η_i. Thus to verify the protocol π under the assumption of the common

knowledge of the goal of the protocols we need to model check the fixed point formula $[(!\![\pi]\phi)^*][\pi]\phi$ instead of $[\pi]\phi$.

For universal verification, we let $\eta'_0 = \langle\!\langle\pi\rangle\!\rangle\phi$ and $\eta'_{i+1} = [!(\eta_0 \wedge \cdots \wedge \eta_i)]\langle\!\langle\pi\rangle\!\rangle\phi$. Similarly, we can show:

3.3.3. P . $\eta'_{i+1} = [\underbrace{!(\langle\!\langle\pi\rangle\!\rangle\phi)\cdots!(\langle\!\langle\pi\rangle\!\rangle\phi)}_{i}]\langle\!\langle\pi\rangle\!\rangle\phi$

Therefore universal verification under the knowledge assumption of the intended goal amounts to checking $\mathcal{M} \vDash [(!\langle\!\langle\pi\rangle\!\rangle\phi)^*]\langle\!\langle\pi\rangle\!\rangle\phi$ which can be simplified as follows:

3.3.4. P .

$$\mathcal{M} \vDash [(!\langle\!\langle\pi\rangle\!\rangle\phi)^*]\langle\!\langle\pi\rangle\!\rangle\phi \iff \mathcal{M} \vDash \langle\!\langle\pi\rangle\!\rangle\phi$$

P $\Rightarrow$ is trivial. For $\Leftarrow$: Suppose $\mathcal{M} \vDash \langle\!\langle\pi\rangle\!\rangle\phi$, then announcing $\langle\!\langle\pi\rangle\!\rangle\phi$ does not change model $\mathcal{M}$, thus the truth values of the formulas are preserved after any iteration of the announcement $\langle\!\langle\pi\rangle\!\rangle\phi$[3]. Therefore $\mathcal{M} \vDash [(!\langle\!\langle\pi\rangle\!\rangle\phi)^*]\langle\!\langle\pi\rangle\!\rangle\phi$. ⋈

We may simplify the verification problem further when looking at *connected* models where every state is connected to all the other states by some path of relations:

3.3.5. P . *If $\mathcal{M}$ is a connected model then for any s in $\mathcal{M}$,*

$$\mathcal{M} \vDash [(!\langle\!\langle\pi\rangle\!\rangle\phi)^*][\pi]\phi \iff \mathcal{M}, s \vDash C\langle\!\langle\pi\rangle\!\rangle\phi$$

P Immediate from Proposition 3.3.3 and the connectivity. ⋈

This means that to verify a protocol is correct under any possible initial information distribution is to check the common knowledge of the correctness of the protocol at an *arbitrary* initial situation. Note that in most applications, the initial epistemic model under consideration is indeed connected, assuming that the agents are perfect reasoners who can imagine the possibilities others may think.

In summary, we define the verification of a protocol specification as follows:

3.3.6. D . **(Verification under Common Knowledge)** Given a protocol specification `Prot` and a pointed model $(\mathcal{M}, s)$ satisfying $T_{\texttt{Prot}}$, the verification of `Prot` against $(\mathcal{M}, s)$ under the common knowledge of the intended goal ϕ is checking:

$$\mathcal{M}, s \vDash [(!\![\pi_{\texttt{Prot}}]\phi_{\texttt{Prot}})^*]\phi_{\texttt{Prot}}$$

[3] cf. [vDK06] for a more general study on *successful updates*.

The *universal verification* of Prot against model $\mathcal{M}, s$ under the common knowledge of the intended goal ϕ is checking:

$$\mathcal{M} \vDash \langle\!\langle \pi \rangle\!\rangle \phi_{\texttt{Prot}}$$

When the model $\mathcal{M}$ is connected, the universal verification problem is equivalent to checking the common knowledge of the correctness of the protocol:

$$\mathcal{M}, s \vDash C\langle\!\langle \pi_{\texttt{Prot}} \rangle\!\rangle \phi_{\texttt{Prot}}$$

$\mathfrak{m}$

Note that since the universal verification problem reduces to simply checking $\mathcal{M} \vDash \langle\!\langle \pi \rangle\!\rangle \phi_{\texttt{Prot}}$, then given a π without iteration the universal verification can also be done in the standard framework of PAL (see page 17).

3.4 Deterministic Protocols for $RCP_{3.3.1}$

In this section we take the Russian Cards Problem as an example to demonstrate the use of our framework in designing and verifying epistemic protocols. In particular, we study deterministic 2-step protocols for $RCP_{3,3,1}$ that can be executed under an *arbitrary* initial distribution of cards. We show that there is a correct, deterministic protocol for $RCP_{3.3.1}$, however with uneven occurrences of the cards in the announcements (i.e. some cards occur more often than others in the announcements).

Recall that in the setting of $RCP_{n.n.k}$, $2n + k$ cards are distributed among three agents according to the distribution $(n.n.k)$ and the agents can only see their own cards (cf. Example 1.1.1). We first formalize the initial assumptions in this setting.

Let $\mathbf{I} = \{A, B, E\}$ be the set of agents, $Dk_{n.n.k} = \{0, 1, \ldots, 2n+k-1\}$ be the set of $2n+k$ cards, Hs_h be the set of h-hands (e.g., $Hs_3 = \{\{x, y, z\} \mid x, y, z \in Dk_{n.n.k} \text{ are different }\}$). Let us consider a tailored set of basic propositions: $\mathbf{P}_{has} = \{has_i x \mid i \in \mathbf{I}, x \in Dk_{n.n.k}\}$ where $has_i x$ intuitively expresses that agent i has card x. We use $has_i X$ as the abbreviation of $\bigwedge_{x \in X} has_i x$. The initial assumptions are formalized as $T_{n.n.k} = \{C\phi\}$ where:

$$\phi = \texttt{OneCardInOneP} \wedge \texttt{EkCards} \wedge \texttt{ABnCards} \wedge \texttt{KnowThyself} \wedge \texttt{DontKnowOthers}$$

- $\texttt{EkCards} := \bigvee_{X \in Hs_k} has_E X$;
- $\texttt{ABnCards} := \bigwedge_{i \in \{A,B\}} \bigvee_{X \in Hs_n} has_i X$;
- $\texttt{OneCardInOneP} := \bigwedge_{i \neq j} (\bigwedge_{x \in Dk_{n.n.k}} (has_i x \to \neg has_j x))$;
- $\texttt{KnowThyself} := \bigwedge_{i \in \mathbf{I}} \bigwedge_{x \in Dk_{n.n.k}} (has_i x \to K_i has_i x)$;
- $\texttt{DontKnowOthers} := \bigwedge_{i \neq j} \bigwedge_{x \in Dk_{n.n.k}} (has_i x \to (\widehat{K}_j has_i x \wedge \widehat{K}_j \neg has_i x)$[4].

[4]Recall that $\widehat{K}\phi$ denotes $\neg K_i \neg \phi$ (i thinks that ϕ is possible).

Intuitively, the formula in T says that it is commonly known that $2n + k$ cards are distributed among three agents according to the distribution $(n.n.k)$ and the agents only know their own cards.

Now we build an initial model which satisfies the above initial assumptions. Let $\mathcal{M}_{n.n.k} = \{S, \mathbf{P}_{has}, \mathbf{I}, \sim, V\}$ where:

- $S = \{(X, Y, Z) \mid X, Y \in Hs_n, Z \in Hs_k, X \cup Y \cup Z = Dk_{n.n.k}\}$;
- $s \sim_i t \iff s_i = t_i$ where $(X, Y, Z)_A = X$, $(X, Y, Z)_B = Y$ and $(X, Y, Z)_E = Z$;
- $V(s) = \{has_i x \mid i \in \mathbf{I}, x \in Dk_{n.n.k}$ and x appears in $s_i\}$.

Note that $\mathcal{M}_{n.n.k}$ is clearly connected. It can be verified that $\mathcal{M}_{n.n.k}$ satisfies $T_{n.n.k}$.

A correct protocol for $RCP_{n.n.k}$ should let A and B eventually know each other's cards (thus also E's cards) while keeping E ignorant about A's and B's cards. This can be formalized as $\phi_{n.n.k} = \phi_1 \wedge \phi_2 \wedge \phi_3$ where:

$$\phi_1 = \bigwedge_{x \in Dk_{n.n.k}} (has_A x \to K_B has_A x)$$

$$\phi_2 = \bigwedge_{x \in Dk_{n.n.k}} (has_B x \to K_A has_B x)$$

$$\phi_3 = \bigwedge_{x \in Dk_{n.n.k}} ((has_A x \to \neg K_E has_A x) \wedge (has_B x \to \neg K_E has_B x))$$

Recall that in Example 1.1.1 we argue that knowing the correctness of the protocol may destroy the correctness of the protocol. Now we can make this claim formal: let $\pi = !(has_A 01 \vee has_A 23 \vee has_A 24 \vee has_A 34)$, it is easy to check that

$$\mathcal{M}_{2.2.1}, (01, 23, 4) \vDash [\pi](\phi_1 \wedge \neg K_E has_A 01)$$

but

$$\mathcal{M}_{2.2.1}, (01, 23, 4) \vDash [![\pi]\phi_1]K_E has_A 01$$

In the rest of this section, we focus on finding a protocol $\pi_{3.3.1}$ such that the following conditions are met:

1. $\pi_{3.3.1}$ is a two-step protocol in the form $\pi_1 \cdot \pi_2$ where $\pi_1 = !_A\psi_1 + \cdots + !_A\psi_m$ and $!_B\psi'_1 + \cdots + !_B\psi'_l$ for some ψ_i and ψ'_i.
2. $\langle \pi_{3.3.1}, \phi_{3.3.1}, T_{3.3.1} \rangle$ is deterministic;
3. $\mathcal{M}_{3.3.1} \vDash \langle\!\langle \pi_{3.3.1} \rangle\!\rangle (\phi_1 \wedge \phi_2 \wedge \phi_3)$;

We say a deterministic, 2-step protocol is *correct* for $RCP_{3.3.1}$ if condition (3) is met. Note that a protocol satisfying (2) can have at most one run at each of the states in $\mathcal{M}_{3.3.1}$. Therefore, assuming (2), checking (3) is then equivalent to checking:

$$\mathcal{M}_{3.3.1} \vDash \langle \pi_{3.3.1} \rangle (\phi_1 \wedge \phi_2 \wedge \phi_3)$$

The following proposition suggests that we can focus on the first step of the protocol with an intermediate goal $\phi_1 \wedge \phi_3$.

3.4.1. P . *If there is a one-step protocol π_1 such that $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle(\phi_1 \wedge \phi_3)$, then there is a π_2 such that $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \cdot \pi_2 \rangle\!\rangle(\phi_1 \wedge \phi_2 \wedge \phi_3)$.*

P Let $\pi_2 = \sum_{X \in Hs_k} !_B has_E X$, we show that if $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle(\phi_1 \wedge \phi_3)$ then $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \cdot \pi_2 \rangle\!\rangle(\phi_1 \wedge \phi_2 \wedge \phi_3)$. Let $K_i Cards_j$ be the abbreviation for $\bigwedge_{x \in Dk}(has_j x \to K_i has_j x)$, then $\phi_1 = K_B Cards_A$ and $\phi_2 = K_A Cards_B$. We can verify that if $i, j, l \in \mathbf{I}$ are different then $K_i Cards_j \to K_i Cards_l(\phi_2)$ is valid in any submodel of $\mathcal{M}_{n.n.k}$. Therefore it is easy to see that if $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle \phi_1$ then $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle K_B Cards_E$. Thus $\pi_1 \cdot \pi_2$ is executable and $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \cdot \pi_2 \rangle\!\rangle K_A Cards_E$. It follows that $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \cdot \pi_2 \rangle\!\rangle \phi_2$. Since ϕ_1 is clearly preserved after executing π_2, we only need to show $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \cdot \pi_2 \rangle\!\rangle \phi_3$. We claim that

$$\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle C(K_B Cards_E)$$

If the claim is true, then $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle K_E(K_B Cards_E)$ thus for any state $s = (X, Y, Z)$: $\mathcal{M}_{n.n.k}, s \vDash \langle\!\langle \pi_1 \rangle\!\rangle K_E(K_B has_E Z)$. Therefore truthfully announcing $K_B has_E Z$ for some $Z \in Hs_k$ does not change the distributions of hands that E considers possible. Therefore ϕ_3 should be preserved after π_2.

Now we prove the claim. First note that π_2 is clearly deterministic: it can have only one run when executed. Now given an arbitrary s in $\mathcal{M}_{n.n.k}$, if $s \sim_{\mathbf{I}} t$ in a model $\mathcal{M}'$ such that $(\mathcal{M}, s)[\![\pi_2]\!](\mathcal{M}', s)$ then we know there is a unique run w such that $w \in \mathcal{L}(\pi_2)$ and $(\mathcal{M}_{n.n.k}, s)[\![w]\!](\mathcal{M}', s)$ and $(\mathcal{M}_{n.n.k}, t)[\![w]\!](\mathcal{M}', t)$. Since $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle K_B Cards_E$ then $\mathcal{M}', t \vDash K_B Cards_E$. Therefore $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle C(K_B Cards_E)$ and this proves the claim. ⋈

To simplify the discussion, we can restrict the form of π_1 further w.l.o.g by adapting a result from [vD03], which states that to announce only A's alternative hands is enough.

3.4.2. P . *If $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle(\phi_1 \wedge \phi_3)$ then there is a π_1' such that $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1' \rangle\!\rangle(\phi_1 \wedge \phi_3)$ and*

$$\pi_1' = !_A Pa_0 + !_A Pa_1 + \cdots + !_A Pa_m$$

where Pa_i is of the form: $has_A X_0 \vee has_A X_1 \vee \cdots \vee has_A X_l$ with $\{X_0, \ldots, X_l\} \subseteq Hs_n$.

P The crucial observation for the proof is that for any formula ϕ such that $\mathcal{M}_{n.n.k}, s \vDash K_A \phi$ at some s, $\{t \mid \mathcal{M}_{n.n.k}, t \vDash \phi\}$ must be the union of some i-equivalence classes in $\mathcal{M}_{n.n.k}$ and each equivalence class of $\mathcal{M}_{n.n.k}$ can be characterized by a formula $has_A X$ for some hand $X \in Hs_n$. Therefore, for each ϕ such that $\mathcal{M}_{n.n.k}, s \vDash K_A \phi$ at some s there is a $\phi' = has_A X_0 \vee has_A X_1 \vee \cdots \vee has_A X_m$ for a set of hands $\{X_0, \ldots, X_m\} \subseteq Hs_n$ such that $\{t \mid \mathcal{M}_{n.n.k}, t \vDash \phi\} = \{t \mid \mathcal{M}_{n.n.k}, t \vDash \phi'\}$. ⋈

In the sequel, we sometimes abuse the notion of Pa_i by viewing it as a set of 3-hands with each hand represented by xyz for $\{x, y, z\} \subset Dk_{n.n.k}$ (the order does not matter). We now prove a lemma for later use in this section.

3.4.3. L . *If there is a deterministic protocol π_1 such that $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle(\phi_1 \wedge \phi_3)$ and $\pi_1 = !_A Pa_0 + !_A Pa_1 + \cdots + !_A Pa_m$ then*

1. *each possible hand appears only once in $Pa_0, !Pa_1, \ldots, !Pa_m$: for each $X \in Hs_n$: there is a unique $i \leq m$ such that $X \in Pa_m$.*
2. *any two hands in one announcement Pa_j can only share at most $n-k-1$ cards.*

P For (1): Since $\mathcal{M}_{n.n.k} \vDash \langle\!\langle \pi_1 \rangle\!\rangle \top$ and $\pi_1 = !_A Pa_0 + !_A Pa_1 + \cdots + !_A Pa_m$ we know that every hand should appear at least once. From the assumption that protocol π is deterministic, every hand can only appear once. In the following, given a hand X of A, let $Pa(X)$ be the set Pa_j such that $X \in Pa_j$.
For (2): To let B know A's cards after A's announcement, we should make sure that given A's hand X, for any B's hand Z, the alternatives in $Pa(X)$ will be ruled out. Namely, for any different hands $X, Y \in Pa(X)$, any hand $Z \subseteq Dk_{n.n.k} \backslash X$ that B may have: $Z \cap Y \neq \emptyset$. This means that for every two hands X, Y in Pa_i, the number of cards different from the cards in $X \cup Y$ must be less than n. Otherwise there is a possible hand Z which does not intersect with both X and Y. Thus, we have $|Dk_{n.n.k}| - |Y \cup X| < n$. Since $|Dk_{n.n.k}| = 2n+k$, $|Y \cup X| > n+k$. Note that $|X| = |Y| = n$, thus $|X \cap Y| < 2n-n-k = n-k$. ⋕

In the following we will concentrate on the specific case $RCP_{3.3.1}$. We first show that there is a deterministic protocol:

3.4.4. T . *There is a correct, 2-step deterministic protocol for $RCP_{3.3.1}$*[5].

P Let Pa_i be the following sets of 3-hands.

Pa_0 :	012	036	045	134	156	235	246
Pa_1 :	013	025	046	126	145	234	356
Pa_2 :	014	026	035	136	245		
Pa_3 :	015	024	123	256	346		
Pa_4 :	016	034	124	135	236	456	
Pa_5 :	023	056	125	146	345		

Let $\pi_1 = \sum_{0 \leq i \leq 5} (!_A Pa_i)$. Note that π_1 is clearly a deterministic protocol: if $has_A 025$ is true then A should announce Pa_2[6]:

$$has_A 013 \vee has_A 025 \vee has_A 046 \vee has_A 126 \vee has_A 145 \vee has_A 234 \vee has_A 356$$

Moreover we can verify that $\mathcal{M}_{3.3.1} \vDash \langle\!\langle \pi \rangle\!\rangle(\phi_1 \wedge \phi_3)$. Thus from Proposition 3.4.1, there is a deterministic 2-step protocol for $RCP_{3.3.1}$. ⋕

However, the above protocol is *biased* in the sense that not all the cards appear evenly in each announcement Pa_i (e.g. in Pa_2, 0 appears three times but others only

[5]The solution was found with the help of the Alloy Analyzer [Jac02] based on Lemma 3.4.3, see Appendix A for the code.
[6]We omit the K_A in front of each $has_A X$.

appear twice). Thus E may learn that some card is more likely than other cards to be held by A. The authors of [AvDR09] showed that we can resolve the harm of *unbiased* protocols by making use of probabilistic selections of announcements. Here we are interested in whether we can have an unbiased deterministic protocol with the same number of occurrences of cards in the alternative announcements. Here are some properties of the unbiased protocol for $RCP_{3.3.1}$, if it exists:

3.4.5. L . *The first step of an unbiased deterministic protocol for $RCP_{3.3.1}$ must satisfy the following:*

1. *each announcement Pa_j must be a set of 7 alternative hands.*
2. *there are in total 5 alternative announcements in the protocol: $Pa_0, \dots Pa_4$.*
3. *every two hands in the same announcement Pa_i have exactly one card in common.*

P For (1): Suppose all the cards appear evenly (suppose g times) in an announcement with h hands. Since each hand has three cards then $3h = 7g$. So the minimal h is 7, and each card appears 3 times. We claim that if h is greater than 7 then there must be two hands which share more than 1 card. Note that there are only $C_7^2 = 21$ different pairs of cards and the three cards in each hand can constitute 3 different pairs. From the second statement in Lemma 3.4.3 any two hands should not have a pair of cards in common, thus 7 hands of three cards then "cover" all the possible 21 pairs of cards without repetition. Therefore adding one more hand of three cards must result in two hands sharing two cards in common.
For (2): From the first statement in Lemma 3.4.3, we know the $C_7^3 = 35$ hands should all appear in some Pa_i once. Thus from (1) the protocol should have 5 alternative announcements with 7 hands each.
For (3): Suppose there are two hands X, Y in an announcement Pa_j such that $X \cap Y = \emptyset$. Without loss of generality let $X = 123$ and $y = 456$. Since each of the possible 21 pairs should appear in some hand in the announcement Pa_i as argued in (1), then the hands $14c$ and $24c'$ must also appear in Pa_j for some cards c, c'. Since every two hands should not have two cards in common and $1, 2 \in X$ and $4 \in Y$, $c, c' \notin X \cup Y$ thus $c = c' = 0$. However then $14c$ and $24c'$ have two cards in common, contradiction. ⋈

In the following, we show that there is no deterministic protocol which is unbiased.

3.4.6. T . *There is no correct deterministic 2-step protocol which is unbiased for $RCP_{3.3.1}$.*

P We prove the theorem by proving the following stronger claim first:
There are no 3 sets with 7 hands, such that: (1) all the 21 hands that appear in these sets are different; (2) every two hands in the same set have one and only one common card; (3) all the cards appear evenly in every set.
Suppose towards a contradiction that there exist 3 sets Pa_2, Pa_3, Pa_4 satisfying (1), (2) and (3). Assume without loss of generality that $012 \in Pa_2$, $013 \in Pa_3$ and $014 \in Pa_4$.

Since for any $x \in \{2,3,4\}$: $01x \in Pa_x$, then from (2) and (3) we know that $xab, xcd \in Pa_x$, $0ac, 0bd \in Pa_x$ and $1ad, 1bc \in Pa_x$ such that $a,b,c,d \in Dk_{3.3.1}\backslash\{0,1,x\}$ are all different. Now we can list all the hands in Pa_x (with $p_i^x \in \{0,1,x\}$ and $p_i^x \neq p_j^x$ if $i \neq j$):

for Pa_2 : 012	$p_1^2 34, p_1^2 56$	$p_2^2 35, p_2^2 46$	$p_3^2 36, p_3^2 45$
for Pa_3 : 013	$p_1^3 24, p_1^3 56$	$p_2^3 25, p_2^3 46$	$p_3^3 26, p_3^3 45$
for Pa_4 : 014	$p_1^4 23, p_1^4 56$	$p_2^4 25, p_2^4 36$	$p_3^4 26, p_3^4 35$

First, there exists an $x \in \{2,3,4\}$ such that $p_1^x = x$, otherwise there must be either two 056 or two 156 in Pa_2, Pa_3, Pa_4, contradictory to (1). Suppose w.l.o.g. that $p_1^2 = 2$. It is easy to see that $p_1^3 \neq 3$ and $p_1^4 \neq 4$, otherwise 234 appears twice in Pa_2, Pa_3, Pa_4. Moreover, since $p_1^3 56$ is in Pa_3 and $p_1^4 56$ is in Pa_4, $p_1^3 \neq p_1^4$. Suppose w.l.g. that $p_2^3 = 3$. Then $p_2^4 \neq 4$ since $346 \in Pa_3$, therefore $p_3^4 = 4$. Now let us fill in the known p_i^x as following:

for Pa_2 : 012	234, 256	$p_2^2 35, p_2^2 46$	$p_3^2 36, p_3^2 45$
for Pa_3 : 013	$p_1^3 24, p_1^3 56$	325, 346	$p_3^3 26, p_3^3 45$
for Pa_4 : 014	$p_1^4 23, p_1^4 56$	$p_2^4 25, p_2^4 36$	426, 435

Now we have $p_2^2, p_3^2, p_1^3, p_3^3, p_1^4, p_2^4 \in \{0,1\}$. We showed that $p_1^3 \neq p_1^4$, thus $p_3^3 \neq p_2^4$ (remember that $p_3^3 \neq p_1^3$ and $p_1^4 \neq p_2^4$). Since $p_3^2, p_3^3, p_2^4 \in \{0,1\}$ then from $p_3^3 \neq p_2^4$ we have: $p_3^2 = p_3^3$ or $p_3^2 = p_2^4$, but in any case, there will be one hand that appears in two announcements, contradiction.
The Theorem follows from the above claim and Lemma 3.4.5. ⋈

3.5 Conclusion and Discussion

The logical framework developed in this chapter made it possible to formally specify and verify announcement protocols under the assumption that the intended goals of the protocols are commonly known. More examples of the protocol verification using this framework can be found in [WKvE09]. In this chapter, we have restricted ourselves to protocols involving public announcement only. This restriction gives us a straightforward model checking algorithm. As we mentioned before, the Kleene star over announcements is the source of the undecidability, thus it is important to understand the behaviour of the Kleene star better. It is shown in [GK03] that the *limit* of an iterated announcement corresponds to a *deflationary* fixed point on a non-monotonic function, thus can be expressed by a formula in the *Modal Iteration Calculus* (MIC). However, it is not clear whether our logic can be translated back to MIC. Moreover, how to fit the iteration of more general action models in a fixed point logic is also open. Some moderate extensions of the language are subgroup announcements and actions for factual change (cf. [vDK08, vBvEK06]). Another interesting extension is to add concurrent actions for modelling simultaneous announcements in a distributed setting which will be addressed in Chapter 5. The restriction to

public announcements also gives some hope for the synthesis problem of epistemic protocols (cf. the ideas mentioned [ABvDS09]).

Another important restriction in the current framework is that we exclude explicit tests in the program language of L_{AP} (we do have an intrinsic test $?\phi$ implicitly for each announcement $!\phi$). Therefore we can not specify guarded announcements like $?K_i(p \wedge q) \cdot !Kp$, where the test is not the announcement itself. A straightforward idea is to introduce the tests with its usual semantics as in PDL. However, as we shall see in Chapter 4, explicit tests require more careful modifications of the semantics, due to the fact that non-intrinsic tests are not publicly observable. As we remarked in the introduction, the tests can explicitly model the preconditions of the actions which make the actions carry more meaning than they seem to have. The link between the precondition and the actions should be established by protocols known to the agents. In Chapter 4, we will have an extensive discussion on how to know and change the protocols.

Chapter 4

Logics of Knowledge and Protocol Change

4.1 Introduction

In Chapter 3 we have shown that knowing that an epistemic protocol would fulfil a certain goal may affect the verification of the protocol. In this chapter, we draw the attention to the knowledge and the dynamics of protocols. As we motivated in Chapter 1, knowing a protocol means *knowing what to do* [HF89] and *knowing the meaning carried by actions* according to the protocols [PR03]. In this chapter, we will make these two observations more precise. More importantly, we address the problem "how to know a protocol?" by modelling the dynamics of protocols.

Knowing what to do In the framework L_{AP} of the previous chapter, a promising candidate for expressing that an agent knows what to do according to an announcement protocol π is the formula $K_i\langle\pi\rangle\top$. However, due to the semantics of L_{AP}, $K_i\langle\pi\rangle\top$ only says that agent i knows that π *can* be executed at any world he considers possible according to the inherent preconditions of the announcements in π. For example, in the Muddy Children scenario (Example 1.1.2) the assumed protocol is to (repeatedly) announce whether you yourself are muddy, and clearly you know you can announce it. However, there are many other true propositions that could be announced by an agent. For example, $K_i\langle!_i m_j + !_i\neg m_j\rangle\top$ is valid in the model for $i \neq j$, but $!_i m_j + !_i\neg m_j$ is not part of the intended protocol. Clearly, we need a constraint telling us which announcements are in accordance with the protocol, in other words, we need to model the role of the *father* as in the original story of the Muddy Children .

The existing work on protocols in DEL enriches the epistemic models with explicit protocols such that the possible behaviours of agents are not only restricted by the inherent preconditions of epistemic events but also restricted by protocol information [HY09, vBGHP09, Hos09, HP10b]. This is similar to the treatment of protocols in ETL [HF89, PR03], where the temporal development of a system is generated from an initial situation by a commonly known protocol[1]. In this chapter we take a different approach: we precisely model the role of the father as in the Muddy Children

[1] However, the framework in [vBGHP09] can also handle the protocols which are not common known.

scenario by introducing *protocol announcements* $[!\pi]$ in the language. For example, we use $[!(a \cdot b)]\neg\langle b\rangle\phi$ to express: after the announcement of the protocol $a \cdot b$, b can not be executed as the first event[2]. The semantics of the language with protocol announcements is defined on standard Kripke models. The extra protocol information is only introduced by protocol announcements while evaluating a formula. Such an approach makes it possible to not only model the "installation" of the initial protocol explicitly but also to handle protocol changes during the execution of the current protocol: we model a true father who may change his mind.

The dynamics of protocols often occur in social interactions. For example, imagine that you were told to close the door and on your way to do it you are told again not to close it. Also as we mentioned in Chapter 1, someone from France may need to update his protocol on cheek kissing when living in Holland. As another example, let consider the yes-no questions which can be viewed as protocols announced by the questioner: answer "Yes!" or answer "No!". In dialogues, a well-trained spokesman may respond to a yes-no question by inserting yet another protocol: "before answering your question, tell me what you meant by ϕ."

Knowing what the actions mean The dynamics of the protocols that carry meanings for actions are even more interesting. Here is yet another example: the Chinese are *non-confrontational* in the sense that they will not overtly say "no", instead they say "I will think about it" or "we will see". For a western businessman, "we will see", according to the standard interpretation, means it is still possible. However, if he is updated with the Chinese protocol: $?p_{no} \cdot a_{\text{will-see}}$ then he should see this is just another way of saying "No". Note that in the standard DEL, the interpretations of events are fixed and implicitly assumed to be common knowledge, e.g., in PAL an announcement $!\phi$ is assumed to have an inherent meaning: ϕ is true. This is because the semantic objects (event models) are explicitly included in the syntax as in the general DEL framework. However, the same utterance $!\phi$ (syntax) may carry different meanings (semantics) as we have seen in the we-will-see example. A closer look at public announcements should separate the utterances and their meanings. In fact, an utterance a only carries the meaning ϕ if the hearer knows that the protocol $?\phi \cdot a$ is carried out (cf. [PR03] for a detailed rationale).

To handle the protocols that carry meanings for actions, it is inevitable to introduce tests in the protocol programming language. Intuitively, the tests are not observable by the agents, unless announced previously, e.g. $[?p_{no} \cdot a_{\text{will-see}}]K_i p_{no}$ should not be valid while $[!(?p_{no} \cdot a_{\text{will-see}})][?p_{no} \cdot a_{\text{will-see}}]K_i p_{no}$ should be valid. We define the formal semantics for this enriched language in this chapter.

[2]Here we assume that if the protocol is announced then it is followed by all the agents. See [PS10] for an interesting discussion on "knowingly following the protocol" by agents in a setting of imperfect information.

Related work Besides the work on DEL protocols we mentioned earlier [HY09, vBGHP09, Hos09], we list some more related work here. Process logic [Pra79, HKP82] extends PDL in adding modalities to specify *progressive* behaviours like "during the execution of program π, ϕ will be true at some point." In this chapter, we not only reason about properties in the middle of an execution of a protocol but also handle the protocol changes during the execution. Moreover, the semantics of our logics will be defined on the states in the models, instead of on paths as in [Pra79, HKP82]. Aucher [Auc09] also proposed an extended DEL, however, the reasoning of the ongoing events is facilitated, in a setting without protocols. Unlike the work of switching strategies in the context of games [PRS09], the change of our protocols can be made at any time without being planned and we also incorporate knowledge in the discussions.

Our treatment for the events that carry meaning is inspired by [PR03], in which the authors give a semantics for messages (events) according to the underlying protocol in the ETL framework. However, we can explicitly express the protocol in the language and design a semantics for the dynamics of protocols. The later feature also distinguishes our work from the work using regular expressions as protocols [BS08b, Mey87, WKvE09]. The semantics of our logics are defined on standard Kripke models, but unlike PAL, we do not use a model-changing semantics for our $[!\pi]$ operator. This gives us the possibility to model radical protocol changes which are not based on the previous ones.

Structure of the chapter In this chapter we develop three logics featuring protocol changing operators, which can all be translated to PDL, but with certain convenience for modelling purposes. As an appetizer, we start in Section 4.2 with the first logic PDL$^!$, a version of test-free PDL equipped with protocol announcements $[!\pi]$. The semantics is given in a non-standard style by using *modes* of satisfaction relations [Gab02, Wan06]. Section 4.3 extends the language PDL$^!$ with knowledge and Boolean tests to handle the cases like the above we-will-see example where knowing a protocol gives meanings to actions. Finally, Section 4.4 proposes a logic PDL$^{\boxtimes}$ with automata as update models, which is powerful for modelling more complicated protocols and various interactions among agents. PDL$^{\boxtimes}$ can be viewed as an extension of DEL [BM04, KvB04, vBvEK06] with more liberal updates.

4.2 Basic Logic PDL$^!$

The formulas of PDL$^!$ are built from the set of basic proposition letters $\mathbf{P}$ and the set of atomic actions $\mathbf{\Sigma}$ as follows:

$$\begin{array}{lll} \phi & ::= & \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid [\pi]\phi \mid [!\pi]\phi \\ \pi & ::= & \mathbf{1} \mid \mathbf{0} \mid a \mid \pi \cdot \pi \mid \pi + \pi \mid \pi^* \end{array}$$

where $p \in \mathbf{P}$ and $a \in \mathbf{\Sigma}$. The intended meaning of the formulas is mostly as in PDL, but "in context" of the protocol constraints: $[\pi]\phi$ now says that "after any run of the

program π which is allowed by the *current protocol*, ϕ holds". The new formula $[!\pi]\phi$ expresses "after the announcement of the new protocol π, ϕ holds."

To give the semantics to $\mathtt{PDL}^!$, we first recall a useful notion of regular expressions. The *input derivative* $\pi\backslash a$ of the regular expression $\pi \in Reg_{\mathbf{\Sigma}}$ is defined as $\mathcal{L}(\pi\backslash a) = \{v \mid av \in \mathcal{L}(\pi)\}$. With the output function $o : Reg_{\mathbf{\Sigma}} \to \{\mathbf{0}, \mathbf{1}\}$ we can axiomatize $\backslash a$ (cf. [Brz64, Con71]):

$$
\begin{array}{ll}
\pi = o(\pi) + \sum_{a\in\mathbf{\Sigma}}(a \cdot \pi\backslash a) & \\
\mathbf{1}\backslash a = \mathbf{0}\backslash a = b\backslash a = \mathbf{0} \quad (a \neq b) & a\backslash a = \mathbf{1} \\
(\pi \cdot \pi')\backslash a = (\pi\backslash a) \cdot \pi' + o(\pi) \cdot (\pi'\backslash a) & (\pi + \pi')\backslash a = \pi\backslash a + \pi'\backslash a \\
(\pi)^*\backslash a = \pi\backslash a \cdot (\pi)^* & o(\pi \cdot \pi) = o(\pi) \cdot o(\pi') \\
o(\pi^*) = \mathbf{1} & o(\mathbf{1}) = \mathbf{1} \\
o(\mathbf{0}) = o(a) = \mathbf{0} & o(\pi + \pi') = o(\pi) + o(\pi')
\end{array}
$$

Given $w = a_0a_1 \cdots a_n \in \mathbf{\Sigma}^*$, let $\pi\backslash w = (\pi\backslash a_0)\backslash a_1 \cdots \backslash a_n$. It is clear that $\pi\backslash w = \{v \mid wv \in \mathcal{L}(\pi)\}$[3]. Together with the axioms of Kleene algebra [Koz91] we can syntactically derive $\pi\backslash w$ which is intuitively the *remaining* protocol of π after executing a run w. For example:

$$(a + (b \cdot c))^*\backslash b = (a\backslash b + (b \cdot c)\backslash b) \cdot (a + b \cdot c)^* = (\mathbf{0} + (\mathbf{1} \cdot c)) \cdot (a + b \cdot c)^* = c \cdot (a + (b \cdot c))^*$$

Note that in general we do not have $w \cdot (\pi\backslash w) = \pi$. We say w is *compliant with* π (notation: $w \propto \pi$) if $\pi\backslash w \neq \mathbf{0}$, namely, executing w is allowed by the protocol π.

Intuitively, to evaluate $[\pi]\phi$ we need to memorize the current protocol in some way. Here we employ a trick similar to the ones used in the semantics developed in [Gab02, Wan06, BE09]: we define the satisfaction relation w.r.t. a *mode* π (notation: $\vDash_\pi$), which is used to record the current protocol. Given the current protocol π, the allowed runs in a program π' w.r.t π are those $w \in \mathbf{\Sigma}^*$ such that $w \in \mathcal{L}(\pi')$ and $w \propto \pi$. Note that if the current protocol is π, then after executing a run w we have to update π by the remaining protocol $\pi\backslash w$. Now we are ready to give the semantics as follows:

$$
\begin{array}{rcl}
\mathcal{M}, s \vDash \phi & \Leftrightarrow & \mathcal{M}, s \vDash_{\mathbf{\Sigma}^*} \phi \\
\mathcal{M}, s \vDash_\pi p & \Leftrightarrow & p \in V(s) \\
\mathcal{M}, s \vDash_\pi \neg\phi & \Leftrightarrow & \mathcal{M}, s \nvDash_\pi \phi \\
\mathcal{M}, s \vDash_\pi \phi \wedge \psi & \Leftrightarrow & \mathcal{M}, s \vDash_\pi \phi \text{ and } \mathcal{M}, s \vDash_\pi \psi \\
\mathcal{M}, s \vDash_\pi [\pi']\phi & \Leftrightarrow & \forall(w, s') : w \in \mathcal{L}(\pi'), w \propto \pi, \text{ and } s \xrightarrow{w} s' \implies \mathcal{M}, s' \vDash_{\pi\backslash w} \phi \\
\mathcal{M}, s \vDash_\pi [!\pi']\phi & \Leftrightarrow & \mathcal{M}, s \vDash \langle\pi'\rangle\top \implies \mathcal{M}, s \vDash_{\pi'} \phi
\end{array}
$$

where $\mathbf{\Sigma}^*$ stands for $(a_0 + a_1 + \cdots + a_n)^*$ if $\mathbf{\Sigma} = \{a_0, a_1, \ldots, a_n\}$. The first clause says that initially everything is allowed and the last says that the newly announced protocol overrides the current one. $[\pi']\phi$ is true w.r.t the current protocol π iff on each s' that is reachable from s by some run w of π' which is allowed by the current protocol π: ϕ holds w.r.t the remaining protocol $\pi\backslash w$. Note that it is important to remember w which denotes *how you get to* s' as the following example shows:

[3] $\pi\backslash w$ is also a regular language cf. [Con71].

4.2.1. E . Consider the following model $\mathcal{M}$:

It can be verified that:

$$\mathcal{M}, s \vDash [!(a \cdot c + b \cdot d)]\langle a + b\rangle(\neg\langle d\rangle\top \wedge \langle c\rangle\top \wedge [!(c + d)]\langle d\rangle\top)$$

The intuition behind this example is as follows. After announcing the protocol $a \cdot c + b \cdot d$, the program $a + b$ can be executed but actually only a can be executed on the model. Thus after executing $a + b$ only c is possible according to the remaining protocol $(a \cdot c + b \cdot d)\backslash a = c$. However, if we now announce a new protocol $(c + d)$ then d becomes available again. Ω

Recall the PDL semantics in Section 2.3.1. It is not hard to see:

4.2.2. P . *For any test-free* PDL *formula* ϕ *and any pointed Kripke model* $(\mathcal{M}, s)$*:*

$$\mathcal{M}, s \vDash_{\text{PDL}} \phi \iff \mathcal{M}, s \vDash \phi$$

A natural question to ask is whether PDL$^!$ is more expressive than test-free PDL. To answer the question, we now take a closer look at the strings w in the semantics of $[\pi']\phi$. Given π, let $C_{\mathcal{L}(\pi)}$ be the set of all the *pre-sequences* of π: $\{w \mid w \propto \pi\}$. We first show that we can partition $C_{\mathcal{L}(\pi)}$ into finitely many regular expressions.

4.2.3. L . *For any regular expression* π *there is a minimal natural number* k *such that* $C_{\mathcal{L}(\pi)}$ *can be finitely partitioned into* $\pi_0, \ldots, \pi_k$ *and for any* $w, v \in \mathcal{L}(\pi_i) : \pi\backslash w = \pi\backslash v$*.*

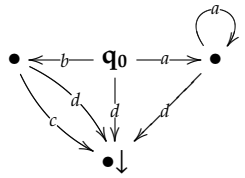
P By Kleene's theorem 2.1.3 we can construct a deterministic finite automaton recognizing the language of π. It is well known that DFA can be minimized, thus we obtain a minimal automaton that recognizes $\mathcal{L}(\pi)$:

$$\mathsf{A}_\pi = (\{q_0, \ldots, q_k\}, \Sigma, q_0, \rightarrowtail, F)$$

where $\{q_0, \ldots, q_k\}$ is a set of states with q_0 being the start state and a subset F being the set of accept states. For each $i \leq k$ such that q_i can reach a state in F: we let π_i be the regular expression corresponding to the automaton $(\{q_0, \ldots, q_k\}, \Sigma, q_0, \rightarrowtail, \{q_i\})$. Since A_π is deterministic, it is not hard to see that these π_i form the partition that we want. ⋈

In the sequel, we call the above unique partition $\pi_0, \ldots, \pi_k$ the *pre-derivatives* of π. For example, the minimal deterministic automaton[4] of $a^* \cdot d + b \cdot (c + d)$ is:

[4]We omit the transitions to the *"trash"* state which can not reach any accept state.



thus the pre-derivatives of $a^* \cdot d + b \cdot (c + d)$ are $\mathbf{1}, a \cdot a^*, b, a^* \cdot d + b \cdot (c + d)$[5].

Now we define the following translation from PDL$^!$ to PDL:

$$\begin{array}{rcl} t(\phi) & = & t_{\Sigma^*}(\phi) \\ t_\pi(p) & = & p \\ t_\pi(\neg\phi) & = & \neg t_\pi(\phi) \\ t_\pi(\phi_1 \wedge \phi_2) & = & t_\pi(\phi_1) \wedge t_\pi(\phi_2) \\ t_\pi([\pi']\phi) & = & \bigwedge_{i=0}^{k}([\theta_i]t_{\pi\backslash\pi_i}(\phi)) \\ t_\pi([!\pi']\phi) & = & \langle\pi'\rangle\top \to t_{\pi'}(\phi) \end{array}$$

where $\pi_0, \ldots, \pi_k$ are the pre-derivatives of π, θ_i is a regular expression corresponding to $\mathcal{L}(\pi') \cap \mathcal{L}(\pi_i)$, and $\pi\backslash\pi_i$ is defined as $\pi\backslash w$ for any $w \in \mathcal{L}(\pi_i)$.

By this translation we can show that PDL and PDL$^!$ are equally expressive.

4.2.4. T . *For any pointed Kripke model* $\mathcal{M}, s$:

$$\mathcal{M}, s \vDash \phi \iff \mathcal{M}, s \vDash_{\mathtt{PDL}} t(\phi).$$

P By induction on ϕ we can show: $\mathcal{M}, s \vDash_\pi \phi \iff \mathcal{M}, s \vDash_{\mathtt{PDL}} t_\pi(\phi)$. The only non-trival case is for $[\pi']\phi$:

$\mathcal{M}, s \vDash_\pi [\pi']\phi$
$\iff \forall(w, s') : w \in \mathcal{L}(\pi), w \propto \pi',$ and $s \xrightarrow{w} s' \implies \mathcal{M}, s' \vDash_{\pi\backslash w} \phi$
$\iff \forall(w, s') :$ if there is a pre-derivative $\pi_i : w \in \mathcal{L}(\pi), w \in \mathcal{L}(\pi_i),$ and $s \xrightarrow{w} s'$ then $\mathcal{M}, s' \vDash_{\pi\backslash w} \phi$
$\iff$ for all pre-derivatives $\pi_i : \forall s' : s \xrightarrow{w} s'$ and $w \in \mathcal{L}(\pi), w \in \mathcal{L}(\pi_i)$ then $\mathcal{M}, s' \vDash_{\pi\backslash w} \phi$
$\iff \mathcal{M}, s \vDash \bigwedge_{i=0}^{k}[\theta_i]t_{\pi\backslash\pi_i}(\phi)$

⋓

Discussion In this section, we take a rather *liberal* view on the "default" protocol, namely we assume that everything is allowed initially, and the announcements may only restrict the possible actions. On the other hand, we can well start with a *conservative* initialization where nothing is allowed unless announced later. It is not hard

[5]Note that $a \cdot a^* \cdot d + b \cdot (c + d) + d = a^* \cdot d + b \cdot (c + d)$.

to see that we can also translate this conservative version of PDL$^!$ to PDL if we let $t(\phi) = t_{\mathbf{1}}(\phi)$ where $\mathbf{1}$ is the constant for the empty sequence i.e., the *skip* protocol. For example, $t_{\mathbf{1}}([a]\bot \wedge [!a]\langle a+b\rangle\top) = [\mathbf{0}]\bot \wedge t_a(\langle a+b\rangle\top) = \langle a\rangle\top$.

Moreover, $[!\pi]$ is rather *radical* in the sense that it changes the protocol completely. We may define a more general operation as follows: Let $\pi(x) \in Reg_{\mathbf{\Sigma}\cup\{x\}}$, namely, $\pi(x)$ is a regular expression with a variable x. Now we define:

$$\boxed{\mathcal{M}, s \vDash_\pi [!\pi'(x)]\phi \quad \Leftrightarrow \quad \mathcal{M}, s \vDash \langle\pi'(\pi)\rangle\top \implies \mathcal{M}, s \vDash_{\pi'(\pi)} \phi}$$

We can then concatenate, add, insert and repeat protocols by announcing $x\cdot\pi'$, $x+\pi'$, $\pi'+x$, and x^* respectively. It is easy to see that the announcement operator $[!\pi]$ introduced previously is a special case of $[!\pi(x)]$. We can still translate the logic with the generalized protocol announcements to PDL with an easy revision of the translation:

$$t_\pi([!\pi'(x)]\phi) \quad = \quad \langle\pi'(\pi)\rangle\top \to t_{\pi'(\pi)}(\phi)$$

4.3 Public Event Logic PDL$^{!?_b}$

In this section, we extend the language of PDL$^!$ with knowledge operator and Boolean tests in programs. We shall see that by announcing a protocol with tests, we can let actions carry propositional information as we motivated in Chapter 1. The language of PDL$^{!?_b}$ is defined as follows:

$$\begin{array}{lll} \phi & ::= & \top \mid p \mid \neg\phi \mid \phi\wedge\phi \mid [\pi]\phi \mid [!\pi]\phi \mid K_i\phi \\ \pi & ::= & ?\phi_b \mid a \mid \pi\cdot\pi \mid \pi+\pi \mid \pi^* \end{array}$$

where $i \in \mathbf{I}$ and ϕ_b are Boolean formulas based on basic propositions in $\mathbf{P}$. Note that we do not include $\mathbf{1}$ and $\mathbf{0}$ as atomic actions since they can be expressed by the Boolean tests $?\top$ and $?\bot$. We call the programs π in PDL$^{!?_b}$ *guarded regular expressions*.

Now we can express the Häagen-Dazs slogan mentioned in Chapter 1 by the protocol: $\pi_{H\text{-}D} = ?p_{love}\cdot a_{buy}$. A suitable semantics should let $[!\pi_{H\text{-}D}][a_{buy}]K_i p_{love}$ be valid. However, without the announcement $!\pi_{H\text{-}D}$, the "secret" love may not be known: $[?p_{love}\cdot a_{buy}]K_i p_{love}$ should not be valid. As we mentioned in the introduction, we assume all the $a \in \mathbf{\Sigma}$ are *public events* which can be observed by all the agents, while the tests, unless announced, are not observable to the agents.

To prepare ourselves for the definition of the semantics, we first interpret regular expressions with Boolean tests as the languages of guarded strings [Koz01]. A *guarded string* over $\mathbf{P}$ and $\mathbf{\Sigma}$ is a sequence $\rho_1 a_1 \rho_2 a_2 \rho \ldots \rho_n a_n \rho_{n+1}$ where $a_i \in \mathbf{\Sigma}$ and $\rho_i \subseteq \mathbf{P}$ represents the valuations of basic propositions in $\mathbf{P}$ ($p \in \rho$ iff p is true according to ρ). For any Boolean formula ψ, let $X_\psi \subseteq 2^{\mathbf{P}}$ be the corresponding set of valuations, represented by subsets of $\mathbf{P}$, that make ψ true. For any $\rho \subseteq \mathbf{P}$, let ϕ_ρ be the formula $\phi_\rho = \bigwedge_{p\in\rho} p \wedge \bigwedge_{p\in(\mathbf{P}-\rho)} \neg p$.

Now we can define the language of guarded strings associated with a guarded regular expression over $\mathbf{\Sigma}$ and $\mathbf{P}$:

$$\begin{array}{l}\mathcal{L}_g(a) = \{\rho a \rho' \mid \rho, \rho' \subseteq \mathbf{P}\}\\ \mathcal{L}_g(?\psi) = \{\rho \mid \rho \in X_\psi\}\\ \mathcal{L}_g(\pi \cdot \pi') = \{w \diamond v \mid w \in \mathcal{L}_g(\pi), v \in \mathcal{L}_g(\pi')\}\\ \mathcal{L}_g(\pi + \pi') = \mathcal{L}_g(\pi) \cup \mathcal{L}_g(\pi')\\ \mathcal{L}_g(\pi^*) = \{\epsilon\} \cup \bigcup_{n>0}(\mathcal{L}_g(\pi^n))\end{array}$$

where $\diamond$ is the *fusion product*: $w \diamond v = w'\rho v'$ when $w = w'\rho$ and $v = \rho v'$; $\pi^n = \underbrace{\pi \cdots \pi}_{n}$.

We write $\pi_1 \equiv_g \pi_2$ if $\mathcal{L}_g(\pi_1) = \mathcal{L}_g(\pi_2)$.

4.3.1. E . We have:

$$?p \cdot ?q \cdot a \equiv_g ?(p \wedge q) \cdot a \equiv_g ?(p \wedge p) \cdot ?q \cdot a$$

$$?(p \wedge q) \cdot a + ?(p \wedge \neg q) \cdot a \equiv_g ?p \cdot a \text{ and } ?p \cdot a \cdot a \not\equiv_g ?p \cdot a$$

Ω

We now define the language of input derivative $\pi\backslash w$ for a guarded string w as:

$$\mathcal{L}_g(\pi\backslash w) = \{v \mid w \diamond v \in \mathcal{L}_g(\pi)\}$$

and we say $w \propto_g \pi$ if $\mathcal{L}_g(\pi\backslash w) \neq \emptyset$. As in the previous section, we let $C^g_\pi = \{w \mid w \propto_g \pi\}$.

Let $\mathcal{L}_!(w)$ be the sequence of public events $a_0 \ldots a_k$ that occurs in w, e.g., $\mathcal{L}_!(?p \cdot a \cdot b) = \mathcal{L}_!(?q \cdot a \cdot ?p \cdot b) = a \cdot b$. Recall that we assume that only the public events can be observed. Thus a guarded string w is indistinguishable from another guarded string v if $\mathcal{L}_!(w) = \mathcal{L}_!(v)$.

According to the standard semantics of PAL, the effect of announcing a formula ϕ is to restrict the model to the ϕ-worlds (see Section 2.3.3). Our public events are like announcements but with preconditions given by the previously announced protocols. However, to model the public events we can also keep the model intact but remember the information induced by the public events. When evaluating epistemic formulas, we let agents only consider possible those worlds which are consistent with the previously recorded information. Since the tests are Boolean, this restriction on accessible worlds works the same as the restriction on models in standard PAL. This motivates us to use $\models^{\phi}_{\pi}$ in the semantics of $\mathtt{PDL}^{!?_b}$ where ϕ is to record the information given by public events according to the protocols. We interpret $\mathtt{PDL}^{!?_b}$ on the S5 models $(S, \mathbf{P}, \mathbf{I}, \sim_i, V)$ as follows:

$$
\begin{array}{rcl}
\mathcal{M}, s \vDash \phi & \Leftrightarrow & \mathcal{M}, s \vDash^{\top}_{\Sigma^*} \phi \\
\mathcal{M}, s \vDash^{\psi}_{\pi} p & \Leftrightarrow & p \in V(s) \\
\mathcal{M}, s \vDash^{\psi}_{\pi} \neg\phi & \Leftrightarrow & \mathcal{M}, s \nvDash^{\psi}_{\pi} \phi \\
\mathcal{M}, s \vDash^{\psi}_{\pi} \phi \wedge \phi' & \Leftrightarrow & \mathcal{M}, s \vDash^{\psi}_{\pi} \phi \text{ and } \mathcal{M}, s \vDash^{\psi}_{\pi} \phi' \\
\mathcal{M}, s \vDash^{\psi}_{\pi} K_i\phi & \Leftrightarrow & \text{for all } v, \text{if } s \sim_i t \text{ and } \mathcal{M}, t \vDash^{\psi}_{\pi} \psi \text{ then } \mathcal{M}, t \vDash^{\psi}_{\pi} \phi \\
\mathcal{M}, s \vDash^{\psi}_{\pi} [\pi']\phi & \Leftrightarrow & \forall w : w \in \mathcal{L}_g(\pi'), w \propto_g \pi, \text{ and } s[\![w]\!]s \implies \mathcal{M}, s \vDash^{\psi \wedge \phi^w_\pi}_{\pi \setminus w} \phi \\
\mathcal{M}, s \vDash^{\psi}_{\pi} [!\pi']\phi & \Leftrightarrow & \mathcal{M}, s \vDash \langle\pi'\rangle\top \implies \mathcal{M}, s \vDash^{\psi}_{\pi'} \phi
\end{array}
$$

where:

$$s[\![w]\!]s \iff w = \rho a_1 \rho a_2 \rho \cdots \rho a_k \rho \text{ and } V(s) = \rho$$

and

$$\phi^w_\pi = \bigvee \{\phi_\rho \mid v = \rho a_1 \rho a_2 \rho \cdots \rho a_k \rho, \mathcal{L}_!(w) = \mathcal{L}_!(v), v \propto_g \pi\}$$

Note that we do not include the transitions labelled by $a \in \Sigma$ in the models since we assume that each public event is executable at each state unless it is not compliant with the current protocol (e.g., you can talk about anything in public unless constrained by some law or conventions). Since the public events are intended to be announcement-like events, we also assume that executing a protocol of such event does not result in changing the real state from one to another. This explains the uniformity of ρ and s in the definition of $[\![w]\!]$. Now we explain the ideas behind ϕ^w_π as follows. First given a $w = \rho a_1 \rho a_2 \rho \cdots \rho a_k \rho$, we collect all the sequences $v = \rho' a_1 \rho' a_2 \rho' \cdots \rho' a_k \rho'$ such that $v \propto_g \pi$. Intuitively ρ' represents the information carried by v according to the protocol π. Since each such v is indistinguishable from w for all the agents, the disjunction ϕ^w_π is then the information which can be derived from the observation of the public events in w according to the protocol π.

Consider the Häagen-Dazs example, let $\mathcal{M}$ be a two-world model representing that a girl i does not know whether a boy loves her or not (she is not sure between a p_{love}-world s and a $\neg p_{love}$ world t). Let $\pi = ?p_{love} \cdot a_{buy}$, and $w_0 = \{p_{love}\}a_{buy}\{p_{love}\}$. It is clear that w_0 is the only guarded string in $\mathcal{L}_g(\pi)$ that have an uniform ρ. Note that $\mathcal{L}_!(w_0) = \mathcal{L}_!(\emptyset a_{buy} \emptyset)$, thus $\phi^{w_0}_{\Sigma^*} = p \vee \neg p$. We now show $\mathcal{M}, s \nvDash [\pi]K_i p_{love}$:

$$
\begin{array}{rl}
 & \mathcal{M}, s \vDash [\pi]K_i p_{love} \\
\iff & \mathcal{M}, s \vDash^{\top}_{\Sigma^*} [\pi]K_i p_{love} \\
\iff & \text{for all } w \in \mathcal{L}_g(\pi), w \propto_g \Sigma^*, \text{ and } s[\![w]\!]s \implies \mathcal{M}, s \vDash^{\phi^w_{\Sigma^*}}_{\Sigma^* \setminus w} K_i p_{love} \\
\iff & s[\![w_0]\!]s \implies \mathcal{M}, s \vDash^{\phi^{w_0}_{\Sigma^*}}_{\Sigma^* \setminus w_0} K_i p_{love} \\
\iff & \mathcal{M}, s \vDash^{\phi^{w_0}_{\Sigma^*}}_{?p \cdot \Sigma^*} K_i p_{love} \\
\iff & \mathcal{M}, s \vDash^{p \vee \neg p}_{?p \cdot \Sigma^*} K_i p_{love}
\end{array}
$$

Since $s \sim_i t$ and $\mathcal{M}, t \vDash p \vee \neg p$ then $\mathcal{M}, s \nvDash [\pi]K_i p_{love}$. On the other hand:

$$
\begin{array}{ll}
 & \mathcal{M}, s \vDash [!\pi][\pi]K_i p_{love} \\
\Longleftrightarrow & \mathcal{M}, s \vDash^{\top}_{\pi^{w_0}_{\pi}} [\pi]K_i p_{love} \\
\Longleftrightarrow & \mathcal{M}, s \vDash^{\phi_\pi}_{\pi \setminus w_0} K_i p_{love} \\
\Longleftrightarrow & \mathcal{M}, s \vDash^{p}_{?\top} K_i p_{love}
\end{array}
$$

Therefore $\mathcal{M}, s \vDash [!\pi][\pi]K_i p_{love}$.

Similarly, for the we-will-see scenario mentioned in the introduction, if $\mathcal{M}$ is a two-world model representing that a Westerner i does know whether p_{no} (state s) or $\neg p_{no}$ (state t) then we can show that:

$$\mathcal{M}, s \vDash [!(?\top \cdot a_{will-see})]([?p_{no} \cdot a_{will-see}]\neg K_i p_{no} \wedge [!(?p_{no} \cdot a_{will-see})][?p_{no} \cdot a_{will-see}]K_i p_{no})$$

where $?\top \cdot a_{will-see}$ is the default protocol a Westerner may have as the standard interpretation for the sentence "we will see" which does not carry any useful information.

In the rest of this section we will show that $\mathtt{PDL}^{!?_b}$ can be translated back to PDL as well. We will follow a similar strategy as in the previous section to finitely partition C^g_π. This time we need to use automata on guarded strings. Given $\mathbf{P}$ let $\mathcal{B}(\mathbf{P})$ be the set $2^{2^{\mathbf{P}}}$. Intuitively, $X \in \mathcal{B}(\mathbf{P})$ represent Boolean formulas over $\mathbf{P}$.

4.3.2. D . **(Automata on guarded strings [Koz01])** A finite *automaton on guarded strings* (or simply *guarded automaton*) over a finite set of actions $\mathbf{\Sigma}$ and a finite set of atomic tests $\mathbf{P}$ is a tuple $\mathsf{A} = (Q, \mathbf{\Sigma}, \mathbf{P}, q_0, \rightarrowtail, F)$ where the transitions are labelled by atomic actions in $\mathbf{\Sigma}$ (action transitions) and sets $X \in \mathcal{B}(\mathbf{P})$ (test transitions). A accepts a finite string w over $\mathbf{\Sigma} \cup \mathcal{B}(\mathbf{P})$ (notation: $w \in \mathcal{L}_{\mathbf{\Sigma} \cup \mathcal{B}(\mathbf{P})}(\mathsf{A})$), if it accepts w as a standard finite automaton over label set $\mathbf{\Sigma} \cup \mathcal{B}(\mathbf{P})$. The acceptance for guarded strings is defined based on the acceptance of normal strings and the following transformation function G which takes a string over $\mathbf{\Sigma} \cup \mathcal{B}(\mathbf{P})$ and outputs a set of guarded strings.

$$
\begin{array}{rcl}
G(a) & = & \{\rho a \rho' \mid \rho, \rho' \subseteq \mathbf{P}\} \\
G(X) & = & \{\rho \mid \rho \in X\} \\
G(ww') & = & \{v\rho v' \mid v\rho \in G(w) \text{ and } \rho v' \in G(w')\}
\end{array}
$$

We say A accepts a finite guarded string $v : \rho_0 a_0 \rho_1 \ldots a_{k-1}\rho_k$ over $\mathbf{\Sigma}$ and $\mathbf{P}$, if $v \in G(w)$ for some string $w \in \mathcal{L}_{\mathbf{\Sigma} \cup \mathcal{B}(\mathbf{P})}(\mathsf{A})$. Let $\mathcal{L}_g(\mathsf{A})$ be the language of guarded strings accepted by A. ℜ

We say a guarded automaton is *deterministic* if the following hold (cf. [Koz01]):

- Each state is either a state that only has outgoing action transitions (*action state*) or a state that only has outgoing test transitions (*test state*).
- The outgoing action transitions are deterministic: for each action state q and each $a \in \mathbf{\Sigma}$, q has one and only one a-successor.
- The outgoing test transitions are deterministic: they are labelled by $\{\{\rho\} \mid \rho \subseteq \mathbf{P}\}$ and for each test state q and each ρ, q has one and only one $\{\rho\}$-successor. Clearly these tests ρ at a test state are logically pairwise exclusive and altogether exhaustive (viewing ρ as the Boolean formula ϕ_ρ).

- The start state is a test state and all accept states are action states.
- Each cycle contains at least one action transition.

The Kleene theorem between guarded automata and guarded regular expressions is proved in [Koz01].

4.3.3. T . *[Koz01, Theorem 3.1, 3.4] For each guarded regular expression π over $\mathbf{P}$ and $\mathbf{\Sigma}$ there is a (deterministic) guarded automaton A over $\mathbf{P}$ and $\mathbf{\Sigma}$ such that $\mathcal{L}_g(\pi) = \mathcal{L}_g(\mathsf{A})$, and vice versa.*

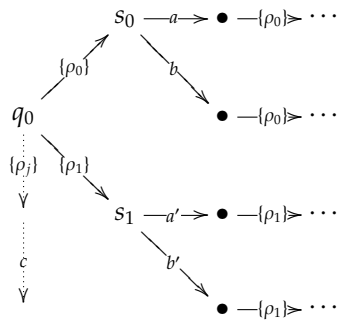
Let L_U be the language of ρ-uniform guarded strings: $\{\rho a_1 \rho a_2 \rho \cdots \rho a_k \rho \mid \rho \subseteq \mathbf{P}, a_i \in \mathbf{\Sigma}\}$. Clearly there is a regular expression for this language: $\sum_{\rho \subseteq \mathbf{P}}((?\phi_\rho \cdot (a_1 + \cdots + a_m))^* \cdot ?\phi_\rho)$ if $\mathbf{\Sigma} = \{a_1, \ldots, a_m\}$. Let $U^g_\pi = C^g_\pi \cap L_U$ be the ρ-uniform part of C^g_π. Following the idea in the previous section, we first need to prove the following lemma:

4.3.4. L . *Given a guarded regular expression π over $\mathbf{\Sigma}$ and $\mathbf{P}$, we can finitely partition U^g_π into $\pi_0, \ldots, \pi_n$ such that for any $i \leq k : w, v \in \mathcal{L}_g(\pi_i) \implies \pi_i \backslash w = \pi_i \backslash v$ and $\phi^w_\pi = \phi^v_\pi$.*

P (Sketch) The strategy for the proof is as follows: we first partition U^g_π into $\pi_0, \ldots, \pi_n$ such that for any $i \leq n$, for any $w, v \in \mathcal{L}_g(\pi_i) : \phi^w_\pi = \phi^v_\pi$, then we further partition each π_i according to the shared derivatives like in Lemma 4.2.3.

From Theorem 4.3.3, we can build deterministic guarded automata A_π and A_U such that $\mathcal{L}_g(\mathsf{A}_\pi) = \mathcal{L}_g(\pi)$ and $\mathcal{L}_g(\mathsf{A}_U) = L_U$. From the definition of deterministic guarded automata, we can assume that in such deterministic automata test states can only have action states as successors, for otherwise the successor test states can be pruned[6]. Now set all the action states in A_π that can reach some accept states as the new accept states, we can obtain a guarded automaton A_C such that $\mathcal{L}_g(\mathsf{A}_C) = C^g_\pi$. Finally we can build an automaton A such that $\mathcal{L}_g(\mathsf{A}) = C^g_\pi \cap L_0$ by the usual automata product of A_C and A_U.

It is not hard to see that if you start with a ρ transition in A then you can never go through a ρ' transition which leads to an accept state such that $\rho \neq \rho'$. Thus the automaton is in the following shape if all the states that are not leading to any accept states are pruned:



[6]Since all the test transitions are labelled by $\{\rho\}$ for some $\rho \subseteq \mathbf{P}$, two consecutive tests are either identical or logically exclusive.

We can then separate ρ_i *"zones"* from each other by taking each s_i as the start state for zone ρ_i. Let B_{s_i} be the standard finite automata over action set $\mathbf{\Sigma}$: $(Q_{act}, \mathbf{\Sigma}, s_i, \rightarrowtail, F)$ where Q_{act} is the set of action states in Q, F is the set of accept states of A that are also in Q_{act}, and $q \overset{a}{\rightarrowtail} q' \iff q \overset{a}{\rightarrowtail}\overset{\rho_i}{\rightarrowtail} q'$ in A. Given $Z \subseteq \{\rho_0, \ldots, \rho_k\}$ (intuitively a Boolean formula), let D_Z be the product automaton $\Pi_{\rho_i \in Z}\mathsf{B}_{s_i} \times \Pi_{\rho_i \notin Z}\overline{\mathsf{B}_{s_i}}$ where $\overline{\mathsf{B}_{s_i}}$ is the complement automaton of B_{s_i}. We can show that D_Z recognizes all the sequences $w \in \mathbf{\Sigma}^*$ such that $\{\rho \mid w = \mathcal{L}_!(v)$ for some $v = \rho a_1 \rho \cdots \rho a_k \rho \in \mathcal{L}_g(\mathsf{A})\} = Z$. Then without much effort, we can turn D_Z into a finite guarded automaton which recognizes guarded strings v in C^g_π, such that:

$$\phi^v_\pi = \bigvee \{\phi_\rho \mid v' = \rho a_1 \rho a_2 \rho \cdots \rho a_k \rho, \mathcal{L}_!(v) = \mathcal{L}_!(v'), v' \propto_g \pi\} = \phi_Z$$

Thus C^g_π can be partitioned into finitely many regular expressions π_i such that for any $w, v \in \mathcal{L}_g(\pi_i) : \phi^w_\pi = \phi^v_\pi$. By the similar techniques as in the previous section, we can further partition each of these regular expressions π_i into finitely many regular expressions $\pi_{i0} \ldots \pi_{im}$ such that for any $w, v \in \pi_{ij}$: $\pi\backslash w = \pi\backslash v$. Thus we can partition C^g_π into $\pi_{00}, \ldots \pi_{km}$ w.r.t $\phi_{00}, \ldots, \phi_{km}$ and $\pi'_{00} \ldots \pi'_{km}$ such that for any $i \leq k, j \leq m$: $w, v \in \mathcal{L}_g(\pi_{ij}) \implies \pi\backslash w = \pi\backslash v = \pi''_{ij}$ and $\phi^w_\pi = \phi^v_\pi = \phi_{ij}$. ⋈

Now we define the following translation from $\mathtt{PDL}^{!?_b}$ to its fragment without $[!\pi]$ which is a $\mathtt{PDL}_{\mathbf{\Sigma}\cup\mathbf{I}}$ language with Boolean tests:

$$\begin{array}{rcl}
t(\phi) & = & t^{\top}_{\mathbf{\Sigma}^*}(\phi) \\
t^\psi_\pi(p) & = & p \\
t^\psi_\pi(\neg\phi) & = & \neg t^\psi_\pi(\phi) \\
t^\psi_\pi(\phi_1 \wedge \phi_2) & = & t^\psi_\pi(\phi_1) \wedge t^\psi_\pi(\phi_2) \\
t^\psi_\pi(K_i\phi) & = & [i](t^\psi_\pi(\psi) \rightarrow t^\psi_\pi(\phi)) \\
t^\psi_\pi([\pi']\phi) & = & \bigwedge^n_{i=0}([\theta_i] t^{\psi\wedge\phi_i}_{\pi\backslash\pi_i}(\phi)) \\
t^\psi_\pi([!\pi']\phi) & = & \langle\pi'\rangle\top \rightarrow t^\psi_{\pi'}(\phi)
\end{array}$$

where $\pi_0, \ldots, \pi_k$ form a partition of U^g_π satisfying the requirements stated by the above lemma. θ_i is a regular expression corresponding to $\mathcal{L}_g(\pi') \cap \mathcal{L}_g(\pi_i)$, and $\pi\backslash\pi_i$ is $\pi\backslash w$ for any $w \in \mathcal{L}_g(\pi_i)$ and ϕ_i is ϕ^w_π for any $w \in \mathcal{L}_g(\pi_i)$.[7]

Note that the translated formulas still have two kinds of modalities: $[\pi]$ and $[i]$. We now argue that we can further eliminate the program modalities. Since we assumed that every event $a \in \mathbf{\Sigma}$ is executable at any state when we disregard the protocol constraint, then we can actually replace each $a \in \mathbf{\Sigma}$ with $?\top$ in the translated formula. Now the program modalities appearing in the translated formula are action-free. Without much effort, we can convert a regular expression of tests into a single test e.g., $(?\phi + ?\psi)^* \cdot ?\chi$ is equivalent to $?(((\phi \vee \psi) \vee \top) \wedge \chi)$. Finally we can eliminate such

[7] Note that if ψ is Boolean then $t^\psi_\pi(\psi) = \psi$, e.g., in $t^\psi_\pi(K_i\phi)$.

test modalities by the validity: $[?\phi]\psi \leftrightarrow (\phi \to \psi)$. Let $t'(\phi)$ be the formula which is obtained from $t(\phi)$ by further eliminating program modalities as described above, then we can translate $\mathtt{PDL}^{!?_b}$ to basic epistemic logic ($\mathtt{EL_I}$) (thus also to $\mathtt{PDL_I}$).

4.3.5. T . *For any pointed S5 Kripke model* $\mathcal{M} = (S, \mathbf{P}, \mathbf{I}, \sim_i, V, s)$:

$$\mathcal{M}, s \vDash_{\mathtt{PDL}^{!?_b}} \phi \iff \mathcal{M}, s \vDash_{\mathtt{EL}} t'(\phi).$$

As an example, consider the formula $\phi = [!((?p \cdot a) + b)][a]K_i p$:

$$\begin{aligned} t(\phi) = \; & t^{\top}_{\Sigma^*}(\phi) \\ = \; & \langle (?p \cdot a) + b\rangle\top \to t^{\top}_{((?p\cdot a)+b)}([a]K_i p) \\ = \; & \langle (?p \cdot a) + b\rangle\top \to [?p \cdot a]t^{p}_{?p}(K_i p) \\ = \; & \langle (?p \cdot a) + b\rangle\top \to [?p \cdot a][i](p \to p) \end{aligned}$$

Therefore by replacing a and b with $?\top$ we have:

$$\langle (?p{\cdot}?\top){+}?\top\rangle\top \to ([?p{\cdot}?\top][i](p \to p))$$

It is easy to see that the above formula is logically equivalent to $\langle ?p\rangle\top \to ([?p][i]\top)$ which is equivalent to $\top$. Indeed, $[!((?p \cdot a) + b][a]K_i p$ is a valid formula.

4.4 Update Logic PDL$^{\boxtimes}$

$\mathtt{PDL}^{!}$ and $\mathtt{PDL}^{!?_b}$ presented in the previous sections are limited in their convenience for modelling epistemic protocols due to the following issues:

- The restriction to Boolean tests excludes the possibility of handling protocols with more complicated pre-conditions, e.g., $?K_i p \cdot a$: *if you know p then do a.*
- The protocols are interpreted as languages of strings, thus we cannot handle branching structures which are useful when considering branching protocols e.g., strategies in games.
- $\mathtt{PDL}^{!?_b}$ does not allow complicated epistemic actions as in DEL [BM04], but only public events.
- The changes of protocols are assumed to be public and agents do not have initial uncertainties about protocols.

As we have seen in the previous sections, operations on finite automata are crucial in proving various results. A natural idea is to use automata directly as modalities in the language. Inspired by [KvB04], in this section we generalize the notion of event models in DEL and introduce a version of PDL with product modalities taking automata as arguments such that the above issues can be handled.

To encode initial uncertainties of protocols we first need to enrich Kripke models with protocol information. Following the notion in process algebra, we use Kripke models with (successful) termination:

4.4.1. D . **(Kripke Model with Termination)** A Kripke model with termination (KMT) is a tuple $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V, F)$ where $(S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V)$ is a standard Kripke model and $F \subseteq S$ is a set of terminating states. We also write $s\downarrow$ for $s \in F$. A pointed KMT is a KMT with a designated state in it. ℳ

Intuitively, the protocol encoded at a state in a KMT can be "read off" by viewing the KMT as an automaton with the designated state as the start state and terminating states as the accept states. A classic Kripke model $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V)$ can be viewed as a KMT with the universal termination: $\mathcal{M}^{\downarrow} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V, S)$. The uncertainties of the initial protocols can be modelled by epistemic relations among those states where different protocols are encoded. Bisimulation on KMTs can be defined in a straightforward way:

4.4.2. D . **(Bisimulation on KMT ($\leftrightarrows$))** A binary relation R between the domain of two KMTs $(S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V, F)$ and $(S', \mathbf{P}, \mathbf{\Sigma}, \rightarrow', V', F')$ is called bisimulation iff $(s, s') \in R$ implies that the following conditions hold:

- $s \in F \iff s' \in F'$;
- $p \in V(s) \iff p \in V(s')$;
- if $s \xrightarrow{a} t$ then there exist t' such that $s' \xrightarrow{a} t'$ and tRt';
- if $s' \xrightarrow{a} t'$ then there exist t such that $s \xrightarrow{a} t$ and tRt'.

ℳ

In this section we build our PDL-style language by using finite automata in two ways: first as program modalities which are alternative representations of modalities with regular expressions with tests as in PDL (cf., e.g., [HKT00]); and second, as update models, the generalized counterpart of the protocol announcements in $\mathtt{PDL}^{!}$ and $\mathtt{PDL}^{!?_b}$. The formulas of our update logic $\mathtt{PDL}^{\boxtimes}$ are built from $\mathbf{P}$ and $\mathbf{\Sigma}$ as follows:

$$\phi \quad ::= \quad \top \mid \downarrow \mid p \mid \neg\phi \mid \phi \wedge \phi \mid [\mathsf{A}]\phi \mid [\boxtimes\mathsf{A}]\phi$$

where $p \in \mathbf{P}$, $\downarrow$ is a constant for successful termination, and each $\mathsf{A} = (Q, \mathbf{\Phi}, \mathbf{\Sigma}, \rightarrowtail, G, q_0)$ is an automaton over actions in $\mathbf{\Sigma}$ and tests in a finite set $\mathbf{\Phi}$ of $\mathtt{PDL}^{\boxtimes}$ formulas[8]. Intuitively, $[\mathsf{A}]\phi$ says "after any execution of the program encoded by A, ϕ holds". $[\boxtimes\mathsf{A}]\phi$ expresses "after updating the current protocol with the one encoded by A, ϕ holds". To simplify the notation, we sometimes use $[\pi^{\mathsf{A}}]$ to denote the automaton modality corresponding to the regular expression with test π.

The semantics for the crucial formulas is given as follows[9]:

$$\begin{array}{rcl} \mathcal{M}, s \vDash \downarrow & \Leftrightarrow & s \in F \\ \mathcal{M}, s \vDash [\mathsf{A}]\phi & \Leftrightarrow & \text{for all } s' : s[\![w]\!]s' \text{ and } w \in \mathcal{L}(\mathsf{A}) \implies \mathcal{M}, s' \vDash \phi \\ \mathcal{M}, s \vDash [\boxtimes\mathsf{A}]\phi & \Leftrightarrow & (\mathcal{M}, s) \boxtimes \mathsf{A} \vDash \phi \end{array}$$

[8] The formulas in $\mathbf{\Phi}$ should be constructed at the earlier stages of the mutual induction on $\mathtt{PDL}^{\boxtimes}$ formulas ϕ and automata A.

[9] According to the semantics, $\boxtimes\mathsf{A}$ is an unconditional update, thus $[\boxtimes\mathsf{A}]\phi \leftrightarrow \langle\boxtimes\mathsf{A}\rangle\phi$ is valid.

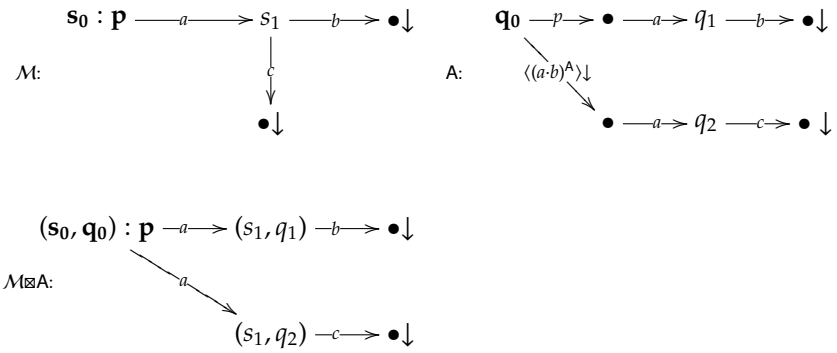
where $s[\![w]\!]s$ is defined as on page 14 for the standard PDL, and the operation $\boxtimes$ is defined as:

4.4.3. D . **(Update Product $\boxtimes$)** Given a KMT $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V, F)$ and a guarded automaton $\mathsf{A} = (Q, \mathbf{\Phi}, \mathbf{\Sigma}, q_0, \rightarrowtail, G)$, the product model is a KMT: $(\mathcal{M} \boxtimes \mathsf{A}) = (S', \mathbf{P}, \mathbf{\Sigma}, \rightarrow', V', F')$ where:

$$\begin{array}{rcl} S' & = & S \times Q \\ \overset{a}{\rightarrow}' & = & \{((s,q),(s',q')) \mid s \xrightarrow{a} s', q \overset{\overline{\phi}}{\rightarrowtail} \overset{a}{\rightarrowtail} q', \text{ and } \mathcal{M}, s \vDash \bigwedge \overline{\phi}\} \\ F' & = & \{(s,q) \mid s \in F, \exists q' \in G : q \overset{\overline{\phi}}{\rightarrowtail} q' \text{ and } \mathcal{M}, s \vDash \bigwedge \overline{\phi}\} \\ V'((s,q)) & = & V(s) \end{array}$$

where $\overline{\phi}$ is a possibly empty sequence of tests in Φ. We let $\bigwedge \overline{\phi}$ be the conjunction of the formulas in $\overline{\phi}$ and let it be $\top$ if $\overline{\phi}$ is empty. For pointed models: $(\mathcal{M}, s_0) \boxtimes \mathsf{A}$ is defined as $\mathcal{M} \boxtimes \mathsf{A}, (s_0, q_0)$. $\mathfrak{m}$

4.4.4. E . We only name a few important states e.g., s_0 in $\mathcal{M}$ where p holds. In the product model $\mathcal{M} \boxtimes \mathsf{A}$ below, we only show the generated submodel w.r.t. (s_0, q_0) :

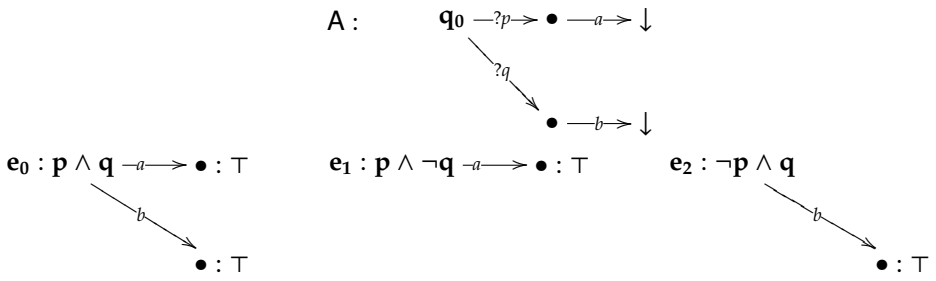


In $\mathcal{M}$, after executing a we can have a choice of b and c (both can lead to the successful termination). A encodes the protocol: *if p then do $a \cdot b$ and if $a \cdot b$ is possible then do $a \cdot c$*. Updating A on $\mathcal{M}$ we obtain $\mathcal{M} \boxtimes \mathsf{A}$ where the choice of b and c after executing a is no longer possible, instead we need to make the choice of $a \cdot b$ and $a \cdot c$ at the beginning. According to the semantics: $\mathcal{M}, s_0 \vDash [a^{\mathsf{A}}](\langle b^{\mathsf{A}}\rangle\downarrow \wedge \langle c^{\mathsf{A}}\rangle\downarrow) \wedge [\boxtimes\mathsf{A}][a^{\mathsf{A}}](\neg(\langle b^{\mathsf{A}}\rangle\downarrow \wedge \langle c^{\mathsf{A}}\rangle\downarrow))$ Ω

As observed in [KvB04], we can view a classic event model of [BMS98] as a automaton where each state with outgoing transitions is guarded by a unique test (the precondition of the state in the event model).

On the other hand, our guarded automata based updates give us more freedom in modelling protocols compared to the event models. Consider the following simple update model denoting the protocol *"if p then you do a and if q then you do b"*. It cannot be mimicked by any single-pointed event model. Instead the update can be simulated by a multi-pointed event model combining three single-pointed event models

with mutually exclusive preconditions at the designated worlds (if we disregard the termination information):



where $(e : \phi)$ denotes that the precondition of e is ϕ.

In the sequel, given an automaton A, we let A_q be the automaton as A but with *start state* q and let A^q be the automaton as A but with q as the only *accept state*.

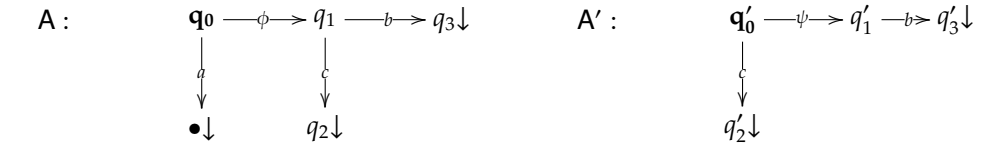
[KvB04] shows that PDL with event model update is equally as expressive as PDL itself by defining a translation pushing the product operators to the inner part of the formulas in order to eliminate them in the end. We will show $\mathtt{PDL}^{\boxtimes}$ can be translated back to PDL as well by following the same idea. In particular, for the formula in the shape of $[\boxtimes\mathsf{A}][\mathsf{B}]\phi$, we need to translate it into some formula in the shape of $[\mathsf{A}'][\boxtimes\mathsf{B}']\psi$. Namely we need to mimic the program B after the update $\boxtimes\mathsf{A}$ by some program before the update. For this purpose, we first define a new product between automata A and A' to handle the interaction between modalities $[\mathsf{A}]$ and $[\boxtimes\mathsf{A}]$.

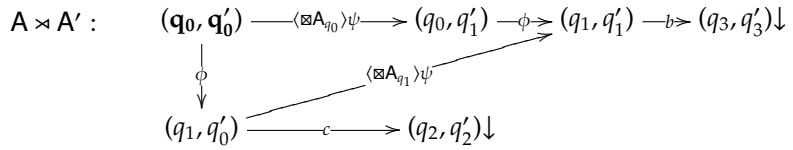
4.4.5. D . **(Sequential Product $\rtimes$)** Given two automata $\mathsf{A} = (Q, \Phi, \mathbf{\Sigma}, \rightarrowtail, q_0, F)$ and $\mathsf{A}' = (Q', \Phi', \mathbf{\Sigma}, \rightarrowtail, q_0', F')$ the sequential product $\mathsf{A} \rtimes \mathsf{A}'$ is again an automaton: $(Q^{\rtimes}, \Phi \cup \Phi'', \mathbf{\Sigma}, \rightarrowtail^{\rtimes}, (q_0, q_0'), F^{\rtimes})$ where:

$$
\begin{array}{rcl}
\Phi'' & = & \{\langle \mathsf{A}_q \rangle \psi \mid \psi \in \Phi', q \in Q\} \\
Q^{\rtimes} & = & Q \times Q' \\
\overset{a}{\rightarrowtail}^{\rtimes} & = & \{((q_1, q_1'), (q_2, q_2')) \mid q_1 \overset{a}{\rightarrowtail} q_2 \text{ and } q_1' \overset{a}{\rightarrowtail} q_2'\} \\
\overset{\phi}{\rightarrowtail}^{\rtimes} & = & \{((q_1, q_1'), (q_2, q_2')) \mid (\phi = \langle \boxtimes \mathsf{A}_{q_1} \rangle \psi, q_1 = q_2 \text{ and } q_1' \overset{\psi}{\rightarrowtail}' q_2') \\
 & & \text{or } (q_1 \overset{\phi}{\rightarrowtail} q_2 \text{ and} q_1' = q_2')\} \\
F^{\rtimes} & = & \{(q, q') \mid q' \in F'\}
\end{array}
$$

$\mathfrak{m}$

Here is an example of a sequential product:





Let PDL$^{\downarrow}$ be the $[\boxtimes \mathsf{A}]$-free fragment of PDL$^{\boxtimes}$. We can then define a translation t from the language of PDL$^{\boxtimes}$ to the language of PDL$^{\downarrow}$ by pushing $\boxtimes \mathsf{A}$ through other modalities (cf. [KvB04]):

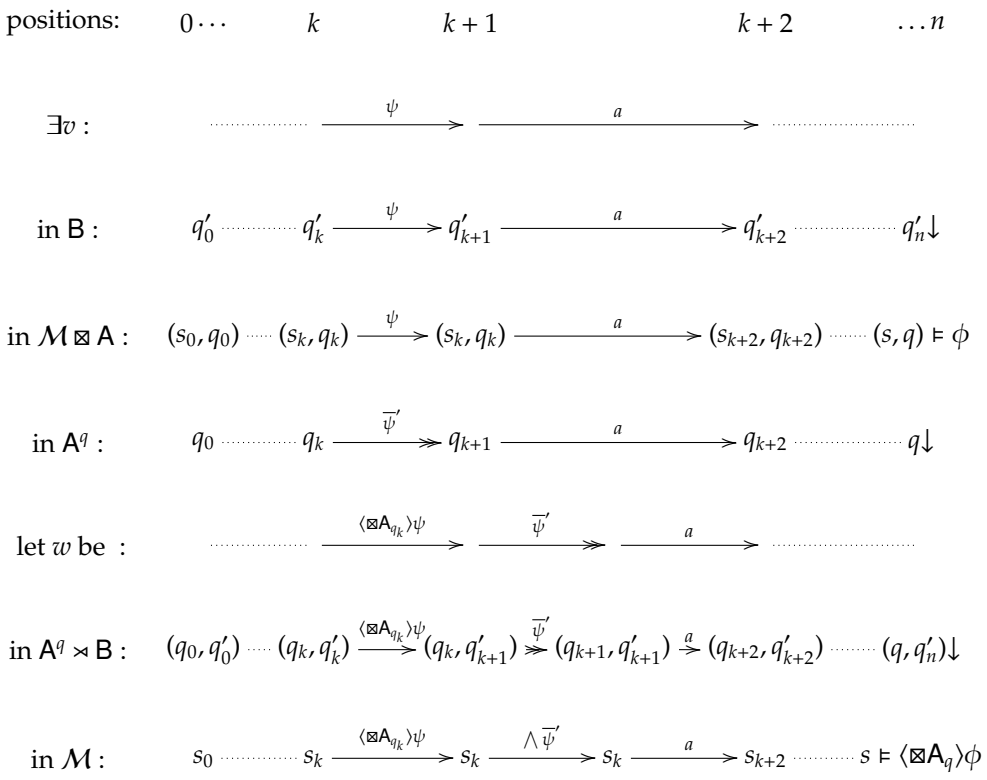
$$\begin{array}{rcl}
t(\top) & = & \top \\
t(\downarrow) & = & \downarrow \\
t(p) & = & p \\
t(\neg\phi) & = & \neg t(\phi) \\
t(\phi_1 \wedge \phi_2) & = & t(\phi_1) \wedge t(\phi_2) \\
t([\mathsf{A}]\phi) & = & [t(\mathsf{A})]t(\phi) \\
t([\boxtimes \mathsf{A}]\top) & = & \top \\
t([\boxtimes \mathsf{A}]\downarrow) & = & \downarrow \wedge t(\chi_{\mathsf{A}}) \\
t([\boxtimes \mathsf{A}]p) & = & p \\
t([\boxtimes \mathsf{A}]\neg\phi) & = & \neg t([\boxtimes \mathsf{A}]\phi) \\
t([\boxtimes \mathsf{A}](\phi_1 \wedge \phi_2)) & = & t([\boxtimes \mathsf{A}]\phi_1) \wedge t([\boxtimes \mathsf{A}]\phi_2) \\
t([\boxtimes \mathsf{A}][\mathsf{B}]\phi) & = & \bigwedge_{q \in Q} t([\mathsf{A}^q \rtimes \mathsf{B}][\boxtimes \mathsf{A}_q]\phi) \\
t([\boxtimes \mathsf{A}][\boxtimes \mathsf{B}]\phi) & = & t([\boxtimes \mathsf{A}]t([\boxtimes \mathsf{B}]\phi))
\end{array}$$

where $t(\mathsf{A})$ is the automaton where each test label ψ in A is replaced by $t(\psi)$, and $\chi_{\mathsf{A}} = \bigvee\{\bigwedge \overline{\phi} \mid \overline{\phi} \in \mathcal{L}(\mathsf{A})\}$. Intuitively, χ_{A} is the "termination test" in A: the disjunction of the combined tests which can lead to an accept state without going through any action transitions. Note that χ_{A} can not be an infinite disjunction essentially since we assume the set of test labels is finite and thus modulo logical equivalence there are only finitely many $\bigwedge \overline{\phi}$. χ_{A} can be computed as follows: we revise A by only keeping the accept states that are reachable from the start state in A via test transitions only; then we can turn this new finite automaton into a regular expression of tests; finally we turn this regular expression into a formula as we mentioned in the end of the previous section. By proving the faithfulness of the translation we can show:

4.4.6. T . PDL$^{\boxtimes}$ *over KMT is equally expressive as* PDL$^{\downarrow}$ *over KMT.*

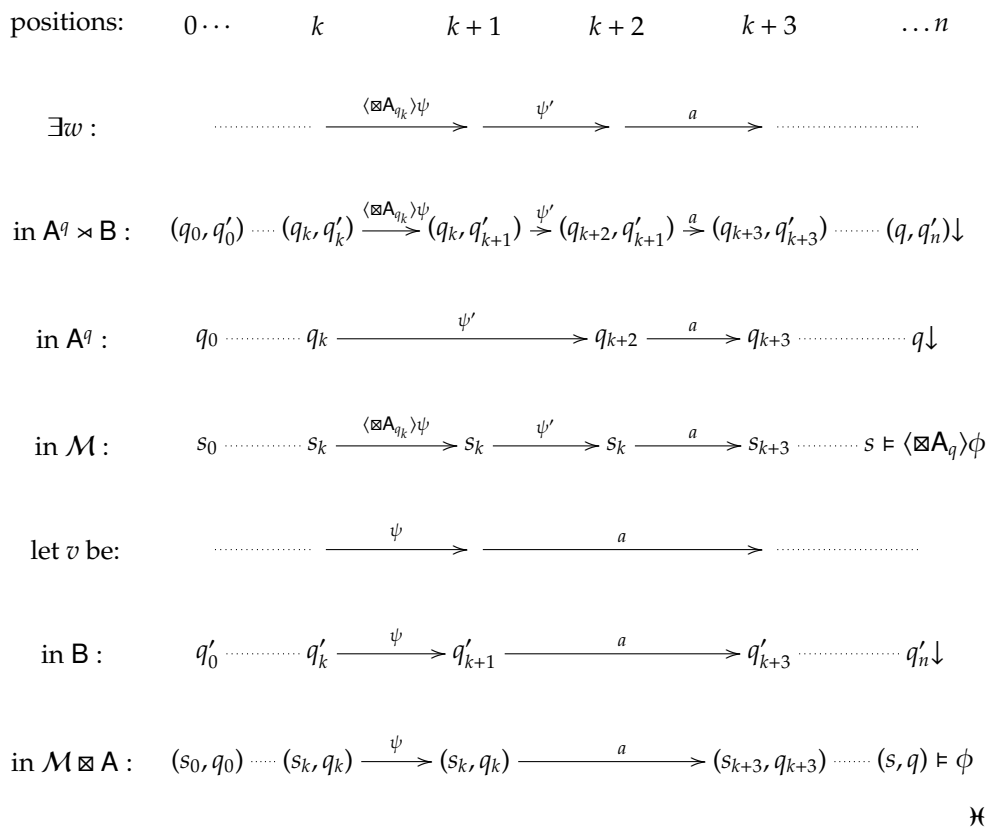
P (sketch) The non-trivial case is to check that $\langle\boxtimes \mathsf{A}\rangle\langle \mathsf{B}\rangle\phi \leftrightarrow \bigvee_{q \in Q}\langle \mathsf{A}^q \rtimes \mathsf{B}\rangle\langle\boxtimes \mathsf{A}_q\rangle\phi$ is valid.

$\Rightarrow$: Suppose $\mathcal{M}, s_0 \vDash \langle \boxtimes \mathsf{A} \rangle \langle \mathsf{B} \rangle \phi$ then $(\mathcal{M}, s_0) \boxtimes \mathsf{A} \vDash \langle \mathsf{B} \rangle \phi$. Thus there is a path $v \in \mathcal{L}(\mathsf{B})$ such that $(s_0, q_0)[\![v]\!](s, q)$ for some (s, q) in $(\mathcal{M}, s_0) \boxtimes \mathsf{A}$ and $\mathcal{M} \boxtimes \mathsf{A}, (s, q) \vDash \phi$ (equivalently $\mathcal{M}, s \vDash \langle \boxtimes \mathsf{A}_q \rangle \phi$). In order to show $\mathcal{M}, s_0 \vDash \langle \mathsf{A}^q \rtimes \mathsf{B} \rangle \langle \boxtimes \mathsf{A}_q \rangle \phi$, we need to find a $w \in \mathcal{L}(\mathsf{A}^q \rtimes \mathsf{B})$ such that $s_0[\![w]\!]s$. The spirit of the proof is to match sequences as follows:



where $\overset{\overline{\psi}'}{\twoheadrightarrow}$ represents a sequence of transitions labelled by the sequence $\overline{\psi}'$.

$\Leftarrow$: Suppose there is a q in A such that $\mathcal{M}, s_0 \vDash \langle \mathsf{A}^q \rtimes \mathsf{B} \rangle \langle \boxtimes \mathsf{A}_q \rangle \phi$. Then there is a $w \in \mathcal{L}(\mathsf{A}^q \rtimes \mathsf{B})$ such that there is an s in $\mathcal{M} : s_0[\![w]\!]s$ and $\mathcal{M}, s \vDash \langle \boxtimes \mathsf{A}_q \rangle \phi$ (equivalently $\mathcal{M} \boxtimes \mathsf{A}, (s, q) \vDash \phi$). To prove $\langle \boxtimes \mathsf{A} \rangle \langle \mathsf{B} \rangle \phi$, we only need to find some $v \in \mathcal{L}(\mathsf{B})$ such that $(s_0, q_0)[\![v]\!](s, q)$ in $\mathcal{M} \boxtimes \mathsf{A}$. We demonstrate the idea of the proof as follows:



4.5 Conclusion and Future Work

Protocols are important components of *social software* [Par02] that govern the human behaviour in social interactions. In this chapter we studied the dynamics of protocols. We proposed three PDL-style logics for reasoning about protocol changes: $\mathtt{PDL}^{!}$ handles protocol changes in the context without knowledge; $\mathtt{PDL}^{!?_b}$ extends $\mathtt{PDL}^{!}$ with knowledge operators and Boolean tests so it can deal with the situations where events carry information according the protocols; $\mathtt{PDL}^{\boxtimes}$ extends the DEL framework with more general product update operations taking guarded automata as update models, which allows us to model branching protocols involving complicated tests. We showed that these three logics can be translated to PDL. What we gain is the explicitness of the language and convenience in modelling scenarios with protocol changes as we demonstrated by various examples. For interested readers who want to see more applications of the protocol changing operations, we refer to [WSvE10]

where we integrated the protocol changing operator as in PDL$^!$ in a specific setting of communications over channels. It is shown in [Lut06] that the public announcement logic, though equally expressive as epistemic logic, is exponentially more succinct than the pure epistemic logic in expressing certain properties on unrestricted models[10]. Here we conjecture that similar results apply to our logics as well. However, we leave the succinctness and complexity analysis for future work.

One thing we did not cover in this chapter is the higher-order change of protocols. For example, *I am asking you to ask her to do something* can be viewed as an announcement of a protocol concerning another protocol announcement. In the logics we presented in this chapter we did not consider protocol updates as basic events, thus excluding protocol announcements such as $!(!\pi \cdot \pi')$. The exact semantics for such announcements can be complicated, and is left for future work[11].

Last but not least, we may introduce more operations on automata other than $\boxtimes$, e.g., continuation or replacement similarly to the generalized protocol announcements mentioned in the end of Section 4.2. It is interesting to see whether PDL is still closed under such extra operations and how the new operator can help to define the existing dynamic operators e.g., the various belief revision operators and preference upgrades as in [BS08a, vBL07, Liu08].

[10] not on $S5$ models as desired though

[11] The propositional coding technique which deals with higher-order event models in [Auc09] may be useful.

Part II

Dynamic Epistemic Modelling

Chapter 5

Composing Models

5.1 Introduction

In Part I, we proposed and studied several variants of PDL as logics for reasoning about knowledge, protocol and change. As we have argued, to verify epistemic protocols by model checking, it is important to have the *right* model. However, as argued in [vB09], to build a model is an *art*. For real life applications, the initial models and the action models can be huge (see, e.g., [DW07] for a case study in a protocol verification setting). Thus some (semi-) automatic method is inevitable in dynamic epistemic modelling. In this part, we make some modest contributions to dynamic epistemic modelling.

Our first inspiration is from the ETL approaches where the temporal epistemic models (interpreted systems) are generated in a distributed fashion with each agent acting as a component (cf. Definition 2.3.1). This distributed feature made ETL very handy in modelling various multi-agent scenarios [FHMV95], for example, adding one extra agent is done by adding one more component. On the other hand, DEL models are apparently not inherently distributed at the first glance: the static models and action models contain information about all the agents. For example, in DEMO [vE07], an implementation of DEL model checking, the initial models are generated by first considering an universal ignorance model, where the agents do not know any atomic proposition, and then restricting it by announcements. It is clear that this method cannot generate all the desired initial models nor is it easy to handle extra agents. To make the DEL approach more applicable for real-life applications, it is crucial to build a static model (or an event model) from local components according to each agent's perspective. This clearly gives rise to the need for a formal way for *composing* models.

In fact, a clue is hinted at in DEL itself: the product update is indeed a way of composing models, though between two different types of models (static models vs. event models). A straightforward idea is to extend this product operation to compositions between two static models or two event models. However, there is a difficulty in defining such a composition operation: the models we want to compose

may have different vocabularies. Note that in practice, one agent may be able to observe only part of the atomic facts (propositions) in the whole vocabulary, e.g., in the Muddy Children scenario each child can see whether other children are muddy but can make no observations about herself. Thus it is reasonable to divide the whole vocabulary into parts (sets of observables) and build "partial models" for agents according to their local vocabularies.

In this chapter, we define and demonstrate the use of *merging composition* between static models and between action models with arbitrary vocabularies. For example, we show how a 2^n worlds Muddy Children model can be viewed as a composition of n two-node models, each talking only about the muddiness of a single child. Next, we extend standard event model update to an update operation which works on static models and event models with different vocabularies, by incorporating vocabulary expansion in the update process. We also look at the models generated in the distributed fashion of ETL and claim that our merging composition with arbitrary vocabularies can achieve the same goal in the DEL setting.

Related work Our merging composition of event models may be viewed as a notion of parallel composition of events. The first concurrent operation in the framework of DEL has been introduced in [vDvdHK03c, vDvdHK03b], where the authors follow the treatment of concurrency as in concurrent PDL [Pel87]. The concurrent operator $\cap$ as in [vDvdHK03c] essentially splits the system into copies with each copy executing a concurrent component (see also [vDvdHK07, Chapter 5] for details). In some sense, composing actions in concurrent DEL may be viewed as merging agents who are acting differently, while in this chapter we focus on merging propositional information which is distributed among agents in both static and event models. Compared to the large body of research about parallel compositions in various process algebra frameworks (e.g., [Mil82, BK85, BHR84, GP94]), the distinct feature of our operator is the merging of different vocabularies and preconditions. The restriction to epistemic models ($S5$ models) also gives specific results meaningful in the epistemic setting.

Structure of the chapter In Section 5.2 we introduce the operator of merging composition on static models, under which the Kripke models form a commutative monoid. We then structurally characterize the induced pre-order by this monoid. Based on the merging composition, we study a natural operation which expands a model with a larger vocabulary. Various logical preservation results between the components and the composed model are proved. Section 5.3 addresses the problem of decomposition by looking at a specific class of models which are useful in a multi-agent setting. We demonstrate that we can decompose a model either by *agents* or by *issues*. We introduce the composition of event models and the extended product update in Section 5.4. We show that under certain conditions the action update distributes over merging composition. We point out some future directions in the last section.

5.2 Composing Static Models

5.2.1 Merging Composition

Recall that an $S5$ Kripke model $\mathcal{M}$ is a tuple $(S, \mathbf{P}, \mathbf{I}, \sim, V)$ where $\mathbf{P}$ is the (finite) vocabulary, $\mathbf{I}$ is the (finite) set of agents, $\sim_i$ is an equivalence relation for each $i \in \mathbf{I}$, and the valuation function $V : S \mapsto 2^{\mathbf{P}}$ assigns a set of atomic propositions to each state. Given a Kripke model $\mathcal{M}$, we use $S_{\mathcal{M}}$, $\mathbf{P}_{\mathcal{M}}, \mathbf{I}_{\mathcal{M}}, \sim_{\mathcal{M}}$ and $V_{\mathcal{M}}$ to denote the corresponding elements in the definition of $\mathcal{M}$. In this chapter we consider the compositions of models with different vocabularies but the same set of agents. We define the *unit* model $\mathcal{E}$ as the model $(\{s\}, \emptyset, \mathbf{I}, \sim, V)$ where $V(s) = \emptyset$ and $\sim_i = \{(s, s)\}$ for any i. In a picture:

Now we define the *merging composition* of two $S5$ models with arbitrary vocabularies.

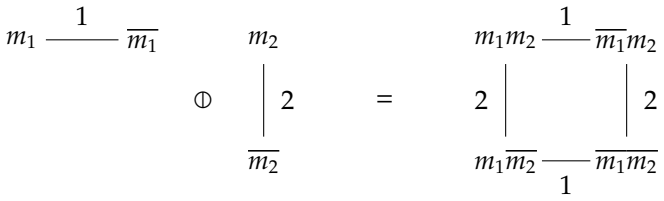
5.2.1. D . **(Merging Composition of Kripke Models)** Given two models with the same set of agents $\mathbf{I} : \mathcal{M} = (S, \mathbf{P}, \mathbf{I}, \sim, V)$ and $\mathcal{N} = (T, \mathbf{P}', \mathbf{I}, \sim', V')$, the *merging composition* $\mathcal{M} \circledcirc \mathcal{N}$ is given by $(S'', \mathbf{P} \cup \mathbf{P}', \mathbf{I}, \sim'', V'')$, where:

- $S'' = \{(s, t) \mid s \in S, t \in T, V(s) \cap \mathbf{P}' = V'(t) \cap \mathbf{P}\}$,
- $(s, s') \sim_i'' (t, t')$ iff $s \sim_i t$ and $s' \sim_i' t'$,
- $V''(s, t) = V(s) \cup V'(t)$.

$\mathfrak{m}$

Intuitively, the accessibility relations in the composed model are defined by "synchronizing" the corresponding relations in the components, in the usual way as in product updates, restricted to the pairs of worlds where the old valuations agree on the common vocabulary $\mathbf{P} \cap \mathbf{P}'$. It is clear that $V(s, s')$ agrees with $V(s)$ on $\mathbf{P}$ and with $V'(s')$ on $\mathbf{P}'$, thus *merging* the two component valuations. We say a state s in $\mathcal{M}$ is *compatible* with a state t in $\mathcal{N}$ if (s, t) is in the composition $\mathcal{M} \circledcirc \mathcal{N}$. It is not hard to verify that any merging composition of $S5$ models is again an $S5$ model.

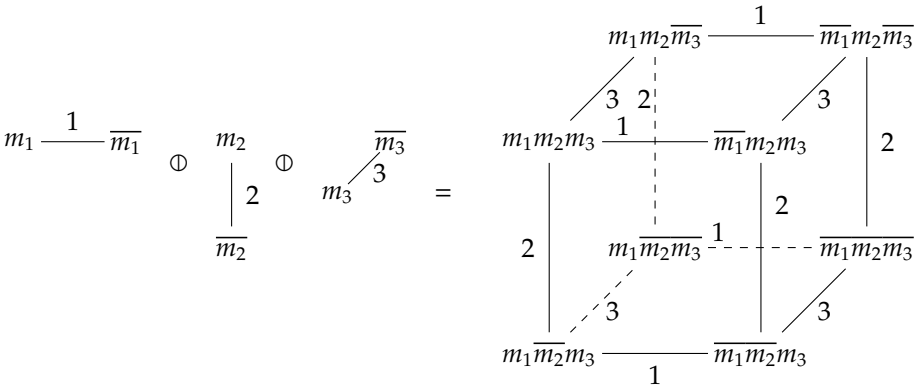
As a first example, here is a "compositional version" of the 2-Muddy Children scenario:



where m_i expresses "child i is muddy", the set of agents is $\{1, 2\}$, the vocabulary of each model $m_i \overset{i}{\longleftrightarrow} \overline{m_i}$ is $\{m_i\}$, and as usual we leave out reflexive arrows which are present for all agents. Intuitively, each 2-world model represents the children's observational power on whether child i is muddy, e.g., $m_i \overset{i}{\longleftrightarrow} \overline{m_i}$ captures the situation that child i does not know whether she herself is muddy while all the others do know whether child i is muddy.

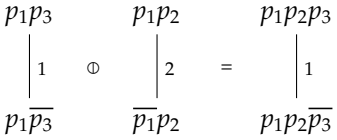
In the same fashion, composing the above models with a third model does give the 3-Muddy Children model[1]:

5.2.2. E . **Composing Muddy Children**

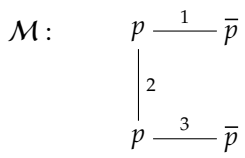


Multidimensional hypercubes with more and more children present can be composed in the same way. **Ω**

Here is an example of composing models with intersected vocabularies:



Note that according to our definition, self-composition $\mathcal{M} \otimes \mathcal{M}$ is not always bisimilar to $\mathcal{M}$. Consider the model $\mathcal{M}$:



[1]Note that the set of agents needs to be extended from $\{1, 2\}$ to $\{1, 2, 3\}$.

Let us call the upper and lower p worlds s and t respectively. It is clear that (s,t) is in the composed model $\mathcal{M} \oplus \mathcal{M}$ and $V_{\mathcal{M}\oplus\mathcal{M}}((s,t)) = \{p\}$. However, according to the definition of relations in the composed model, (s,t) cannot reach a $\neg p$ world by just one step in the composed model. Therefore (s,t) is not bisimilar to any world in $\mathcal{M}$. Nevertheless, Kripke models with different vocabularies do form a commutative monoid:

5.2.3. T . *Kripke models with the same set of agents form a commutative monoid under the $\oplus$ operation, with total bisimilarity (see Definition 2.2.2) as the appropriate equality notion. In particular, we have:*

$$\begin{array}{rcl} \mathcal{E} \oplus \mathcal{M} & \leftrightarrows & \mathcal{M} \\ \mathcal{M} \oplus \mathcal{E} & \leftrightarrows & \mathcal{M} \\ \mathcal{M} \oplus (\mathcal{N} \oplus \mathcal{K}) & \leftrightarrows & (\mathcal{M} \oplus \mathcal{N}) \oplus \mathcal{K} \\ \mathcal{M} \oplus \mathcal{N} & \leftrightarrows & \mathcal{N} \oplus \mathcal{M} \end{array}$$

P Commutativity and axioms about the unit are immediate. We only check associativity here. Let $A(l)^y_x$ be the abbreviation of $V_x(l) \cap \mathbf{P}_y$ for $l \in \{s,t,k\}$ and $x, y \in \{\mathcal{M}, \mathcal{N}, \mathcal{K}\}$, e.g., $A(s)^{\mathcal{N}}_{\mathcal{M}}$ represents $V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{N}}$. Thus the condition

$$\mathsf{PC} := (A(s)^{\mathcal{N}}_{\mathcal{M}} = A(t)^{\mathcal{M}}_{\mathcal{N}} \text{ and } A(s)^{\mathcal{K}}_{\mathcal{M}} = A(k)^{\mathcal{M}}_{\mathcal{K}} \text{ and } A(t)^{\mathcal{K}}_{\mathcal{N}} = A(k)^{\mathcal{N}}_{\mathcal{K}})$$

expresses that s, t, k are pairwise compatible. A moment of reflection should assure that:

$$(s,(t,k)) \in S_{\mathcal{M}\oplus(\mathcal{N}\oplus\mathcal{K})} \iff \mathsf{PC} \iff ((s,t),k)) \in S_{(\mathcal{M}\oplus\mathcal{N})\oplus\mathcal{K}}$$

Then it is straightforward to see that $\mathcal{M} \oplus (\mathcal{N} \oplus \mathcal{K}) \leftrightarrows (\mathcal{M} \oplus \mathcal{N}) \oplus \mathcal{K}$. ⋈

Note that $\leftrightarrows$ is indeed a congruence of this monoid:

5.2.4. P . *If $\mathcal{M}_1 \leftrightarrows \mathcal{M}_2$ and $\mathcal{N}_1 \leftrightarrows \mathcal{N}_2$ then $\mathcal{M}_1 \oplus \mathcal{N}_1 \leftrightarrows \mathcal{M}_2 \oplus \mathcal{N}_2$*

P Let Z_1 Z_2 be the total bisimulations witnessing $\mathcal{M}_1 \leftrightarrows \mathcal{M}_2$ and $\mathcal{N}_1 \leftrightarrows \mathcal{N}_2$ respectively. Then the relation $Z \subseteq S_{\mathcal{M}_1\oplus\mathcal{N}_1} \times S_{\mathcal{M}_2\oplus\mathcal{N}_2}$ defined by:

$$(s_1,t_1)Z(s_2,t_2) \iff s_1 Z_1 s_2 \text{ and } t_1 Z_2 t_2$$

is clearly a total bisimulation between $\mathcal{M}_1 \oplus \mathcal{N}_1$ and $\mathcal{M}_2 \oplus \mathcal{N}_2$. ⋈

The commutative monoid yields the algebraic preordering $\leq$ on the class of Kripke models with different vocabularies:

$$\mathcal{M} \leq \mathcal{N} \text{ iff there is a } \mathcal{K} \text{ with } \mathcal{M} \oplus \mathcal{K} \leftrightarrows \mathcal{N}.$$

We proceed to give a structural characterization of this relation. For this, let a *left-simulation* between two restricted static models $\mathcal{M}$ and $\mathcal{N}$ be a bisimulation with the

invariance condition restricted to proposition letters in the vocabulary of $\mathcal{M}$, and without the Zig condition (see Definition 2.2.2). Formally, given two models $\mathcal{M}$ and $\mathcal{N}$ such that $\mathbf{P}_{\mathcal{M}} \subseteq \mathbf{P}_{\mathcal{N}}$, a left-simulation between $\mathcal{M}$ and $\mathcal{N}$ is a relation $R \subseteq S_{\mathcal{M}} \times S_{\mathcal{N}}$ such that sRt implies that the following hold:

Restricted Invariance $V_{\mathcal{M}}(s) = V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{M}}$;

Zag If for some $i \in \mathbf{I}$ there is a $t' \in S_{\mathcal{N}}$ with $t \xrightarrow{i} t'$ then there is a $s' \in S_{\mathcal{M}}$ with $s \xrightarrow{i} s'$ and $s'Rt'$.

We will use $\mathcal{M}, s \leftleftarrows \mathcal{N}, t$ to indicate that there is a left-simulation that connects s and t, and $\mathcal{M} \leftleftarrows \mathcal{N}$ to indicate that there is a *total* left-simulation between $\mathcal{M}$ and $\mathcal{N}$: there is a left-simulation R that links *every* world in $\mathcal{N}$ to *some* world in $\mathcal{M}$.[2]

Ditmarsch and French [vDF09] prove that for finite static models $\mathcal{M}$ and $\mathcal{N}$:

$$\mathcal{M} \text{ is a simulation of } \mathcal{N} \iff \text{there exists an event model } \mathcal{A} \text{ st. } \mathcal{N} \leftrightarrows \mathcal{M} \otimes \mathcal{A}$$

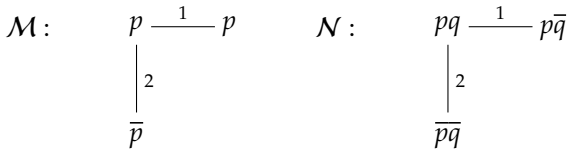
Here we prove a similar result in our setting:

5.2.5. T . *For any models $\mathcal{M}, \mathcal{N}$ with arbitrary vocabularies:*

$$\mathcal{M} \leq \mathcal{N} \implies \mathcal{M} \leftleftarrows \mathcal{N}$$

P $\Rightarrow$: Assume $\mathcal{M} \leq \mathcal{N}$. Then there is a model $\mathcal{K}$ with $\mathcal{M} \oplus \mathcal{K} \leftrightarrows \mathcal{N}$. Let Z be a total bisimulation between $S_{\mathcal{M}\oplus\mathcal{K}}$ and $S_{\mathcal{N}}$. Define R as sRt iff there is some world $x \in S_{\mathcal{K}}$ with $(s, x)Zt$. R is easily seen to be a total left-simulation between $\mathcal{M}$ and $\mathcal{N}$. The restricted invariance property follows from the definition of the valuation on $\mathcal{M} \oplus \mathcal{K}$. The zag property follows from the definition of the accessibility relations on $\mathcal{M} \oplus \mathcal{K}$. Thus, $\mathcal{M} \leftleftarrows \mathcal{N}$. ⋈

Note that the converse does not hold without restrictions on the models. For example, let $\mathcal{M}$ and $\mathcal{N}$ be the following two $S5$ models:



It is clear that $\mathcal{M} \leftleftarrows \mathcal{N}$. Now suppose towards a contradiction that there exists an $\mathcal{M}'$ such that $\mathcal{M} \oplus \mathcal{M}' \leftrightarrows \mathcal{N}$. Since there is a pq world in $\mathcal{N}$, there must be a world t in $\mathcal{M}'$ such that $q \in V_{\mathcal{M}'}(t)$ and t is compatible with any p world in $\mathcal{M}$. Let us denote the upper-right world in $\mathcal{M}$ as s. Then (s, t) must be in the composed model $\mathcal{M} \oplus \mathcal{M}'$ and $V((s, t)) = \{p, q\}$. However, according to the definition of $\oplus$, (s, t) cannot reach a

[2] Note that the *totality* here is different from the totality of bisimulation which requires that any world in any one of the two models is linked to some world in the other model.

$\neg p$ world in one step, thus (s,t) is not bisimilar to any pq world in $\mathcal{N}$. It follows that (s,t) is not bisimilar to any world in $\mathcal{N}$.

Now we look at a subclass of models. A model $\mathcal{M}$ is called *propositionally differentiated* if for any $s,t \in \mathcal{M}$: $V_{\mathcal{M}}(s) \neq V_{\mathcal{M}}(t)$. For example, the models for Muddy Children and Russian Cards Problem as in Chapter 3 are indeed propositionally differentiated. Restricted to this simple but useful class of models we have the exact correspondence of $\leftleftarrows$ and $\leq$:

5.2.6. T . *Let $\mathcal{M}$ be a propositionally differentiated model. Then*

$$\mathcal{M} \leq \mathcal{N} \iff \mathcal{M} \leftleftarrows \mathcal{N}$$

P $\Rightarrow$ follows from Theorem 5.2.5.

For $\Leftarrow$: Assume $\mathcal{M} \leftleftarrows \mathcal{N}$. Let R be a left-simulation between $\mathcal{M}$ and $\mathcal{N}$. Since R is total, for each $t \in S_{\mathcal{N}}$ there is at least one world s in $S_{\mathcal{M}}$ such that sRt. Note that since $\mathcal{M}$ is propositionally differentiated and $\mathbf{P}_{\mathcal{M}} \subseteq \mathbf{P}_{\mathcal{N}}$, for each $t \in S_{\mathcal{N}}$ there is at most one world $s \in S_{\mathcal{M}}$ such that $V_{\mathcal{M}}(s) = V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{M}}$. Therefore for each $t \in \mathcal{N}$ there is one and only one world s such that sRt.

We will show that $\mathcal{M} \oplus \mathcal{N} \leftrightarrows \mathcal{N}$. Let the relation Z between $\mathcal{M} \oplus \mathcal{N}$ and $\mathcal{N}$ be defined as:

$$(s,t)Zt' \text{ iff } t = t'$$

We claim Z is a total bisimulation. Totality is straightforward. Suppose $(s,t)Zt$. We now check the three conditions for bisimulation.

By the construction of $\mathcal{M} \oplus \mathcal{N}$, $V_{\mathcal{M}\oplus\mathcal{N}}((s,t)) = V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) = V_{\mathcal{N}}(t)$. This proves the invariance property.

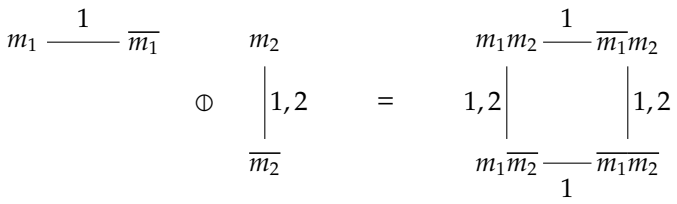
Suppose $(s,t) \sim_i (s',t')$. By the construction of $\mathcal{M} \oplus \mathcal{N}$ this means $s \sim_i s'$ and $t \sim_i t'$. By the definition of Z, $(s',t')Zt'$. This proves the Zig property.

Suppose $t \sim_i t'$. Recall that since $\mathcal{M} \leftleftarrows \mathcal{N}$ and $\mathcal{M}$ is propositionally differentiated, there must be a unique s in $\mathcal{M}$ such that sRt. Then since R is left-simulation, there must be some s' such that $s \sim_i s'$ and $s'Rt'$. Since $s'Rt'$, $V_{\mathcal{M}}(s') = V_{\mathcal{N}}(t') \cap \mathbf{P}_{\mathcal{M}}$ so $(s',t') \in \mathcal{M} \oplus \mathcal{N}$. Since $s \sim_i s'$ and $t \sim_i t'$ then $(s,t) \sim_i (s',t')$. By the definition of Z, $(s',t')Zt'$. This proves the Zag property. ⋈

5.2.2 Expansion

Based on the merging composition operation, we can define the expansions of models with new vocabularies. Let $\mathcal{M}^{\mathbf{I}}_{\mathbf{P}}$ be the universal ignorance model for $\mathbf{P}$, i.e. $\mathcal{M}^{\mathbf{I}}_{\mathbf{P}} = (S, \mathbf{P}, \mathbf{I}, \sim, V)$ with $S = \mathcal{P}(\mathbf{P})$, $\sim_i = S \times S$, $V = \mathrm{id}$. Given $\mathcal{M}$ we define the *expansion* of $\mathcal{M}$ w.r.t. vocabulary $\mathbf{P}'$ as follows: $\mathcal{M} \triangleleft \mathbf{P}' = \mathcal{M} \oplus \mathcal{M}^{\mathbf{I}}_{\mathbf{P}'}$. Note that $\mathbf{P}_{\mathcal{M}\triangleleft\mathbf{P}'} = \mathbf{P}_{\mathcal{M}} \cup \mathbf{P}'$ and the states in the expansion are of the form (s,X) where $s \in S_{\mathcal{M}}$ and $X \subseteq \mathbf{P}$. In the sequel, we will use variables X, Y to denote (possibly empty) subsets of the whole vocabulary in the context of expansions.

Here is an example of expanding with a single new proposition letter m_2. Note: Here and henceforth, worlds are i-linked if there is an i-path in the picture.



Model expansions will be used in Section 5.4 to define the event model update on models with arbitrary vocabularies. We now show that expansions w.r.t. different vocabularies are bisimilar to each other, as long as the expanded vocabulary stays the same:

5.2.7. P . *For any model $\mathcal{M}$, and vocabularies X, Y of proposition letters, if $X \cup \mathbf{P}_\mathcal{M} = Y \cup \mathbf{P}_\mathcal{M}$ then $\mathcal{M} \triangleleft X \leftrightarrows \mathcal{M} \triangleleft Y$.*

P Let relation $Z \subseteq S_{\mathcal{M}\triangleleft X} \times S_{\mathcal{M}\triangleleft Y}$ be given by:

$$(s, X')Z(s', Y') \iff s = s' \text{ and } V_\mathcal{M}(s) \cup X' = V_\mathcal{M}(s') \cup Y'$$

We claim that Z is a total bisimulation. Totality follows from the fact that $X \cup \mathbf{P}_\mathcal{M} = Y \cup \mathbf{P}_\mathcal{M}$. Now we check the three conditions of bisimulation. Suppose $(s, X')Z(s, Y')$ then by definition of Z, $V_\mathcal{M}(s) \cup X' = V_\mathcal{M}(s) \cup Y'$, namely the invariance condition holds. Then based on totality, it is easy to show the Zig and Zag conditions also hold. ⋈

Also the expansion is monotonic in the sense that the expansion with a larger extra vocabulary is restricted bisimilar (see Definition 2.2.2) to the expansion with a smaller extra vocabulary:

5.2.8. P . *For any model $\mathcal{M}$, any vocabularies X, Y such that $Y \subseteq X$, if $X \cap \mathbf{P}_\mathcal{M} = \emptyset$ then $\mathcal{M} \triangleleft X \leftrightarrows_{\mathbf{P}_\mathcal{M} \cup Y} \mathcal{M} \triangleleft Y$.*

P Let relation $Z \subseteq S_{\mathcal{M}\triangleleft X} \times S_{\mathcal{M}\triangleleft Y}$ be given by:

$$(s, X')Z(s', Y') \iff s = s' \text{ and } Y' = X' \cap Y$$

It is not hard to verify that Z is a total bisimulation restricted to the vocabulary $\mathbf{P}_\mathcal{M} \cup Y$. ⋈

If $\mathcal{M}$ is left-similar to $\mathcal{N}$ then the expansion of $\mathcal{M}$ with $\mathbf{P}_\mathcal{N}$ is also left-similar to $\mathcal{N}$.

5.2.9. P . *If $\mathcal{M}, s \leftleftarrows \mathcal{N}, t$ then $\mathcal{M} \triangleleft \mathbf{P}_\mathcal{N}, (s, V_\mathcal{N}(t)) \leftleftarrows \mathcal{N}, t$*

P Let R be a left-simulation which witnesses $\mathcal{M}, s \leftleftarrows \mathcal{N}, t$. Let

$$R' = \{((s, V_{\mathcal{N}}(t)), t) \mid (s, t) \in R\}$$

Note that when $(s, t) \in R$, $(s, V_{\mathcal{N}}(t))$ is indeed in the model $\mathcal{M} \triangleleft \mathbf{P}_{\mathcal{N}}$ due to the restricted invariance condition of R. Thus R' is well-defined. Totality follows from the totality of R. We claim that R' is a left-simulation between $\mathcal{M} \triangleleft \mathbf{P}_{\mathcal{N}}, (s, V_{\mathcal{N}}(t))$ and $\mathcal{N}, t$. The condition of restricted invariance is obvious. For the Zag condition, suppose $t \xrightarrow{i} t' \in \mathcal{N}$ then there is an s' such that $s \xrightarrow{i} s'$ in $\mathcal{M}$ and $s' R t'$. Since $\mathcal{M} \triangleleft \mathbf{P}_{\mathcal{N}} = \mathcal{M} \oplus \mathcal{M}^{\mathbf{I}}_{\mathbf{P}_{\mathcal{N}}}$, we have $(s, V_{\mathcal{N}}(t)) \xrightarrow{i} (s', V_{\mathcal{N}}(t'))$ in $\mathcal{M} \triangleleft \mathbf{P}_{\mathcal{N}}$ and $(s', V_{\mathcal{N}}(t')) R' t'$. ⋈

5.2.3 Preservation

Now let us consider the PDL language over $\mathbf{P}, \mathbf{I}$ (notation: $\mathtt{PDL}_{\mathbf{P},\mathbf{I}}$):

$$\begin{array}{lcl} \phi & ::= & \top \mid p \mid \neg\phi \mid \phi \vee \phi \mid \langle\pi\rangle\phi \\ \pi & ::= & i \mid ?\phi \mid \pi \cdot \pi \mid \pi + \pi \mid \pi^* \end{array}$$

The semantics for $\mathtt{PDL}_{\mathbf{P},\mathbf{I}}$ is defined as usual (see Section 2.3.1). Note that the truth value of a $\mathtt{PDL}_{\mathbf{P},\mathbf{I}}$ formula may not be defined on a model with a different vocabulary other than $\mathbf{P}$. We will study a three valued semantics in chapter 7, while in this chapter, we stick to the 2-valued semantics and make sure the formulas are evaluated on the models where the semantics is defined.

Since PDL is bisimulation invariant, as a straightforward consequence of Proposition 5.2.8, we have:

5.2.10. P . *For any model $\mathcal{M}$, if $X \cap \mathbf{P}_{\mathcal{M}} = \emptyset$ and $Y \subseteq X$ then for any $\phi \in \mathtt{PDL}_{\mathbf{P}\cup Y,\mathbf{I}}$: $\mathcal{M} \triangleleft X, (s, X') \vDash \phi \iff \mathcal{M} \triangleleft Y, (s, X' \cap Y) \vDash \phi$.*

We will use this proposition to prove Theorem 5.4.5 in Section 5.4

The diamond fragment of $\mathtt{PDL}_{\mathbf{P},\mathbf{I}}$ is given by the following ϕ form of formulas:

$$\begin{array}{lcl} \psi & ::= & \top \mid p \mid \neg\psi \mid \psi \vee \psi \\ \pi & ::= & i \mid ?\phi \mid \pi \cdot \pi \mid \pi + \pi \mid \pi^* \\ \phi & ::= & \psi \mid \langle\pi\rangle\phi \mid \phi \vee \phi \mid \phi \wedge \phi. \end{array}$$

We can define the box fragment of $L_{\mathbf{P},\mathbf{I}}$ i.e. the collection of formulas which are logically equivalent to $\neg\phi$ for some ϕ in the diamond fragment.

It is well-known that diamond formulas are preserved under simulation. The following theorem generalizes this to cases where the vocabularies of the two models may be different.

5.2.11. T . *If $\mathcal{M}, s \leq \mathcal{N}, t$ then all formulas ϕ in the diamond fragment of $\mathtt{PDL}_{\mathbf{P}_{\mathcal{M}},\mathbf{I}}$ are preserved from right to left under left simulation: if $\mathcal{N}, t \vDash \phi$ then $\mathcal{M}, s \vDash \phi$. Equivalently, the box fragment of $\mathtt{PDL}_{\mathbf{P}_{\mathcal{M}},\mathbf{I}}$ is preserved from left to right under left simulation.*

P Suppose $\mathcal{M}, s \leq \mathcal{N}, t$. From Theorem 5.2.5, we know that $\mathcal{M}, s \leftleftarrows \mathcal{N}, t$. Let R be a left simulation with sRt. We prove the property by induction on the construction of ϕ. The purely boolean cases are trivial according to the invariance property of R. We only need to show the case of $\langle\pi\rangle\phi$. Suppose $\mathcal{N}, t \vDash \langle\pi\rangle\phi$, then there is a path w starting from t in $\mathcal{M}$ such that $w \in L(\pi)$ which leads to some ϕ world t'. By the Zag property of R, it is not hard to see that there is a matching w path in $\mathcal{M}$ starting from s to some world s' such that $s'Rt'$. By the induction hypothesis, $\mathcal{M}, s' \models \phi$, and therefore $\mathcal{M}, s \vDash \langle\pi\rangle\phi$. ⋈

Based on Proposition 5.2.9 we can relax the restrictions of $\mathbf{P}_{\mathcal{M}}$ on the formulas:

5.2.12. C . *If $\mathcal{M}, w \leq \mathcal{N}, t$ then all formulas ϕ in the diamond fragment of $\mathtt{PDL}_{\mathbf{P}_{\mathcal{N}},\mathbf{I}}$ are preserved from right to left under left simulation: if $\mathcal{N}, t \vDash \phi$ then $\mathcal{M} \triangleleft \mathbf{P}_{\mathcal{N}}, (w, V(t)) \vDash \phi$.*

Theorem 5.2.11 suggests a way of checking the properties (in terms of formulas in the box fragment) of a big model by looking at its components. The following theorem shows that the components can carry more information about the composed models, if we restrict ourselves to certain decomposition of the models. Formally we say model $\mathcal{M}$ is *decomposable* into $\mathcal{M}_0, \ldots, \mathcal{M}_n$ if $\mathcal{M} \leftrightarrows \mathcal{M}_0 \oplus \ldots \oplus \mathcal{M}_n$. A pointed model $(\mathcal{M}, s)$ is decomposable into $(\mathcal{M}_0, s_0), \ldots, (\mathcal{M}_n, s_n)$ if $\mathcal{M}, s \leftrightarrows \mathcal{M}_1 \oplus \ldots \oplus \mathcal{M}_n, (s_0, \ldots, s_n)$.

5.2.13. T (P). *If a pointed model $(\mathcal{M}, s)$ is decomposable into models $(\mathcal{M}_0, s_0), \ldots, (\mathcal{M}_n, s_n)$ with disjoint vocabularies $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$, then for any i: $\mathcal{M}_i, s_i \leftrightarrows_{\mathbf{P}_i} \mathcal{M}, s$. Therefore for any ϕ in $\mathtt{PDL}_{\mathbf{P}_i,\mathbf{I}}$: $\mathcal{M}_i, s_i \vDash \phi \iff \mathcal{M}, s \vDash \phi$.*

P Suppose without loss of generality that $\mathcal{M}, s = \mathcal{M}_0 \oplus \cdots \oplus \mathcal{M}_n, (s_0, \ldots, s_n)$ where $s_i \in \mathcal{M}_i$. Given a tuple $\vec{t} = (t_0, t_1, \ldots, t_n) \in \mathcal{M}$, we let $\vec{t}[i]$ be the ith element in the tuple t. Let Z_i be the relation on $S_{\mathcal{M}} \times S_{\mathcal{M}_i}$ given by $\vec{t} Z_i t$ iff $\vec{t}[i] = t$. We show that Z_i is a $\mathbf{P}_i$-restricted bisimulation. It is clear that sZ_is_i. Assume $\vec{t} Z_i t$ for some $\vec{t} \in \mathcal{M}$ and $t \in \mathcal{M}_i$. Then $V_{\mathcal{M}}(\vec{t}) \cap \mathbf{P}_i = V_{\mathcal{M}_i}(t)$, by the definition of the merging composition. Thus, $\mathbf{P}_i$-restricted invariance holds. Next suppose $\vec{t} \xrightarrow{k} \vec{t'}$. Then by the definition of the accessibility relations on $\mathcal{M}$, $\vec{t}[i] \xrightarrow{k} \vec{t'}[i]$, whence, by definition of Z_i, there is a t'' in the domain of $\mathcal{M}_i$ with $\vec{t'} Z_i t''$. It follows that the Zig condition holds. Finally, assume $t \xrightarrow{k} t'$ in $\mathcal{M}_i$ and $\vec{t} Z_i t$. Now consider the state $\vec{t'}$ given by $\vec{t'}[i] = t'$ and $\vec{t'}[j] = \vec{t}[j]$ for $j \neq i$. Since $\mathbf{P}_i$ is disjoint from any other vocabulary $\mathbf{P}_j$ for $j \neq i$, $\vec{t'}$ must be in $\mathcal{M}$ and $\vec{t'} Z_i t'$. Then by the reflexivity of the $S5$ component models and the fact that $t \xrightarrow{k} t'$, we have $\vec{t} \xrightarrow{k} \vec{t'}$. This proves the Zag condition. ⋈

As an example, consider the case of Muddy Children. From the above theorem, we know that any epistemic statement that talks about the muddiness of a single child in the big model can be checked in a two-world component, e.g., at the component model $m_1 \overset{1}{\longleftrightarrow} \overline{m_1}$, we can verify that agent 2 knows that agent 1 does not know whether she herself is muddy.

5.3 Decomposition

At this stage a natural question to ask is: what kind of model can be decomposed into what kind of form? In this section we look at a particular class of models which is useful in multi-agent systems. In the interpreted systems literature, a basic proposition $p \in \mathbf{P}$ is *i-local* for $i \in \mathbf{I}$ in a model $\mathcal{M}$, if for any s, t in $S_\mathcal{M}$: $s \sim_i t$ implies that $(p \in V_\mathcal{M}(s) \iff p \in V_\mathcal{M}(t))$ (cf., e.g., [EvdMM98]). Intuitively, the i-local propositions are the *atomic observables* of agent i and thus agent i also knows whether they are true. Here we extend this idea by considering not only basic propositions but also their boolean combinations. We say $\mathcal{M}$ is *locally generated* if, for every agent i, there is a non-empty set of boolean formulas Φ_i (the set of local observables) based on $\mathbf{P}_\mathcal{M}$ such that:

$$\text{for all } s, s' \in S_\mathcal{M}, s \sim_i s' \text{ iff for all } \varphi \in \Phi_i,\ \mathcal{M}, s \models \varphi \Leftrightarrow \mathcal{M}, s' \models \varphi$$

Intuitively, a model is locally generated if those local observables determine the epistemic relations in the model. The Muddy Children model is a typical example of a locally generated model (the set of local observables for i is $\{m_j \mid j \neq i, j \in \mathbf{I}\}$). As the following two propositions will show, locally generated models are essentially propositionally differentiated models, which we considered in Theorem 5.2.6.

5.3.1. P . *A locally generated model is bisimilar to a propositionally differentiated model. More precisely, its bisimulation contraction (see Definition 2.2.3) is propositionally differentiated.*

P Given a locally generated model $\mathcal{M}$, suppose Φ_i is the set of local observables for i. Let $Z = \{(s, t) \mid V_\mathcal{M}(s) = V_\mathcal{M}(t)\}$. We show Z is a bisimulation. Assume sZt. The invariance condition is trivial. For Zig, suppose $s \overset{i}{\sim} s'$. Since $\mathcal{M}$ is locally generated, for any $\phi \in \Phi_i : \mathcal{M}, s \vDash \phi \iff \mathcal{M}, s' \vDash \phi$. Since Φ_i contains only boolean formulas and $V_\mathcal{M}(s) = V_\mathcal{M}(t)$, we have for any $\phi \in \Phi_i : \mathcal{M}, t \vDash \phi \iff \mathcal{M}, s' \vDash \phi$. Again due to the definition of the relations in a locally generated model, we have $t \sim_i s'$. Obviously $s'Zs'$, thus it proves the Zig condition. The same argument works for the Zag condition. Therefore it is easy to see that the bisimulation contraction of $\mathcal{M}$ is propositionally differentiated. ⋈

We also have:

5.3.2. P . *Propositionally differentiated models are locally generated.*

P Suppose $\mathcal{M}$ is propositionally differentiated. Let $|S_\mathcal{M}|_{\sim_i}$ be the partitioning of $S_\mathcal{M}$ according to the equivalence relation $\sim_i$. Since $\mathcal{M}$ is propositionally differentiated, we can characterize each world by a conjunction of literals. Then we can characterize each equivalence class in $|S_\mathcal{M}|_{\sim_i}$ by a disjunction of these characterizations. Let Φ_i be the set of these disjunctions. Then clearly $\mathcal{M}$ is locally generated from these Φ_i. ⋈

We can decompose a locally generated model into certain components in an intuitive way.

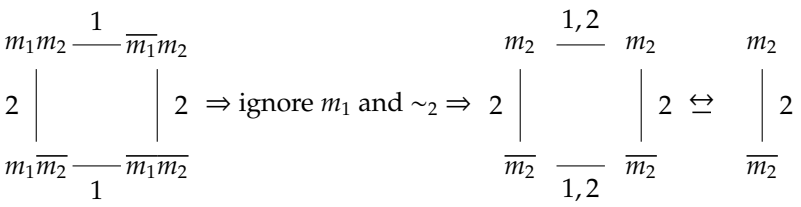
5.3.3. T (D). *Given a set of agent* $\mathbf{I} = \{1, 2, \dots n\}$. *If* $\mathcal{M} = (S, \mathbf{P}, \mathbf{I}, \sim, V)$ *is locally generated w.r.t.* $\Phi_1, \dots, \Phi_n$, *then there are models* $\mathcal{M}_1, \dots, \mathcal{M}_n$ *and* $\mathcal{M}_0$ *such that:*

- $\mathcal{M} \leftrightarrows (\mathcal{M}_0 \oplus \mathcal{M}_1 \oplus \cdots \oplus \mathcal{M}_n)$;
- $|S_{\mathcal{M}_j}| \leq |S|$ *and* $\mathcal{M}_i$ *is bisimulation contracted model;*
- $\mathbf{P}_{\mathcal{M}_j} = \{p \in \mathbf{P}_{\mathcal{M}} \mid p \text{ appears in } \Phi_j\}$ *for* $j > 0$;

P Let $\mathbf{P}_i = \{p \in \mathbf{P}_{\mathcal{M}} \mid p \text{ appears in } \Phi_i\}$ and $\mathbf{P}_0 = \mathbf{P}_{\mathcal{M}}$ then we define $\mathcal{N}_i = (S_{\mathcal{M}}, \mathbf{P}_i, \mathbf{I}, \sim_{\mathcal{N}_i}, V_i)$ where V_i is the restriction of $V_{\mathcal{M}}$ to $\mathbf{P}_i$ and

$$s \overset{j}{\sim}_{\mathcal{N}_i} s' \iff \begin{cases} s \overset{j}{\sim}_{\mathcal{M}} s' & \text{if } j = i \\ \text{always} & \text{if } i \neq j \end{cases}$$

Intuitively, for each $i \in \mathbf{I}$: $\mathcal{N}_i$ is a "local" model for agent i obtained by ignoring the non-local information: atomic propositions not mentioned in the i-observables and epistemic accessibility relations for agents other than i. Note that by ignoring the epistemic relations for j we mean setting $\sim_j$ to be universal. For example:



By our definition, the relations in $\mathcal{N}_0$ are universal. Intuitively, $\mathcal{N}_i$ captures all the possible *states of affairs* in $\mathcal{M}$. Let a relation $Z \subseteq S_{\mathcal{M}} \times S_{\mathcal{N}_0 \oplus \cdots \oplus \mathcal{N}_n}$ be given as follows:

$$sZ(s_0, s_1, \dots, s_n) \iff s = s_0$$

Now let us verify that Z is indeed a total bisimulation. Totality and invariance are trivial by definition of Z.

For Zig: Suppose $s \overset{i}{\sim}_{\mathcal{M}} s'$ and $sZ(s_0, s_1, \dots, s_n)$ then $s = s_0$. Since $(s, s_1, \dots, s_n)$ exists, then $V_{\mathcal{M}}(s) \cap \mathbf{P}_i = V_{\mathcal{N}_i}(s_i)$. Therefore s and s_i satisfy the same set of boolean formulas based on $\mathbf{P}_i$. Since $\mathcal{M}$ is locally generated then we know that s' and s_i agree on the formulas in Φ. Therefore s_i and s' must also agree on the truth values of the formulas in Φ, thus $s_i \overset{i}{\sim}_{\mathcal{M}_i} s'$. Since $\overset{j}{\sim}_{\mathcal{N}_i}$ is universal for $j \neq i$, it is clear that $(s, s_1, \dots, s_n) \overset{i}{\sim} (s', s', \dots, s')$ in $\mathcal{N}_0 \oplus \cdots \oplus \mathcal{N}_n$ and $s'Z(s', s', \dots, s')$.

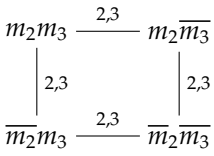
For Zag: Suppose $(s_0, s_1, \dots, s_n) \overset{i}{\sim}_{\mathcal{N}_i} (s'_0, s'_1, \dots, s'_n)$ and $sZ(s_0, s_1, \dots, s_n)$, we then have $s = s_0$ and $s_i \overset{i}{\sim}_{\mathcal{N}_i} s'_i$. By the definition of $\mathcal{N}_i$, we have $s_i \overset{i}{\sim}_{\mathcal{M}} s'_i$. Thus s_i and s'_i agree on formulas in Φ_i. Since s_0 in $\mathcal{M}$ is compatible with s_i in $\mathcal{N}_i$ and s'_i in $\mathcal{N}_i$ is

compatible with s'_0 in $\mathcal{M}$ thus s'_0 also agrees with s_0 on formulas in Φ_i. Therefore $s \overset{i}{\sim}_{\mathcal{M}} s'_0$ and $s'_0 Z(s'_0, s'_1, \ldots, s'_n)$. This proves the Zag condition.

Now we have shown $\mathcal{M} \leftrightarrows \mathcal{N}_0 \oplus \cdots \oplus \mathcal{N}_n$. Let $\mathcal{M}_i$ be the bisimulation contraction of $\mathcal{N}_i$. From Proposition 5.2.4, we know $\mathcal{M} \leftrightarrows (\mathcal{M}_0 \oplus \mathcal{M}_1 \oplus \cdots \oplus \mathcal{M}_n)$. ⋈

In the above proof, $\mathcal{M}_0$ is used to rule out unnecessary worlds created by the merging composition. It is not hard to see that if $\bigcup_{i\in\mathbf{I}} \mathbf{P}_i = \mathbf{P}_{\mathcal{M}}$ and for any $\mathbf{P} \subseteq \mathbf{P}_{\mathcal{M}}$ there is an $s \in S_{\mathcal{M}}$ such that $V_{\mathcal{M}}(s) = \mathbf{P}$, then we can drop the $\mathcal{M}_0$ in the decomposition.

The above theorem gives another way to decompose the Muddy Children models different from the one in Example 5.2.2. Recall that an n-Muddy Children model is locally generated by sets of observables $\Phi_1, \ldots, \Phi_n$ where $\Phi_i = \{m_j \mid j \neq i, j \in \mathbf{I}\}$. For example, if $n = 3$ then the set of observables for agent 1 is $\{m_2, m_3\}$. We can then decompose the 3-Muddy Children model $\mathcal{M}$ by $\mathcal{M}_1, \ldots, \mathcal{M}_3$, where e.g., $\mathcal{M}_1$ is as follows:



Compared to the two-world model decomposition in Example 5.2.2, the above decomposition requires bigger size components (2^{n-1} worlds for the n children case). This is because we decompose the model in an *agent-based* fashion: each component represents one agent's observational power regardless of the others. Thus if the vocabulary of the set of observables Φ_i is big then so is the component model. In the Muddy Children example, if there are more children then the vocabulary of the observables for each child also increases (e.g., new m_j), therefore the component model for this agent also grows bigger. However, in other applications the vocabulary of observables may not increase even when the initial model grows bigger. For example, in the Russian Cards scenario, the agents can only observe their own cards, no matter how many other agents there are (cf. the locally generated initial model of $RCP_{n,n,k}$ in Section 3.4). Therefore, the size of each component can be constant and relatively small.

To decompose a Muddy Children model as in Example 5.2.2, we decompose the model in an *issue-based* fashion (every proposition is an issue), as the following theorem demonstrates:

5.3.4. T (D). *Given a set of agent* $\mathbf{I} = \{1, 2, \ldots n\}$ *and a set of proposition letters* $\mathbf{P} = \{p_1, \ldots, p_k\}$, *if* $\mathcal{M} = (S, \mathbf{P}, \mathbf{I}, \sim, V)$ *is locally generated by* $\Phi_1, \ldots, \Phi_n$ *such that* Φ_i *only contains atomic propositions (i.e.,* $\Phi_i \subseteq \mathbf{P}$*), then there are models* $\mathcal{M}_1, \ldots, \mathcal{M}_k$ *and* $\mathcal{M}_0$ *such that:*

- $\mathcal{M} \leftrightarrows (\mathcal{M}_0 \oplus \mathcal{M}_1 \oplus \cdots \oplus \mathcal{M}_k)$;
- $\mathbf{P}_{\mathcal{M}_j} = \{p_j\}$ *for* $j > 0$ *and* $\mathbf{P}_0 = \mathbf{P}$;

- $|S_{\mathcal{M}_j}| = 2$ *for* $j > 0$

P Let $\mathcal{M}_0$ be the same as in the proof of Theorem 5.3.3. For $j > 0$, let $\mathcal{M}_j = (S_j, \mathbf{P}_j, \mathbf{I}, \sim_{\mathcal{M}_j}, V_j)$ where:

- $S_j = \{p_j, \overline{p_j}\}$;
- $\mathbf{P}_j = \{p_j\}$;
- $V_j(p_j) = \{p_j\}$ and $V_j(\overline{p_j}) = \emptyset$ with the obvious interpretation;
- for $j > 0 : s \overset{i}{\sim}_{\mathcal{M}_j} t \iff s = t$ or $(s \neq t$ and $p_j \notin \Phi_i)$.

Let a relation $Z \subseteq S_{\mathcal{M}} \times S_{\mathcal{M}_0 \oplus \cdots \oplus \mathcal{M}_k}$ be given as follows:

$$sZ(s_0, s_1, \ldots, s_k) \iff s = s_0$$

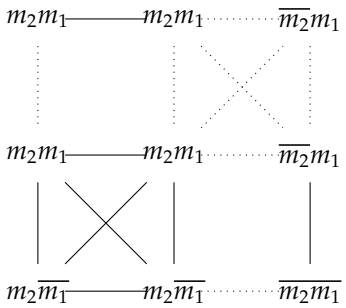
If $sZ(s_0, s_1, \ldots, s_k)$ then for $0 < j \leq k : s_j$ intuitively represents the truth value of p_j in s. Now let us verify that Z is indeed a total bisimulation. Totality and invariance are trivial.

For Zig: Suppose $s \overset{i}{\sim}_{\mathcal{M}} s'$ and $sZ(s_0, s_1, \ldots, s_k)$. Since $(s_0, s_1, \ldots, s_k)$ exists, then $s_0 = s$ and $(s_j = p_j \iff p_j \in V_{\mathcal{M}}(s))$. It is easy to see there is some world $(s', s'_1, \ldots, s'_k)$ in $\mathcal{M}_0 \oplus \cdots \oplus \mathcal{M}_k$ such that $s'Z(s', s'_1, \ldots, s'_k)$. We need to show that for each $j > 0 : s_j \overset{i}{\sim}_{\mathcal{M}_j} s'_j$. If $s_j = s'_j$ then $s_j \overset{i}{\sim}_{\mathcal{M}_j} s'_j$ by definition of $\overset{i}{\sim}_{\mathcal{M}_j}$. Now suppose $s_j \neq s'_j$. Since $\mathcal{M}$ is locally generated by $\Phi_1, \ldots, \Phi_n$, s and s' agree on the truth values of the atomic propositions in Φ_i. Therefore $p_j \notin \Phi_i$, thus $s_j \overset{i}{\sim}_{\mathcal{M}_j} s'_j$.

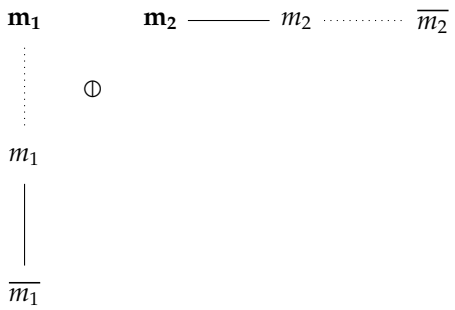
For Zag: Suppose $(s_0, s_1, \ldots, s_k) \overset{i}{\sim}_{\mathcal{M}_i} (s'_0, s'_1, \ldots, s'_k)$ and $sZ(s_0, s_1, \ldots, s_k)$, we have $s_0 = s$ and for any $j \leq k : s_j \overset{i}{\sim}_{\mathcal{M}_j} s'_j$. By the definition of $\mathcal{M}_i$, we have $s_j = s'_j$, or $s_j \neq s'_j$ and $p_j \notin \Phi_i$. Namely, for $p_j \in \Phi_i : s_j = s'_j$. Therefore s and s' agree on the truth values of the propositions in Φ_i. Since $\mathcal{M}$ is locally generated by $\Phi_1, \ldots, \Phi_n$ then $s \overset{i}{\sim}_{\mathcal{M}} s'$.This proves the Zag condition. ⋈

According to the above theorem, a locally generated model by sets of atomic propositions can be decomposed by components based on each atomic proposition. This gives us the desired decomposition of the n-Muddy Children models as in Example 5.2.2.

Theorems 5.3.3 and 5.3.4 show that we can decompose a locally generated model. On the other hand, there are models which are not locally generated but decomposable in a non-trivial way. For example, consider the following model (to ease the presentation, we use solid lines for agent 1 and dotted lines for agent 2):



This model is not bisimilar to any propositionally differentiated model. From Proposition 5.3.1 follows that it is not bisimilar to any locally-generated model. Nevertheless, $\mathcal{M}$ can be decomposed into two models as follows:



If we take the boldface states as the real worlds in these two models respectively, then the two models capture the situations where agent 2 is not sure whether 1 knows m_1 and agent 1 is not sure whether 2 knows m_2. If we interpret m_1 and m_2 as in Muddy Children, then the composed model, when taking the top-left corner state as the real world, captures the situation where the children can see each other's faces but are not sure whether the other has a mirror (actually they do have mirrors). Since the vocabularies of the above two models are disjoint, from Proposition 5.2.13, we know that any true claim about only m_2 or m_1 will be preserved at the components. For example, agent 1 knows agent 2 does not know whether agent 1 knows m_1 can be verified in the left hand component model.

5.4 Composing Updates

Recall Definition 2.3.2 that an (S5) event model $\mathcal{A} = (E, \mathbf{I}, \leftrightarrow, Pre)$ is like a static model, but with valuations replaced by precondition formulas taken from an appropriate language. Let $\mathbf{P}_{\mathcal{A}}$ be the set of proposition letters appearing in the preconditions of $e \in E$ according to Pre.

Note that the standard product update as in Definition 2.3.3 is defined on the pairs of a static model and an event model where $\mathbf{P}_{\mathcal{A}} \subseteq \mathbf{P}_{\mathcal{M}}$. We will now generalize the standard product update to an operation that works on Kripke models and event models with arbitrary vocabularies.

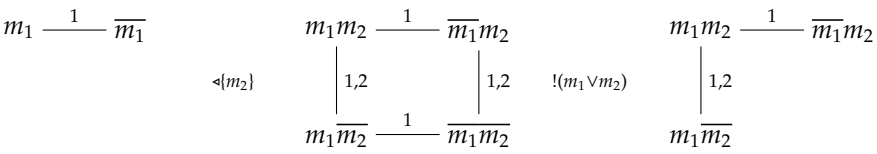
Model expansion is used in the following definition of product update to ensure that no matter what the vocabulary of the static model is, we can always check the preconditions of the events model on the static model. The vocabulary of the resulting updated model is the union of the vocabulary of the static model and the vocabulary of the event model.

5.4.1. D . **(Extended Product Update)** Given a static model $\mathcal{M} = (S, \mathbf{P}, \mathbf{I}, \sim, V)$ and an event model $\mathcal{A} = (E, \mathbf{I}, \leftrightarrow, Pre)$ for the same set of agents $\mathbf{I}$. Let X be the differential vocabulary, i.e., $X = \mathbf{P}_{\mathcal{A}} - \mathbf{P}$. Then the extended product update $\mathcal{M} \circledcirc \mathcal{A}$ is the static model $(S', \mathbf{P} \cup \mathbf{P}_{\mathcal{A}}, \mathbf{I}, \sim', V')$ given by $(\mathcal{M} \triangleleft X) \otimes \mathcal{A}$, where $\otimes$ denotes the usual update product. ℳ

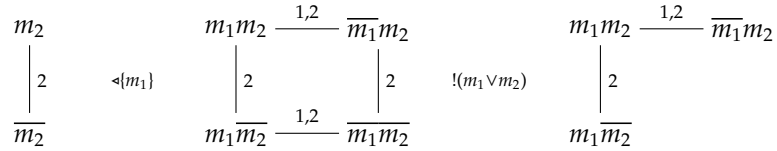
This definition boils down to the following:

1. $S' = \{(s, X', e) \mid s \in S, e \in E, X' \subseteq X, \mathcal{M} \triangleleft X, (s, X') \vDash Pre(e)\}$,
2. $(s, X', e) \sim'_i (t, X'', f)$ iff $s \sim_i t$ and $e \leftrightarrow_i f$,
3. $V'(s, X', e) = V(s) \cup X'$.

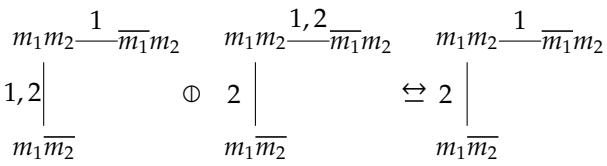
Note that the definition of accessibility relations does not require $X = X'$ since all the different values for the novel atomms are the same for i. From Proposition 5.2.8, we can equivalently (modulo bisimulation) define the update as $(\mathcal{M} \triangleleft \mathbf{P}_{\mathcal{A}}) \otimes \mathcal{A}$. However, for the ease in proofs we will stick to the above definition where $\mathcal{M}$ is expanded with $\mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}}$. Here is an example of an update with a public announcement "at least one of you is muddy" (i.e., an event model with only one world whose precondition is $m_1 \vee m_2$). As usual, we denote this event model as $!(m_1 \vee m_2)$. Note that the update involves model expansion:



Here is an update of the other component of the 2-Muddy Children model:



And here is the outcome of composing the two update results:



This is the same as the result of public announcement of $m_1 \vee m_2$ on the composition of $m_1 \overset{1}{\longleftrightarrow} \overline{m_1}$ and $m_2 \overset{2}{\longleftrightarrow} \overline{m_2}$. In the following, we will show that this outcome is not accidental: updating a composed model yields the same result (modulo bisimulation) as composing the updates of its components, provided the event model has certain property. We call an event model $\mathcal{A}$ *propositionally differentiated* if the preconditions are purely boolean formulas and any two states in $\mathcal{A}$ have mutually exclusive preconditions. For a boolean precondition $Pre(e)$ of e in $\mathcal{A}$, a vocabulary $\mathbf{P}$ such that $\mathbf{P}_{\mathcal{A}} \subseteq \mathbf{P}$, and a set $X \subseteq \mathbf{P}$, we write $X \vDash_{\mathbf{P}} Pre(e)$ if X (viewed as a valuation for $\mathbf{P}$) makes $Pre(e)$ true. It is clear that $X \cap \mathbf{P}_{\mathcal{A}} \vDash_{\mathbf{P}} Pre(e)$ iff $X \vDash_{\mathbf{P}} Pre(e)$. In case $\mathbf{P} = \bigcup_{j\in J} \mathbf{P}_{\mathcal{M}_j}$ we write $X \vDash_{\{\mathcal{M}_j | j\in J\}} Pre(e)$.

5.4.2. T . *If $\mathcal{A}$ is a propositionally differentiated event model then:*

$$(\mathcal{M} \odot \mathcal{N}) \otimes \mathcal{A} \leftrightarrows (\mathcal{M} \otimes \mathcal{A}) \odot (\mathcal{N} \otimes \mathcal{A}).$$

P Let $\mathcal{M}_1 = (\mathcal{M} \odot \mathcal{N}) \otimes \mathcal{A}$ and $\mathcal{M}_2 = (\mathcal{M} \otimes \mathcal{A}) \odot (\mathcal{N} \otimes \mathcal{A})$. Let relation $Z \subseteq S_{\mathcal{M}_1} \times S_{\mathcal{M}_2}$ be given by:

$$((s,t,X),e)Z((s',X_1,e'),(t',X_2,e'')) \text{ iff } s = s', t = t', e = e' = e'' \text{ and } X = X_1 \cap X_2$$

We need to show that Z is a total bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$. The totality of Z is proved in Lemma 5.4.3. Here we focus on the three conditions of bisimulation. Suppose $((s,t,X),e)$ and $((s,X_1,e'),(t,X_2,e''))$ exist in $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively and $((s,t,X),e)Z((s,X_1,e'),(t,X_2,e''))$.

For invariance we need to show

$$V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup (X_1 \cap X_2) = V_{\mathcal{M}}(s) \cup X_1 \cup V_{\mathcal{N}}(t) \cup X_2$$

Since the only difference between the left hand side and right hand side is about $X, X_1, X_2 \subseteq \mathbf{P}_{\mathcal{A}}$, then showing the following suffices:

$$(V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup (X_2 \cap X_2)) \cap \mathbf{P}_{\mathcal{A}} = (V_{\mathcal{M}}(s) \cup X_1 \cup V_{\mathcal{N}}(t) \cup X_2) \cap \mathbf{P}_{\mathcal{A}} \qquad (\star)$$

Since (s,X_1,e) in $\mathcal{M} \otimes \mathcal{A}$ and (t,X_2,e) in $\mathcal{N} \otimes \mathcal{A}$ are compatible, we have:

$$(V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}} = (V_{\mathcal{N}}(t) \cup X_2) \cap \mathbf{P}_{\mathcal{A}} \qquad (\star\star)$$

Now let us massage the left hand side of $(\star)$:

$$
\begin{aligned}
&(V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup (X_1 \cap X_2)) \cap \mathbf{P}_{\mathcal{A}} \\
&= ((V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X_1) \cap (V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X_2)) \cap \mathbf{P}_{\mathcal{A}} \\
&= ((V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X_1) \cap \mathbf{P}_{\mathcal{A}} \cap (V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X_2)) \cap \mathbf{P}_{\mathcal{A}} \\
&= (((V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{A}})) \cap (((V_{\mathcal{N}}(t) \cup X_2) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{A}})) \\
&= (((V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{A}})) \cap ((V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{A}})) \text{ (by } (\star\star)) \\
&= ((V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{N}}(t) \cap V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{A}}) \\
&= ((V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}}) \qquad (\text{ since } V_{\mathcal{N}}(t) \cap V_{\mathcal{M}}(s) \subseteq V_{\mathcal{M}}(s)) \\
&= ((V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}}) \cup ((V_{\mathcal{N}}(t) \cup X_2) \cap \mathbf{P}_{\mathcal{A}}) \qquad (\text{ from } (\star\star)) \\
&= (V_{\mathcal{M}}(s) \cup X_1 \cup V_{\mathcal{N}}(t) \cup X_2) \cap \mathbf{P}_{\mathcal{A}}
\end{aligned}
$$

This proves the invariance requirement.

Now assume $((s,t,X),e)Z((s,X_1,e_1),(t,X_2,e_2))$ and $((s,t,X),e) \overset{i}{\sim} ((s',t',X'),e')$ in $\mathcal{M}_1$, then $e = e_1 = e_2$, $s \overset{i}{\sim} s'$ in $\mathcal{M}$, $t \overset{i}{\sim} t'$ in $\mathcal{N}$ and $e \overset{i}{\sim} e'$ in $\mathcal{A}$. From totality (Lemma 5.4.3), in $\mathcal{M}_2$ there exists $((s',X_1',e'),(t',X_2',e'))$ for some X_1' and X_2' such that

$$((s',t',X'),e')Z((s',X_1',e'),(t',X_2',e')).$$

According to the definition of relations in $\mathcal{M}_2$, it is not hard to see that

$$((s,X_1,e),(t,X_2,e)) \overset{i}{\sim} ((s',X_1',e'),(t',X_2',e'))$$

This proves the Zig requirement.

Suppose $((s,t,X),e)Z((s,X_1,e_1),(t,X_2,e_2))$ and

$$((s,X_1,e),(t,X_2,e)) \overset{i}{\sim} ((s',X_1',e'),(t',X_2',e'))$$

Therefore $s \overset{i}{\sim}_{\mathcal{M}_1} s'$, $t \overset{i}{\sim}_{\mathcal{M}_2} t'$, and $e \overset{i}{\sim}_{\mathcal{A}} e'$. From Lemma 5.4.3, in $\mathcal{M}_1$ there exists (s',X',e') for some X' such that:

$$((s',t',X'),e')Z((s',X_1',e'),(t',X_2',e'))$$

It follows that $((s,t,X),e) \overset{i}{\sim} ((s',t',X'),e')$. This proves the Zag condition.

⋓

5.4.3. L . *The relation Z defined above is total.*

P We need to show for any state u that exists in $\mathcal{M}_1$ there is a state v exists in $\mathcal{M}_2$ such that uZv, and for any v exists in $\mathcal{M}_2$ there is an u in $\mathcal{M}_1$ such that uZv.

Suppose $((s,t,X),e)$ exists in $\mathcal{M}_1$ then the following hold:

Fact 1 $X \subseteq \mathbf{P}_{\mathcal{A}} - (\mathbf{P}_{\mathcal{M}} \cup \mathbf{P}_{\mathcal{N}})$;

Fact 2 $V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{N}} = V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{M}}$;

Fact 3 $V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X \vDash_{\mathcal{M},\mathcal{N},\mathcal{A}} Pre(e)$.

Now we let:

$$X_1 = X \cup ((V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}}) \text{ and } X_2 = X \cup ((V_{\mathcal{M}}(s) - V_{\mathcal{N}}(t)) \cap \mathbf{P}_{\mathcal{A}}).$$

Clearly $X = X_1 \cap X_2$. To show $((s, X_1, e), (t, X_2, e))$ exists in $\mathcal{M}_2$, we need to show:

1. X_1 and X_2 are well-defined: $X_1 \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}}$ and $X_2 \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{N}}$.
2. e can be executed on both (s, X_1) and (t, X_2): $V_{\mathcal{M}}(s) \cup X_1 \vDash_{\mathcal{M},\mathcal{A}} Pre(e)$ and $V_{\mathcal{N}}(t) \cup X_2 \vDash_{\mathcal{N},\mathcal{A}} Pre(e)$.
3. (s, X_1, e) and (t, X_2, e) can be composed: $(V_{\mathcal{M}}(s) \cup X_1) \cap (\mathbf{P}_{\mathcal{N}} \cup \mathbf{P}_{\mathcal{A}}) = (V_{\mathcal{N}}(t) \cup X_2) \cap (\mathbf{P}_{\mathcal{M}} \cup \mathbf{P}_{\mathcal{A}})$.

For (1): Recall that $V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{N}} = V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{M}}$, thus we have $V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{M}} \subseteq V_{\mathcal{M}}(s)$ and $V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{N}} \subseteq V_{\mathcal{N}}(t)$. Therefore, $(V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{M}} = \emptyset$ and $(V_{\mathcal{M}}(s) - V_{\mathcal{N}}(t)) \cap \mathbf{P}_{\mathcal{N}} = \emptyset$. It means $((V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}}) \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}}$ and $((V_{\mathcal{M}}(s) - V_{\mathcal{N}}(t)) \cap \mathbf{P}_{\mathcal{A}}) \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{N}}$. Also note that $X \subseteq \mathbf{P}_{\mathcal{A}} - (\mathbf{P}_{\mathcal{M}} \cup \mathbf{P}_{\mathcal{N}}) \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}}$ and similarly $X \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{N}}$. Therefore by the definitions of X_1 and X_2 we have $X_1 \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}}$ and $X_2 \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{N}}$.

For (2): By the definition of X_1:

$$V_{\mathcal{M}}(s) \cup X_1 = V_{\mathcal{M}}(s) \cup X \cup ((V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}})$$

Then we have:

$$\begin{array}{ll} & (V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}} \qquad (\#) \\ = & (V_{\mathcal{M}}(s) \cup X \cup ((V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}})) \cap \mathbf{P}_{\mathcal{A}} \\ = & ((V_{\mathcal{M}}(s) \cup X) \cap \mathbf{P}_{\mathcal{A}}) \cup ((V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}}) \\ = & (V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X) \cap \mathbf{P}_{\mathcal{A}} \end{array}$$

Since $V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X \vDash_{\mathcal{M},\mathcal{N},\mathcal{A}} Pre(e)$ we have

$$(V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X) \cap \mathbf{P}_{\mathcal{A}} \vDash_{\mathcal{M},\mathcal{N},\mathcal{A}} Pre(e)$$

Therefore from the derivation (#), $(V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}} \vDash_{\mathcal{M},\mathcal{N},\mathcal{A}} Pre(e)$ and then $V_{\mathcal{M}}(s) \cup X_1 \vDash_{\mathcal{M},\mathcal{A}} Pre(e)$. Similarly we can prove $V_{\mathcal{N}}(t) \cup X_2 \vDash_{\mathcal{N},\mathcal{A}} Pre(e)$.

For (3): By the definition of X_1:

$$(V_{\mathcal{M}}(s) \cup X_1) \cap (\mathbf{P}_{\mathcal{N}} \cup \mathbf{P}_{\mathcal{A}}) = ((V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}}) \cup ((V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{N}})$$

From (#), we know that: $(V_{\mathcal{M}}(s) \cup X_1) \cap \mathbf{P}_{\mathcal{A}} = (V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X) \cap \mathbf{P}_{\mathcal{A}}$, thus

$$(V_{\mathcal{M}}(s) \cup X_1) \cap (\mathbf{P}_{\mathcal{N}} \cup \mathbf{P}_{\mathcal{A}}) = ((V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{N}}) \cup (X_1 \cap \mathbf{P}_{\mathcal{N}}) \quad (\dagger)$$

Note that

$$\begin{array}{ll} & X_1 \cap \mathbf{P}_{\mathcal{N}} \\ = & (X \cup ((V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}})) \cap \mathbf{P}_{\mathcal{N}} \\ = & ((V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}}) \cap \mathbf{P}_{\mathcal{N}} \qquad (\text{ since } X \cap \mathbf{P}_{\mathcal{N}} = \emptyset) \\ = & (V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}} \qquad (\text{ since } V_{\mathcal{N}}(t) \subseteq \mathbf{P}_{\mathcal{N}}) \end{array}$$

Therefore go back to (†) we have:

$$\begin{aligned}&(V_{\mathcal{M}}(s) \cup X_1) \cap (\mathbf{P}_{\mathcal{N}} \cup \mathbf{P}_{\mathcal{A}})\\ &= ((V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{N}}) \cup ((V_{\mathcal{N}}(t) - V_{\mathcal{M}}(s)) \cap \mathbf{P}_{\mathcal{A}})\\ &= ((V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{N}}) \qquad (\ddagger)\end{aligned}$$

Similarly we can show

$$(V_{\mathcal{N}}(t) \cup X_2) \cap (\mathbf{P}_{\mathcal{M}} \cup \mathbf{P}_{\mathcal{A}}) = ((V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup X) \cap \mathbf{P}_{\mathcal{A}}) \cup (V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{M}}) \qquad (\S)$$

From the **Fact 1** ($V_{\mathcal{N}}(t) \cap \mathbf{P}_{\mathcal{M}} = V_{\mathcal{M}}(s) \cap \mathbf{P}_{\mathcal{N}}$), (‡) and (§) we have:

$$(V_{\mathcal{M}}(s) \cup X_1) \cap (\mathbf{P}_{\mathcal{N}} \cup \mathbf{P}_{\mathcal{A}}) = (V_{\mathcal{N}}(t) \cup X_2) \cap (\mathbf{P}_{\mathcal{M}} \cup \mathbf{P}_{\mathcal{A}})$$

This proves (3).

Till now we have proved that for any state u that exists in $\mathcal{M}_1$ there is a state v exists in $\mathcal{M}_2$ such that uZv. Now suppose $((s, X_1, e), (t, X_2, e'))$ exists in $\mathcal{M}_2$ we need to show that there is an u in $\mathcal{M}_1$ such that uZv. Since $\mathcal{A}$ is propositionally differentiated, no two actions can be executed under the same valuation over $\mathbf{P}_{\mathcal{A}}$, thus events $e = e'$. We now only need to show that $((s, t, X_1 \cap X_2), e)$ exists in $\mathcal{M}_1$. Formally we need to verify the following claims:

1. s and t are compatible.
2. X is well-defined: $X_1 \cap X_2 \subseteq \mathbf{P}_{\mathcal{A}} - (\mathbf{P}_{\mathcal{M}} \cup \mathbf{P}_{\mathcal{N}})$.
3. e can be executed on $(s, t, X_1 \cap X_2)$: $V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup (X_1 \cap X_2) \vDash_{\mathcal{M},\mathcal{N},\mathcal{A}} Pre(e)$.

From the existence of $((s, X_1, e), (t, X_2, e))$, clearly s and t can be composed. Since $X_1 \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}}$ and $X_2 \subseteq \mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{N}}$, (2) is also straightforward. Now we prove (3). Since $((s, X_1, e)$ and $(t, X_2, e'))$ can be composed

$$V_{\mathcal{M}}(s) \cup X_1 \cup V_{\mathcal{N}}(t) = V_{\mathcal{M}}(s) \cup X_2 \cup V_{\mathcal{N}}(t)$$

Now we have:

$$\begin{aligned}&V_{\mathcal{M}}(s) \cup X_1 \cup V_{\mathcal{N}}(t)\\ &= (V_{\mathcal{M}}(s) \cup X_1 \cup V_{\mathcal{N}}(t)) \cap (V_{\mathcal{M}}(s) \cup X_2 \cup V_{\mathcal{N}}(t)) = (V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup (X_1 \cap X_2))\end{aligned}$$

Since $((s, X_1, e), (t, X_2, e))$ exists, it is not hard to see that $(V(s) \cup V(t) \cup X_1) \cap \mathbf{P}_{\mathcal{A}} \vDash_{\mathcal{M},\mathcal{N},\mathcal{A}} Pre(e)$. Thus$V_{\mathcal{M}}(s) \cup V_{\mathcal{N}}(t) \cup (X_1 \cap X_2) \vDash_{\mathcal{M},\mathcal{N},\mathcal{A}} Pre(e)$. ⋓

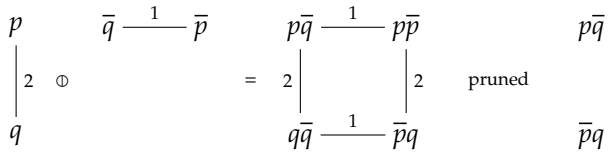
Event models are very similar to static models, and it turns out that composition on event models can be defined in a natural way.

5.4.4. D . **(Merging Composition of Event Models)** The composition $\mathcal{A} \odot \mathcal{B}$ of two event models $\mathcal{A}$ and $\mathcal{B}$ with the same set of agents $\mathbf{I}$ is given by $(E, \mathbf{I}, \leftrightarrow, Pre)$, where:

- $E = \{(e, f) \mid e \in E_{\mathcal{A}}, f \in E_{\mathcal{B}}\}$
- $(e, f) \leftrightarrow_i (e', f')$ iff $e \leftrightarrow_i e'$ in $\mathcal{A}$ and $f \leftrightarrow_i f'$ in $\mathcal{B}$
- $Pre((e, e')) = Pre_{\mathcal{A}}(e) \wedge Pre_{\mathcal{B}}(e')$.

$\mathfrak{m}$

Note that in the composed event model, some e may have an unsatisfiable precondition. We do not delete such non-executable actions in the composed model. The simplest example is composing two announcements $!\phi$ and $!\psi$, which results in an announcement of $\phi \wedge \psi$. The composition operator presented here can be viewed as a kind of parallel compositions of events. Consider the following example (where $\mathbf{I} = \{1, 2\}$ and the propositions are preconditions):



The first model captures the event that agent 1 is being told that either p or q is true, while agent 2 can only see it without hearing the exact message. Similarly, the second model reflects the event that 2 is being told either p or q is false without 1 hearing the message. The composition of the two captures that both events are happening at the same time. As we can see, the effect of updating this composed event is the same as updating an announcement $p \wedge \neg q$ or $\neg p \wedge q$. Intuitively, if agent 1 is told p and he knows that 2 is (truthfully) told either $\neg q$ or $\neg p$ then he actually knows that $\neg p$.

Updating with a composite event model should yield the same outcome as first updating with its components and then composing the results. The following theorem says that it does, notably, without any restriction to certain class of models.

5.4.5. T . $\mathcal{M} \otimes (\mathcal{A} \oplus \mathcal{B}) \leftrightarrows (\mathcal{M} \otimes \mathcal{A}) \oplus (\mathcal{M} \otimes \mathcal{B})$.

P Let $\mathcal{M}_1 = \mathcal{M} \otimes (\mathcal{A} \oplus \mathcal{B})$ and $\mathcal{M}_2 = (\mathcal{M} \otimes \mathcal{A}) \oplus (\mathcal{M} \otimes \mathcal{B})$. Let the relation $Z \subseteq S_{\mathcal{M}_1} \times S_{\mathcal{M}_2}$ be given by:

$(s, X, (e, f))Z((s', X_1, e'), (s'', X_2, f'))$ iff $s = s' = s'', e = e', f = f'$ and $X = X_1 \cup X_2$

We first show Z is total.

$\Rightarrow$: Suppose that $(s, X, (e, f))$ is in $S_{\mathcal{M}_1}$, we need to show that there are X_1 and X_2 with $X = X_1 \cup X_2$ such that $((s, X_1, e), (s, X_2, f))$ exists in $S_{\mathcal{M}_2}$. Notice that:

$(s, X, (e, f)) \in S_{M \otimes (\mathcal{A} \oplus \mathcal{B})}$
$\iff \mathcal{M} \triangleleft ((\mathbf{P}_{\mathcal{A}} \cup \mathbf{P}_{\mathcal{B}}) - \mathbf{P}_{\mathcal{M}}), (s, X) \models Pre_{\mathcal{A}}(e) \wedge Pre_{\mathcal{B}}(f)$
$\iff \mathcal{M} \triangleleft (\mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}}), (s, X \cap (\mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}})) \models Pre_{\mathcal{A}}(e)$
and $\mathcal{M} \triangleleft (\mathbf{P}_{\mathcal{B}} - \mathbf{P}_{\mathcal{M}}), (s, X \cap (\mathbf{P}_{\mathcal{B}} - \mathbf{P}_{\mathcal{M}})) \models Pre_{\mathcal{B}}(f)$ (From Proposition 5.2.10)

Now let $X_1 = X \cap (\mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}})$ and $X_2 = X \cap (\mathbf{P}_{\mathcal{B}} - \mathbf{P}_{\mathcal{M}})$. Since $X \subseteq (\mathbf{P}_{\mathcal{A}} \cup \mathbf{P}_{\mathcal{B}}) - \mathbf{P}_{\mathcal{M}}$, $X_1 \cup X_2 = X \cap ((\mathbf{P}_{\mathcal{A}} - \mathbf{P}_{\mathcal{M}}) \cup (\mathbf{P}_{\mathcal{B}} - \mathbf{P}_{\mathcal{M}})) = X$. From the above derivation, $(s, X_1, e) \in S_{M \otimes \mathcal{A}}$ and $(s, X_2, f) \in S_{M \otimes \mathcal{B}}$ exist and they are compatible. Therefore $((s, X_1, e), (s, X_2, f)) \in S_{\mathcal{M}_2}$.

$\Leftarrow$: Suppose $((s, X_1, e), (s, X_2, f))$ exists in $S_{\mathcal{M}_2}$. Let $X = X_1 \cup X_2$, clearly $X \subseteq (\mathbf{P}_{\mathcal{A}} \cup \mathbf{P}_{\mathcal{B}}) - \mathbf{P}_{\mathcal{M}}$. It is then easy to show that $(s, X, (e, f)) \in S_{\mathcal{M}_1}$.

The invariance is straightforward since $X = X_1 \cup X_2$. Based on totality, Zig and Zag properties of Z are immediate. ⋈

5.5 Discussion and Future Work

In this chapter, we presented a preliminary report on composing static models and event models with different vocabularies. We studied the algebraic properties of the newly introduced composition operator with the presence of the product update. We gave some results on the decomposition of locally generated models[3]. Definitely more questions about decomposability of models should be asked. For example, in [CLDQ09] a symmetry reduction technique is proposed in the setting of multi-agent systems, while it is not very clear whether every symmetric model can be decomposed in a non-trivial way by merging compositions. Note that non-symmetric models can also be decomposed: there are locally-generated models which do not have non-trivial symmetric structures[4]. More ambitious agenda is to logically characterize some decomposable classes of models. To systematically answer such decomposition questions, we may need to find help in both graph theory and modal logic.

Ditmarsch and French [vDF09] studied $\leftleftarrows$ between models with the same vocabulary in the context of product updates. They prove that when restricted to finite models:
$\mathcal{M}, e \leftleftarrows \mathcal{N}, t$ iff there exists an event model $(\mathcal{A}, e)$ such that $\mathcal{N}, t \leftrightarrows \mathcal{M} \otimes \mathcal{A}, (s, e)$. Compared to this result, our Theorem 5.2.6 requires a much stronger condition. The reason is that essentially we only have very weak propositional "preconditions" when composing the model: matching the values of the basic propositions in the common vocabulary. This motivates the future extension with more complicated *matching* conditions when composing the models.

In Theorem 5.3.3 we decompose a locally-generated model w.r.t. the merging composition only. However, we can also decompose the model by composing $\mathcal{M}_1, \ldots, \mathcal{M}_n$ and then, instead of composing $\mathcal{M}_0$, restrict the resulting model by a public announcement of the possible states of affairs. Here a more general question arises naturally: what are the natural basic operations to construct a model besides

[3] We gave a characterization of models that can be decomposed into two-world building blocks in [vEWS10]. However, the criteria are not very intuitive, thus we have chosen to omit it here.

[4] When the sets of observables are not symmetric, the locally-generated model may be asymmetric.

composition? Is there a *normal form* of any model $\mathcal{M}$ by composition, relativization (public announcement) and perhaps general product updates? Again some clues were hinted at in [vDF09]: the author showed that a simulation can be seen as a bisimulation transformation followed by a model restriction. We leave further studies on this topic to future work.

The combination of communicative actions and vocabulary expansion deserves further study. There is an obvious connection to the dynamics of awareness, as studied in [vBVQ09, dJ09], while our expansion operation can be seen as an action for (publicly) extending the awareness set. In this chapter we fix the set of agent in our discussion, while in other applications (e.g., about awareness) expansions with extra agents may be also relevant.

Finally our investigation is encouraging for epistemic model checking with dynamic epistemic logic, for it suggests ways to check relevant epistemic properties on small components of large models. We will pursue this line of research further in Chapter 7 of this thesis.

Chapter 6

Counting Models

6.1 Introduction

In Chapter 3, we argued that a formal definition of a protocol should come with a finite set of formulas which specify the initial setting in which the protocol is to be executed. The verification of the protocol should be performed against the models which satisfy these initial requirements. By making the initial requirements explicit in terms of logical formulas, we may narrow down the gap between the informal scenario and the formal initial model, which is supposed to be a mathematical abstraction of the former. This may help us to gain more grip on the *"model hacking"* which precedes model checking. A natural question to ask at this point is whether the initial assumptions induce a *unique* model? Moreover, if not, *how many* different models are there? The precise meaning of the above questions depends on the logical language we use and the notion of equivalence between models. Since we are interested in modal logics which are bisimulation invariant, we fix bisimulation as the equivalence notion.

First note there are formulas that have unique models modulo bisimulation. We say a formula *characterizes* a pointed model $(\mathcal{M}, s)$ if any model of it is bisimilar to $(\mathcal{M}, s)$. It is shown in [BM96, vB98] that a modal logic equipped with iteration or fixed point operators can characterize arbitrary finite models up to bisimulation[1]. For example, given $\mathbf{P} = \{p\}$ and $\mathbf{I} = \{1, 2\}$, interested readers can verify that the following formula has a unique $S5$ model modulo bisimulation[2]:

$$C_{1,2}((p \to (\widehat{K}_1 p \wedge \widehat{K}_1 \neg p \wedge K_2 p \wedge \widehat{K}_2 p) \wedge (\neg p \to (\widehat{K}_1 p \wedge \widehat{K}_1 \neg p \wedge K_2 \neg p \wedge \widehat{K}_2 \neg p)))$$

Similar results in computer science with respect to branching time temporal logics can be found, e.g., in [BCG87]. In the context of dynamic epistemic modelling,

[1] It is not hard to see that a logic that enjoys the finite model property can only characterize finite models up to bisimulation.

[2] The model should represent a scenario where 1 does not know whether p but 2 knows, and this is common knowledge.

[vDvdHK03a, vD02] demonstrate that there are intuitive epistemic formulas (*descriptions*) that characterize the initial models in the case of the card games. However, in general, a set of formulas translated from an informal description of the scenario may not have a unique model.

In this chapter, we make the initial steps towards an answer for the question: how many non-bisimilar models are there for a given formula? To make our results general enough, we take the *Propositional Modal μ-Calculus* (Mu) as the logical language for specifying the initial requirements.

Since its invention by Kozen [Koz83], Mu has received great interest in computer science due to its neat syntax, strong expressive power and nice model-theoretical properties (see, e.g., [BS06] for an introduction). The language of Mu includes general least (greatest) fixed-point formulas in the shape of $\mu X.\phi$ ($\nu X.\phi$), thus superseding the usual temporal logics in expressive power. For example, a PDL formula $[a^*]p$ can be translated as $\nu X.(p \wedge [a]X)$ in Mu. It is shown in [vBI08] that adding product updates to Mu does not increase its expressive power. Therefore, Mu can actually be considered as a very powerful logic of communication, bearing in mind the potential dynamic epistemic applications. This is the rationale behind the choice of this very powerful logic as the logic in focus in this chapter.

Many theoretical issues of Mu have been studied extensively during the past three decades (cf., e.g., [Wal00, JW95, BS06]). In particular, this chapter is inspired by [Niw91], in which Niwiński tackled the cardinality question of the tree languages recognized by Rabin tree automata. It is also suggested in [Niw91] that the results induce a method to evaluate the number of models of a given formula, due to the fact that formulas of temporal logics can be reduced to Rabin automata. However, in the original paper, the author focused on automata on binary trees and counted models modulo *isomorphism*, which limited the use of the results in the setting of modal logics which are invariant under bisimulation on Kripke models.

In this chapter, we want to pursue this line of research further by counting models modulo bisimulation. Note that we will not work with the Mu formulas directly, but consider the corresponding *alternating tree automata* (*ATA*) ([JW95] shows the equivalence of formulas of Mu and *ATA*). Like [Niw91], but with non-trivial complication of the proof, we show that an ATA recognizable set of image-finite models modulo bisimulation is uncountable iff it is of the cardinality continuum iff it contains a non-regular tree. We also give a normal form of the countable languages recognized by an ATA. These results constitute the first steps towards an algorithm to output the number of models modulo bisimulation.

Related Work In the field of epistemic logic, an important question to ask is: how many different *states* of knowledge of a given fact are there? This question has been addressed by [Aum89, Har96] and [PK92, Par03] in different settings. It is interesting to see that these discussions in the literature can be unified from a perspective of counting models. For example, though presented in the setting of the *information structures* of [Aum76], the set-theoretical counterpart of epistemic logic for economists, [Har96] essentially shows that there are continuum many $S5$ Kripke

models such that any two models are separated by an epistemic formula based on a basic proposition p. If we only consider image-finite models, then bisimulation on $S5$ models coincides with the logical equivalence of epistemic logic, therefore the above result can be equally rephrased as: there are continuum many non-bisimilar $S5$ models of p.

On the other hand, the main result in [PK92, Par03] says that there are countably many different realizable *"levels"* of knowledge, where a level of knowledge realized by a certain pointed $S5$ Kripke model $(\mathcal{M}, s)$ is a set:

$$\{(i_1, i_2, \ldots, i_n) \in \mathbf{I}^* \mid \mathcal{M}, s \vDash K_{i_1} K_{i_2} \ldots K_{i_n} p\}$$

In our setting, this amounts to counting models of p modulo an equivalence notion which is much weaker than bisimulation: it is not hard to see that two models have the same level of knowledge iff they have the same set of labelled paths to $\neg p$ worlds[3]. Therefore, it is expected that there are less levels of knowledge than states of knowledge in the sense of [Har96].

A generalization of the results in [Niw91] is presented in [BKR09], where the focus is on the elimination of the *uncountability quantifiers* in the setting of the monadic second-order logic of order over image-finite trees.

The structure of the chapter In the next section, we will recall some standard definitions and results on alternating tree automata. Section 6.3 presents our main result on the cardinality of ATA recognizable languages. In Section 6.4 we give a normal form of the countable languages recognized by an ATA. In the last section, we discuss some interesting implications and further extensions of our result in an epistemic setting.

6.2 Preliminaries

We first define *trees*. For a non-empty set $\mathbf{\Delta}$, a $\mathbf{\Delta}$ labelled-tree is a tuple:

$$\mathcal{T} = (W, \mathbf{\Delta}, \mathbf{\Sigma}, \rightarrow, L, r)$$

where W is a set of vertices with designated node $r \in W$ as the *root*, $L : W \rightarrow \mathbf{\Delta}$ is a labelling function for the nodes and $\{\rightarrow_a\}_{a \in \mathbf{\Sigma}} \subset W \times W$ is a set of edges such that the root has no incoming edges, and there is a unique directed *finite* path from the root to every node. We also write $W_{\mathcal{T}}$ for the domain of $\mathcal{T}$, similarly for $L_{\mathcal{T}}$, $\rightarrow_{\mathcal{T}}$ and $r_{\mathcal{T}}$. We call two nodes w and w' *siblings* if they are two successors of the same node. By abusing the notation, we sometimes write $v \in \mathcal{T}$ for $v \in W_{\mathcal{T}}$.

Notation The reader should not be confused when seeing w, v as the vertices in a tree while they are reserved to denote sequences in other parts of the thesis. In fact,

[3] The case for common knowledge in [Par03], can be viewed in the same light with little adaptation.

a tree is sometimes represented as a prefix closed subset of $\mathbb{N}^*$ and thus every vertex in a tree is essentially a prefix of a branch (a sequence).

Let $dep(\mathcal{T}, v)$ be the *depth* of v in $\mathcal{T}$, namely the length of the path from $r_{\mathcal{T}}$ to v. We say $v \to^* w$ if $w = v$ or w is reachable from v by following the edges, and $v \to^n w$ if it is reachable in n steps. Intuitively, the nodes that are of the same depth form a *level* of the tree. If $\mathcal{T}$ is a tree then $\mathcal{T}^v$ denotes its *subtree* rooted at v. Let $Sub_{\underline{\leftrightarrow}}(\mathcal{T})$ be the set of subtrees of $\mathcal{T}$ modulo bisimulation: $\{[\mathcal{T}^v]_{/\underline{\leftrightarrow}} | v \in \mathcal{T}\}$. We say $\mathcal{T}$ is *bisimulation-regular* (*B-regular* for short) if $|Sub_{\underline{\leftrightarrow}}(\mathcal{T})|$ is finite. For any u, s in $\mathcal{T}$, let $\mathcal{T}[u \backslash s]$ be the tree constructed from $\mathcal{T}$ by replacing the subtree $\mathcal{T}^u$ with $\mathcal{T}^s$.

It is clear that every tree can be viewed as a pointed Kripke model with labelling function instead of valuations. On the other hand, for an arbitrary Kripke model $\mathcal{M}, s_0$ we can associate its *tree unravelling*, the tree $\mathcal{T}^{\mathcal{M}} = (W, 2^{\mathbf{P}}, \mathbf{\Sigma}, \to, L, r)$ where W is the set of all possible finite paths $s_0 a_0 s_1 a_1 \dots a_n s_n$ starting with s_0 in $\mathcal{M}$, and $w' \xrightarrow{b} w \iff w = w'bs$ in $\mathcal{M}$. It is not hard to see that each Kripke model is bisimilar to its tree unravelling.

Recall that given a Kripke model $\mathcal{M} = (S, \mathbf{\Sigma}, \to, V)$, the bisimulation contraction of $\mathcal{M}$ is the quotient model $\mathcal{M}_{/\equiv_b}$ (see Definition 2.2.3). It follows that if a tree is B-regular, then its bisimulation contraction is a finite Kripke model.

In [JW95], general alternating tree automata (ATA) on Kripke models are defined. For technical convenience, we work with the μ-automata as in [DN05], which can be viewed as an equivalent but intuitive exposition of ATA. These μ-automata run on $2^{\mathbf{P}}$–labelled (infinite) trees, where $\mathbf{P}$ is the set of basic propositions.

6.2.1. D . **(μ-automata [JW95, DN05])** A μ-automaton A on set of basic propositions $\mathbf{P}$ and set of basic actions $\mathbf{\Sigma}$ is a tuple:

$$\mathsf{A} = (Q, B, q_0, \to_{\mathtt{OR}}, \to_{\mathtt{BR}}, L, \Omega)$$

where:

- Q is a non-empty, finite set of OR (choice) states,
- B is a finite set of BR (branching) states such that $B \cap Q = \emptyset$,
- $q_0 \in Q$ is an start state,
- $\to_{\mathtt{OR}} \subseteq Q \times B$ is an unlabelled choice relation from Q to B,
- $\to_{\mathtt{BR}} \subseteq B \times \mathbf{\Sigma} \times Q$ is a labelled branching relation from B to Q,
- $L : B \to 2^{\mathbf{P}}$ is a labelling function mapping each branching state to a set of basic propositions,
- $\Omega : Q \to \mathbb{N}$ is an indexing function.

$\mathfrak{m}$

Let $\mathtt{BR}(b,a) = \{q \mid (b,a,q) \in \rightarrow_{\mathtt{BR}}\}$ and $\mathtt{OR}(q) = \{b \mid (q,b) \in \rightarrow_{\mathtt{OR}}\}$. A q-run $\mathcal{R}$ of $\mathsf{A} = (Q, B, q_0, \rightarrow_{\mathtt{OR}}, \rightarrow_{\mathtt{BR}}, L, \Omega)$ on a $(2^{\mathbf{P}})$-labelled tree $\mathcal{T} = (W_{\mathcal{T}}, 2^{\mathbf{P}}, \Sigma, \rightarrow_{\mathcal{T}}, L_{\mathcal{T}}, r)$ is a $(W_{\mathcal{T}} \times (Q \cup B))-$labelled tree $\mathcal{R} = (W_{\mathcal{R}}, W_{\mathcal{T}} \times (Q \cup B), \Sigma \cup \{\tau\}, \rightarrow_{\mathcal{R}}, L_{\mathcal{R}}, r')$ such that the following conditions are satisfied:

- $L_{\mathcal{R}}(r') = (r, q_0)$.
- (OR) If $L_{\mathcal{R}}(w) = (v,q)$, where $q \in Q$ then w has exactly one $\tau-$child w' such that $L_{\mathcal{R}}(w') = (v,b)$ for some $b \in \mathtt{OR}(q)$.
- (BR) If $L_{\mathcal{R}}(w) = (v,b)$ where $b \in B$ then:
  - $L_{\mathcal{T}}(v) = L(b)$.
  - For every $a-$child v' of v in $\mathcal{T}$, there is an $a-$child w' of w in $\mathcal{R}$ such that $L_{\mathcal{R}}(w') = (v', q')$ for some $q' \in \mathtt{BR}(b,a)$.
  - For every $q' \in \mathtt{BR}(b,a)$ there is an $a-$child w' of w in $\mathcal{R}$ such that $L_{\mathcal{R}}(w') = (v', q')$ for some $a-$child v' of v in $\mathcal{T}$.

For a path P of $\mathcal{T}$, we define $Q^{\mathcal{R}}(P) = \{q \mid L_{\mathcal{R}}(w) = (v,q)$ for some $w \in \mathcal{R}$ and v on $P\}$. Note that $Q^{\mathcal{R}}(v)$ is not always a singleton since one node in $\mathcal{T}$ may correspond to more than one state in the automaton. For an infinite path $P = v_0, v_1, ...$ we define:

$$Inf(\mathcal{R}, P) = \{q \mid q \in Q^{\mathcal{R}}(v_i) \text{ for infinitely many } v_i\}$$

The acceptance of runs is defined by the *parity* condition: A q-run $\mathcal{R}$ of A on $\mathcal{T}$ is *accepting* iff for every infinite path P in $\mathcal{R}$: the greatest value of $\Omega^{\mathcal{R}}(q)$, for $q \in Inf(\mathcal{R}, P)$, is even. We denote such greatest value as $\Omega^{\mathcal{R}}(P)$. A tree $\mathcal{T}$ is accepted by A if there is a q_0-accepting run on $\mathcal{T}$. Let $\mathcal{L}(\mathsf{A})$ be the set of trees which are accepted by A. A pointed Kripke model $\mathcal{M}, s$ is accepted by A iff its tree unravelling is accepted by A.

Given a run $\mathcal{R}$ of A on $\mathcal{T}$ and a state $v \in \mathcal{T}$, we let $W^{\mathcal{R}}(v) = \{w \mid L_{\mathcal{R}}(w) = (v,q)$ for some $q \in Q_{\mathsf{A}}\}$. By the definition of the run, we can verify that any two nodes in $\mathcal{R}$ labelled by the same node in $\mathcal{T}$ are at the same level in $\mathcal{R}$: for any $w, w' \in W^{\mathcal{R}}(v)$: $dep(\mathcal{R}, w) = dep(\mathcal{R}, w')$.

The following fundamental theorem relates the Mu formulas with $\mu-$automata.

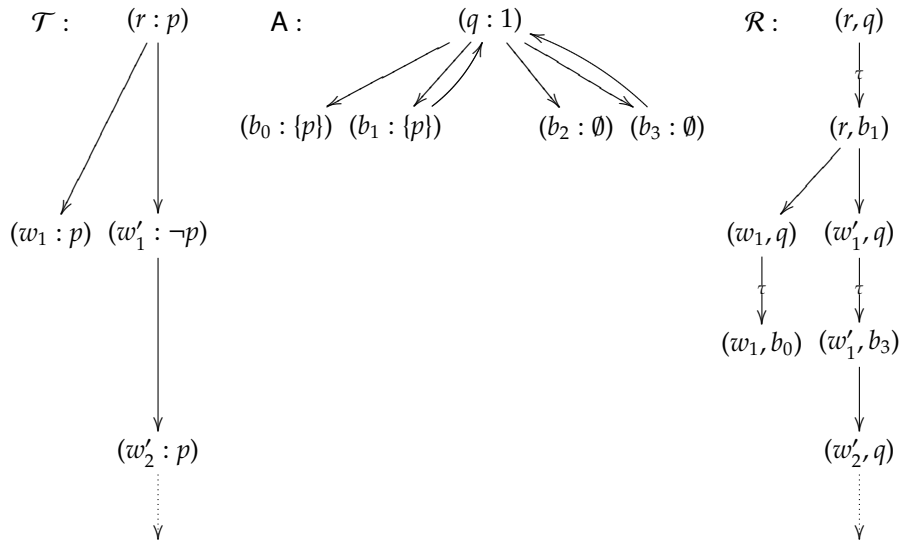
6.2.2. T . *[JW95] For each $\mu-$automaton there is an equivalent Mu$-$formula. For each Mu$-$formula there is an equivalent $\mu-$automaton .*

Since Mu is invariant under bisimulation then we have:

6.2.3. C . *If $\mathcal{M}, s \leftrightarrows N, t$, then $\mathcal{M}, s$ is accepted by a $\mu-$ automaton A iff $\mathcal{N}, t$ is accepted by A.*

We end this section with an example to illustrate how the μ-automata work:

6.2.4. E .



where we fix $\mathbf{P} = \{p\}$. In the tree $\mathcal{T}$, $(x : y)$ indicates that y holds at state x; In A, $(q : 1)$ indicates that $\Omega(q) = 1$ and $(b_i : Z)$ means $L_{\mathsf{A}}(b_i) = Z$; In the run $\mathcal{R}$, (x, y) are the labels associated to the nodes. It is not hard to see that $\mathcal{R}$ is a run of A on $\mathcal{T}$, however, it is not accepting. Actually A is the μ-automaton corresponding to the Mu formula $\mu X.\Box X$ which expresses well-foundedness: there is no infinite descending chain (cf., e.g., [GO06]). Ω

6.3 Cardinality of the Tree Languages

The collection of the models recognized by an arbitrary $\mu-$automaton is not always a set, even up to bisimulation. For instance, take the $\mu-$automaton A in Example 6.2.4 which is equivalent to the $\mu-$calculus formula $\mu X.\Box X$ that expresses well-foundedness. Now for any ordinal α, consider the Kripke structure $\mathcal{M}_\alpha = \{\{\beta \mid \beta \leq \alpha\}, \rightarrow, V\}$ where $\beta \rightarrow \beta' \iff \beta' < \beta \leq \alpha$ (the inverse of the order relation). It is clear that for all α, $\mathcal{M}_\alpha$ is well-founded thus recognized by the automaton A. By induction, one can prove that $\mathcal{M}_\alpha$ is not bisimilar to $\mathcal{M}_\beta$ if $\alpha \neq \beta$. Therefore there are $\mu-$automata which recognize class-size tree languages.

In this chapter, we concentrate on image-finite models in which each state has only finitely many successors w.r.t the same label in $\mathbf{\Sigma}$. It is clear that the tree unravelling of an image-finite model is again an image-finite model. Note that the class of image-finite models has the Hennessy-Milner property for Mu: for any image-finite models $\mathcal{M}, s_0$ and $\mathcal{N}, t_0$: $\mathcal{M}, s_0 \leftrightarrows \mathcal{N}, t_0 \iff \mathcal{M}, s_0$ and $\mathcal{N}, t_0$ satisfy the same Mu formulas (cf., e.g, [BdRV02]). Therefore, if a Mu-formula ϕ has n non-bisimilar image-finite models, then all these different models can be told apart by Mu$-$formulas.

The rest of this section is devoted to our main result (Theorem 6.3.5): a $\mu-$automata recognizable set of image-finite models modulo bisimulation is uncountable iff it has the cardinality of the continuum iff it contains a non-B-regular tree. This can be seen as an analogy of the main result in [Niw91], with Rabin tree automata on ranked infinite-trees replaced by μ-automata on Kripke models, and isomorphism[4] replaced by bisimulation. The new setting significantly complicates the proof of the main theorem by two reasons:

1. A path in a tree that is accepted by a $\mu-$automaton, may correspond to a *tree* in the accepting run, not always a *path* as in the case of Rabin automata on ranked trees.
2. Bisimulation requires much more care than isomorphism, as we will see in the proof of Theorem 6.3.5, where we need infinitely many non-bisimular trees with complicated construction, while two non-isomorphic trees are enough for the corresponding proof in [Niw91].

To simplify the discussion, we focus on trees without action labels. Therefore we omit $\mathbf{\Sigma}$ in the definition of Kripke models and trees. To generalize the result to the case with finitely many action labels is a standard exercise. In the context of unlabelled trees, it is conventional to say a tree is *image-finite* if it is finitely branching. We also fix a label set $2^{\mathbf{P}}$ for the trees and omit the label set $W_{\mathcal{T}} \times (Q_{\mathsf{A}} \cup B_{\mathsf{A}})$ in runs of A on a tree $\mathcal{T}$ when the context is clear.

The following two propositions and the later lemma intend to deal with the first complication we mentioned earlier. First, we show that for image-finite trees the accepting runs (if they exist) can also be image-finite.

6.3.1. P . *If an image-finite tree $\mathcal{T}$ is accepted by A then there is an image-finite accepting run $\mathcal{R}'$ of A on $\mathcal{T}$, such that there are no two sibling nodes w and w' in $\mathcal{R}'$ satisfying $L_{R'}(w) = L_{R'}(w')$.*

P Suppose there is an accepting run $\mathcal{R}$ of A on $\mathcal{T}$ and there is a node w in $\mathcal{R}$ which has infinitely many successors. It is clear that w must be labelled by a $\mathtt{BR}$ state in A, thus $L_{\mathcal{R}}(w) = (v, b)$ for some v in $\mathcal{T}$ and $b \in B_{\mathsf{A}}$. We define an equivalence relation $\sim$ among the successors of w such that $w' \sim w'' \iff L_{\mathcal{R}}(w) = L_{\mathcal{R}}(w')$. With the presence of the Axiom of Choice, we pick one w' as a representative in each of the equivalence classes w.r.t $\sim$. Since $\mathcal{T}$ is image-finite and Q_{A} is finite, there are only finitely many such equivalence classes. A moment's reflection should confirm that we can then prune the successors other than the representatives safely, such that the resulting run $\mathcal{R}'$ is still an accepting run of A on $\mathcal{T}$. ⋈

Given a $\mu-$automaton A, we call a run $\mathcal{R}$ of A on $\mathcal{T}$ *non-parallel* if for any $v \in \mathcal{T}$: $Q^R(v)$ is a singleton. We sometimes write $Q^R(v) = q$ instead of $Q^R(v) = \{q\}$.

[4] more precisely, some notion of tree identity.

6.3.2. P . *If a μ-automaton A accepts an image-finite tree $\mathcal{T}$, then A accepts an image-finite tree $\mathcal{T}'$ such that there is a non-parallel accepting run of A on $\mathcal{T}'$ and $\mathcal{T} \leftrightarrows \mathcal{T}'$.*

P Suppose $\mathcal{R} = \{W_\mathcal{R}, \to_\mathcal{R}, L_\mathcal{R}, r_\mathcal{R}\}$ is a successful run of $\mathcal{T} = \{W_\mathcal{T}, \to_\mathcal{T}, L_\mathcal{T}, r_\mathcal{T}\}$ on A. Let $\mathcal{T}' = \{W_{\mathcal{T}'}, \to_{\mathcal{T}'}, L_{\mathcal{T}'}, r_{\mathcal{T}'}\}$ where:

- $W_{\mathcal{T}'} = \{w \in W_\mathcal{R} \mid L_\mathcal{R}(w) = (v, q) \text{ for some } v \in \mathcal{T}, q \in Q_\mathsf{A}\}$.
- $v \to_{\mathcal{T}'} v' \iff v \xrightarrow{\tau}_\mathcal{R} w \to_\mathcal{R} v'$ for some $w \in \mathcal{R}$.
- $L_{\mathcal{T}'}(w) = L_\mathcal{T}(v)$ for $v \in \mathcal{T}$ such that $L_\mathcal{R}(w) = (v, q)$.
- $r_{\mathcal{T}'} = r_\mathcal{R}$.

It is not hard to see that $\mathcal{T}'$ is accepted by A through a successful run

$$\mathcal{R}' = \{W_\mathcal{R}, \to_\mathcal{R}, L', r_\mathcal{R}\}$$

where:

$$L'(w) = \begin{cases} (w, q) & if \quad L_\mathcal{R}(w) = (v, q) \text{ for some } v \in \mathcal{T} \\ (w', b) & if \quad L_\mathcal{R}(w) = (v, b) \text{ for some } v \in \mathcal{T} \text{ and } w' \xrightarrow{\tau} w \text{ in } \mathcal{R} \end{cases}$$

Given $u \in \mathcal{T}'$, suppose $L'(w) = (u, q)$ and $L'(w') = (u, q')$ for some $w, w' \in \mathcal{R}'$. By the definition of L', we have $w = u = w'$ thus $q = q'$. Therefore, for all $u \in \mathcal{T}' : Q^{\mathcal{R}'}(u)$ is a singleton.

We now define binary relations $\sim$ on $W_\mathcal{T} \times W_{\mathcal{T}'}$ as follows:

$$v \sim u \iff L_\mathcal{R}(u) = (v, q) \text{ for some } q \in Q_\mathsf{A}.$$

We claim $\sim$ is a bisimulation between $\mathcal{T}$ and $\mathcal{T}'$. For every pair $(v, u) \in \sim$ with $L_\mathcal{R}(u) = (v, q)$ the three conditions of bisimulation hold:

1. $L_\mathcal{T}(v) = L_{\mathcal{T}'}(u)$: by definition.
2. Suppose $v \to_\mathcal{T} v'$. Then by the definition of the run $\mathcal{R}$, there must be a node $u' \in \mathcal{R}$ such that $L_\mathcal{R}(u') = (v', q')$ for some q' and $u \xrightarrow{\tau}_\mathcal{R} \to_\mathcal{R} u'$. It is clear that $u \to_{\mathcal{T}'} u'$ in $\mathcal{T}'$ and $v' \sim u'$.
3. Suppose $u \to_{\mathcal{T}'} u'$ in $\mathcal{T}'$. Then by the definition of $\mathcal{T}'$, $u \xrightarrow{\tau}_\mathcal{R} \to_\mathcal{R} u'$ in $\mathcal{R}$. By the definition of $\mathcal{R}$, there is some $v' \in \mathcal{T}$ such that $L_\mathcal{R}(u') = (v', q')$ and $v \to_\mathcal{T} v'$.

⋓

We call a run $\mathcal{R}$ of A on a tree $\mathcal{T}$ *simple* if $\mathcal{R}$ is image-finite and $W^\mathcal{R}(v)$ is a singleton for any $v \in T$. Now based on Propositions 6.3.1 and 6.3.2, we can prove the following lemma:

6.3.3. L . *If a μ-automaton A accepts an image-finite tree $\mathcal{T}$, then A accepts an image-finite tree $\mathcal{T}'$ such that $\mathcal{T} \leftrightarrows \mathcal{T}'$ and there is an accepting simple run $\mathcal{R}$ of A on $\mathcal{T}'$. Therefore, a path in $\mathcal{T}'$ has a unique corresponding path in $\mathcal{R}$.*

P Given $\mathcal{T}$, we first build an image-finite accepting run as in the proof of Proposition 6.3.1. Then we can convert this run into a tree $\mathcal{T}'$ which is bisimilar to $\mathcal{T}$ according to Proposition 6.3.2, with a non-parallel run $\mathcal{R}$. Now we show that $W^{\mathcal{R}}(v)$ is a singleton for any $v \in T'$. Suppose not, then there is a node $v \in \mathcal{T}'$ such that there are w and $w' \in \mathcal{R}$ such that $L_{\mathcal{R}}(w) = (v, q)$ and $L_{\mathcal{R}}(w') = (v, q)$. By the definition of the run, it is clear that w and w' are at the same level of $\mathcal{R}$. According to Proposition 6.3.1, w and w' can not be siblings. Therefore there must be a *departure node* w_0 with $L_{\mathcal{R}}(w_0) = (v', b)$ for some $v' \in \mathcal{T}'$ and $b \in B_{\mathsf{A}}$, such that $w_0 \to w_1 \to^n w$ and $w_0 \to w_2 \to^n w'$ for some $w_1 \neq w_2$ and a natural number $n > 0$. Suppose $L(w_1) = (v_1, q_1)$ and $L(w_1) = (v_2, q_2)$, then it is not hard to see that $v_1 \neq v_2$, since w_1 and w_2 are siblings. It s clear that v_1 and v_2 are also siblings in $\mathcal{T}'$. However, it is impossible to reach the same point v from two sibling nodes in $\mathcal{T}'$, according to the definition of the trees. Contradiction.

For any $v \in \mathcal{T}'$, since $W^{\mathcal{R}}(v)$ is a singleton, we let $\mathcal{R}(v)$ be the unique element in $W^{\mathcal{R}}(v)$. Now given an infinite path $P : v_0, v_1, v_2 \dots$ in $\mathcal{T}'$ we can find the corresponding unique path in $\mathcal{R} : \mathcal{R}(v_0) \xrightarrow{\tau} w_0 \to \mathcal{R}(v_1) \xrightarrow{\tau} w_1 \to \mathcal{R}(v_1) \xrightarrow{\tau} \dots$, where w_i are unique successors of $\mathcal{R}(v_i)$ in $\mathcal{R}$.

⋓

Before we go to the main theorem, we need the following lemma which helps to provide a source tree to be pumped in the later proof.

6.3.4. L . *If an image-finite tree $\mathcal{T}$ is not B-regular then there is an infinite path $P = v_0, v_1, \dots$ such that:*

$$\textit{for any } k \in \mathbb{N}, \textit{ any } v \textit{ such that } dep(\mathcal{T}, v) < dep(\mathcal{T}, v_k)\textit{: } \mathcal{T}^{v_k} \not\leftrightarrows \mathcal{T}^{v}.$$

Namely, for each v_k, $\mathcal{T}^{v_k}$ is a "new" subtree which does not appear up to bisimulation at earlier levels of the tree.

P Let $V = \{v \in \mathcal{T} \mid \forall w \in \mathcal{T} : \mathcal{T}^v \leftrightarrows \mathcal{T}^w \implies dep(\mathcal{T}, v) \leq dep(\mathcal{T}, w)\}$. Intuitively, V is the set of nodes where a "new" tree appears, in top-bottom order. Let $V' = \{v : v \to^* w \text{ in } \mathcal{T}, \text{ for some } w \in V\}$, then $\mathcal{T}' = \{V', \to|_{V' \times V'}, L|_{V'}, r\}$ is a tree. Due to the fact that T is image-finite and non-B-regular, T' is an image-finite infinite tree. With the presence of Axiom of Choice, we recall König's Lemma on unordered trees:

An image-finite infinite tree has an infinite path.

Thus there is an infinite path $P = v_0, v_1, v_2, \dots$ in T'. Clearly P is also an infinite path in $\mathcal{T}$. Now suppose towards contradiction that there are $v_k \in P$ and $v \in \mathcal{T}$ such that $dep(\mathcal{T}, v) < dep(\mathcal{T}, v_k)$ and $\mathcal{T}^{v_k} \leftrightarrows \mathcal{T}^{v}$. By the definition of V', there is a w such that $v_k \to^n w$ and $w \in V$ for some n. Since $\mathcal{T}^{v} \leftrightarrows \mathcal{T}^{v_k}$ then there is a $w' : v \to^n w'$ in $\mathcal{T}$

such that $\mathcal{T}^w \leftrightarrows \mathcal{T}^{w'}$. However, it is clear that $dep(\mathcal{T}, w') < dep(\mathcal{T}, w)$, thus $w \notin V$, contradiction. ⋈

Now we come to our main theorem. Note that we only consider image-finite models in the sequel.

6.3.5. T . *Let* A *be a* μ*-automaton. Then the following are equivalent:*

1. $|\mathcal{L}(\mathsf{A})|_{/\underline{\leftrightarrow}} = 2^{\aleph_0}$,
2. $|\mathcal{L}(\mathsf{A})|_{/\underline{\leftrightarrow}} > \aleph_0$,
3. $\mathcal{L}(\mathsf{A})$ *contains a non-B-regular tree.*

P (1) $\Longrightarrow$ (2) is straightforward.

(2) $\Longrightarrow$ (3) : Suppose $\mathcal{L}(\mathsf{A})$ only contains B-regular trees. Then by the definition of B-regular trees, each tree in $\mathcal{L}(\mathsf{A})$ is bisimilar to its bisimulation contraction, which is finite. However, there are only countably many such finite Kripke models, given a finite set $\mathbf{P}$ of basic propositions.

(3) $\Longrightarrow$ (1) : Observe that each ω−branching tree[5] can be viewed as a downward closed subset of $\mathbb{N}^*$. Since $|\mathbb{N}^*| = \aleph_0$, clearly $|\mathcal{L}(\mathsf{A})|_{/\underline{\leftrightarrow}} \leq 2^{\aleph_0}$. We will prove (3) $\Longrightarrow$ $|\mathcal{L}(\mathsf{A})|_{/\underline{\leftrightarrow}} \geq 2^{\aleph_0}$. The idea of the proof is to *pump* $2^{\aleph_0}$ many non-bisimilar acceptable trees out of a non-B-regular tree.

Suppose $\mathcal{L}(\mathsf{A})$ contains an image-finite non-B-regular tree $\mathcal{T}^o$. By Lemma 6.3.3, $\mathcal{L}(\mathsf{A})$ contains an image-finite tree $\mathcal{T} = (W, \rightarrow, L, r)$ bisimilar to $\mathcal{T}^o$, such that there is an accepting run $\mathcal{R}$ of A on $\mathcal{T}$ and $W^{\mathcal{R}}(v)$ is a singleton for each $v \in \mathcal{T}$. Clearly, $\mathcal{T}$ is also non-B-regular.

By Lemma 6.3.4, there is an infinite path $P : v_0 = r, v_1, v_2, \ldots$ such that for any $k \in \mathbb{N}$ and any $v \in \mathcal{T} : dep(\mathcal{T}, v) < dep(\mathcal{T}, v_k) \implies \mathcal{T}^{v_k} \not\underline{\leftrightarrow} \mathcal{T}^{v}$. It is obvious that for any $j \neq i : \mathcal{T}^{v_i} \not\underline{\leftrightarrow} \mathcal{T}^{v_j}$. Since $\mathcal{R}$ is non-parallel, $Q_{\mathcal{R}}(v_i)$ is a singleton for any $v_i \in P$.

We now pick a *distant* node $v_m \in P$ such that $Q^{\mathcal{R}}(v_m) = \{q\} \subseteq Inf(\mathcal{R}, P)$ for some $q \in Q_{\mathsf{A}}$ and $Q^{\mathcal{R}}(P^{v_m}) = Inf(\mathcal{R}, P)$, where P^{v_m} is the suffix of P starting at v_m. Intuitively, we pick v_m in such a way that all the points in P after v_m are only matched with the states in the automaton that appear infinitely often according to the labelling of P in $\mathcal{R}$. We then find an infinite subsequence $P' : v'_0, v'_1, v'_2 \ldots$ of P such that $v'_0 = v_m$ and for each k: $Q^{\mathcal{R}}(v'_{k+1}) = \{q\}$ and $Q^{\mathcal{R}}(P^{v'_k}_{v'_{k+1}}) = Inf(\mathcal{R}, P)$, where $P^{v'_k}_{v'_{k+1}}$ is the segment of P between v'_k and v'_{k+1}. Note that such P' exists since $q \in Inf(\mathcal{R}, P)$.

Now we are ready to pump the tree $\mathcal{T}$ into $2^{\aleph_0}$ many non-bisimilar trees. For each infinite sequence α of 0s and 1s, we construct a tree $\mathcal{T}_\alpha$ which is accepted by A. We do this by building the sequence of triples $(\mathcal{T}_{\alpha_n}, u_{\alpha_n}, s_{\alpha_n})$ where α_n is a prefix of α of the length n and u_{α_n},$s_{\alpha_n} \in P'$. Intuitively u_{α_n} is the"*replacing point*" and $\mathcal{T}^{s_{\alpha_n}}$ is the "*substitution*".

[5] ω−branching trees are the trees that have at most $\aleph_0$ many successors at each node.

Before defining the pumped trees formally, recall Fact 2.2.5: if two pointed models $(\mathcal{M}, s_0)$ and $(\mathcal{N}, t_0)$ are not bisimilar then Spoiler has a winning strategy in some n-round bisimulation game $\mathcal{G}_n((\mathcal{M}, s), (\mathcal{N}, t))$. We let $g \underline{\leftrightarrow} ((\mathcal{M}, s), (\mathcal{N}, t))$ be the minimal number n_0 such that the spoiler has a winning strategy for the n_0-round bisimulation game $\mathcal{G}_{n_0}((\mathcal{M}, s), (\mathcal{N}, t))$.

We start from $\mathcal{T}_\epsilon = \mathcal{T}, u_\epsilon = v'_0, s_\epsilon = v'_1$. For any finite binary sequence β, $\mathcal{T}_{\beta\cdot 0}$ and $\mathcal{T}_{\beta\cdot 1}$ are constructed as follows:

- $\mathcal{T}_{\beta\cdot 1} = \mathcal{T}_\beta[u_\beta \backslash s_\beta]$, $u_{\beta\cdot 1} = next(s_\beta, D(\mathcal{T}_\beta, u_\beta, s_\beta))$ and $s_{\beta\cdot 1} = next(u_{\beta\cdot 1}, 0)$.
- $\mathcal{T}_{\beta\cdot 0} = \mathcal{T}_\beta$, $u_{\beta\cdot 0} = u_{\beta\cdot 1}$ and $s_{\beta\cdot 0} = s_{\beta\cdot 1}$.

where

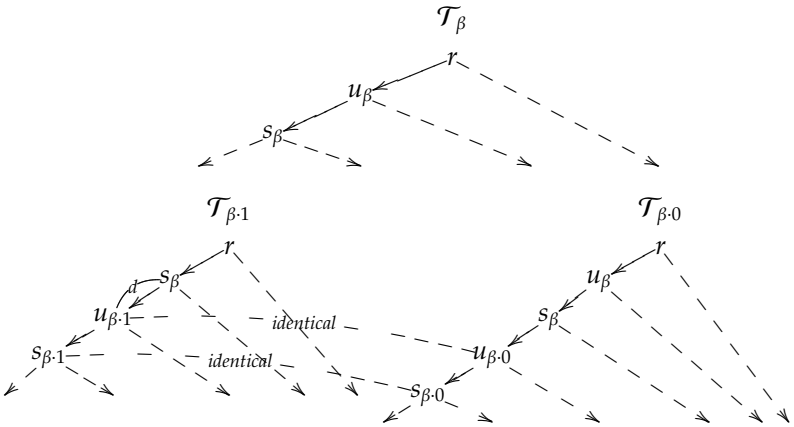
- for u, s in P' such that $dep(\mathcal{T}, u) < dep(\mathcal{T}, s)$, $D(\mathcal{T}, u, s)$ is the depth of the zone where the non-bisimilarity between $\mathcal{T}^s$ and any other subtree at the same level of $\mathcal{T}^u$ can be detected (by bisimulation games). Formally:

$$D(\mathcal{T}, u, s) = max\{g \underline{\leftrightarrow} (\mathcal{T}^{u'}, \mathcal{T}^s) \mid dep(\mathcal{T}, u') = dep(\mathcal{T}, u)\}.$$

- for s in P', $n \in \mathbb{N}$: $next(s, n)$ is defined as v'_k in P' such that k is the minimal index satisfying $dep(\mathcal{T}, v'_k) - dep(\mathcal{T}, s) > n$. Intuitively, $next(s, n)$ is the *next* point in P' after s, such that the zone with depth n can be preserved.

Note that $g \underline{\leftrightarrow} (\mathcal{T}^{u'}, \mathcal{T}^s)$ is well-defined since for any s on P', $dep(\mathcal{T}, u) < dep(\mathcal{T}, s)$ implies $\mathcal{T}^u \not\underline{\leftrightarrow} \mathcal{T}^s$. Moreover $\{g \underline{\leftrightarrow} (\mathcal{T}^{u'}, \mathcal{T}^s) \mid dep(\mathcal{T}, u') = dep(\mathcal{T}, u)\}$ is a finite set since $\mathcal{T}$ is image-finite, thus the maximal element of this set exists.

Let $d = D(\mathcal{T}_\beta, u_\beta, s_\beta)$, the intuition behind the above construction is illustrated as follows:



We build $\mathcal{T}_{\beta\cdot 1}$ by placing the substitution $\mathcal{T}^{s_\beta}$ at the replacing point u_β. Then we let the next replacing point $u_{\beta\cdot 1}$ be far away enough from the previous substitution point s_β, such that the non-bisimilarity of $\mathcal{T}^{s_\beta}$ and its new neighbour subtrees at the same

level in $\mathcal{T}_\beta[u_\beta \backslash s_\beta]$ can be detected by a bisimulation game before reaching the new replacing point. Finally we let the new substitution be the"next" suitable point after $s_{\beta\cdot 1}$ in P'. For $\mathcal{T}_{\beta\cdot 0}$ we simply do not execute the substitution but change the $s_{\beta\cdot 0}$ and $u_{\beta\cdot 0}$ as in the case of $\mathcal{T}_{\beta\cdot 1}$.

Given an infinite binary sequence α, we can now build $\mathcal{T}_{\alpha_n}$ for each n. To build $\mathcal{T}_\alpha$, we define the *stable* part of $\mathcal{T}_{\alpha_n}$ as follows:

- *stable domain:* $sdom(\mathcal{T}_{\alpha_n}) = W_{\mathcal{T}_{\alpha_n}} - \{v \mid u_{\alpha_n} \to^* v \text{ in } \mathcal{T}_{\alpha_n}\}$
- *stable edges:* $sedge(\mathcal{T}_{\alpha_n}) = \to_{\mathcal{T}_{\alpha_n}} |_{sdom(\mathcal{T}_{\alpha_n})\times sdom(\mathcal{T}_{\alpha_n})}$.
- *stable label:* $slabel(\mathcal{T}_{\alpha_n}) = L_{\mathcal{T}_{\alpha_n}}|_{sdom(\mathcal{T}_{\alpha_n})}$.

It is not hard to see that the above defined stable part of $\mathcal{T}_{\alpha_n}$ does not get altered in $\mathcal{T}_{\alpha_m}$ for $m > n$. Due to such monotonicity, we can now build the limit tree:

$$\mathcal{T}_\alpha = (\bigcup_{n<\omega} sdom(\mathcal{T}_{\alpha_n}), \bigcup_{n<\omega} sedge(\mathcal{T}_{\alpha_n}), \bigcup_{n<\omega} slabel(\mathcal{T}_{\alpha_n}))$$

Since $\mathcal{R}$ is an accepting run of A on $\mathcal{T}$ such that $W^{\mathcal{R}}(v)$ is a singleton for each $v \in \mathcal{T}$, every node in $\mathcal{T}$ corresponds to a two-node path $w \xrightarrow{\tau} w'$ in $\mathcal{R}$ such that $L_{\mathcal{R}}(w) = (v, q)$ and $L_{\mathcal{R}}(w') = (v, b)$ for some $b \in B_{\mathsf{A}}$ and $\{q\} = Q^{\mathcal{R}}(w)$. Then it is easy to see that we can build each $\mathcal{R}_{\alpha_n}$ based on $\mathcal{R}_{\alpha_{n-1}}$, by the corresponding substitution as in the construction for $\mathcal{T}_{\alpha_n}$. Such $\mathcal{R}_{\alpha_n}$ is indeed a run of $\mathcal{T}_{\alpha_n}$ since the replacing point and the substitution point in $\mathcal{T}_{\alpha_{n-1}}$ are all labelled with the same $q \in Q_{\mathsf{A}}$. Similarly, we can define the stable parts of each $\mathcal{R}_{\alpha_n}$ and let:

$$\mathcal{R}_\alpha = (\bigcup_{n<\omega} sdom(\mathcal{R}_{\alpha_n}), \bigcup_{n<\omega} sedge(\mathcal{R}_{\alpha_n}), \bigcup_{n<\omega} slabel(\mathcal{R}_{\alpha_n}))$$

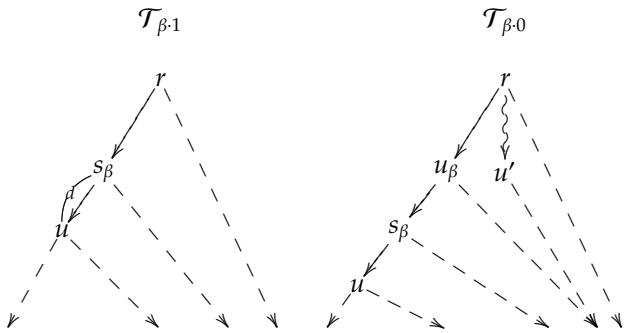
To see the limit run $\mathcal{R}_\alpha$ is also an accepting run of A on $\mathcal{T}_\alpha$, we need to check the parity condition for every infinite path in $\mathcal{R}_\alpha$.

Note that if an infinite path P_1 in $\mathcal{R}_\alpha$ is contained in a stable part of $\mathcal{R}_{\alpha_n}$ for some n, then we can find a path P_1' in $\mathcal{R}$ such that P_1 and P_1' share an infinite suffix. Then it is clear that P_1 and P_1' only differ in their finite prefixes, thus $\Omega^{\mathcal{R}_\alpha}(P_1) = \Omega^{\mathcal{R}_{\alpha_n}}(P_1) = \Omega^{\mathcal{R}}(P_1')$ which is even. The non-trivial case is the *limit path* P_α which is not contained in the stable part of any $\mathcal{T}_{\alpha_n}$, thus goes through infinitely many substitution points $r, u_{\alpha_0}, u_{\alpha_1}, u_{\alpha_2} \cdots$. Since $Q_{\mathcal{R}}(P^{v_0}) = Inf(\mathcal{R}, P)$ and $Q_{\mathcal{R}}(P^{v_k'}_{v_{k+1}'}) = Inf(\mathcal{R}, P)$ for any $k < \omega$, then the construction of the limit run can neither make any $q' \notin Inf(\mathcal{R}, P)$ occur infinitely often nor make any $q' \in Inf(\mathcal{R}, P)$ occur only finitely often. Therefore $\Omega^{\mathcal{R}_\alpha}(P_\alpha) = \Omega^{\mathcal{R}}(P)$. In sum, $\mathcal{R}_\alpha$ is indeed an accepting run of A on $\mathcal{T}_\alpha$.

To complete the above proof, we need to show that for any $\alpha \neq \alpha' \in 2^*$, $\mathcal{T}_\alpha \not\leftrightarrow \mathcal{T}_{\alpha'}$, which is proved by the following lemma. ⋈

6.3.6. L . *If $\alpha \neq \alpha' \in 2^\omega$, then Spoiler can win the bisimulation game $G_n(\mathcal{T}_\alpha, \mathcal{T}_{\alpha'})$ for some $n < \omega$.*

P Since $\alpha \neq \alpha'$, there is a sequence $\beta \in 2^*$ such that $\alpha_m = \beta \cdot 0$ and $\alpha'_m = \beta \cdot 1$ for some m. We recall the construction of $\mathcal{T}_{\beta\cdot 1}$ and $\mathcal{T}_{\beta\cdot 0}$ as follows:



where $u = u_{\beta\cdot 0} = u_{\beta\cdot 1}$. Note that in both trees, the parts which are not reachable from u will be preserved in $\mathcal{T}_\alpha$ and $\mathcal{T}_{\alpha'}$, according to the construction of the limit tree. We claim that there is a winning strategy for spoiler in the game $\mathcal{G}_n(\mathcal{T}_\alpha, \mathcal{T}_{\alpha'})$ where $n = dep(\mathcal{T}_{\beta\cdot 1}, u) = dep(\mathcal{T}_\beta, u_\beta) + d$.

It is clear that Spoiler has a strategy to reach the node s_β in $\mathcal{T}'_\alpha$ by following a path in $\mathcal{T}_{\alpha'}$ from r. To match Spoiler's moves, the verifier, if it has not lost already, will reach another point u' in $\mathcal{T}_\alpha$ with the same depth. Since $d > max\{g_{\underline{\leftrightarrow}}(\mathcal{T}^v, \mathcal{T}^{s_\beta}) \mid dep(\mathcal{T}, v) = dep(\mathcal{T}, u_\beta)\}$ as in the construction of $\mathcal{T}_{\beta\cdot 1}$, Spoiler can win the bisimulation game $\mathcal{G}_d(\mathcal{T}^{u'}, \mathcal{T}^{s_\beta})$. Therefore Spoiler has a strategy to show the difference between $\mathcal{T}^{u'}$ and $\mathcal{T}^{s_\beta}$ in d steps, namely within the stable parts of $\mathcal{T}_{\beta\cdot 1}$ and $\mathcal{T}_{\beta\cdot 0}$. Thus Spoiler can win the bisimulation game $\mathcal{G}_d(\mathcal{T}^{u'}_\alpha, \mathcal{T}^{s_\beta}_{\alpha'})$. In sum, Spoiler has a winning strategy for the game $\mathcal{G}_n(\mathcal{T}_\alpha, \mathcal{T}_{\alpha'})$. ⋈

6.4 Normal Form of the Countable Languages

Following [Niw91], we call a tree $\mathcal{T}$ *alive* if $\mathcal{T}_{/\underline{\leftrightarrow}} \in Sub_{\underline{\leftrightarrow}}(\mathcal{T})$, namely $\mathcal{T}$ can *regenerate* itself up to bisimulation. Given a μ−automaton A, let $alive(\mathcal{L}(\mathsf{A}))$ be the collection of all the alive subtrees of the trees accepted by A:

$$alive(\mathcal{L}(\mathsf{A})) = \{\mathcal{T} \mid \mathcal{T} \text{ is an alive subtree of some } \mathcal{T}' \in \mathcal{L}(\mathcal{A})\}.$$

It is easy to see that a tree is alive if and only if there is a cycle in its bisimulation contraction starting from $[r]_{/\underline{\leftrightarrow}}$. An infinite path P is called a *regenerating path* of an alive tree $\mathcal{T}$, if P goes through infinitely many nodes: $v_0, v_1 \ldots$ such that each $\mathcal{T}^{v_i} \underline{\leftrightarrow} \mathcal{T}$. We call those v_i *regenerating points* of P. It is clear that every alive tree has at least one regenerating path starting at the root. We say an alive tree $\mathcal{T}$ is *q-lively-accepted* by A, if there is an accepting simple q−run $\mathcal{R}$ of A on $\mathcal{T}$, such that there are infinitely many regenerating points $v_0, v_1 \ldots$ in a regenerating path st. $Q^{\mathcal{R}}(v_i) = \{q\}$ for each $i \in \mathbb{N}$.

6.4.1. P . *For every tree $\mathcal{T}_0 \in alive(\mathcal{L}(\mathsf{A}))$, there is a tree $\mathcal{T}_1 \leftrightarrows \mathcal{T}_0$ such that $\mathcal{T}_1$ is q-lively-accepted by A for some state $q \in Q_\mathsf{A}$ which is reachable from the start state q_0 of A.*

P Given a tree $\mathcal{T}_0 \in alive(\mathcal{L}(\mathsf{A}))$, let $\mathcal{T} \in \mathcal{L}(\mathsf{A})$ be a tree such that $\mathcal{T}_0$ is a subtree of $\mathcal{T}$. By Lemma 6.3.3, there is a tree $\mathcal{T}' \leftrightarrows \mathcal{T}$ such that there is a q_0-run $\mathcal{R}'$ which is accepting and $W^{\mathcal{R}'}(v)$ is a singleton for each $v \in \mathcal{T}'$. It is clear that $\mathcal{T}'$ has an alive subtree $\mathcal{T}'_0$ which is bisimilar to $\mathcal{T}_0$. Then there is an infinite regenerating path P starting at the root of $\mathcal{T}'_0$. Consider the corresponding path P' in $\mathcal{R}'$, it is easy to see that there are infinitely many nodes corresponding to the regenerating points in P which are labelled with the same state $q \in Q_\mathsf{A}$.

⋈

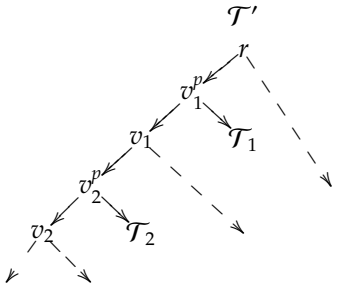
We now show a countable language only involves finitely many alive trees up to bisimulation. Differing from the case of Rabin Automata on ranked tress in [Niw91], our μ−automata work on unranked trees, thus we can now construct new acceptable trees by *inserting* branches to an existing acceptable tree, as shown in the proof of the following lemma.

6.4.2. L . *Given a μ−automaton A, if $\mathcal{L}(\mathsf{A})$ is countable up to bisimulation then $|alive(\mathcal{L}(\mathsf{A}))|_{/\leftrightarrows}$ is finite.*

P Suppose towards contradiction that $\mathcal{L}(\mathsf{A})$ is countable but $|alive(\mathcal{L}(\mathsf{A}))|_{/\leftrightarrows}$ is infinite. By Proposition 6.4.1 and the pigeon hole argument, there are infinitely many non-bisimilar alive trees $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots$ that are q−lively accepted by A for some $q \in Q_\mathsf{A}$ which is reachable from the start state of A.

By Proposition 6.4.1 let $\mathcal{R}$ be a legal simple run of A on a tree $\mathcal{T}$ with some node $v \in \mathcal{T}$ such that $Q^{\mathcal{R}}(v) = \{q\}$. Then $\mathcal{T}[v \backslash \mathcal{T}_0]$ is accepted by the simple run $\mathcal{R}[W^{\mathcal{R}}(v) \backslash \mathcal{R}_0]$, where $\mathcal{R}_0$ is the q-alive-accepting run of $\mathcal{A}$ on $\mathcal{T}_0$.

Since $\mathcal{T}_0$ is alive, we can find an infinite path in $\mathcal{T}[v \backslash \mathcal{T}_0]$ containing an infinite subsequence of regenerating points $P : v_1, v_2 \ldots$ where $\mathcal{T}^{v_i} \leftrightarrows \mathcal{T}_0$ for each $i \in \mathbb{N}$. Based on $\mathcal{T}_0$, we now build a non-B-regular tree $\mathcal{T}'$ by "inserting" $\mathcal{T}_i$ as a child of the parent node of v_i for each $i > 0$:



where v_i^p is the parent node of v_i. Take the simple run $\mathcal{R}[W^{\mathcal{R}}(v) \backslash \mathcal{R}_0]$ of A on $\mathcal{T}[v \backslash \mathcal{T}_0]$. Since $\mathcal{T}_i$ are all q-accepted, we can build a run $\mathcal{R}'$ for $\mathcal{T}'$ by inserting the corresponding

$q-$run $\mathcal{R}_i$ of $\mathcal{T}_i$ as a child of the unique w_i in $\mathcal{R}'$ such that $L_{\mathcal{R}}(w_i) = (v_i^p, b)$ for some b in B_{A}.

It is not hard to see that $\mathcal{R}'$ is legal run and all the infinite paths satisfy the parity condition thus $\mathcal{R}'$ is accepting. Thus $\mathcal{L}(\mathsf{A})$ contains a non-B-regular tree. But then by Theorem 6.3.5, $\mathcal{L}(\mathsf{A})$ is uncountable. Contradiction. ⋈

Let $\mathfrak{F}_n$ be the set of finite trees with some leaves possibly labelled by variables $x_1, x_2, \ldots, x_n$. Given a $\mathcal{T}_f \in \mathfrak{F}_n$, let $\mathcal{T}_f[x_1 \backslash \mathcal{T}_1, \ldots, x_1 \backslash \mathcal{T}_n]$ be the tree obtained by replacing each x_i-labelled node in $\mathcal{T}_f$ by the tree $\mathcal{T}_i$. We can show that each regular tree can be turned into a normal form:

6.4.3. L . *If $\mathcal{T}$ is an image-finite B-regular tree, then there exist alive trees $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and a $\mathcal{T}_f \in \mathfrak{F}_n$ for some $n \in \mathbb{N}$ such that: $\mathcal{T} \leftrightarrows \mathcal{T}_f[x_1 \backslash \mathcal{T}_1, \ldots, x_1 \backslash \mathcal{T}_n]$.*

P Given a B-regular tree $\mathcal{T}$, suppose there are n different subtrees modulo bisimulation (call them $\mathcal{T}_1, \ldots \mathcal{T}_n$). We now turn $\mathcal{T}$ into the normal form by *"rewriting"* the first n levels of $\mathcal{T}$, in the top-bottom fashion starting from the root: if we reach a node v such that $\mathcal{T}^v$ is bisimilar to some alive $\mathcal{T}_i$, then we turn v into x_i and discard all the nodes reachable from v. Since $\mathcal{T}$ is image-finite, we only need to rewrite finitely many nodes. We claim that if we reach some node at level n which has not been discarded in an earlier state of the rewriting, then this node is a leaf in the tree $\mathcal{T}$. Suppose not, then there is a child w of this node. Since $dep(\mathcal{T}, w) > n$, along the path from the root to n there must be a w' such that $\mathcal{T}^w \leftrightarrows \mathcal{T}^{w'}$, due to the fact that there are only n different subtrees modulo bisimulation. However, then $\mathcal{T}^{w'}$ is an alive tree and thus the nodes reachable from w' should be discarded. Contradiction.

Let $\mathcal{T}_f$ be the resulting tree. It is clear that $\mathcal{T} \leftrightarrows \mathcal{T}_f[x_1 \backslash \mathcal{T}_1, \ldots, x_1 \backslash \mathcal{T}_n]$.

⋈

Given $F_n \subseteq \mathfrak{F}_n$, we let $F_n[x_1 \backslash \mathcal{T}_1, \ldots, x_1 \backslash \mathcal{T}_n] = \{\mathcal{T}[x_1 \backslash \mathcal{T}_1, \ldots, x_1 \backslash \mathcal{T}_n] \mid \mathcal{T} \in F_n\}$. We can now show the normal form theorem as follows:

6.4.4. T . *Given a $\mu-$automaton A, if $\mathcal{L}(\mathsf{A})$ is countable up to bisimulation, then it can be represented by*

$$F_n[x_1 \backslash \mathcal{T}_1, \ldots, x_1 \backslash \mathcal{T}_n]$$

for some $n < \omega$, $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\} \subseteq alive(\mathcal{L}(\mathsf{A}))$, and some $F_n \subseteq \mathfrak{F}_n$ which is recognizable by an finite automaton B on finite trees in $\mathfrak{F}_n$.

P (Sketch) From Lemma 6.4.2, we can list the finitely many different representatives of the alive trees in $\mathcal{L}(\mathsf{A})$ as $\mathcal{T}_1, \ldots, \mathcal{T}_n$ for some n. We now build the finite automaton B on finite unranked trees based on A. Let $\mathsf{B} = (Q_{\mathsf{A}}, B_{\mathsf{A}} \cup \{x_1, \ldots, x_n\}, q_0, \rightarrow_{\mathtt{OR}}{}^{\mathsf{A}} \cup \rightarrow'_{\mathtt{OR}}, \rightarrow_{\mathtt{BR}}{}^{\mathsf{A}}, L')$ where:

$$q \rightarrow'_{\mathtt{OR}} x_i \text{ iff there is an accepting } q-\text{run of } \mathsf{A} \text{ on } \mathcal{T}_i$$

and

$$L'(b) = \begin{cases} L_{\mathsf{A}}(b) \text{ if } b \in B_{\mathsf{A}} \\ x_i \text{ if } b = x_i \end{cases}$$

We say a finite tree $\mathcal{T}_f$ with variables is accepted by B, if there is a legal run of B as a μ-automaton on $\mathcal{T}_f$. Let $\mathcal{L}_f(\mathsf{B})$ be the set of trees in $\mathfrak{F}_n$ that are accepted by B. It is not hard to verify that $\mathcal{L}(\mathsf{A})$ is equivalent (up to bisimulation) to $\mathcal{L}(\mathsf{B})[x_1 \backslash \mathcal{T}_1, \ldots, x_1 \backslash \mathcal{T}_n]$. ⋈

6.5 Discussion and Future Work

This chapter extends a result by Niwiński [Niw91] on the cardinality of tree languages recognized by automata. We showed that a $\mu-$automata recognizable set of image-finite models modulo bisimulation is uncountable iff it is of the cardinality continuum iff it contains a non-B-regular tree. As in [Niw91], we give a normal form of the countable languages modulo bisimulation.

A straightforward consequence of Theorem 6.3.5 is that a $\mathtt{Mu}$-formula has countably many image-finite bisimulation contracted models iff all these models are finite. Another interesting consequence in the case of countable languages is implied by Lemma 6.4.2: there are only finitely many non-bisimilar $S5$ or $KD45$ models of a $\mathtt{Mu}$ formula, if it has only finite bisimulation contracted models. To see this, first note that the $S5$ models are reflexive, thus their unravellings are alive trees themselves, therefore there are only finitely many of them due to Lemma 6.4.2. For the case of $KD45$ models, observe that the only state that is not reflexive for any labelled relations in a connected $KD45$ pointed model can only be the designated state. Therefore, any $KD45$ model of the given formula can be generated (modulo bisimulation) by linking an start state to some of the alive trees. If the set of labels and the set of basic propositions are finite, then we only have finitely many such $KD45$ models.

However, it should be noted that our main theorem does not imply the following: a $\mu-$automata recognizable set of image-finite $S5$ models modulo bisimulation is uncountable iff it is of the cardinality continuum iff it contains a non-B-regular tree. Our pumping construction in the proof of Theorem 6.3.5 does not preserve $S5$ conditions, i.e., we may pump a tree which is not bisimilar to any $S5$ models from an unravelling of a $S5$ model. Interested readers may try to verify that transitivity is not preserved by our pumping construction in this sense. This calls for a closer look at finer classes of models which may require more sophisticated pumping constructions. In fact, the proof of Lemma 6.4.2 already suggests that a pumping construction which *adds* trees instead of *substituting* trees may also work to prove the main theorem in a way that may preserve the properties. We leave this for future work.

On the other hand, we may also look at finer classes of automata (or, say, classes of modal μ-calculus formulas) to see whether we can have a better understanding for specific fragments of $\mathtt{Mu}$. An interesting question is to characterize the maximal fragment of $\mathtt{Mu}$ in which each formula has a unique model up to bisimulation.

Note that in the proof of our Lemma 6.4.2, we did not give a bound on the number of alive trees as in [Niw91]. It is not yet clear how we can tighten the proof in order to obtain a finite bound. The difficulty actually lies in the proof of our main theorem,

where using infinitely many non-bisimilar trees is essential, while in [Niw91] two *"different"* trees are enough. This may be an obstacle to an algorithm, as an analogy to the algorithm given in [Niw91], to output the number of non-bisimilar models of a given modal μ-calculus formula. We also leave this for future work.

Part III

Model Checking

Chapter 7

Making Models Smaller

7.1 Introduction

In this chapter, we import the 3-valued abstraction-refinement techniques developed for temporal logics to DEL model checking (see, e.g., [BG99, GHJ01]), with new features particularly relevant for a multi-agent epistemic setting. The abstraction-refinement method intuitively relates a detailed model (refined model) with a coarser one (abstract model) in which some information may be lost, but the information kept is faithful to the detailed model. In the Kripke models of the epistemic setting, there are often transitions with different labels that might be similar to each other, for instance, if they express uncertainties of agents playing similar roles in a multi-agent system. Another specific characteristic of epistemic Kripke models is that in modelling practical situations numerous different basic propositions might be used as we have seen in the Russian Cards or Muddy Children examples in the previous chapters. We may expect to lump together some of those transitions with different labels or combine states with different propositional valuations to obtain a more compact abstraction. However, the traditional abstraction techniques do not perform these types of reductions, therefore an adaptation is needed. Moreover, to apply the abstraction on DEL, it is a challenge to design a reasonable 3-valued semantics of DEL which facilitates faithful reasoning on abstract models.

Specifically, in this chapter, we extend the abstraction-refinement theory for *Kripke Modal Labelled Transition Systems* (KMLTSs) [HJS01], incorporating not only state mapping but also label and proposition lumping, in order to obtain compact but informative abstractions. We develop a 3-valued Public Announcement Logic (PAL) and prove that the abstraction relation on *static* models *can* assure us to safely verify any *dynamic* properties in terms of PAL-formulas on the abstractions of a KMLTS. Thus the theory can be used to abstract Kripke models, since Kripke models can be regarded as a special case of KMLTSs. This theory is in particular applicable for an epistemic setting as the example of the Muddy Children shows. We shall also see that under certain conditions, the components as in Chapter 5 can be viewed as abstractions of the composed model.

Related work In the flourishing field of abstraction techniques, to the best of our knowledge, no work on the abstraction of Kripke models exists yet with reducing both the number of labels and of basic propositions. The literature related most closely to the current chapter is the work on abstraction of LTSs [vdPE04] in which the labels could be grouped. In the related field of Epistemic Temporal Logic, although the computational complexity of ETL model checking has been well-addressed in the literature (see e.g., [vdMS99, SG02, vdMS04] and the survey on page 157 of this thesis), the state space reduction techniques, such as symmetry reductions and abstractions, have not been used until recently [CDLR09, CLDQ09]. The multi-valued semantics of model logic has been discussed in [Fit91, Fit92] in a very general setting while we focus on the 3-valued semantics of PAL based on KMLTSs.

Structure of the chapter Section 7.2 introduces Kripke Modal Labelled Transition Systems, together with a 3-valued interpretation of PAL. In Section 7.3, the notions of refinement and abstraction are introduced and the preservation results are proven. Section 7.4 contains two examples of applying abstraction to some real epistemic models. We conclude in Section 7.5.

7.2 Preliminaries

In this section we introduce the 3-valued Public Announcement Logic interpreted on 3-valued Kripke Modal Labelled Transition Systems.

7.2.1 Kripke Modal Labelled Transition System

In order to define abstractions of Kripke models the standard definition is extended in the following sense:

- To incorporate the approximation of propositional information in the abstract model, we use 3-valued valuations instead of 2-valued ones. Besides *true* and *false*, atomic propositions can now have a third truth value $\uparrow$ which is intended to mean *unknown*.

- To incorporate the approximation of relations, two types of relations *must* and *may* are introduced as in *Modal Transition Systems* [LT88][1], where *must*-relations are under-approximations (the relations are necessarily there in the concrete model) and *may*-relations are over-approximations (there are possibly such relations). Since necessarily existing relations should be at least possible, we require that the *must*-relations are included in the *may*-relations. Essentially, *may*- and *must*-relations together also assign *"truth values"* to the relations in the model: a relation from s to s' is "true" if there is a *must*-relation between s and s', it is "false" if there is no *may*-relation between the states, and it is "unknown"

[1] See also [AHL+08] for a survey on such systems.

when there is a *may*-relation between s and s' without a corresponding *must*-relation.

Formally, similar to the definition of Kripke Modal Transition Systems in [HJS01, GJ02], we have:

7.2.1. Definition (Kripke Modal Labelled Transition System) A *Kripke Modal Labelled Transition System* (KMLTS) is a tuple $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \dashrightarrow, \rightarrow, V)$ where:

- $S, \mathbf{P}, \mathbf{\Sigma}$ are as usual;
- $\dashrightarrow$ is a set of transitions of the form $s \overset{i}{\dashrightarrow} s'$ where $i \in \mathbf{\Sigma}$;
- $\rightarrow$ is a set of transitions of the form $s \overset{i}{\rightarrow} s'$ where $i \in \mathbf{\Sigma}$;
- V is a valuation function: $V : S \rightarrow \{true, false, \uparrow\}^{\mathbf{P}}$.

We require that $\rightarrow \subseteq \dashrightarrow$. We call $(\mathbf{P}, \mathbf{\Sigma})$ the signature of $\mathcal{M}$. A *pointed KMLTS* $(\mathcal{M}, s)$ is a pair of a KMLTS $\mathcal{M}$ and a distinguished state s in it. ℜ

We include the signature $(\mathbf{P}, \mathbf{\Sigma})$ in the specification of the models as, in general, the signatures of a model and its abstractions will be different.

A standard Kripke model can be regarded as a special kind of KMLTS, where *must* and *may* coincide and the valuation is essentially 2-valued:

7.2.2. Definition (Concrete model) A KMLTS $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \dashrightarrow, \rightarrow, V)$ is a concrete model if:

- $\dashrightarrow = \rightarrow$;
- for all $s \in S$, all $p \in P : V(s)(p) \neq \uparrow$.

ℜ

7.2.2 Three-valued Public Announcement Logic

Public Announcement Logic (PAL), initiated in [Pla89, GG97], is a convenient language for describing announcements and their informational consequences for (a group of) agents. Based on the standard language of epistemic logic (logic of knowledge), a new modality $[!\phi]$ is introduced into the language, with $[!\phi]\psi$ intended to express "*if ϕ is true then after the announcement of ϕ, ψ is true.*" (see page 16 of this thesis). Various case studies show this logic to be powerful in helping to understand complicated higher order reasoning about knowledge and announcements such as in the cases of Muddy Children, Sum and Product and the protocol of Dining Cryptographers.[2]

[2] we refer interested readers to [vDvdHK07] for detailed explanations

Formally, given a signature $(\mathbf{P}, \mathbf{\Sigma})$, the formulas of the *Public Announcement Logic* $\mathtt{PAL}_{\mathbf{\Sigma},\mathbf{P}}$ are defined by

$$\phi \quad ::= \quad p \mid \phi \wedge \psi \mid \neg\phi \mid \Box_i\phi \mid [!\phi]\phi$$

where $p \in \mathbf{P}$, $i \in \mathbf{\Sigma}$. As usual, we define $\phi \vee \psi$, $\phi \to \psi$ and $\Diamond_i\phi$ as abbreviations of $\neg(\neg\phi \wedge \neg\psi)$, $\neg\phi \vee \psi$ and $\neg\Box_i\neg\phi$ respectively.

As we will see in the next section, our overall approach is not constrained to be used only in epistemic settings, as it does not require the model to be *S*5.[3] Not constrained within *S*5 models, we have more freedom to find suitable abstractions, as we will see in the Muddy Children example.

The semantics for 2-valued public announcement logic is the extension of standard modal logic with relativization operators $[!\phi]$: $\mathcal{M}, s \vDash [!\phi]\psi \iff [\mathcal{M}, s \vDash \phi$ implies $\mathcal{M}|_\phi, s \vDash \psi]$, where the relativized model $\mathcal{M}|_\phi$ is the restriction of $\mathcal{M}$ to the states where ϕ holds. We extend such relativization, which we call "update" in the context of PAL, to the 3-valued case and take the usual semantics for $\Box$ as in the logics on Modal Transition Systems:

7.2.3. D . **(3-valued Semantics of PAL)** The truth value of a $\mathtt{PAL}_{\mathbf{\Sigma},\mathbf{P}}$ formula ϕ in a state s of a KMLTS $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \dashrightarrow, \to, V)$, written $[\![\phi]\!]^{\mathcal{M},s}$, is defined by:

$$\begin{array}{rcl}
[\![p]\!]^{\mathcal{M},s} & = & V(s)(p) \\
[\![\neg\phi]\!]^{\mathcal{M},s} & = & \neg_3[\![\phi]\!]^{\mathcal{M},s} \\
[\![\phi \wedge \psi]\!]^{\mathcal{M},s} & = & [\![\phi]\!]^{\mathcal{M},s} \wedge_3 [\![\psi]\!]^{\mathcal{M},s} \\
[\![\Box_i\phi]\!]^{\mathcal{M},s} & = & \begin{cases} \mathit{true} & \text{if } \forall s' : s \overset{i}{\dashrightarrow} s' \implies [\![\phi]\!]^{\mathcal{M},s'} = \mathit{true} \\ \mathit{false} & \text{if } \exists s' : s \overset{i}{\to} s' \text{ and } [\![\phi]\!]^{\mathcal{M},s'} = \mathit{false} \\ \uparrow & \text{otherwise} \end{cases} \\
[\![[!\phi]\psi]\!]^{\mathcal{M},s} & = & \begin{cases} \mathit{true} & \text{if } [\![\phi]\!]^{\mathcal{M},s} = \mathit{false} \text{ or } [\![\psi]\!]^{\mathcal{M}|_\phi,s} = \mathit{true} \\ \mathit{false} & \text{if } [\![\phi]\!]^{\mathcal{M},s} = \mathit{true} \text{ and } [\![\psi]\!]^{\mathcal{M}|_\phi,s} = \mathit{false} \\ \uparrow & \text{otherwise} \end{cases}
\end{array}$$

where:

- $\neg_3(\mathit{true}) = \mathit{false}, \neg_3(\mathit{false}) = \mathit{true}$ and $\neg_3(\uparrow) = \uparrow$, and for any $x, y \in \{\mathit{true}, \mathit{false}, \uparrow\}$: $x \wedge_3 y = \mathit{min}(x, y)$ w.r.t. $\leq_v$: $\mathit{false} \leq_v \uparrow \leq_v \mathit{true}$.
- $\mathcal{M}|_\phi = (\mathbf{\Sigma}, \mathbf{P}, S' \dashrightarrow', \to', V')$ is defined as follows:
  - $S' = \{s \in S \mid [\![\phi]\!]^{\mathcal{M},s} \neq \mathit{false}\}$;
  - $\dashrightarrow' = \dashrightarrow \cap (S' \times \mathbf{\Sigma} \times S')$;
  - $\to' = \to \cap (S' \times \mathbf{\Sigma} \times \{s \in S' \mid [\![\phi]\!]^{\mathcal{M},s} = \mathit{true}\})$;
  - $V'(s) = V(s)$ for $s \in S'$.

[3] *S*5 is a set of formulas axiomatizing the reading of $\Box$ as knowledge. *S*5 characterizes models in which the relations are equivalence relations.

Note that the above 3-valued semantics for the propositional fragment of PAL is essentially Kleene's *strong* 3-valued logic [Kle50] which is the strongest 3-valued propositional logic satisfying the following property:
[monotonicity] the behaviour of $\uparrow$ is compatible with any increase in information, i.e. if the truth value of a basic proposition appearing in ϕ is changed from $\uparrow$ to *true* or *false* then the truth value of ϕ should not be inherently changed from *false* to *true* or from *true* to *false*.

From the perspective of abstraction, monotonicity guarantees that if we have a definite truth value (*true* or *false*) of a formula in the 3-valued valuation (abstract model) then this truth value should be the same w.r.t any 2-valued valuation (concrete model) obtained by turning $\uparrow$ values into either *true* or *false*. Thus we can *correctly* reason about the concrete model by looking at the abstract model[4].

The semantics of $\Box_i\phi$ is given as in [HJS01][5]. The intuitive idea behind the semantics of $\Box_i$ is that $\Box_i\phi$ is true if all the possible (*may*) *i*-relations lead to ϕ-true states, and is false if there exists a necessary (*must*) *i*-relation leading to a ϕ-false state. A moment of reflection should confirm that this semantics for modal formulas also complies with the above monotonicity in spirit: turning the *3rd* truth value of propositions and transitions into definite truth values in a model will not change the definite truth value of any modal formula.

The semantics of $[!\phi]\psi$ is given with the similar concern of monotonicity. The updated model $\mathcal{M}|_\phi$ defined above keeps all the ϕ-*not-false* states and all the relations among them, except for the *must*-relations that lead to ϕ-unknown states in $\mathcal{M}$. Recall that the *must*-relations signify *necessary* relations. However, a ϕ-unknown state s is not necessarily there in the updated model, as $\uparrow$ leaves the possibility open that ϕ could 'actually' be *false*. A relation directed at a possibly but not necessarily existent state, cannot be a necessary relation, so *must*-relations to ϕ-unknown states are removed.

Note that $\mathcal{M}|_\phi$ is still a KMLTS since $\rightarrow' \subseteq \dashrightarrow'$ by definition. It is not hard to check that this 3-valued semantics "coincides" with the standard 2-valued semantics on concrete models. Formally, for any $\mathtt{PAL}_{\Sigma,\mathtt{P}}$ formula ϕ, any concrete model $\mathcal{M}$:

$$[\![\phi]\!]^{\mathcal{M},s} = \mathit{true} \iff \mathcal{M}', s \models \phi \qquad [\![\phi]\!]^{\mathcal{M},s} = \mathit{false} \iff \mathcal{M}', s \not\models \phi$$

where $\mathcal{M}'$ is the standard Kripke model converted from $\mathcal{M}$ by lumping *may* and *must* relations together and $\models$ is the satisfaction relation for the standard 2-valued semantics of PAL.

[4] Although Kleene's strong 3-valued logic is the *strongest* one satisfying monotonicity, it does *not* mean we can get all the possible definite truth values w.r.t. the concrete model by looking at the abstract ones, e.g., the truth value of $p \vee \neg p$ is $\uparrow$ if the truth value of p is $\uparrow$, although $p \vee \neg p$ is valid under 2-valued valuation.

[5] [HJS01] presented the 3-valued semantics in an equivalent form by assigning each formula a pair of sets of states: $[\![\phi]\!]^{nec}$ (the states where ϕ is *necessarily* true) and $[\![\phi]\!]^{pos}$ (the states where ϕ is *possibly* true). See [GJ03] for discussions on these two forms.

For 2-valued PAL the following reduction axioms hold:

(At)	$[!\phi]p$	$\leftrightarrow$	$\phi \rightarrow p$
(PF)	$[!\phi]\neg\psi$	$\leftrightarrow$	$\phi \rightarrow \neg[!\phi]\psi$
(Dist)	$[!\phi](\psi_1 \wedge \psi_2)$	$\leftrightarrow$	$[!\phi]\psi_1 \wedge [!\phi]\psi_2$
(Seq)	$[!\phi][!\psi]\chi$	$\leftrightarrow$	$[!\phi \wedge [!\phi]\psi]\chi$
(KA)	$[!\phi]\Box_i\psi$	$\leftrightarrow$	$\phi \rightarrow \Box_i[!\phi]\psi$

A natural question to ask is, in the above axioms, whether the formula at the left hand side of $\leftrightarrow$ always has the same truth value as the right hand side formula given an arbitrary KMLTS? The answer is "No." For example, consider the axiom PF and an KMLTS $\mathcal{M}_1$ with a single state s where p is $\uparrow$ and q is *false*, then $[\![[!p]\neg q]\!]^{\mathcal{M}_1,s} = true$ but $[\![p \rightarrow \neg[!p]q]\!]^{\mathcal{M}_1,s} = \uparrow$.

Moreover, we can show that any other reasonable 3-valued semantics of PAL can not reduce the left hand side of $\leftrightarrow$ to the right hand side. First note that an *informative* 3-valued semantics should indeed make $[\![[!p]\neg q]\!]^{\mathcal{M}_1,s} = true$, since changing the valuation of p in $\mathcal{M}_1$ to *true* or *false* will only make $[!p]\neg q$ true. We call a 3-valued semantics of PAL on KMLTSs *reasonable* if:

1. it coincides with strong Kleene 3-valued logic on its propositional fragment;
2. it coincides with the standard 2-valued semantics of PAL on concrete KMLTSs;
3. it is monotonic with respect to basic propositions.

Now we can show:

7.2.4. P . *There is no reasonable 3-valued semantics for* PAL *such that* $[\![p \rightarrow \neg[p]q]\!]^{\mathcal{M}_1,s} = true$.

P Suppose towards a contradiction that $[\![p \rightarrow \neg[!p]q]\!]^{\mathcal{M}_1,s} = true$ w.r.t to a reasonable 3-valued semantics of PAL. Note that $[\![p]\!]^{\mathcal{M}_1,s} = \uparrow$, then according to the strong Kleene semantics, we have $[\![\neg[!p]q]\!]^{\mathcal{M}_1,s} = true$, thus $[\![[!p]q]\!]^{\mathcal{M}_1,s} = false$. However if we change the valuation of p in $\mathcal{M}_1$ to *false* then $[!p]q$ would be *true* according to the standard 2-valued semantics of PAL, which contradicts monotonicity. ⋈

The above result also shows that we can not translate the 3-valued PAL back to its modal logic fragment by just applying the reduction axioms of the 2-valued PAL.

Although our concern in this chapter is primarily to develop the theory of epistemic abstractions, the ultimate goal is to enable automatic verification of large epistemic models. Designing efficient algorithms for checking the satisfaction of 3-valued PAL formulae on KMLTSs, based on the definition above, is an interesting topic in itself and we leave it as further work. We now only note that, looking at similar results in the literature [BG04], we expect that such a model checking algorithm will not be more complex than the ones for checking (2-valued) PAL on Kripke models.

7.3 Abstraction and Logical Characterization

In this section we extend the classic definition of abstraction with label and proposition mappings in order to reduce the number of labels and possibly achieve smaller abstraction models. We show that we can reason about properties of the more refined model by model checking the more abstract model.

7.3.1 Abstraction

As observed in [vdPE04], to do model checking on infinitely-labelled systems, one needs abstraction to obtain a model with a reduced finite number of labels. We aim for an abstraction method that reduces the labels also in the finite case, by lumping similar transitions with different labels together into a unified one. This is often applicable in the epistemic case, as several agents may play a similar role and therefore have similar uncertainties. On the other hand, different propositions may also have a similar role on different states, in which case abstractions may combine propositions together as well. In the following, we use two mappings between signatures to capture the above intuitions of lumping labels and propositions. It is important to note that these abstractions produce models with a different signature than the original one.

Notation For a function h and x in its range, we use $h^{-1}[x]$ to denote the pre-image of x.

7.3.1. D . (Abstraction and Refinement) Let $\mathcal{M} = (S, \mathbf{\Sigma}, \mathbf{P}, \dashrightarrow, \rightarrow, V)$ and $\mathcal{N} = (T, \mathbf{\Sigma}', \mathbf{P}', \dashrightarrow', \rightarrow', V')$ be two KMLTSs. Given two surjective functions $f : \mathbf{\Sigma}' \to \mathbf{\Sigma}$ and $g : \mathbf{P}' \to \mathbf{P}$, a binary relation $R \subseteq T \times S$ is called an f, g-abstraction relation between $\mathcal{N}$ and $\mathcal{M}$, if for all $t \in T, s \in S$ with $(t, s) \in R$ the following hold:

- for any $p \in \mathbf{P} : V(s)(p) \neq \uparrow$ implies $\forall p' \in g^{-1}[p] : V'(t)(p') = V(s)(p)$;
- $t \overset{i'}{\dashrightarrow}' t'$ implies $\exists s' \in S$: $s \overset{f(i')}{\dashrightarrow} s'$ and $R(t', s')$;
- $s \overset{i}{\rightarrow} s'$ implies $\forall i' \in f^{-1}[i] : \exists t' \in T$ such that $t \overset{i'}{\rightarrow}' t'$ and $R(t', s')$.

We say $\mathcal{M}$ is an f, g-abstraction of $\mathcal{N}$ (notation: $\mathcal{N} \Subset_{f,g} \mathcal{M}$) if there exists an f, g-abstraction relation R between $\mathcal{N}$ and $\mathcal{M}$. We say $(\mathcal{M}, s)$ is an f, g-abstraction of $(\mathcal{N}, t)$ (notation: $(\mathcal{N}, t) \Subset_{f,g} (\mathcal{M}, s)$) if there exists an f, g-abstraction relation R between $\mathcal{N}$ and $\mathcal{M}$ such that $(t, s) \in R$.

Correspondingly, $(\mathcal{N}, t)$ is called an $f, g-$ *refinement* of $(\mathcal{M}, s)$ iff $(\mathcal{M}, s)$ is an f, g-abstraction of $(\mathcal{N}, t)$. ℜ

The first condition says that the valuation in the more abstract model can be less informative by making some propositions *unknown* (↑), but never unfaithful. The second condition requires that if an i'-may-transition in the more refined model then

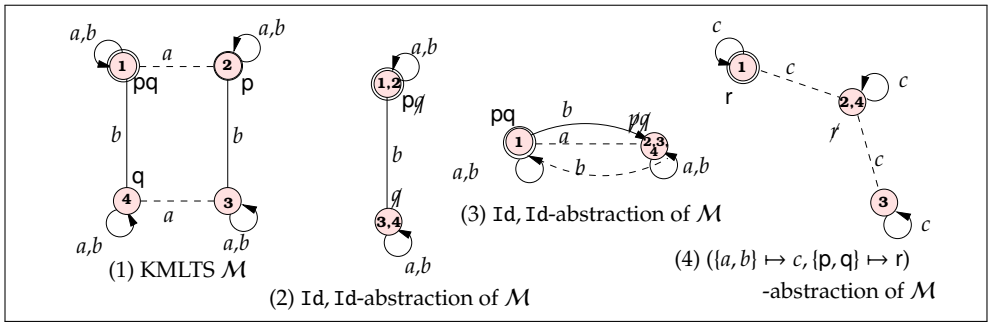


Figure 7.1: A pointed KMLTS and three possible abstractions of it. Dotted lines are for *may*-relations and solid lines for *must*. *May*-relations that coincide with corresponding *must* ones are omitted. If there is no arrow on a relation then it is bidirectional.

there is a matching transition in the more abstract model w.r.t the label mapping. The last condition says if there is an i-must-transition in the more abstract model then there is a corresponding i'-must-transition in the more refined model for *each* i' such that $f(i') = i$.

Note that for two 2-valued Kripke models with the same signature $(\mathbf{P}, \mathbf{\Sigma})$, $\mathcal{N}$ is a refinement of $\mathcal{M}$ in the classical sense of [Lar90] iff $\mathcal{N}$ is an $(Id_{\mathbf{\Sigma}}, Id_{\mathbf{P}})$−refinement of $\mathcal{M}$ where Id_X is identity function on the domain X.

Fig. 7.1 shows an example of a KMLTS $\mathcal{M}$ and several abstractions of it. In the picture, $\not{p}$ is to mean the value of p is *unknown* ($\uparrow$) at the current state while the absence of a proposition at a state means it is *false* there. For clarity, the states of $\mathcal{M}$ are numbered and the numbers on the states of the abstracted models indicate which original states they represent. In (2), the mappings are the identity functions, and the valuation of proposition q is mapped to $\uparrow$ for all worlds. In (3), the abstraction is given by the identity functions as well, but collapsing different worlds. In (4), there is an abstraction obtained by lumping both agents and both propositions.

Since $\rightarrow \subseteq \dashrightarrow$, we can make a concrete refinement of any KMLTS by dropping *may* relations that do not have a *must* counterpart (i.e. $\dashrightarrow'$, $\rightarrow' := \rightarrow$) and by adapting the valuation to become two-valued (e.g., by defining $V'(s)(p) = \mathit{false}$ whenever $V(s)(p) = \uparrow$ and $V'(s)(p) = V(s)(p)$ otherwise). Therefore:

7.3.2. P . *A KMLTS $\mathcal{M}$ always has a concrete refinement.*

7.3.2 Logical Characterization

We will prove a preservation result of satisfaction of formulas between a pointed model $(\mathcal{N}, t)$ and its abstraction $(\mathcal{M}, s)$. Intuitively we want a formula to be true/false at $\mathcal{N}$ if it is true/false at $\mathcal{M}$ respectively, such that we can safely model check the more abstract model to get the information of the more refined one. However, as

these models may have different signatures due to the f, g mappings attached to the abstraction relation, we need to check different formulas on these two models. Given two pointed models $(\mathcal{M}, s), (\mathcal{N}, t)$, and two formulas ϕ, ψ, we say $[\![\psi]\!]^{\mathcal{M},s} \leq [\![\phi]\!]^{\mathcal{N},t}$ if the following hold:

1. $[\![\psi]\!]^{\mathcal{M},s} = \mathit{true} \implies [\![\phi]\!]^{\mathcal{N},t} = \mathit{true}$;
2. $[\![\psi]\!]^{\mathcal{M},s} = \mathit{false} \implies [\![\phi]\!]^{\mathcal{N},t} = \mathit{false}$.

Then our goal is to check whether $(\mathcal{N}, t) \Subset_{f,g} (\mathcal{M}, s)$ implies for all ϕ: $[\![\ulcorner\phi\urcorner]\!]^{\mathcal{M},s} \leq [\![\phi]\!]^{\mathcal{N},t}$ where $\ulcorner\phi\urcorner$ is a formula in the signature of $\mathcal{M}$ corresponding to ϕ. To pinpoint the right formulas to check, we introduce the following translation:

7.3.3. D . **(Translation of formulas)** Given signatures $(\mathbf{P}', \mathbf{\Sigma}'), (\mathbf{P}, \mathbf{\Sigma})$, and surjective functions $f : \mathbf{\Sigma}' \to \mathbf{\Sigma}, g : \mathbf{P}' \to \mathbf{P}$, we define the translation of an $\mathtt{PAL}_{\mathbf{P}',\mathbf{\Sigma}'}$-formula ϕ into an $\mathtt{PAL}_{\mathbf{P},\mathbf{\Sigma}}$-formula $\ulcorner\phi\urcorner_{f,g}$ inductively as follows:

$$\begin{array}{lcl}
\ulcorner p'\urcorner_{f,g} & = & g(p') \\
\ulcorner\neg\psi\urcorner_{f,g} & = & \neg\ulcorner\psi\urcorner_{f,g} \\
\ulcorner\psi_1 \wedge \psi_2\urcorner_{f,g} & = & \ulcorner\psi_1\urcorner_{f,g} \wedge \ulcorner\psi_2\urcorner_{f,g} \\
\ulcorner\Box_{i'}\psi\urcorner_{f,g} & = & \Box_{f(i')}\ulcorner\psi\urcorner_{f,g} \\
\ulcorner[!\chi]\psi\urcorner_{f,g} & = & [!\ulcorner\chi\urcorner_{f,g}]\ulcorner\psi\urcorner_{f,g}
\end{array}$$

$\mathfrak{m}$

Before proving the main result of this chapter, we first prove a result establishing the abstraction relation between the updated models $(\mathcal{N}|_\chi, t)$ and $(\mathcal{M}|_{\ulcorner\chi\urcorner_{f,g}}, s)$ for some $\mathcal{L}_{\mathbf{P},\mathbf{\Sigma}}$-formula χ, given that $(\mathcal{N}, t) \Subset_{f,g} (\mathcal{M}, s)$

7.3.4. L . *Suppose $(\mathcal{N}, t), (\mathcal{M}, s)$ are pointed KMLTSs with signatures $(\mathbf{P}', \mathbf{\Sigma}')$ and $(\mathbf{P}, \mathbf{\Sigma})$ and sets of states T and S respectively, such that $(\mathcal{N}, t) \Subset_{f,g} (\mathcal{M}, s)$. Then for any $\mathtt{PAL}_{\mathbf{\Sigma}',\mathbf{P}'}$ formula χ such that t is in $\mathcal{N}|\chi$ and s is in $\mathcal{M}|_{\ulcorner\chi\urcorner_{f,g}}$, the following (1) implies (2):*

1. *for each $t' \in T, s' \in S : (\mathcal{N}, t') \Subset_{f,g} (\mathcal{M}, s') \implies [\![\ulcorner\chi\urcorner_{f,g}]\!]^{\mathcal{M},s'} \leq [\![\chi]\!]^{\mathcal{N},t'}$* ($\star$)
2. *$(\mathcal{N}|_\chi, t) \Subset_{f,g} (\mathcal{M}|_{\ulcorner\chi\urcorner_{f,g}}, s)$*

P Suppose $(\mathcal{N}, t) \Subset_{f,g} (\mathcal{M}, s)$ then there is a relation R which constitutes an f, g-refinement between $\mathcal{N}$ and $\mathcal{M}$ with $(t, s) \in R$. Now given a $\mathtt{PAL}_{\mathbf{\Sigma}',\mathbf{P}'}$ formula χ such that t is in $\mathcal{N}|\chi$ and s is in $\mathcal{M}|_{\ulcorner\chi\urcorner_{f,g}}$, let $T|_\chi$ and $S|_{\ulcorner\chi\urcorner_{f,g}}$ be the sets of states of $\mathcal{N}|_\chi$ and $\mathcal{M}|_{\ulcorner\chi\urcorner_{f,g}}$. We claim that $R' = R \cap (T|_\chi \times S|_{\ulcorner\chi\urcorner_{f,g}})$ is an f, g-abstraction relation between $\mathcal{N}|_\chi$ and $\mathcal{M}|_{\ulcorner\chi\urcorner_{f,g}}$. Note that $(t, s) \in R'$ since $t \in \mathcal{N}|_\chi$ and $s \in \mathcal{M}|_{\ulcorner\chi\urcorner_{f,g}}$. Now we check the three conditions of the abstraction relation:

- for the condition on p: follows from the first item in the definition of R and the fact that the valuation of an updated model is just the restriction of the original valuation to the remaining states.

- suppose $t \overset{i'}{\dashrightarrow} t'$ in $\mathcal{N}|_\chi$, then $t \overset{i'}{\dashrightarrow} t'$ in $\mathcal{N}$ according to the definition of the update. Since $(t,s) \in R$ and R is an abstraction relation, there exists $s' \in \mathcal{M}$: $s \overset{f(i')}{\dashrightarrow} s'$ and $(t',s') \in R$. We must still show that $s' \in \mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}}$. Suppose not, then $[\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s'} = \mathit{false}$. Because $(t',s') \in R$ ensures $(\mathcal{N},t') \Subset_{f,g} (\mathcal{M},s')$, it then follows from $(\star)$ that $[\![\chi]\!]^{\mathcal{N},t'} = \mathit{false}$. But then $t' \notin \mathcal{N}|_\chi$, contradiction.

- suppose $s \overset{i}{\rightarrow} s'$ in $\mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}}$, then $[\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s'} = \mathit{true}$ and $s \overset{i'}{\rightarrow} s'$ in $\mathcal{M}$. Because R is an f,g-abstraction relation and $(t,s) \in R$, for any $i' \in f^{-1}[i]$ there exists $t' \in \mathcal{N}$ such that $t \overset{i'}{\rightarrow} t'$ and $(t',s') \in R$. To show that $(t',s') \in R'$ for such t', it remains to show that $t' \in \mathcal{N}|_\chi$. Since $[\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s'} = \mathit{true}$ and $(t',s') \in R$, it then follows from condition $(\star)$ that $[\![\chi]\!]^{\mathcal{N},t'} = \mathit{true}$. Hence, $t' \in \mathcal{N}|_\chi$.

⋓

Now come our main results (Theorem 7.3.5 and Theorem 7.3.7) based on the above lemma.

7.3.5. T . *Suppose $\mathcal{N},\mathcal{M}$ are two KMLTSs w.r.t. $\mathbf{I}',\mathbf{P}'$ and $\mathbf{I},\mathbf{P}$ respectively. s and t are two states in $\mathcal{M}$ and $\mathcal{N}$ respectively. Then:*

$$(\mathcal{N},t) \Subset_{f,g} (\mathcal{M},s) \textit{ implies that for all } \phi \in \mathtt{PAL}_{\mathbf{\Sigma}',\mathbf{P}'} : [\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} \leq [\![\phi]\!]^{\mathcal{N},t}.$$

P We prove the theorem by induction on the structure of ϕ :

- $\phi = p'$: trivial, follows from the first condition of the definition of $\Subset_{f,g}$.

- $\phi = \neg\psi$: suppose $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true}$ then according to the semantics $[\![\ulcorner \psi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$. Thus by induction hypothesis $[\![\psi]\!]^{\mathcal{N},t} = \mathit{false}$. Therefore $[\![\phi]\!]^{\mathcal{N},t} = \mathit{true}$. For the case $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$, similar.

- $\phi = \psi_1 \wedge \psi_2$:

  - suppose $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true}$ then by the semantics: $[\![\ulcorner \psi_1 \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true}$ and $[\![\ulcorner \psi_2 \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true}$. Thus by induction hypothesis $[\![\psi_1]\!]^{\mathcal{N},t} = \mathit{true}$ and $[\![\psi_2]\!]^{\mathcal{N},t} = \mathit{true}$. Therefore $[\![\phi]\!]^{\mathcal{N},t} = \mathit{true}$.
  - suppose $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$ then by the semantics either $[\![\ulcorner \psi_1 \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$ or $[\![\ulcorner \psi_2 \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$. Without loss of generality, suppose the latter. Thus by induction hypothesis $[\![\psi_2]\!]^{\mathcal{N},t} = \mathit{false}$. Therefore $[\![\phi]\!]^{\mathcal{N},t} = \mathit{false}$.

- $\phi = \Box_{i'}\psi$: then $\ulcorner \phi \urcorner_{f,g} = \Box_{f(i')} \ulcorner \psi \urcorner_{f,g}$.

  - suppose $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true}$ then according to the semantics for all s' with $s \overset{f(i')}{\dashrightarrow} s'$ we have $[\![\ulcorner \psi \urcorner_{f,g}]\!]^{\mathcal{M},s'} = \mathit{true}$. Suppose in $\mathcal{N}$ there is a world t' such

that $t \overset{i'}{\dashrightarrow} t'$ then according to the definition of refinement, there is a $s'' \in \mathcal{M}$ such that $s \overset{f(i')}{\dashrightarrow} s''$ and $(\mathcal{N}, t') \Subset_{f,g} (\mathcal{M}, s'')$. Thus $[\![\ulcorner \psi \urcorner_{f,g}]\!]^{\mathcal{M},s''} = \mathit{true}$. By induction hypothesis, $[\![\psi]\!]^{\mathcal{N},t'} = \mathit{true}$. Therefore $[\![\Box_{i'} \psi]\!]^{\mathcal{N},t} = \mathit{true}$.

- suppose $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$ then according to the semantics, there is s' with $s \overset{f(i')}{\rightarrow} s'$ such that $[\![\ulcorner \psi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$. By definition of refinement, for any $i'' \in f^{-1}[f(i')]$ there is a $t' \in \mathcal{N}$ such that $t \overset{i''}{\rightarrow} t'$ and $(\mathcal{N}, t') \Subset_{f,g} (\mathcal{M}, s')$. By induction hypothesis, for all such t' : $[\![\psi]\!]^{\mathcal{N},t'} = \mathit{false}$. Thus for all $i'' \in f^{-1}[f(i')] : [\![\Box_{i''} \psi]\!]^{\mathcal{N},t} = \mathit{false}$. In particular: $[\![\Box_{i'} \psi]\!]^{\mathcal{N},t} = \mathit{false}$.

- $\phi = [!\chi]\psi$

  - if $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true}$ then $[\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$ or $[\![\ulcorner \psi \urcorner_{f,g}]\!]^{\mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}},s} = \mathit{true}$. If $[\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$ then $[\![\chi]\!]^{\mathcal{N},t} = \mathit{false}$ by induction hypothesis, hence $[\![\phi]\!]^{\mathcal{N},t} = \mathit{true}$. Otherwise, $[\![\ulcorner \psi \urcorner_{f,g}]\!]^{\mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}},s} = \mathit{true}$ and $[\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s} \neq \mathit{false}$, then $s \in \mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}}$. Now suppose $[\![\chi]\!]^{\mathcal{N},t} \neq \mathit{false}$, so: $t \in \mathcal{N}|_\chi$. We need to show that $[\![\psi]\!]^{\mathcal{N}|_\chi,t} = \mathit{true}$. By induction hypothesis $(\mathcal{N}, t') \Subset_{f,g} (\mathcal{M}, s') \implies [\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s'} \leq [\![\chi]\!]^{\mathcal{N},t'}$ for each $s' \in S, t' \in T$. Therefore from Lemma 7.3.4 we have $(\mathcal{N}|_\chi, t) \Subset_{f,g} (\mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}}, s)$. By induction hypothesis, $[\![\psi]\!]^{\mathcal{N}|_\chi,t} = \mathit{true}$. Thus $[\![\phi]\!]^{\mathcal{N},t} = \mathit{true}$.

  - if $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false}$ then $[\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true}$ and $[\![\ulcorner \psi \urcorner_{f,g}]\!]^{\mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}},s} = \mathit{false}$. Since $[\![\ulcorner \chi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true}$ then $[\![\chi]\!]^{\mathcal{N},t} = \mathit{true}$ by induction hypothesis. We only need to show $[\![\psi]\!]^{\mathcal{N}|_\chi,s} = \mathit{false}$. It is clear that $t \in \mathcal{N}|_\chi$ and $s \in \mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}}$, then by induction hypothesis the condition of Lemma 7.3.4 holds, and it follows that $(\mathcal{N}|_\chi, t) \Subset_{f,g} (\mathcal{M}|_{\ulcorner \chi \urcorner_{f,g}}, s)$. Thus by the induction hypothesis we have $[\![\psi]\!]^{\mathcal{N}|_\chi,t} = \mathit{false}$. Therefore: $[\![\phi]\!]^{\mathcal{N},t} = \mathit{false}$.

⋈

7.3.6. C . *Suppose* $(\mathcal{N}, t), (\mathcal{M}, s)$ *are two pointed KMLTSs w.r.t.* $(\mathbf{I}', \mathbf{P}')$ *and* $(\mathbf{I}, \mathbf{P})$ *respectively. If* $(\mathcal{N}, t) \Subset_{f,g} (\mathcal{M}, s)$ *and* $\mathcal{N}$ *is a Kripke model converted from a concrete KMLTS then for any formula* $\phi \in \mathtt{PAL}_{\mathbf{\Sigma}', \mathbf{P}'}$:

- $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{true} \implies \mathcal{N}, t \vDash \phi$
- $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} = \mathit{false} \implies \mathcal{N}, t \vDash \neg\phi$

By the above corollary, to know whether ϕ is satisified at a pointed Kripke model, we can instead model check $\ulcorner \phi \urcorner_{f,g}$ on its $f, g-$abstraction.

To justify the logical characterization, we prove the converse of Theorem 7.3.5.

7.3.7. T . *Suppose* $(\mathcal{N}, t)$ *and* $(\mathcal{M}, s)$ *are pointed KMLTSs with signatures* $(\mathbf{P}', \mathbf{\Sigma}')$ *and* $(\mathbf{P}, \mathbf{\Sigma})$*, and suppose they enjoy* image finiteness *(see page 11). Then:*

$$\textit{If for every } \phi \in \mathtt{PAL}_{\mathbf{P}',\mathbf{\Sigma}'} : [\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} \leq [\![\phi]\!]^{\mathcal{N},t} \textit{ then } (\mathcal{N}, t) \Subset_{f,g} (\mathcal{M}, s).$$

P Assume that for every formula $\phi \in \mathtt{PAL}_{\mathbf{\Sigma}',\mathbf{P}'}$: $[\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s} \leq [\![\phi]\!]^{\mathcal{N},t}$, and let $R = \{(t', s') \mid \text{for every } \phi : [\![\ulcorner \phi \urcorner_{f,g}]\!]^{\mathcal{M},s'} \leq [\![\phi]\!]^{\mathcal{N},t'}\}$. Then $(t, s) \in R$, and we check the three conditions of definition 7.3.1 for R. Suppose $(t', s') \in R$, then:

- The first condition follows from $[\![\ulcorner p' \urcorner_{f,g}]\!]^{\mathcal{M},s'} \leq [\![p']\!]^{\mathcal{N},t'}$ for $p' \in \mathbf{P}'$.
- Suppose towards contradiction that $\exists t'' : t' \overset{i'}{\dashrightarrow} t''$ in $\mathcal{N}$ but for any $s'' \in S$: $s' \overset{f(i')}{\dashrightarrow} s''$ implies $(t'', s'') \notin R$. According to image finiteness, we have only finitely many such s'' (call them $s''_0 \ldots s''_n$). For each s''_k, since $(t'', s''_k) \notin R$, there must be a formula $\psi_{s''_k}$ such that $[\![\ulcorner \psi_{s''_k} \urcorner_{f,g}]\!]^{\mathcal{M},s''_k} = \textit{true}$ but $[\![\psi_{s''_k}]\!]^{\mathcal{N},t''} \neq \textit{true}$.[6] Now $\Box_{f(i')}(\bigvee_{k=0}^{n} \ulcorner \psi_{s''_k} \urcorner_{f,g})$ is *true* at s' but $\Box_{i'}(\bigvee_{k=0}^{n} \psi_{s''_k})$ is not *true* at t', contradicting the assumption that $(t', s') \in R$.
- Suppose towards contradiction that $s' \overset{f(i')}{\rightarrow} s''$ in $\mathcal{M}$, but there exists $i'' \in f^{-1}[f(i')]$ such that $\forall t'' \in T$: $t' \overset{i''}{\rightarrow} t''$ implies $(t'', s'') \notin R$. According to image finiteness, there are only finitely many such t''(call them $t''_0 \ldots t''_n$). For each t''_k, since $(t''_k, s'') \notin R$, there must be a formula $\psi_{t''_k}$ such that $[\![\ulcorner \psi_{t''_k} \urcorner_{f,g}]\!]^{\mathcal{M},s''} = \textit{false}$ but $[\![\psi_{t''_i}]\!]^{\mathcal{N},t''_i} \neq \textit{false}$. Note that $\Box_{f(i')}(\bigvee_{k=0}^{n} \ulcorner \psi_{t''_i} \urcorner_{f,g})$ is *false* at s' but $\Box_{i''}(\bigvee_{k=0}^{n} \psi_{t''_i})$ is not *false* at t', contradicting the assumption that $(t', s') \in R$.

⋈

7.4 The Muddy Children and Abstraction

Recall the discussions we had in Section 5.2 of Chapter 5: we can decompose the model for n-Muddy Children (see Example 1.1.2) into n two-world models (with disjoint vocabularies) $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_n$ where each $\mathcal{M}_i$ represents the children's observation about whether child i is dirty:

$$m_i \text{ —}i\text{— } \overline{m_i}$$

It is clear that we can view each $\mathcal{M}_i$ as a KMLTS with signatures $\mathbf{P} = \{m_1, m_2, \ldots, m_n\}$ and $\mathbf{\Sigma} = \{1, 2, \ldots, n\}$, where may- and must-relations coincide but propositions in $\mathbf{P} - \{m_i\}$ are assigned the third truth value $\uparrow$. It is not hard to see that each $\mathcal{M}_i$ is an (id, id)-abstraction of the composed model $\mathcal{M}$. Thus we can verify the properties

[6] If $[\![\ulcorner \psi_{s''_k} \urcorner_{f,g}]\!]^{\mathcal{M},s''} = \textit{false}$ but $[\![\psi_{s''_k}]\!]^{\mathcal{N},t''} \neq \textit{false}$ then $[\![\ulcorner \neg\psi_{s''_k} \urcorner_{f,g}]\!]^{\mathcal{M},s''} = \textit{true}$ but $[\![\neg\psi_{s''_k}]\!]^{\mathcal{N},t''} \neq \textit{true}$.

about the composed model by looking at its components. For example, let ϕ be the common knowledge formula $C(\neg K_i m_i \wedge \neg K_i \neg m_i)$, universally verifying ϕ against $\mathcal{M}$ (checking $\mathcal{M} \vDash \phi$) is then reduced to checking $\mathcal{M}_i \vDash \phi$, which is obviously true.

More generally, we can prove Theorem 5.2.13 in Chapter 5 as an easy application of our Theorem 7.3.5:

7.4.1. T . *If a pointed S5 model $(\mathcal{M}, s)$ is decomposable (w.r.t $\mathbb{D}$) into S5 models $(\mathcal{M}_0, s_0), (\mathcal{M}_1, s_1), \ldots, (\mathcal{M}_n, s_n)$ with disjoint vocabularies $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$, then for any epistemic formula ϕ based on $\mathbf{P}_i$: $\mathcal{M}_i, s_i \vDash \phi \iff \mathcal{M}, s \vDash \phi$.*

P Again we consider the models $\mathcal{M}_i$ and $\mathcal{M}$ as KMLTSs where may- and must-relations coincide as above. We let R be the relation linking a world s in $\mathcal{M}$ with a world t in $\mathcal{M}_i$ iff s is composed by t and other worlds from other models. We need to show that R is indeed an (id, id)-abstraction relation. The first and second conditions of Definition 7.3.1 are trivial, according to the definition of $\mathbb{D}$ in Section 5.2. Now suppose $t \xrightarrow{i} t'$ in $\mathcal{M}_i$ and sRt. Since sRt then we can assume that s is a tuple $\langle t_0, \ldots, t_i, \ldots, t_n \rangle$ where $t_i = t$ and each t_j is from the model $\mathcal{M}_j$. Since $\mathbf{P}_i$ are disjoint from each other, then there must be a state $s' = \langle t_0, \ldots, t', \ldots, t_n \rangle$ in $\mathcal{M}$ differing from s only in the ith place in the tuple. Since all the $\mathcal{M}_i$ are S5 models, $t_j \xrightarrow{i} t_j$ in $\mathcal{M}_j$ for $j \neq i$. Because $t_i \xrightarrow{i} t'$ then by the definition of the composed model, $s \xrightarrow{i} s'$ in $\mathcal{M}$. ⋈

Note that the above abstraction of the model of n-Muddy Children by decomposition with two-world models is somehow too coarse, since it does not reflect the dynamics of the story. For example, on a two-world abstraction of n-Muddy Children, the announcement of $m_1 \vee m_2 \vee m_3$ (one of you is muddy) simply does not change anything since the truth value of $m_1 \vee m_2 \vee m_3$ on these two-world abstractions is either *true* or $\uparrow$. In the sequel, let us consider more sophisticated abstractions of the model of Muddy Children which reflect the dynamics of announcements. We will focus on the 3-children case from now on.

The left column of Fig. 7.2 shows the standard epistemic model and its dynamics for 3-Muddy Children. The middle and right columns of Fig. 7.2 show abstracted versions of the concrete model on the left. The abstraction relation underlying both abstractions relates three pairs of worlds in the concrete model to three single worlds in the abstraction, while the world with all propositions *false* and the world with only m_3 *true* are kept (for example, the world with m_2 *true* and the world with m_2, m_3 *true* in the concrete model are related to the one world in the abstracted model where m_2 is *true* and m_3 *unknown*). In the middle column, the parameters f, g for the refinement are identities, in the right column f maps both 1 and 2 to abstract label A. Let ϕ_m be the abbreviation of the first announcement ($m_1 \vee m_2 \vee m_3$) and ϕ_K be the abbreviation of the next ones ($\neg\Box_1 m_1 \wedge \neg\Box_2 m_2 \wedge \neg\Box_3 m_3$). Notice the following significant properties can be verified to be *true* in the two abstractions: (1) In both abstractions, $\ulcorner[!\phi_m][!\phi_K][!\phi_K](\Box_1 m_1 \wedge \Box_2 m_2)\urcorner_{f,g}$ is *true* at the worlds that correspond to the world which makes m_1, m_2 and m_3 *true* in the original model. Thus by Theorem 7.3.5, $[!\phi_m][!\phi_K][!\phi_K](\Box_1 m_1 \wedge \Box_2 m_2)$ is *true* in that world in the

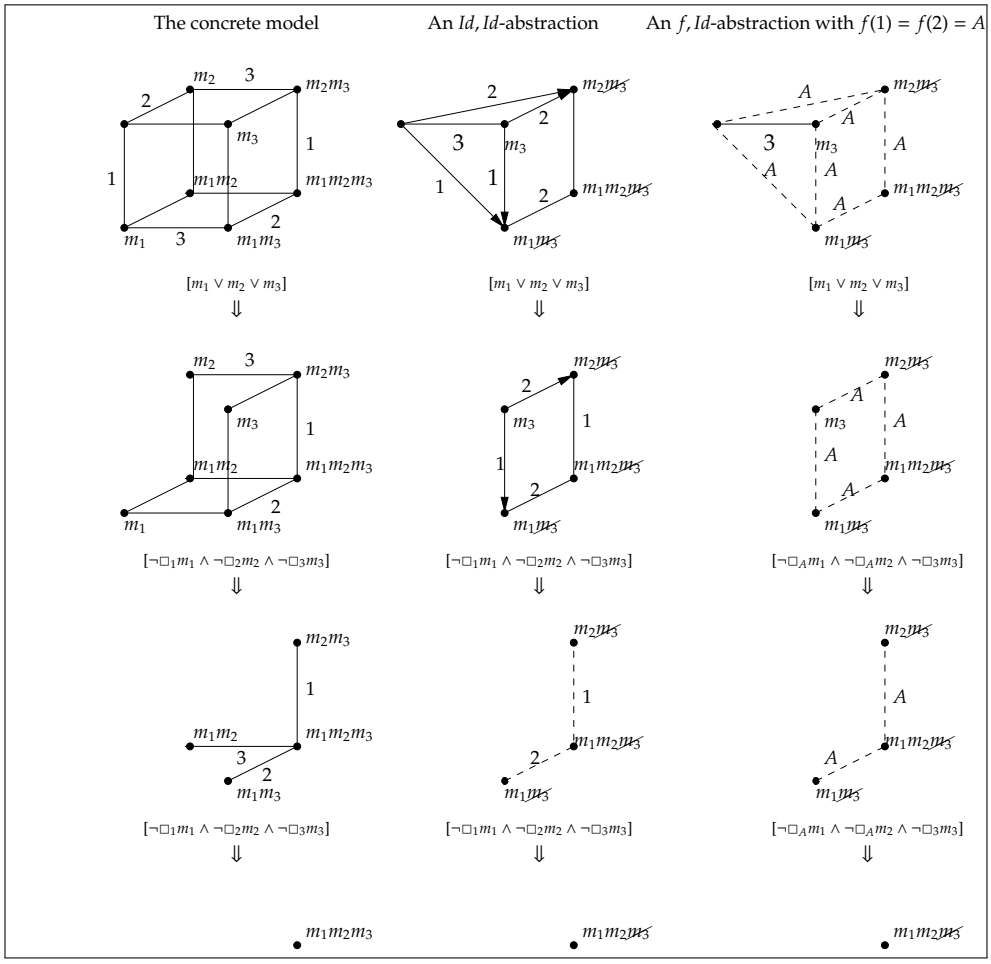


Figure 7.2: Abstractions of the Muddy Children for $n = 3$ children. Each world has reflexive *may*-relations for each $i \in \mathbf{\Sigma} = \{1, 2, 3\}$, some have reflexive *must*-relations, but for simplicity of presentation, all reflexive relations are omitted as usual.

original model. Namely, in the case all three children are muddy, children 1 and 2 will know they are muddy after three announcements. (2) In both abstractions, $\ulcorner[!\phi_m][!\phi_K]\Box_1 m_1\urcorner_{f,g}$ is *true* at the worlds that correspond to the original world where only m_1 and m_3 are *true*. Namely in the case children 1 and 3 are muddy, child 1 will know he is muddy after 2 updates. (3) $\ulcorner[!\phi_m]\Box_3 m_3\urcorner_{f,g}$ is *true* at the worlds with only m_3 *true*. Namely when child 3 is the only muddy child, he will know after the first announcement. For the generalization to the n children case, similar abstractions can be made.

Note that whereas all relations in the concrete model are equivalence relations (S5), this is no longer the case for the abstractions: in the middle abstraction, the *must* relations can be seen to be non-symmetric, and in the right abstraction, the relation labelled A is no longer transitive (in general the union of two equivalence relations is not necessarily transitive)[7].

7.5 Conclusion and Future work

We have developed an abstraction framework for KMLTSs, which allows us to verify properties that involves public announcements on smaller abstract models instead of on big concrete models. We demonstrate the use of our framework by looking at the example of Muddy Children. Another example of abstracting a model for encoded broadcast can be found in [DOW08].

The theoretical novelty of this chapter is the extension of traditional abstraction techniques to both the label and proposition mappings and to a logic containing dynamic modalities (*public announcements*) which change the models. Both features are of fundamental importance in (epistemic) modelling and verification, which is the main motivation of our work. In order to incorporate the full power of dynamic epistemic modelling, more research is needed on integrating general update constructions as formalized by action models [BM04]. The abstraction of action models is also useful, as it is shown in [DW07] that the action models can be quite large when modelling protocols. Another goal is to adapt this framework to Interpreted Systems [FHMV95], which combines both epistemic and temporal characteristics.

On a practical side, our framework opens a way to dynamic epistemic verification of large or even infinite models. Future research should be dedicated to practical problems like generating abstract models automatically from formal, but compact, model specifications [CGJ+03].

[7] From Theorem 7.3.5, the truth values of some S5 axioms are $\uparrow$ rather than *true* in the non-S5 abstractions of this example.

Chapter 8

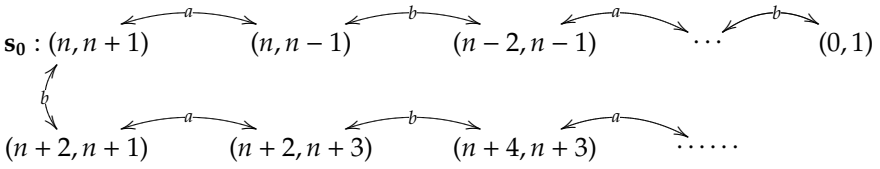
Accelerating the Transitions

8.1 Introduction

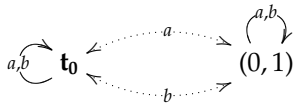
In this chapter we look at a particular technique of $\mathtt{PDL}_\Sigma$ abstraction. Since DEL can be translated back to PDL, the technique we will develop here can be adapted to DEL model checking. As demonstrated in the previous chapter, we can use three-valued logic to reason about properties using abstract models with *may*- and *must*-transitions. According to the 3-valued semantics of modal logic, universal (safety) properties $\Box\phi$ are checked w.r.t the over-approximation (may-transitions), while existential (liveness) properties $\Diamond\phi$ are checked w.r.t the under-approximation (must-transitions). This works fine for *safety* properties, but the verification of *liveness* properties is problematic. The problem comes from the lack of guaranteed ("must") behaviours, due to the non-determinism introduced by abstraction. Consider the following example:

8.1.1. E . **(Guessing the other number)** Agents a and b have the natural numbers n and $n+1$ respectively. They only know their own numbers. They are told that what they have are two consecutive natural numbers, but they do not know who has the bigger one. **Ω**

We can build the following model (suppose n is an even number):

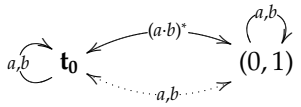


If we want to abstract away all the intermediate states except the *terminal* one $(0, 1)$ then we have the following model:



where t_0 is the abstraction of all the states in the original model except $(0,1)$. It is not hard to see that there cannot be any must relations in such an abstract model except the reflexive ones at $(0,1)$. Therefore we cannot get a definite answer to the model checking problem of the formula $\widehat{C}_{a,b}has_a0$ (in PDL: $\langle a+b\rangle^* has_a 0$) on the abstract model.

To deal with this problem, Espada and van de Pol proposed *accelerated modal LTS* (accModal-LTS), a new formalism to represent abstractions [EvdP06]. They enhance KMLTSs by labelling must-transitions with *regular expressions* over basic actions. These so-called *accelerated* transitions capture the idea that a state *must* be reached from another state by *some* finite computation contained in the language of the corresponding regular expression. This extension of the abstraction enables us to capture the concrete models more accurately and infer more *liveness* properties. Consider example 8.1.1 again. One could introduce an accelerated must-transition from t_0 to $(0,1)$ in the abstract model as follows:



Intuitively, this must-transition means, in any state abstracted by t_0 there is a finite $\xrightarrow{a}\xrightarrow{b}\xrightarrow{a}\xrightarrow{b}\ldots$ path to the state $(0,1)$. According to the 3-valued $\mathtt{PDL}_\Sigma$ semantics defined in [EvdP06], $\langle\pi\rangle\phi$ is true at a point s if there is a *must*-path $s \xrightarrow{\pi_1} s_1 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_n} s_n$ to a state where ϕ is true, such that $\mathcal{L}(\pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi)$. Thus we can verify that $\langle(a+b)^*\rangle has_a 0$ is indeed true on the above abstract model. Based on a suitable abstraction relation, [EvdP06] shows that we can safely reason about the properties of the concrete models by model checking the same properties on their abstractions.

In [EvdP06], the authors also gave a non-trivial model checking algorithm for 3-valued $\mathtt{PDL}_\Sigma$ on accModal-LTSs, showing decidability of the model checking problem. The most intricate part of the algorithm deals with finding must-paths that comply with the regular expression π in a diamond formula $\langle\pi\rangle\phi$ which is quite different from the usual $\mathtt{PDL}_\Sigma$ model checking algorithm (see, e.g., [Lan06]). A hard problem left open in [EvdP06] is the precise complexity and optimality of the algorithm.

To understand the behaviour of the accelerated transitions better, in this chapter, we consider $\mathtt{PDL}_\Sigma$ defined on 2-valued models with accelerated transitions only, which we call *Accelerated Kripke Models (AKM)*. Note that in our AKM, we have only one type of relation as in the standard Kripke models. Thus an AKM can be viewed as the *must* part of an accModal-LTS with 2-valued valuations on each state. The model checking algorithm of $\mathtt{PDL}_\Sigma$ on AKM can be easily adapted to the original three-valued setting as in [EvdP06].

Main contributions. As we will see, $\mathtt{PDL}_\Sigma$ interpreted over an AKM behaves quite differently from standard PDL. Developing a model checking algorithm is of utmost importance. Moreover, for an in-depth understanding of the logic, axiomatization and satisfiability checking are two central questions. We address all of these problems.

In Section 3, we first reduce model checking $\mathtt{PDL}_\Sigma$ on AKM to model checking PDL on standard Kripke models, by exploiting the notion of *regular expression rewriting* studied extensively in [CDGLV02]. We then provide an automata theoretical model checking algorithm whose complexity can be easily analyzed, namely, in E . Furthermore, we prove an E lower bound for the model checking problem. These results solve the open problem on model checking left in [EvdP06] and establish a strong link between model checking $\mathtt{PDL}_\Sigma$ over AKM and regular expression rewriting. In Section 4, we provide an axiomatization of $\mathtt{PDL}_\Sigma$ on AKM, which employs *Kleene Algebra* [Koz91] as an oracle. The soundness and completeness of this system are shown. This result shows very clearly the differences with standard $\mathtt{PDL}_\Sigma$ on Kripke models. Furthermore, in Section 5, we study the decision problem of satisfiability. For satisfiability, again, by resorting to the notion of regular expression rewriting, we reduce this problem to the satisfiability of $\mathtt{PDL}_\Sigma$ in the standard semantics over Kripke models. We show that the satisfiability of a $\mathtt{PDL}_\Sigma$ formula over AKM can be checked in 3-E .

Related work. Finite-state automata that allow more complex transition labels recently received a resurgence of attention. These include *generalized automata* [GM99] (a.k.a. string or lazy automata) with strings (or blocks) as transition labels rather than merely characters or the null string and *expression automata* [HW05], finite-state automata whose transition labels are regular expressions over the input alphabet. However, these have been studied from the automata and language perspectives. In particular, the determinism and minimization problems are explored there. In logic, [Mat03] studies μ-calculus with regular expressions in the modalities. It is shown that in this case, regular expressions in formulae can be easily eliminated by the fixpoint construction. [LS07] introduces the notion of regular linear temporal logic, which is a logic that generalizes linear temporal logic with the ability to use regular expressions arbitrarily as sub-expressions. The expressiveness and satisfiability of this logic are investigated there. These works are orthogonal to the use of regular expressions in LTSs, which is the main focus of this chapter. Another work which extends the transitions in LTS is [Lod95], in which the authors study $\mathtt{PDL}_\Sigma$ on *distributed transition systems* where the transition relations are labelled with a finite set of actions, representing the fact that the actions occur as a concurrent step. However, the semantics of $\mathtt{PDL}_\Sigma$ in [Lod95] is quite different from ours, due to the different interpretation of the non-standard transitions.

8.2 Preliminaries

8.2.1 PDL on AKM

8.2.1. D . **(Accelerated Kripke model)** An *Accelerated Kripke model* (AKM) is a tuple $\mathcal{M} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V)$ where:

- $S, \mathbf{P}, \mathbf{\Sigma}, V$ are as usual;

- $\rightarrow$ is a possibly infinite set of *accelerated* transitions of the form $s \xrightarrow{\pi} s'$ with $s, s' \in S$, and $\pi \in Reg_{\mathbf{\Sigma}}$ where recall that $Reg_{\mathbf{\Sigma}}$ is the set of (test-free) regular expressions over alphabet $\mathbf{\Sigma}$ (with $a \in \mathbf{\Sigma}$):

$$\pi ::= \mathbf{0} \mid \mathbf{1} \mid a \mid \pi + \pi \mid \pi \cdot \pi \mid \pi^*$$

Following the tradition in modal logic, we shall call $\mathcal{F} = (S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow)$ an AKM *frame*. As usual, a pointed AKM is an AKM with a designated state $s_0 \in S$: $(S, \mathbf{P}, \mathbf{\Sigma}, \rightarrow, V, s_0)$. 𝔐

In this chapter, we fix a vocabulary $\mathbf{P}$, thus we refer to an AKM as $(S, \mathbf{\Sigma}, \rightarrow, V)$.

Recall the *test-free* PDL language:

$$\phi ::= \top \mid p \mid \phi \wedge \phi \mid \neg\phi \mid \langle\pi\rangle\phi$$

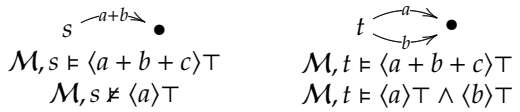
where π is a regular expression over some alphabet $\mathbf{\Sigma}$. When $\mathbf{\Sigma}$ is not fixed, we use $\mathtt{PDL}_{\mathbf{\Sigma}}$ to denote the test-free PDL language w.r.t the action set $\mathbf{\Sigma}$.

$\langle\pi\rangle\phi$ is intended to express that there *must* be an execution of π which entails ϕ. Recall that an accelerated transition $s \xrightarrow{\pi} t$ intuitively means that there must be an execution of π from s to t, however, we do not know which execution can do the job. Therefore, assuming $s \xrightarrow{\pi} t$, we can only be sure that there must be an execution of π' to a t world if $\mathcal{L}(\pi) \subseteq \mathcal{L}(\pi')$. The following semantics fleshes out this intention:

$\mathcal{M}, s \vDash p$	$\Leftrightarrow$	$p \in V(s)$
$\mathcal{M}, s \vDash \neg\phi$	$\Leftrightarrow$	$s \nvDash \phi$
$\mathcal{M}, s \vDash \phi \wedge \psi$	$\Leftrightarrow$	$\mathcal{M}, s \vDash \phi$ and $\mathcal{M}, s \vDash \psi$
$\mathcal{M}, s \vDash \langle\pi\rangle\phi$	$\Leftrightarrow$	there exists a path $s = s_0 \xrightarrow{\pi_1} s_1 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_n} s_n$ in $\mathcal{M}$ such that $\mathcal{M}, s_n \vDash \phi$ and $\mathcal{L}(\pi_0 \cdot \pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi)$

To illustrate the semantics, we present two simple examples:

8.2.2. E .



$\mathcal{M}, s \vDash \langle a + b + c\rangle\top$ $\quad$ $\mathcal{M}, t \vDash \langle a + b + c\rangle\top$

$\mathcal{M}, s \nvDash \langle a\rangle\top$ $\quad$ $\mathcal{M}, t \vDash \langle a\rangle\top \wedge \langle b\rangle\top$

Ω

It is clear that on Kripke models (AKM with only atomic actions as labels), the above semantics coincides with the standard $\mathtt{PDL}_{\mathbf{\Sigma}}$ semantics.

8.2.2 Regular Expression Rewriting

The notion of *regular expression rewriting* is introduced in [CDGLV02], and turns out to play an essential role in solving model checking and satisfiability checking problems in this chapter.

Given a regular expression π over an alphabet $\mathbf{\Sigma}$ and a finite set $\mathcal{E} = \{\pi_0, \pi_1, \ldots, \pi_n\}$ of regular expressions over the same alphabet $\mathbf{\Sigma}$, the goal of regular expression rewriting of π over $\mathcal{E}$ is to re-express π, if possible, by a suitable combination of $\pi_0, \ldots, \pi_n$ using operations $\cdot, +$, and *. Given $\mathcal{E}$, let $\mathbf{\Sigma}_\mathcal{E}$ be an alphabet containing exactly one unique symbol e_π for each π in $\mathcal{E}$. We shall use $exp_\mathbf{\Sigma}(e)$ to denote the regular expression associated with the symbol $e \in \mathbf{\Sigma}_\mathcal{E}$. We can lift $exp_\mathbf{\Sigma}$ to any regular expression α over alphabet $\mathbf{\Sigma}_\mathcal{E}$ in a straightforward way: $exp_\mathbf{\Sigma}(\alpha)$ is the regular expression over $\mathbf{\Sigma}$ obtained by replacing each occurrence of $e \in \mathbf{\Sigma}_\mathcal{E}$ with $exp_\mathbf{\Sigma}(e)$. We let $\mathcal{L}_\mathcal{E}(\alpha)$ be the language of α over $\mathbf{\Sigma}_\mathcal{E}$ (see Definition 2.1.2). Given a regular expression α over $\mathbf{\Sigma}_\mathcal{E}$, $exp_\mathbf{\Sigma}(\alpha)$ is called the *expansion* of α. It is clear that $exp_\mathbf{\Sigma}(e_{\pi_1} \cdot e_{\pi_2} \cdots e_{\pi_n}) = \mathcal{L}(\pi_1 \cdot \pi_2 \cdots \pi_n)$ for $\{\pi_1, \ldots, \pi_n\} \subseteq \mathcal{E}$.

Now we define the concept of regular expression rewriting formally:

8.2.3. D . **(Regular Expression Rewriting)** Given a regular expression π over $\mathbf{\Sigma}$, a set of regular expressions over $\mathbf{\Sigma}$: $\mathcal{E} = \{\pi_0, \pi_1, \ldots \pi_n\}$, and another regular expression α over the alphabet $\mathbf{\Sigma}_\mathcal{E}$, we say α is an *$\mathcal{E}$-rewriting* of π if $exp_\mathbf{\Sigma}(\alpha) \subseteq \mathcal{L}(\pi)$. α is called a *maximal $\mathcal{E}$-rewriting* (notation: $\widehat{\pi}_\mathcal{E}$) if for any other $\mathcal{E}$-rewriting β of π: $\mathcal{L}_\mathcal{E}(\beta) \subseteq \mathcal{L}_\mathcal{E}(\alpha)$ (thus $exp_\mathbf{\Sigma}(\beta) \subseteq exp_\mathbf{\Sigma}(\alpha)$). We say that a rewriting α is *empty* if $\mathcal{L}_\mathcal{E}(\pi) = \emptyset$. ₥

Note that given $\mathcal{E}$ and π, there is a unique maximal $\mathcal{E}$-rewriting of π (modulo language equivalence over $\mathbf{\Sigma}_\mathcal{E}$), for otherwise suppose there are two different maximal rewritings α, β. Then $\alpha + \beta$ is also an $\mathcal{E}$-rewriting of π, which contradicts the maximality of α and β. Moreover, we have the following straightforward result:

8.2.4. P . *If $\mathcal{L}(\pi_1 \cdot \pi_2 \cdots \pi_n) \subseteq \mathcal{L}(\pi)$ and $\{\pi_1, \pi_2, \ldots, \pi_n\} \subseteq \mathcal{E}$ then $e_{\pi_1} \cdots e_{\pi_n} \in \mathcal{L}_\mathcal{E}(\widehat{\pi}_\mathcal{E})$,*

P Towards a contradiction suppose $\mathcal{L}(\pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi)$ and $\{\pi_1, \ldots, \pi_n\} \subseteq \mathcal{E}$, but $e_{\pi_1} \cdots e_{\pi_n} \notin \mathcal{L}_\mathcal{E}(\widehat{\pi}_\mathcal{E})$. Let $\alpha = \widehat{\pi}_\mathcal{E} + (e_{\pi_1} \cdots e_{\pi_n})$. It is clear that α is another $\mathcal{E}-$rewriting of π such that $\mathcal{L}_\mathcal{E}(\widehat{\pi}_\mathcal{E}) \subset \mathcal{L}_\mathcal{E}(\alpha)$, which contradicts the maximality of $\widehat{\pi}_\mathcal{E}$. ₭

The following two theorems are from [CDGLV02]:

8.2.5. T . *([CDGLV02]) The problem of verifying the existence of a non-empty rewriting of a regular expression π' w.r.t. a set $\mathcal{E}$ of regular expressions is* E -complete.

8.2.6. T . *([CDGLV02]) There is an essentially optimal algorithm to compute the maximal $\mathcal{E}$-rewriting of a given π w.r.t a given set $\mathcal{E}$ in* 2-E .

8.3 Model Checking

In this section, we tackle the model checking problem. At first sight, one might think this is a simple problem: an immediate idea might be to first transform an AKM into a Kripke model by replacing every accelerated transition labelled by π with the corresponding (deterministic) automaton of π, then run a traditional model checking algorithm. However, this does *not* work, at least not in a naive way. Let us look at the left figure in Example 8.2.2. Suppose one wants to check $\langle a \rangle \top$, which is *false* at s, following the above idea, one can obtain a Kripke model in the right figure. However, the result will be *true*. This example suggests that the model checking cannot be performed in a very simple way.

8.3.1 A Reduction to Standard $\mathtt{PDL}_\Sigma$ Model Checking

We now present a non-trivial method to reduce the model checking problem of $\mathtt{PDL}_\Sigma$ over AKM to the one over Kripke models. Here, as said, the notion of regular expression rewriting is crucially exploited.

Given an AKM $\mathcal{M} = (S, \mathbf{\Sigma}, \rightarrow, V)$, let :

$$\langle\rangle_{\mathcal{M}} = \{\pi \mid \pi \in Reg_{\mathbf{\Sigma}} \text{ and } \pi \text{ appears as a label for some transition in } \mathcal{M}\}$$

We define $\ulcorner \mathcal{M} \urcorner$ as $(S, \{e_\pi \mid \pi \in \langle\rangle_{\mathcal{M}}\}, \rightarrow', V)$ where $s \xrightarrow{e_\pi}' s'$ iff $s \xrightarrow{\pi} s'$. If $\langle\rangle_{\mathcal{M}}$ is finite then we can compute the maximal $\langle\rangle_{\mathcal{M}}$-rewriting of any regular expressions $\pi \in Reg_{\mathbf{\Sigma}}$ in 2-E , according to Theorem 8.2.6. Then we can rewrite a $\mathtt{PDL}_\Sigma$-formula w.r.t to a model as follows:

8.3.1. D . **(Rewriting w.r.t an AKM)** Given an AKM $\mathcal{M}$ and a $\mathtt{PDL}_\Sigma$ formula ϕ, $\mathfrak{R}_{\mathcal{M}}(\phi)$ is the rewriting of ϕ in the language $\mathtt{PDL}_{\Sigma_{\langle\rangle_{\mathcal{M}}}}$ defined by :

- $\mathfrak{R}_{\mathcal{M}}(p) = p$ for $p \in \mathbf{P}$;
- $\mathfrak{R}_{\mathcal{M}}(\neg\psi) = \neg\mathfrak{R}_{\mathcal{M}}(\psi)$;
- $\mathfrak{R}_{\mathcal{M}}(\psi_1 \wedge \psi_2) = \mathfrak{R}_{\mathcal{M}}(\psi_1) \wedge \mathfrak{R}_{\mathcal{M}}(\psi_2)$;
- $\mathfrak{R}_{\mathcal{M}}(\langle\pi\rangle\psi) = \langle\widehat{\pi}_{\langle\rangle_{\mathcal{M}}}\rangle\mathfrak{R}_{\mathcal{M}}(\psi)$.

$\mathfrak{m}$

Let $\Vdash$ denote the standard $\mathtt{PDL}_\Sigma$ semantics on Kripke models (cf. Section 2.3.1), the we have:

8.3.2. T . *For any pointed AKM $\mathcal{M}, s$ and any $\mathtt{PDL}_\Sigma$ formula ϕ,*

$$\mathcal{M}, s \vDash \phi \iff \ulcorner \mathcal{M} \urcorner, s \Vdash \mathfrak{R}_{\mathcal{M}}(\phi).$$

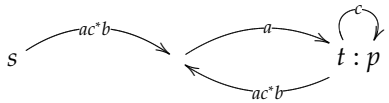


Figure 8.1: Accelerated LTS

P By induction on the structure of ϕ. The only interesting case is $\phi = \langle\pi\rangle\psi$.

($\Rightarrow$:) Suppose $\mathcal{M}, s \vDash \langle\pi\rangle\psi$ then there exists some t in $\mathcal{M}$ such that $s \xrightarrow{\pi_1} \cdots \xrightarrow{\pi_n} t$ in $\mathcal{M}$, $\mathcal{L}(\pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi)$, and $\mathcal{M}, t \vDash \psi$. From proposition 8.2.4, it is not hard to see that $e_{\pi_1} \cdots e_{\pi_n} \in \mathcal{L}_{\langle\rangle_\mathcal{M}}(\widehat{\pi}_{\langle\rangle_\mathcal{M}})$. By the induction hypothesis, $\ulcorner\mathcal{M}\urcorner, t \Vdash \mathfrak{R}_\mathcal{M}(\psi)$ and thus $\ulcorner\mathcal{M}\urcorner, s \Vdash \langle\widehat{\pi}_{\langle\rangle_\mathcal{M}}\rangle\mathfrak{R}_\mathcal{M}(\psi)$. Namely $\ulcorner\mathcal{M}\urcorner, s \Vdash \mathfrak{R}_\mathcal{M}(\phi)$.

($\Leftarrow$:) Suppose $\ulcorner\mathcal{M}\urcorner, s \Vdash \langle\widehat{\pi}_{\langle\rangle_\mathcal{M}}\rangle\mathfrak{R}_\mathcal{M}(\psi)$, then there exits a path $s \xrightarrow{e_{\pi_1}} \cdots \xrightarrow{e_{\pi_n}} t$ in $\ulcorner\mathcal{M}\urcorner$ such that $e_{\pi_1} \cdots e_{\pi_n} \in \mathcal{L}_{\langle\rangle_\mathcal{M}}(\widehat{\pi}_{\langle\rangle_\mathcal{M}})$ with $\{\pi_1, \ldots, \pi_n\} \subseteq \langle\rangle_\mathcal{M}$. It follows that $exp_\Sigma(e_{\pi_1} \cdots e_{\pi_n}) \subseteq exp_\Sigma(\widehat{\pi}_{\langle\rangle_\mathcal{M}})$. Since $\widehat{\pi}_{\langle\rangle_\mathcal{M}}$ is a $\langle\rangle_\mathcal{M}$-rewriting, $\mathcal{L}(\pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi)$. By the definition of $\ulcorner\mathcal{M}\urcorner$, $s \xrightarrow{\pi_1} \cdots \xrightarrow{\pi_n} t$ in $\mathcal{M}$. By the induction hypothesis, $\mathcal{M}, t \vDash \psi$, and thus $\mathcal{M}, s \vDash \langle\pi\rangle\psi$. ⋈

Theorem 8.3.2 allows us to use the standard $\mathtt{PDL}_\Sigma$ model checking algorithm (e.g. [Lan06]) to solve the problem over AKM in a straightforward manner. We present an example here. Let us consider the AKM $\mathcal{M}$ depicted in Fig. 8.1. Suppose we need to check whether the formula $\phi = \langle a \cdot (b \cdot a + c)^*\rangle p$ holds at state s. We first collect the set $\langle\rangle_\mathcal{M} = \{a, a \cdot c^* \cdot b, c\}$; then we compute the maximal rewriting of $a \cdot (b \cdot a + c)^*$ w.r.t $\langle\rangle_\mathcal{M}$, following the algorithm of generating the maximal rewriting in [CDGLV02]. It follows that $\mathfrak{R}_\mathcal{M}(\phi) = \langle(e_{a \cdot c^* \cdot b})^* \cdot e_a \cdot (e_c)^*\rangle p$[1]. According to Theorem 8.3.2, we only need to check whether $\ulcorner\mathcal{M}\urcorner, s \Vdash \mathfrak{R}_\mathcal{M}(\phi)$, where $\ulcorner\mathcal{M}\urcorner$ is the same graph as in Fig.8.1 except that the labels become $e_{a \cdot c^* \cdot b}, e_a$ and e_c respectively. A standard $\mathtt{PDL}_\Sigma$ model checking algorithm will return and thus we can conclude that $\mathcal{M}, s \vDash \phi$.

8.3.2 A Direct Algorithm

Due to Theorem 8.2.6, the above translation $\mathfrak{R}_\mathcal{M}$ is quite expensive (2-E). To avoid generating the maximal rewriting explicitly, we may process the rewriting and the model checking at the same time and check non-emptiness of the rewriting when needed. Based on this idea, we now present a more efficient direct algorithm, which shares the same basic structure as those proposed in literature for branching-time temporal logic (typically CTL, see e.g. [CGP99] for a clear exposition). In a nutshell, given a formula ϕ, the algorithm recursively evaluates the truth-values of the subformulas ψ of ϕ at all states, starting from the propositional formulas of ϕ and following the recursive definitions of each modality. The whole process will be gathered up in a global labelling algorithm. It turns out that the central part of the algorithm is to solve the following question:

[1] Actually $e^*_{a \cdot c^* \cdot b} \cdot e_a \cdot e^*_c$ is a exact $\langle\rangle_\mathcal{M}$-rewriting of $a \cdot (b \cdot a + c)^*$ (cf. [CDGLV02]).

Exists$(\mathcal{M}, s, T, \pi_0)$: Given a pointed AKM $\mathcal{M} = (S, \mathbf{\Sigma}, \rightarrow, V, s_0)$, a set of states $T \subseteq S$ and a regular expression π_0, check whether there exists a sequence of transitions $s \xrightarrow{\pi_1} s_1 \cdots \xrightarrow{\pi_n} t$ such that $\mathcal{L}(\pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi_0)$ and $t \in T$. **Exists**$(\mathcal{M}, s, T, \pi_0)$ returns $\top$ if the answer is yes and returns $\bot$ otherwise.

In the sequel, we deal with this problem by an automata-theoretic approach. A sketch is as follows:

1. Construct a *deterministic* automaton (DFA) A_{π_0} such that $\mathcal{L}(\mathsf{A}_\pi) = \mathcal{L}(\pi_0)$;
2. Define a suitable product $\mathcal{M} \otimes \mathsf{A}_{\pi_0}$ of $\mathcal{M}$ and A_{π_0};
3. Run the emptiness check on $\mathcal{M} \otimes \mathsf{A}_{\pi_0}$.

Now, we present the detailed construction step by step. Step 1 is standard (cf., e.g., [Con71]). We start from Step 2.

8.3.3. D . **(Product $\otimes_T$)** Given a pointed AKM $\mathcal{M} = (S, Act, \rightarrow, s_0)$ and $T \subseteq S$, a DFA $\mathsf{A} = (Q, \Sigma, \delta, q_0, F)$ (cf., Definition. 2.1.1), define $\mathcal{M} \otimes_T \mathsf{A}$ as the nondeterministic automaton $(G, \Sigma', \rho, q_0', F')$ where:

- $G = S \times \mathcal{P}(Q)$;
- $\Sigma' = \{e_\pi \mid \pi \text{ appears in the transition of } \mathcal{M}\}$;
- $(\langle s, U\rangle, e_\pi, \langle t, R\rangle) \in \rho \iff s \xrightarrow{\pi} t$ and $R = \bigcup_{u \in U} \bigcup_{w \in \mathcal{L}(\pi)} \{\delta^*(u, w)\}$;
- $q_0' = \langle s_0, \{q_0\}\rangle$; and
- $F' = \{\langle s, U\rangle \mid s \in T \text{ and } U \subseteq F\}$.

δ^* is the extended transition function in a deterministic automaton such that given $u \in Q$ and word $w \in \mathbf{\Sigma}^*$, $\delta^*(q, w)$ gives the unique state that can be reached from q by a w path. ℜ

In the above construction, although $\mathcal{L}(\pi)$ may be infinite, we can still compute $\bigcup_{w \in \mathcal{L}(\pi)} \{\delta^*(u, w)\}$. Given an $u \in Q_\mathsf{A}$ and a regular expression π, let A^u be the DFA just as A but with the new start state u, and let A' be a nondeterministic automaton such that $\mathcal{L}(\mathsf{A}') = \mathcal{L}(\pi)$. It is not hard to see that we can compute $\bigcup_{w \in \mathcal{L}(\pi)} \delta^*(u, w)$ by collecting the $u' \in Q_\mathsf{A}$ in the reachable final states of the standard product of A^u and A' (cf. e.g., [Con71]).

Step 3, the emptiness checking for a nondeterministic automaton can be done in a standard and efficient way. We are in a position to present the correctness of the whole procedure.

8.3.4. P . *Given a pointed AKM* $\mathcal{M} = (S, \mathbf{\Sigma}, \to, V, s_0)$ *and* $T \subseteq S$, *if* $T = \{t \mid \mathcal{M}, t \vDash \phi\}$, *then we have:*

$$\mathcal{M}, s_0 \vDash \langle \pi_0 \rangle \phi \iff \mathcal{L}(\mathcal{M} \otimes_T \mathsf{A}_{\pi_0}) \neq \emptyset$$

where A_{π_0} *denotes the deterministic automaton corresponding to* π_0.

P Let $\mathsf{A}_{\pi_0} = (Q, \Sigma, \delta, q_0, F)$ be the deterministic automaton of π_0.

($\Rightarrow$:) Since $\mathcal{M}, s_0 \vDash \langle \pi_0 \rangle \phi$, there is a path $s_0 \xrightarrow{\pi_1} s_1 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_n} s_n$ in $\mathcal{M}$ such that $s_n \in T$ and $\mathcal{L}(\pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi_0)$. It follows that for any $w_1 \cdots w_n \in \mathcal{L}(\pi_1 \cdots \pi_n)$ where $w_i \in \mathcal{L}(\pi_i)$, $w_1 \cdots w_n \in \mathcal{L}(\mathsf{A}_{\pi_0})$.

We define $U_0 = \{q_0\}$ and $U_{i+1} = \bigcup_{u \in U_i} \bigcup_{w \in \mathcal{L}(\pi_i)} \{\delta^*(u, w)\}$. Now consider the trace:

$$\langle s_0, U_0 \rangle \xrightarrow{e_{\pi_1}} \langle s_1, U_1 \rangle \xrightarrow{e_{\pi_2}} \cdots \xrightarrow{e_{\pi_n}} \langle s_n, U_n \rangle \qquad (\#)$$

Clearly, this is a path in $\mathcal{M} \otimes_T \mathsf{A}_{\pi_0}$. Moreover, for each state $q \in U_n$, there must exist $q_0, q_1, \cdots, q_n = q$ with $w_1, w_2, \cdots, w_n$ such that for each i, $q_i \in U_i$ and $q_{i+1} = \delta^*(q_i, w_i)$ and $w_i \in \mathcal{L}(\pi_i)$. Therefore $w_1 \cdot w_2 \cdots w_n \in \mathcal{L}(\pi_0)$. and then $w_1 \cdot w_2 \cdots w_n \in \mathcal{L}(\pi_0)$. Thus $w_1 \cdot w_2 \cdots w_n$ is accepted by A_{π_0}. Since A_{π_0} is deterministic, q must be an accept state, namely, $q \in F$. It follows that $U_n \subseteq F$. Since $s_n \in T_n$, (s_n, U_n) is an accept state in $\mathcal{M} \otimes_T \mathsf{A}_{\pi_0}$. Therefore the above path (#) is an accepting path, i.e. $\mathcal{L}(M \otimes_T \mathsf{A}_{\pi_0}) \neq \emptyset$.

($\Leftarrow$:) Suppose $\mathcal{L}(\mathcal{M} \otimes_T \mathsf{A}_{\pi_0}) \neq \emptyset$, then there exists some path:

$$\langle s_0, U_0 \rangle \xrightarrow{e_{\pi_1}} \langle s_1, U_1 \rangle \xrightarrow{e_{\pi_2}} \cdots \xrightarrow{e_{\pi_n}} \langle s_n, U_n \rangle$$

such that (s_n, U_n) is an accept state, namely, $s_n \in T$ and $U_n \subseteq F$. It follows from the definition that $s_0 \xrightarrow{\pi_1} s_1 \xrightarrow{\pi_2} \cdots \xrightarrow{\pi_n} s_n$ in $\mathcal{M}$, and for each $w_i \in \mathcal{L}(\pi_i)$, each $u_i \in U_i$, $\delta^*(u_i, w_i) \in U_{i+1}$. Hence for any $w_1 \cdots w_n \in \mathcal{L}(\pi_1 \cdots \pi_n)$, we can construct a sequence of states $q_0, q_1, \cdots, q_n$ such that $\delta^*(q_i, w_i) = q_{i+1}$ and $q_i \in U_i$. Since $U_n \subseteq F$, $q_n \in F$. Namely, $w_1 \cdots w_n$ is accepted by A_{π_0} and thus $w_1 \cdots w_n \in \mathcal{L}(\pi_0)$. Therefore $\mathcal{L}(\pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi_0)$, namely $\mathcal{M}, s_0 \vDash \langle \pi_0 \rangle \phi$. ⋈

Note that in the above correctness proof, we rely on the property of *determinism*. It is crucial that the transition function assigns to each state and each label one and only one successor.

Algorithm. We have presented how to construct the function **Exists**$(\mathcal{M}, s, T, \pi)$ and now we are in the position to give the full algorithm, as defined in Algorithm 1. The termination of the algorithm is clear and thus the correctness can be ensured by the following theorem:

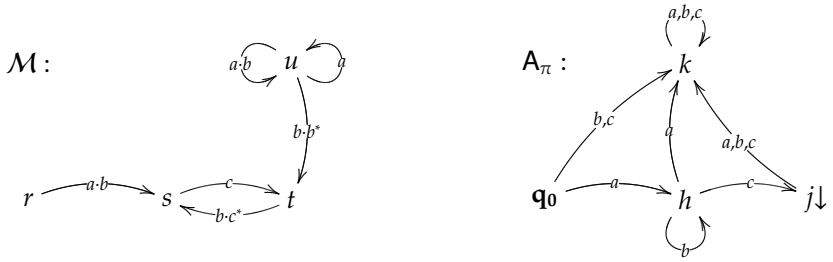
8.3.5. T . *Given a pointed AKM* $\mathcal{M} = (S, Act, \to, s_0)$, *and a* PDL *formula* ϕ.

$$s_0 \in \mathit{eval}(\phi) \iff s_0 \vDash \phi$$

Algorithm 1 Model Checking Algorithm

Input: A pointed AKM $\mathcal{M} = (S, Act, \rightarrow, s_0)$, and a $\mathtt{PDL}_\Sigma$ formula ϕ.
return $s_0 \in eval(\phi)$;
where
Function $eval(\phi)$
  If $\phi = \top$, then **return** S;
  If $\phi = p$, then **return** $\{s \in S \mid p \in V(s)\}$;
  If $\phi = \neg\phi'$, then **return** $S \setminus eval(\phi')$;
  If $\phi = \phi' \wedge \phi''$, then **return** $eval(\phi') \cap eval(\phi'')$;
  If $\phi = \langle\pi\rangle\phi'$, then **return**
    $\{s \in S \mid \mathbf{Exists}(\mathcal{M}, s, eval(\phi'), \pi) = \top\}$;

We end this section by presenting an example, originally appearing in [EvdP06]. Let us consider the AKM $\mathcal{M}$ below (the valuation function is not essential for this example and we name the states as r, s, t, etc.):



We demonstrate how to compute $\mathbf{Exists}(\mathcal{M}, r, T, \pi)$, where $\pi = a \cdot b^* \cdot c$ and $T = \{t\}$. The first step is to transform π into a complete deterministic automaton A_π (the right graph above, where q_0 is the start state and j is the accept state). Then we can compute the product, which is simply:

$$(r, \{q_0\}) \xrightarrow{e_{a\cdot b}} (s, \{h\}) \xrightarrow{e_c} (t, \{j\}) \xrightarrow{e_{b\cdot c^*}} (s, \{k\}) \underset{e_{b\cdot c^*}}{\overset{e_c}{\rightleftarrows}} (t, \{k\})$$

with $(t, \{j\})$ the accept state (note that by an on-the-fly construction, the unreachable parts are omitted). At last, the emptiness checking yields YES. It is clear that our algorithm is much simpler than the one presented in [EvdP06].

8.3.3 Complexity Analysis

Upper Bounds. Let us analyse the complexity of the algorithm presented above for model checking $\mathtt{PDL}_\Sigma$ over AKMs. As we mentioned before, the essential part of the algorithm is the oracle $\mathbf{Exists}(\mathcal{M}, s, T, \pi)$. We observe that

(1) for a regular expression π, the deterministic automaton A_π is of exponential size $O(2^{|\pi|})$;

(2) the product $\mathcal{M} \otimes_T \mathsf{A}_\pi$ is of the size $O(|\mathcal{M}| \cdot 2^{2^{|\pi|}})$;

(3) Checking emptiness can be done in nondeterministic logarithmic space.

To glue them together, we obtain a nondeterministic E bound, and using Savitch's theorem, we get a deterministic E bound for the oracle[2]. Moreover, as in the traditional algorithm for CTL, the main algorithm presented in Algorithm 1 can be done in time with an E -bounded oracle. So the complexity is $^{\text{E}}$, which is E .

One might think the complexity is a bit too high in practice. However, Lichtenstein and Pnueli argued that when analyzing the complexity of model checking, a distinction should be made between complexity in the size of the input structure and complexity in the size of the input formula. And it is often the complexity in the size of the structure that is typically the computational bottleneck [LP85]. In a nutshell, *program complexity* refers to the complexity of the problem in terms of the size of the input module, assuming the formula is *fixed*. Clearly, in our case, the program complexity turns out to be L . This is important in practice since people might argue that the complexity of our algorithm is too high to be practical. However, in practice, usually the logic formula is small and in this case the algorithm still performs very well.

Lower Bound. We show that the upper bound established above is essentially optimal. We shall exploit the regular expression rewriting problem (see Section 8.2.2) to prove the E lower bound of the problem of model checking AKM w.r.t. a $\texttt{PDL}_{\mathbf{\Sigma}}$ formula.

We present a reduction as follows:

8.3.6. L . *Given a set of non-empty regular expressions $\mathcal{E} = \{\pi_1, \cdots, \pi_k\}$ and another regular expression π over $\mathbf{\Sigma}$, there exists a pointed AKM model $(\mathcal{M}_\mathcal{E}, s)$ and a $\texttt{PDL}_{\mathbf{\Sigma}}$ formula ϕ such that:*

$$\mathcal{M}_\mathcal{E}, s \vDash \langle\pi\rangle\phi \iff \textit{there is a non-empty rewriting of } \pi \textit{ w.r.t. } \mathcal{E}.$$

P Given $\mathcal{E} = \{\pi_1, \cdots, \pi_k\}$ and π, we define the AKM $\mathcal{M}_\mathcal{E}$ as

$$(\{s\}, \mathcal{E}, \rightarrow, V)$$

where $\rightarrow = \{(s, \pi_i, s) \mid \pi_i \in \mathcal{E}\}$, V is an arbitrary valuation. Let $\phi = \langle\pi\rangle\top$.

($\Rightarrow$:) Suppose $\mathcal{M}_\mathcal{E}, s \vDash \langle\pi\rangle\top$. According to the semantics, there is a path in $\mathcal{M}_\mathcal{E}$ with $s \xrightarrow{\pi'_1} s \cdots \xrightarrow{\pi'_m} s$ where $\{\pi'_1, \cdots, \pi'_m\} \subseteq \mathcal{E}$ and $\mathcal{L}(\pi'_1 \cdots \pi'_m) \subseteq \mathcal{L}(\pi)$. It follows that $e_{\pi'_1} \cdots e_{\pi'_m}$ is a non-empty rewriting of π w.r.t. $\mathcal{E}$.

[2]Note that some care is needed to get the claimed space bound. We cannot simply construct A_π since it is of doubly exponential size. Instead, we construct A_π on the fly.

($\Leftarrow$:) Suppose there is a non-empty rewriting α of π w.r.t. $\mathcal{E}$. Since α is non-empty, there is a possibly empty word $e_{\pi'_1} \cdots e_{\pi'_m} \in \mathcal{L}(\alpha)$ where for each $1 \leq i \leq m$, $\pi'_i \in \mathcal{E}$. It is easy to see that $exp_{\Sigma}(e_{\pi'_1} \cdots e_{\pi'_m}) \subseteq exp_{\Sigma}(\alpha)$. Furthermore, according to the definition of the rewriting, $exp_{\Sigma}(\alpha) \subseteq \mathcal{L}(\pi)$ and thus $\mathcal{L}(\pi'_1 \cdots \pi'_m) \subseteq \mathcal{L}(\pi)$. Clearly there exists a path in $\mathcal{M}_{\mathcal{E}}$ with $s \xrightarrow{\pi'_1} s \cdots \xrightarrow{\pi'_m} s$ thus $\mathcal{M}_{\mathcal{E}}, s \vDash \langle\pi\rangle\top$. This completes the proof. ⋈

Based on the E upper bound, Theorem 8.2.5 and Lemma 8.3.6 yield the main result of the current section, as follows:

8.3.7. T . *The problem of model checking a* $\mathtt{PDL}_{\Sigma}$ *formula w.r.t. an AKM is* E - *complete.*

8.4 Axiomatization

In this section, we give a complete axiomatization of $\mathtt{PDL}_{\Sigma}$ over AKM. Although the syntax of $\mathtt{PDL}_{\Sigma}$ does not change, the interpretation over AKM results in a new semantics which differs from standard $\mathtt{PDL}_{\Sigma}$ considerably. For instance, the following axioms are valid in standard $\mathtt{PDL}_{\Sigma}$. However, most of them are *not* valid any more (if a $\leftarrow$ appears in the right column, this indicates that $\leftrightarrow$ should be replaced by $\leftarrow$ to keep the formula valid).

Axioms	In our semantics
$[\pi](\phi \to \psi) \to ([\pi]\phi \to [\pi]\psi)$	valid
$\langle\pi_1 \cdot \pi_2\rangle\phi \leftrightarrow \langle\pi_1\rangle\langle\pi_2\rangle\phi$	$\leftarrow$
$\langle\pi_1 + \pi_2\rangle\phi \leftrightarrow \langle\pi_1\rangle\phi \vee \langle\pi_2\rangle\phi$	$\leftarrow$
$\langle\pi^*\rangle\phi \leftrightarrow (\phi \vee \langle\pi\rangle\langle\pi^*\rangle\phi)$	$\leftarrow$
$[\pi^*](\phi \to [\pi]\phi) \to (\phi \to [\pi^*]\phi)$	invalid

In view of this, instead of the standard $\mathtt{PDL}_{\Sigma}$ axioms we propose the following new conditional axiomatization.

8.4.1. D . A deductive system AS

TAUTOLOGY	all the tautologies
K	$[\pi](\phi \to \phi') \to ([\pi]\phi \to [\pi]\phi')$
SEQ	$[\pi_1 \cdot \pi_2]\phi \to [\pi_1][\pi_2]\phi$
STAR	$[\pi^*]\phi \to \phi$
Rules	
$\Box$	$\dfrac{\phi}{[\pi]\phi}$
MP	$\dfrac{\phi, \phi \to \psi}{\psi}$
INCL	$\dfrac{\vdash_{\mathtt{KA}} \pi + \pi' = \pi'}{[\pi']\phi \to [\pi]\phi}$

where KA is a complete Kleene algebra, for example as in [Koz91], acting as an oracle.

The rest of this section is devoted to showing that AS is sound and complete w.r.t the class of all AKM frames. First let us consider a special class of AKM frames on which we can use an equivalent simple semantics for technical convenience. An AKM frame is called *normal* if it satisfies the following properties:

- sequentiality: For any $\pi, \pi' \in Reg_{\Sigma}$: $\xrightarrow{\pi} \circ \xrightarrow{\pi'} \subseteq \xrightarrow{\pi \cdot \pi'}$ where $\circ$ is concatenation of binary relations;
- *-reflexivity: For any $\pi \in Reg_{\Sigma}$: if $\{\mathbf{1}\} \in \mathcal{L}(\pi)$ then $s \xrightarrow{\pi} s$ for any $s \in S$;
- regularity: For any $\pi, \pi' \in Reg_{\Sigma}$: $\mathcal{L}(\pi) \subseteq \mathcal{L}(\pi')$ implies that $\xrightarrow{\pi} \subseteq \xrightarrow{\pi'}$.

Models based on the normal AKM frames are called *normal* AKM models. Now we can define an equivalent semantics $\vDash_0$ on the normal AKM models as follows:

- For boolean cases: as before;
- For modal case:
  $\mathcal{M}, s \vDash_0 \langle \pi \rangle \phi \iff \exists t : s \xrightarrow{\pi} t$ and $t \vDash_0 \phi$.

We can saturate an arbitrary AKM frame of PDL_{π}: $\mathcal{F} = (S, \mathbf{\Sigma}, \rightarrow)$ into a normal frame $R(\mathcal{F}) = (S, \mathbf{\Sigma}, \rightarrow_r)$ by adding transitions[3]:

$$s \xrightarrow{\pi}_r t \iff \exists s \xrightarrow{\pi_1} s_1 \xrightarrow{\pi_1} \cdots \xrightarrow{\pi_n} s_n \text{ and } \mathcal{L}(\pi_1 \pi_2 \ldots \pi_n) \subseteq \mathcal{L}(\pi)$$

$R(\mathcal{M})$ is the saturated model which keeps the valuation the same but saturates the frame of $\mathcal{M}$. It is easy to see that $\vDash_0$ coincides with $\vDash$ on normal models:

8.4.2. P . *Given an AKM $\mathcal{M} = (S, \mathbf{\Sigma}, \rightarrow, V)$, for any $\mathrm{PDL}_{\mathbf{\Sigma}}$ formula ϕ:*

$$\mathcal{M}, s \vDash \phi \iff R(\mathcal{M}), s \vDash_0 \phi \iff R(\mathcal{M}), s \vDash \phi$$

Since all the normal AKM frames are AKM frames and all the AKM frames can be saturated into normal AKM frames, it follows from the above proposition that $\Delta \vDash \phi \iff \Delta \vDash_0 \phi$, where Δ is a set of $\mathrm{PDL}_{\mathbf{\Sigma}}$ formulas.

It is easy to check the following lemmata:

8.4.3. L . *For any normal AKM frame $\mathcal{F}$ and any two regular expressions π and π', if $\vdash_{KA} \pi + \pi' = \pi'$ then $\mathcal{F} \vDash_0 [\pi']p \rightarrow [\pi]p$.*

Here $\mathcal{F} \vDash_0 \phi$ iff for any model $\mathcal{M}$ based on $\mathcal{F}$: $\mathcal{M} \vDash_0 \phi$.

8.4.4. L . *For any normal AKM frame $\mathcal{F}$: $\mathcal{F}$ satisfies sequentiality $\iff \mathcal{F} \vDash_0$ SEQ.*

8.4.5. L . *For any normal AKM frame $\mathcal{F}$: $\mathcal{F}$ satisfies *-reflexivity implies $\mathcal{F} \vDash_0$ STAR.*

[3]Note that $\mathbf{1}$ denotes for the empty sequence, and for any s: $s \xrightarrow{\mathbf{1}} s$.

From the above lemmata, and the completeness of Kleene Algebra [Koz91], it is straightforward to establish:

8.4.6. T (S). AS *is sound for normal AKM frames.*

Note that the STAR axiom does not correspond to $*$-reflexivity by itself, but in the presence of the other two properties[4]:

8.4.7. L . *If an AKM frame $\mathcal{F}$ satisfies regularity, sequentiality and $\mathcal{F} \vDash$ STAR then $\mathcal{F}$ is normal.*

P Suppose $\mathcal{F}$ satisfies regularity and sequentiality, we only need to show that $\mathcal{F}$ satisfies *-reflexivity: for any regular expression $\pi \in Reg_{\Sigma}$, if $\mathbf{1} \in \mathcal{L}(\pi)$ then $\xrightarrow{\pi}$ is reflexive. We prove this by induction on the structure of π.

- If $\pi = \pi'^{*}$ then it is straightforward to check that $\xrightarrow{\pi}$ is reflexive since $\mathcal{F} \vDash$ STAR.
- If $\pi = \pi_1 + \pi_2$ then $\mathbf{1} \in \mathcal{L}(\pi_1)$ or $\mathbf{1} \in \mathcal{L}(\pi_2)$. By the induction hypothesis $\xrightarrow{\pi_1}$ is reflexive or $\xrightarrow{\pi_2}$ is reflexive. From regularity, $\xrightarrow{\pi_1} \subseteq \xrightarrow{\pi}$ and $\xrightarrow{\pi_2} \subseteq \xrightarrow{\pi}$. So $\xrightarrow{\pi}$ is reflexive.
- If $\pi = \pi_1 \cdot \pi_2$ then $\mathbf{1} \in \mathcal{L}(\pi_1)$ and $\mathbf{1} \in \mathcal{L}(\pi_2)$. By the induction hypothesis $\xrightarrow{\pi_1}$ and $\xrightarrow{\pi_2}$ are reflexive. From sequentiality, $\xrightarrow{\pi_1} \circ \xrightarrow{\pi_2} \subseteq \xrightarrow{\pi}$. So $\xrightarrow{\pi}$ is reflexive.

Completeness follows from the standard canonical model construction.

8.4.8. T (C). *For any set of $\mathtt{PDL}_{\Sigma}$ formulas $\Delta \cup \{\phi\}$: $\Delta \vDash_0 \phi \implies \Delta \vdash_{\mathsf{AS}} \phi$. Namely* AS *is strongly complete for normal AKM frames w.r.t $\vDash_0$. Thus* AS *is strongly complete for all AKM frames.*

P Recall that a logic theory is a *normal modal logic* if it contains all the instances of tautologies, the K axiom and is closed under MP and $\Box$.Therefore AS induces a normal modal logic. Thus it is strongly complete with respect to its canonical model $\mathcal{M}^c = (S^c, Reg_{\Sigma}, \longrightarrow^c, V^c)$ according to the canonical model theorem(see e.g., [BdRV02, Theorem 4.22]), where S^c is the set of all AS−maximal consistent sets, $s \xrightarrow{\pi} t$ if for all ψ, $\psi \in s \Rightarrow \langle\pi\rangle\psi \in t$, $V^c(s) = \{p \mid p \in s\}$. We only need to show that the canonical model $\mathcal{M}^c$ is indeed a model based on a normal AKM frame. Since S^c is the set of AS-maximal consistent sets, $\mathcal{M}^c \vDash_0$ STAR $\wedge$SEQ. From Lemma 8.4.4 and 8.4.7, we only need to show the canonical model satisfies regularity:

$$\text{For any } \pi, \pi' \in \pi^{*}, \mathcal{L}(\pi) \subseteq \mathcal{L}(\pi') \text{ implies } \xrightarrow{\pi}{}^{c} \subseteq \xrightarrow{\pi'}{}^{c} .$$

[4]That is why we don't include a rule like: $\frac{[\pi]\phi}{\phi}$ if $\epsilon \in \mathcal{L}(\pi)$.

Suppose there are regular expressions π, π' such that $\mathcal{L}(\pi) \subseteq \mathcal{L}(\pi')$ and $\exists s, t : s \overset{\pi}{\longrightarrow}^c t$ in the canonical model. From the definition of $\overset{\pi}{\longrightarrow}^c$, we have for all $\psi : \psi \in t \Rightarrow \langle\pi\rangle\psi \in s$. Since $\mathcal{L}(\pi) \subseteq \mathcal{L}(\pi')$ and KA is complete, we have $\vdash_{\mathtt{KA}} \pi + \pi' = \pi'$. Since s is a maximal consistent set, then from INCL we have for all $\psi : \langle\pi\rangle\psi \rightarrow \langle\pi'\rangle\psi \in s$. Therefore by applying MP we have for all $\psi \in t : \langle\pi'\rangle\psi \in s$. It follows, by definition, that $s \overset{\pi'}{\longrightarrow}^c t$. ⋈

Strong completeness implies compactness:

8.4.9. C (C). $\mathtt{PDL}_\Sigma$ *w.r.t AKM is model compact. Namely if all the finite subsets of* Γ *are satisfiable then* Γ *is satisfiable.*

8.4.10. R . Recall that the standard $\mathtt{PDL}_\Sigma$ is not model compact: considering the set $\Gamma = \{\langle a^*\rangle p, \neg p, \neg\langle a\rangle p, \neg\langle a\rangle\langle a\rangle p, \cdots\}$, any finite subset of Γ is satisfiable, yet not the whole Γ. However, Γ is satisfiable in the following AKM model:

$$\neg p \overset{a^*}{\rightharpoonup} p$$

8.5 Satisfiability

In this section, we turn to the satisfiability checking problem. The basic idea is to reduce this problem to traditional $\mathtt{PDL}_\Sigma$ satisfiability checking. However, clearly this can not be done in a straightforward way, since their semantics do not coincide, as observed in the previous section.

For technical convenience, let us consider the equivalent positive $\mathtt{PDL}^+_\Sigma$ language

$$\phi \quad ::= \quad \top \mid \bot \mid p \mid \overline{p} \mid \phi \wedge \phi \mid \phi \vee \phi \mid [\pi]\phi \mid \langle\pi\rangle\phi$$

where p and $\overline{p}$ (negation of p) are in a set *lit* of literals of basic propositions and $\pi \in Reg_\Sigma$. It is a standard exercise to transform a $\mathtt{PDL}_\Sigma$ formula to an equivalent $\mathtt{PDL}^+_\Sigma$ formula and vice versa.

Given a $\mathtt{PDL}^+_\Sigma$ formula ϕ, let $\langle\rangle_\phi$ be the set $\{\pi \mid \langle\pi\rangle$ appears in $\phi\}$. We now prove that if a formula is satisfiable then it is satisfiable in a certain class of models.

8.5.1. P . *Given a* $\mathtt{PDL}^+_\Sigma$ *formula* ϕ, ϕ *is satisfiable on an AKM* $\iff$ ϕ *is satisfiable in an AKM that only contains* π*-transitions for* $\pi \in \langle\rangle_\phi$.

P $\Leftarrow$ is straightforward. We now prove $\Rightarrow$:
Suppose there is an AKM $\mathcal{M} = (S, \Sigma, \rightarrow, V)$ such that $\mathcal{M}, s \vDash \phi$ for some $s \in S$. From proposition 8.4.2, $R(\mathcal{M}), s \vDash \phi$. Based on $R(\mathcal{M})$ we build the model $\mathcal{M}' = \{S, \Sigma', \rightarrow', V\}$ where:

$$\Sigma' = \langle\rangle_\phi \text{ and } s \overset{\pi}{\longrightarrow}' t \text{ in } \mathcal{M}' \iff s \overset{\pi}{\longrightarrow}_r t \text{ in } R(\mathcal{M}).$$

Namely we eliminate all the transitions in $R(\mathcal{M})$ except the ones labelled by some $\pi \in \langle\rangle_\phi$. We claim: $\mathcal{M}', s \vDash \phi$. We prove it by induction on the structure of ϕ :

- For atomic and boolean cases, trivial.
- $\phi = \langle\pi\rangle\psi$: since $R(\mathcal{M}), s \vDash \phi$ then $\exists t \in S$ such that $R(\mathcal{M}), t \vDash \psi$ and $s \xrightarrow{\pi}_r t$. By the definition of $\rightarrow'$, $s \xrightarrow{\pi}' t$. Now by the induction hypothesis we have $\mathcal{M}', t \vDash \psi$, thus $\mathcal{M}', s \vDash \phi$.
- $\phi = [\pi]\psi$: since $R(\mathcal{M}), s \vDash \phi$ then for all t such that $s \xrightarrow{\pi}_r t$, $R(\mathcal{M}), t \vDash \psi$. By the induction hypothesis, $\mathcal{M}', t \vDash \psi$. Note that if there exists t such that $s \xrightarrow{\pi_1}' \cdots \xrightarrow{\pi_n}' t$ in $\mathcal{M}'$, and $\mathcal{L}(\pi_1 \cdot \cdots \cdot \pi_n) \subseteq \mathcal{L}(\pi)$ then $s \xrightarrow{\pi}_r t$ in $R(\mathcal{M})$. Therefore for all $\pi-$reachable states t in S, $\mathcal{M}', t \vDash \psi$. It follows that $\mathcal{M}', s \vDash \phi$.

Given a $\mathtt{PDL}^+_\Sigma$ formula ϕ, we define a rewriting of ϕ, obtained by replacing every instance of π in $[\pi]\psi$ with its maximal $\langle\rangle_\phi$-rewriting $\widehat{\pi}_{\langle\rangle_\phi}$. Recall that $\widehat{\pi}_{\langle\rangle_\phi}$ is a regular expression over the alphabet $\Sigma_{\langle\rangle_\phi} = \{e_\pi \mid \pi \in \langle\rangle_\phi\}$ where each e_π is a new symbol.

8.5.2. D . **(Rewriting of a $\mathtt{PDL}^+_\Sigma$ formula)** Given a $\mathtt{PDL}^+_\Sigma$ formula ϕ, $\mathfrak{R}(\phi)$ is the rewriting of ϕ in the language $\mathtt{PDL}^+_{\Sigma_{\langle\rangle_\phi}}$ defined by:

- $\mathfrak{R}(p) = p$ where $p \in lit \cup \{\top, \bot\}$.
- $\mathfrak{R}(\psi_1 \wedge \psi_2) = \mathfrak{R}(\psi_1) \wedge \mathfrak{R}(\psi_2)$.
- $\mathfrak{R}(\psi_1 \vee \psi_2) = \mathfrak{R}(\psi_1) \vee \mathfrak{R}(\psi_2)$.
- $\mathfrak{R}(\langle\pi\rangle(\psi)) = \langle e_\pi\rangle\mathfrak{R}(\psi)$.
- $\mathfrak{R}([\pi]\psi) = [\widehat{\pi}_{\langle\rangle_\phi}]\mathfrak{R}(\psi)$.

8.5.3. T . *Given a $\mathtt{PDL}^+_\Sigma$ formula ϕ, ϕ is satisfiable on an AKM $\iff$ $\mathfrak{R}(\phi)$ is satisfiable on a Kripke model w.r.t. the standard $\mathtt{PDL}_\Sigma$ semantics.*

P
($\Rightarrow$:) Suppose ϕ is satisfiable, then from proposition 8.5.1, we know that ϕ is satisfiable in an AKM model $\mathcal{M}$ that only contains π-transitions for $\pi \in \langle\rangle_\phi$. Note that we can also treat $\mathcal{M}$ as a Kripke model over the action set $\Sigma_{\langle\rangle_\phi}$, which we denote by $\mathcal{G}$. Namely, $\mathcal{G}$ is the same as $\mathcal{M}$ except that the transition is renamed. Assuming $\mathcal{M}, s \vDash \phi$, we now show $\mathcal{G}, s \Vdash \mathfrak{R}(\phi)$ by induction on the structures of $\mathfrak{R}(\phi)$:

- For atomic and boolean cases, trivial.
- Suppose $\phi = \langle\pi\rangle\psi$ thus $\mathfrak{R}(\phi) = \langle e_\pi\rangle\mathfrak{R}(\psi)$, where $\pi \in \langle\rangle_\phi$. Since $\mathcal{M}, s \vDash \phi$, there exists some $s \xrightarrow{\pi} s'$ with $\mathcal{M}, s' \models \psi$. According to our construction, in $\mathcal{G}$, $s \xrightarrow{e_\pi} s'$. By the induction hypothesis, $\mathcal{G}, s' \Vdash \mathfrak{R}(\psi)$ in $\mathcal{G}$. It follows from the standard semantics of $\mathtt{PDL}_\Sigma$ that $\mathcal{G}, s \Vdash \mathfrak{R}(\phi)$.

- Suppose $\phi = [\pi]\psi$ thus $\mathfrak{R}(\phi) = [\widehat{\pi}_{\langle\rangle_\phi}]\mathfrak{R}(\psi)$. Since $\mathcal{M}, s \models \phi$, for any sequence of transitions $s \xrightarrow{\pi_1} \cdots \xrightarrow{\pi_n} s'$ with $n \geq 0$, $\mathcal{L}(\pi_1 \cdots \pi_n) \subseteq \mathcal{L}(\pi)$ implies $\mathcal{M}, s' \models \psi$. Now let us consider any sequence of transitions $s \xrightarrow{e_{\pi'_1}} \cdots \xrightarrow{e_{\pi'_m}} s_m$ with $e_{\pi'_1} \cdots e_{\pi'_m} \in \mathcal{L}_{\langle\rangle_\phi}(\widehat{\pi}_{\langle\rangle_\phi})$ in $\mathcal{G}$. It is clear that $exp_{\mathbf{\Sigma}}(e_{\pi'_1} \cdots e_{\pi'_m}) \subseteq \exp_{\mathbf{\Sigma}}(\widehat{\pi}_{\langle\rangle_\phi})$. Since $\widehat{\pi}_{\langle\rangle_\phi}$ is a $\langle\rangle_\phi$-rewriting of π, we have:

$$\mathcal{L}(\pi'_1 \cdots \pi'_m) = exp_{\mathbf{\Sigma}}(e_{\pi'_1} \cdots e_{\pi'_m}) \subseteq \mathcal{L}(\pi)$$

Therefore $\mathcal{M}, s_m \models \psi$. By the induction hypothesis, $\mathcal{G}, s_m \Vdash \mathfrak{R}(\psi)$. It follows that $\mathcal{G}, s \Vdash \phi$.

($\Leftarrow$:) Suppose $\mathfrak{R}(\phi)$ is satisfiable at a pointed Kripke model $(\mathcal{G}, s)$ over action set $\mathbf{\Sigma}_{\langle\rangle_\phi}$ such that $\mathcal{G}, s \Vdash \mathfrak{R}(\phi)$. Clearly, we can construct a corresponding AKM $\mathcal{M}$ which is the same as $\mathcal{G}$ except that for any transition $e_\pi \in \mathbf{\Sigma}_{\langle\rangle_\phi}$ in $\mathcal{G}$, we rename the label e_π by π. We now show $\mathcal{M}, s \vDash \phi$ by the induction on the structure of ϕ:

- For atomic and boolean cases, trivial.
- Suppose $\phi = \langle\pi\rangle\psi$, where $\pi \in \langle\rangle_\phi$. Since $\mathcal{G}, s \Vdash \mathfrak{R}(\phi)$, namely $\mathcal{G}, s \Vdash \langle e_\pi\rangle\mathfrak{R}(\psi)$, there exists some $s \xrightarrow{e_\pi} s'$ in $\mathcal{G}$ with $s' \Vdash \mathfrak{R}(\psi)$. According to our construction, $s \xrightarrow{\pi} s'$ in $\mathcal{M}$. By induction hypothesis, $\mathcal{M}, s' \models \psi$. It follows from our semantics that $\mathcal{M}, s \vDash \phi$.
- Suppose $\phi = [\pi]\psi$: Since $\mathcal{G}, s \Vdash \mathfrak{R}(\phi)$, namely $\mathcal{G}, s \Vdash [\widehat{\pi}_{\langle\rangle_\phi}]\mathfrak{R}(\psi)$, we have $s \xrightarrow{e_{\pi_1}} \cdots \xrightarrow{e_{\pi_m}} s'$ and $e_{\pi_1} \cdots e_{\pi_m} \in \mathcal{L}_{\langle\rangle_\phi}(\widehat{\pi}_{\langle\rangle_\phi})$ implies $\mathcal{G}, s' \vDash \mathfrak{R}(\psi)$. Take an arbitrary t such that $s \xrightarrow{\pi'_1} \cdots \xrightarrow{\pi'_n} t$ in $\mathcal{M}$ and $\mathcal{L}(\pi'_1 \cdots \pi'_n) \subseteq \mathcal{L}(\pi)$. Since $\widehat{\pi}_{\langle\rangle_\phi}$ is the maximal $\langle\rangle_\phi-$rewriting of π, from Proposition 8.2.4 we have $e_{\pi'_1} \cdots e_{\pi'_n} \in \mathcal{L}_{\langle\rangle_\phi}(\widehat{\pi}_{\langle\rangle_\phi})$. Hence $\mathcal{G}, t \Vdash \mathfrak{R}(\psi)$. By the induction hypothesis, $\mathcal{M}, t \vDash \psi$. Therefore $\mathcal{M}, s \vDash \phi$.

⋓

8.5.4. R . This result is somewhat surprising. Note that our semantics and traditional $\mathtt{PDL}_{\mathbf{\Sigma}}$ semantics differs as shown in the previous section. However, they coincide after the rewriting. For example, $\phi = \langle a \cdot b\rangle p \wedge [a][b]\neg p$ is satisfiable w.r.t our semantics, but not in standard PDL, while $\mathfrak{R}(\phi) = \langle e_{a\cdot b}\rangle p \wedge [\mathbf{0}][\mathbf{0}]\neg p$ is satisfiable in traditional PDL semantics, where constant $\mathbf{0}$ denotes the empty language.

From Theorem 8.5.3, we can reduce satisfiability checking of $\mathtt{PDL}_{\mathbf{\Sigma}}$ over AKM to standard $\mathtt{PDL}_{\mathbf{\Sigma}}$ satisfiability checking, which has been extensively studied in the literature (see, e.g., [HKT00]) and is known to be E -complete. Note that the regular expression rewriting can be done in 2-E as in Theorem 8.2.6. These entail that the satisfiability checking of $\mathtt{PDL}_{\mathbf{\Sigma}}$ over AKM can be done in 3-E [5].

[5] The length of the output of a 2-E algorithm is essentially at most doubly exponential of the size of the input. A moment of reflection should confirm the desired complexity.

8.6 Conclusion and Future Work

We have performed a thorough study of $\mathtt{PDL}_\Sigma$ over accelerated labelled transition systems. We investigated three problems: model checking, axiomatization and satisfiability checking. We showed that the model checking problem of this logic is E -complete while the program complexity turns out to be N -complete. This answers an open question in [EvdP06]. We also provided a sound and complete axiomatization for $\mathtt{PDL}_\Sigma$ which involves Kleene Algebra as an Oracle. Furthermore, we show the satisfiability problem is decidable in 3-E by giving a reduction to the satisfiability of $\mathtt{PDL}_\Sigma$ w.r.t. the standard $\mathtt{PDL}_\Sigma$ semantics on Kripke models.

There are many avenues for future study. First, although we conjecture that our reduction method is optimal, the exact complexity of the satisfiability problem is left open. In [CvdPW08], we claimed the satisfiability problem is E -complete. However, the argument was, in retrospect, based on a misunderstanding of Theorem 8.2.6. There are a number of extensions of $\mathtt{PDL}_\Sigma$ (e.g. the test operator) and we are interested in what will happen if the accelerated transitions are labelled by expressions containing extra operators. Furthermore, some open problems remain in applying AKM to abstract model checking of liveness properties, as sketched in [EvdP06]. For instance, how can an abstraction with accelerated transitions be computed automatically? [EvdP06] hints at the relation to automated termination provers. Our study shows that the model checking problem with accelerated transitions is hard. So another interesting question is how to add the minimal number of accelerated transitions, in order to prove a certain liveness property.

Part IV

Modelling Security Protocols

Chapter 9

Epistemic Approaches to Security Protocol Verification

9.1 Knowledge in Security Protocols

Security protocols are rules (often based on cryptography) that govern communications in hostile environments in order to guarantee certain security goals. Many such goals are *naturally expressed* in terms of knowledge: only the right agents should get to *know* the right things. This has to do with the fact that many security properties are about hiding information from the *bad guys* or making sure the *good guys* get their information. For example, here are some intuitive epistemic readings of the security properties mentioned in [RS01]:

- *Sender authentication*: the receiver *knows* the sender of a message;
- *Mutual authentication*: both parties (commonly) *know* they are talking to each other;
- *Anonymity*: the sender is *unknown* (to an eavesdropper);
- *Secrecy*: an intruder does not *know* certain information.

More specifically, in the area of voting protocols, which recently drew much attention, more involved properties are considered, for example (cf. [DKR07]):

- *Vote-privacy*: nobody other than the voter herself *knows* that a particular voter voted in a particular way;
- *Receipt-freeness*: a voter does not gain any information (a *receipt*) which can be used to let another *know* for sure that she voted in a certain way.
- *Coercion-resistance:* a voter cannot cooperate with a coercer to let him *know* that she voted in a certain way.

The above list is only indicative, and by no means exhaustive to cover the security properties that have epistemic readings. Although the precise formal meaning of the security properties as above is debatable, the relevance of epistemics in such settings is undeniable (cf. also [Kra07] (Slogan 8): "The purpose of a cryptographic protocol is to interactively compute, via message passing, knowledge of the truth of desired and, dually, knowledge of the falsehood of undesired cryptographic states of affairs").

However, security protocols are deceptively simple-looking objects with very subtle behaviours, which require extremely precise formal analysis. Designing a correct protocol can be thought of as *programming Satan's computer* as [AN95] put it. Consider the 3-line Needham-Schroeder authentication protocol [NS78]:[1]

$$\begin{array}{lll} 1. & \mathtt{A} \to \mathtt{B} & : \{\mathtt{n_A}, \mathtt{A}\}_{\mathtt{PK_B}} \\ 2. & \mathtt{B} \to \mathtt{A} & : \{\mathtt{n_A}, \mathtt{n_B}\}_{\mathtt{PK_A}} \\ 3. & \mathtt{A} \to \mathtt{B} & : \{\mathtt{n_B}\}_{\mathtt{PK_B}} \end{array}$$

which prescribes a set of action patterns with *roles* of agents to authenticate two agents with each other. BAN logic provided a correctness proof of the above protocol, which was later proven flawed due to a man-in-the-middle attack [Low96]:

$$\begin{array}{llllll} 1 & A \to I & : \{n_A, A\}_{PK_I} & & & \\ 1' & & & I(A) \to B & : & \{n_A, A\}_{PK_B} \\ 2' & & & B \to I(A) & : & \{n_A, n_B\}_{PK_A} \\ 2 & I \to A & : \{n_A, n_B\}_{PK_A} & & & \\ 3 & A \to I & : \{n_B\}_{PK_I} & & & \\ 3' & & & I(A) \to B & : & \{n_B\}_{PK_B} \end{array}$$

where A, B, I are concrete agents playing different roles according to the specification of the protocol. After A contacts I, the intruder I can pretend to be A towards B by forwarding A's special number to B. After B's reply, I can use A to obtain B's number and confirm B according to the protocol. Thus B may believe he is talking to A while in fact he is talking to I.

The lack of a proper semantics for its epistemic language and its high level reasoning limit the value of correctness proofs in BAN-logic. This proves the need for a closer look at the meaning of knowledge and the cryptographic operations used in security protocols.

9.1.1 Different Aspects of Knowledge

As an appetizer, consider the property of *Secrecy*:

*"an intruder does not know certain **information**".*

[1] A generates a random number (a *nonce*), and then sends it to B in a "locked box" that only B can open with his private key. B then sends A's number back with a random number of his own, in a box that only A can open. A then confirms by sending B his number back. The intended goal is that both A and B know that they are talking to each other.

If the information concerned is a bit string s (a *piece of information*), then to *know* it amounts to possessing this piece of bit string, while s itself does not have any truth value. On the other hand, if the information concerned has the form *"it was B who sent the message"* (call it ϕ), then to *know* ϕ means knowing the fact that *it was B who sent the message*, or in other words knowing that the *proposition* ϕ is true. Clearly, the same word *knowledge* can be used for different aspects of what there is to learn. We will, following [RS05b], refer to the first type of knowledge (in the sense of possession of bit strings) as *knowledge of explicit data*,[2] and to the second type of knowledge as *propositional knowledge*.

More subtleties regarding knowledge in a security context, are related to the cryptographic operations used in the security protocols. First of all, based on the bit strings that agents possess and the cryptographic operations available, they can *know* more bit strings by constructing complex message terms from what they possess, or decomposing a composed one into simpler ones. Such knowledge, in terms of possession of bit strings obtained by cryptographic operations, can be classified as *algorithmic knowledge* [HP03]. More intricately, if an agent A does not possess the symmetric key k, then the encrypted message of m by k (denoted by $\{m\}_k$) should mean no more than a random bit string to A, even though she possesses it. Thus we need a notion of knowledge to denote that an agent can *see* the inherent structure of bit strings. We call the last type *"certain knowledge"* following [BRS07].

These different ways of using the term "knowledge" (and the verb 'to know') suggest different structures and treatments in the formal models, which we will discuss in Section 9.2.2.

9.1.2 Tension Between Epistemic and Temporal Structure

Despite the epistemic flavour in expressing security goals, the interchange of messages, which constitutes protocols, occurs over *time*. Thus a rigourous epistemic approach to security protocol verification needs to harmonise the epistemic and temporal aspects. However, the intuition about the expressivity of epistemic logics does not quite coincide with the practice of security protocol verification so far: most of the successful approaches usually model the protocols formally with purely temporal structures, and try to capture the properties in a temporal formalism (cf. e.g.,[RS01, AF01, FGM04]). The tension between the natural temporal essence of the formal model of protocols, and the natural epistemic formalisation of the security requirements has proven to be a challenge. This raises two natural questions:

1. Does introducing epistemics into the language indeed boost the expressive power in formalising security properties?

2. What is the computational cost of combining epistemic and temporal aspects in security protocol verifications?

[2] [Kra07] uses the term "individual knowledge" for this.

Fortunately, recent years have seen a growing interest in epistemic approaches connected to the study of certain security properties that are not easily expressed in terms of events which did or did not happen along a *single* run of the protocol. A list of such properties includes, for example, anonymity [SS99, HO05], receipt-freeness [JdV06, JP06, BRS07], and coercion-resistance [DKR06, DKR07]. The verification of such properties depends on whether agents are able to distinguish between different courses of events, which is exactly the idea behind the standard Kripke semantics of knowledge. Formally, this involves the addition of equivalence relations to the temporal model, where it is useful, natural or even necessary as we will argue in Section 9.4.

Moreover, despite the apparent disguises of the formalisations, the epistemic logical approaches proposed by different research communities do have some important common features, where careful comparisons are needed to pinpoint the differences. Our goal of this chapter is two-fold. First, we give a brief overview of several epistemic proposals in Section 9.2 and compare the essential techniques they employ in Section 9.3. The survey in these sections is intended to be an introduction to this developing field of epistemic verification. Second, in Section 9.4, we try to give partial answers to the questions we proposed above. The survey will be presented mostly in a high level fashion and will only get to some technical details in Sections 9.3 and 9.4 when truly necessary.

While we intend this chapter to give a brief overview of approaches to modelling knowledge in the analysis of security protocols, we cannot cover all different aspects. The focus in this chapter will be on model checking approaches to verification, based on modal logics of *knowledge* rather than belief, that are *possibilistic* rather than probabilistic. For those interested in the other aspects, our introductory text in Subsection 9.2.1 contains pointers to some work in the areas outside of our focus.

9.2 Epistemic Approaches: A Brief Survey

9.2.1 BAN logic

The starting point of formal verification of security protocols is often attributed to the development of BAN-logic [BAN89], named after its inventors Burrows, Abadi and Needham. The syntax of this logic includes predicates of *belief*[3] and *actions*, thus it is able to express message passing actions and security goals. In fact, BAN-logic presents a calculus (proof system) by giving a number of inference rules to derive statements. For example, here is a rule for "if A believes he shares a secret key k with B, and A has received a message X encrypted with k, then A believes that it was B who sent the message":

$$\frac{A \textit{ believes } (A \overset{k}{\leftrightarrow} B), \quad A \textit{ sees } \{X\}_k}{A \textit{ believes } (B \textit{ said } X)}.$$

[3]However, it is essentially *knowledge*, following the intuition given by the authors.

To handle protocols in this framework, the protocol first needs to be *idealised*, then the initial assumptions are spelled out in the BAN-language, after which each step in the protocol is annotated with a BAN-formula asserting the state of affairs after that step. The statement after the final step describes the outcome of the protocol. The goal of the analysis is to derive a final assertion that implies the protocol is correct.

However, the *soundness* of the inference rules in the BAN-approach was questionable due to the lack of a formal semantics and clear underlying assumptions of the "idealisation" which led BAN-logic to an abstraction level too high to capture the consequences of all the possible intruder behaviours[4]. These drawbacks made the BAN-logic analysis of the Needham-Schroeder authentication protocol overlook the possibility of the man-in-the-middle attack exposed by Lowe [Low96], who used a process theoretic analysis in the process algebra CSP [BHR84]. At the same time, model checking approaches [CGP99] began to flourish and later became prominent.To do model checking on security protocols with epistemic logic, it is necessary to have a suitable formal semantics for knowledge in the security setting.

Despite the efforts made in the literature [GNY90, Bie90, Syv92], the main hurdle to a reasonable semantics of BAN-like logics was the so-called *logical omniscience* problem, an inherent issue of the standard possible-world semantics of epistemic logics [VW51, Hin62]: agents know all the valid propositions and all logical consequences of what they know. According to the Kripke semantics, if a message m is indeed of the form $\{m'\}_k$ (thus $m = \{m'\}_k$ is true everywhere in the model), then an agent knows it, even when she does not possess the key k. This sounds contradictory to our intuition in security analysis.

Many approaches have been suggested to avoid the logical omniscience problem (see [FHMV95] [Ch.9] and [HP10a] for surveys). In the context of security analysis, the most relevant one is the approach of algorithmic knowledge [HMV94], which is prominent in our later introduction of various epistemic approaches. The idea is that an agent *knows* a message term only if it is derivable by some algorithm with respect to a deductive system capturing idealised cryptographic operations [HP03, Puc06]. For propositional knowledge, a more sophisticated way of avoiding the logical omniscience problem can be obtained by deviating from the standard Kripke semantics in the definition of reachable possible worlds, as demonstrated in [CD05b, CD07]. Essentially, such an approach introduces extra possible worlds which may not be in the model when evaluating epistemic formulas. Awareness can also be used to deal with logical omniscience in the security setting (for instance, see [ABV03]), but we will not elaborate on this here.

Before moving on from BAN to the modern model checking epistemic approaches, we should mention that several authors have proposed analyses for security protocols involving belief rather than knowledge, e.g. [HD07, vdMW07, BS08a]. Also, the epistemic approaches that we survey are *possibilistic* in the sense that an agent knows a fact if he does not consider it possible to be false, while in certain security contexts this may be inappropriate. For example, can we rightfully say that A anonymously

[4] See [Tee06] for a more elaborate discussion on the soundness of BAN-logic.

sent a message, if A is the sender of the message in 99 out of 100 runs considered possible? This suggests a *probabilistic* approach to knowledge or belief to analyse certain security properties, as in [RR98, SS99, HO02, HO05, Shm04, BP05]. These *doxastic* and *probabilistic* approaches are not covered in our survey.

9.2.2 Basics of Epistemic Approaches

In this section, we will list the commonly used components of most epistemic approaches in the post-BAN era. We first need a logical language $\mathcal{L}_{\mathbf{I}}$ to specify properties of models, where $\mathbf{I}$ is a (finite) set of agents. Due to the fact that we are talking about message passing in a protocol setting, we need to mention messages in our language. This is often done by introducing the message terms as follows:

$$m ::= \quad c \quad | \quad k \quad | \quad \{m\}_k \quad | \quad (m, m')$$

where c stands for some basic plain terms which may in general have many sorts (e.g., names, integers, etc.), $\{m\}_k$ is the encryption of m with key k, and (m, m') intuitively represents pairing of m and m'. In general, arbitrary cryptographic operations f can be introduced in this way.

Associated with the message terms there is a derivation system to capture the cryptographic functions in the message terms [Pau97, Pau98, CJM98]. For example the following derivation rules capture the symmetric encryption and pairing of the messages:

$$\texttt{synth}: \frac{m \quad m'}{(m, m')} \quad \frac{m \quad k}{\{m\}_k} \qquad \texttt{analz}: \quad \frac{(m, m')}{m} \quad \frac{(m, m')}{m'} \quad \frac{\{m\}_k \quad k}{m}$$

where `synth` rules govern the application of cryptographic operations to form new terms from the old, while `analz` rules intuitively extract information from complex terms. We can alternatively represent `analz` rules by an equational theory E, e.g., $\texttt{dec}(\texttt{enc}(x, y), y) = x$ for the last rule above, if `dec`, `enc` are introduced as cryptographic operations with the obvious meaning in the language of message terms. Given a set of messages M, we say $M \vdash m$ if either $m \in M$ or m is *derivable* from M by applying the rules. We write $m =_E m'$, if $m = m'$ is an instantiation of an equation induced by E. [HP03] argues that a derivation system may not be convenient to model certain powerful adversary operations, and proposes to use arbitrary algorithms instead of derivation systems. For simplicity, we will not cover this more general case here.

We build formulas based on message terms which are not formulas themselves. Following the observations in Section 9.1.1, we need different knowledge operators in the language to cope with various types of knowledge:

1. *Knowledge of explicit data* (possession of bit strings): We build basic propositions in the shape of $has_i m$, where m is a message term, meaning that agent i *possesses* m. For such knowledge we have the *de dicto* reading: $has_i\{m\}_k$ means that the bit

string of $\{m\}_k$ is possessed by i. However, i may be unsure about the structure of the message.

2. *Algorithmic knowledge* (possession of derivable bit strings): In the literature, the knowledge of explicit data can be viewed as a special case of algorithmic knowledge. We can use $\overline{has}_i m$ to express that m as a bit string can be derived from the information agent i possesses, by applying corresponding cryptographic operations modelled by `synth` and `analz` rules (see [HP03] and [RS05b] for the detailed rationale)[5].

3. *Propositional knowledge* (what facts are known to the agents): As in the standard epistemic logic, we use $K_i\phi$ to express that "agent i knows that ϕ is true." Thus the logical language $\mathcal{L}_{\mathrm{I}}$ may look like:

$$has_i m \quad | \quad \overline{has}_i m \quad | \quad \phi \wedge \psi \quad | \quad \neg\phi \quad | \quad K_i\phi \quad | \quad O\phi$$

where $m \in M$, and O can be any modal operator other than K_i, depending on what properties we want to specify. On the other hand, given an existing modal logic language, we can turn it into a language about message passing by adding epistemic operators and taking $has_i m$ as the basic propositions[6].

4. *Certain knowledge* (the understanding of the bit strings). This kind of knowledge sits in between algorithmic knowledge and propositional knowledge, since it is not only about message terms itself but also about the observational power of agents [BRS07]. We may use $K_i has_i m$ to express that agent i knows that m is of certain structure, e.g., $K_i has_i \{c\}_k$ means i knows that he has a bit string which stands for $\{c\}_k$.[7] Thus knowledge operator K_i induces somehow a *de re* reading of $has_i m$.

To evaluate the basic formulae in the shape of $has_i m$ on Kripke models, we need to associate a set of message terms for each i at each state. Then $has_i m$ is true at a state s if m is in the set of messages associated with i on s. The semantics of $\overline{has}_i m$ is also straightforward by considering the derivable messages at a state.

According to the standard Kripke semantics, $K_i\phi$ is true at a state if ϕ is true anywhere reachable from the current state. The equivalence relations naturally model the epistemic uncertainties of agents. Thus the actual formal meaning of propositional knowledge and certain knowledge depends on the definition of the equivalence relation in the model and the message terms possessed by agents at various states. We will compare different equivalence relations in Section 9.3.1.

Given an epistemic language in the above style, an epistemic verification framework should give a general way to build up models from a protocol description in

[5] Here the "overline" in $\overline{has}_i m$ shows that m is in the *closure* of derivation.

[6] In addition to $has_i m$, it is also common to introduce special propositions to denote the actions happened in the past, e.g., $send^i_j(m)$ (see, for instance [HP03]).

[7] Different semantics for K_i operator may cause subtly different readings for such statements. We will see different semantics in Section 9.3.1.

order to do model checking. Two approaches are discussed in the next subsections following the traditions of Epistemic Temporal Logic and Dynamic Epistemic Logic.

9.2.3 Epistemic Temporal Approaches

To ease the exposition we now equip the interpreted systems defined in Definition 2.3.1 with explicit events. As usual, given a set of agents $\mathbf{I}$ with ϵ for the environment and the sets of local states $L_1, \ldots, L_n, L_\epsilon$, a set S of global states is a subset of $L_\epsilon \times L_1 \times \cdots \times L_n$. Given a set of events E and a set of global states S, we associate with each $e \in E$ a transition relation $\xrightarrow{e} \subseteq S \times S$. An infinite *run* r on S is a function $r : \mathbb{N} \mapsto S \times E$. Let $r_S(u)$ and $r_E(u)$ be the corresponding global state and event (to happen) at the uth point of the run r respectively. We say a run r is *admissible* if $\forall u \geq 0 : r_S(u) \xrightarrow{r_E(u)} r_S(u+1)$. An interpreted system $\mathcal{I}$ is then defined as a pair (R, V) where R is a set of admissible runs, and $V : S \mapsto 2^{\mathbf{P}}$ is a valuation function assigning to each proposition atom in $\mathbf{P}$ a truth value. We denote by $(\mathcal{I}, r, u)$ the point $r(u)$ in interpreted system $\mathcal{I}$.

To verify a protocol in the presence of an adversary, [8] we need to formalise the protocols and the adversary model, describing the possible actions of an adversary. Here we show an example of a formalisation of the Needham-Schroeder authentication protocol mentioned in Section 9.1, with the Dolev-Yao adversary model [DY83] where all the messages are delivered via the intruder role (`E`) acting as a *buffer* (see, e.g., [Cre06] for rationale):[9]

for A :	1.	A	send	E	:	$\{n_A, A\}_{PK_B}$	for B :	1.	B	rec	E	:	$\{n_A, A\}_{PK_B}$
	2.	A	rec	E	:	$\{n_A, n_B\}_{PK_A}$		2.	B	send	E	:	$\{n_A, n_B\}_{PK_A}$
	3.	A	send	E	:	$\{n_B\}_{PK_B}$		3.	B	rec	E	:	$\{n_B\}_{PK_B}$

Here the action patterns in a protocol are broken down and grouped into *local protocols* by roles. Note that, in the above formalisation, the intruder implicitly *eavesdrops* on all the messages and the agents will accept any message that the intruder may possess, as long as it is in the forms specified (thus modelling the intruder's ability to *manipulate* messages).

Despite differences in details in each specific framework, e.g. [HP03, vdMS04, RS05b, BCL09], we can summarise the merit of the general `ETL` approach for modelling protocols under an adversary model, as the following steps.

Step 1. Suppose the set of agents is $\mathbf{I} = \{1, 2, \ldots, n, \epsilon\}$, where ϵ indicates the intruder. We start from a set S_0 (usually a singleton) of initial states which are tuples of local states $\langle l_1, \ldots l_n, l_\epsilon \rangle$. An initial local state for agent i should, among other things, encode a set of message terms representing the messages that agent i initially

[8] Sometimes one intruder is enough, and we can give a small finite bound on the number of other agents, see, for instance, [LC03].

[9] For simplicity, we do not go into the details of the various specification languages and adversary models proposed in the literature. For example, [HP03] provide the possibility of modelling different adversaries in the IS-framework.

possesses (i.e. the *information states* of agents [RS05b]). In such a setting, we can retrieve the information state of i at global state s by $info_i(s)$. We can then define the semantics of $has_i m$ and $\overline{has_i m}$ at $(\mathcal{I}, r, u)$ by $info_i(r_S(u))$ in a straightforward way.

Step 2. We can generate a temporal structure, based on the initial states, by collecting all the admissible sequences of global states according to the protocol under the adversary model. The protocol specification and the adversary model define a set of events (instantiated action patterns). To give the transition relation $\xrightarrow{e}$ for the events on the global states, we can give each event e a precondition and a postcondition. The first specifies when the event can happen and the latter one changes the local states of agents to model information updates by the events. In the above example, an instantiated action: $(j \ \texttt{send} \ \epsilon : \{n_j, j\}_{PK_i})$ has the precondition that $\{n_j, j\}_{PK_i}$ is in the current information set of j and the postcondition that $\{n_j, j\}_{PK_i}$ is added to the information state of the intruder. In general, agents can send a message only if they possess it, and the effect of a `send` action is that the message is delivered to the intruder (under the Dolev-Yao model). The order of the actions according to the protocol can be encoded also by preconditions requiring that a certain action happened in the earlier stage of the run. We call the resulting set of runs the *generated temporal structure* $T(S_0)$.

We choose to let each e be *observable* to an agent i iff i herself is involved, e.g., $(j \ \texttt{send} \ \epsilon : m)$ is only observable by ϵ and j. Similarly, the i-observable subsequence of $(j \ \texttt{send} \ \epsilon : m)(i \ \texttt{rec} \ \epsilon : m')$ is $(i \ \texttt{rec} \ \epsilon : m')$. In the Dolev-Yao setting we presented above, the intruder can observe all the events. In a more sophisticated analysis, the events are composed by synchronising local events with respect to each agent according to their local protocols, cf. e.g., [BCL09].

Step 3. From $T(S_0)$, we build up the epistemic temporal model $E(T(S_0))$ by defining epistemic relations $\sim_i$ between points $(T(S_0), r, k)$. The standard way of defining $\sim_i$ in IS is by matching local states of i, or local views of i of the histories of events. However, the information sets and local histories in the protocol setting do not capture how the messages are understood by the agents (recall what we called *certain knowledge*, Section 9.2.2). It is possible that two message terms are different, but still *regarded* as the same by an agent e.g., events $\texttt{rec} : \{m\}_k$ and $\texttt{rec} : \{m'\}_k$ are not distinguishable to an agent who does not have the key k. Moreover, if an agent later obtains the key k, then she can tell $\{m\}_k$ and $\{m'\}_k$ apart by *"looking back with a fresh eye"*. Thus we need to build $\sim_i$ on some sophisticated equivalence relation on messages ($\approx$). In Section 9.3.1, we will discuss different existing definitions for $\approx$ on *lists* of message terms, since we usually assume that the agents can remember the order of the messages passing actions that she can observe.

It is not hard to see that we can lift $\approx$ to equivalence between points in an IS. Suppose each information set is represented by a list of messages. Let $M_i(e_0, \ldots, e_u)$ be the list of messages occurring in i's observable subsequence of events in $e_0, \ldots, e_u$. Two obvious possibilities are:

- *Asynchronous*: $(s_0 \xrightarrow{e_0} s_1 \ldots s_{u-1} \xrightarrow{e_{u-1}} s_u) \sim_i (s'_0 \xrightarrow{e'_0} s'_1 \ldots s'_{u'-1} \xrightarrow{e'_{u'-1}} s'_{u'})$ iff $info_i(s_u) \approx$

$info_i(s'_{u'})$.[10]

- *Synchronous*:$(s_0 \xrightarrow{e_0} s_1 \dots s_{u-1} \xrightarrow{e_{u-1}} s_u) \sim_i (s'_0 \xrightarrow{e'_0} s'_1 \dots s'_{u'-1} \xrightarrow{e'_{u'-1}} s'_{u'})$ iff $u' = u$ and $\langle info_i(s_0), M_i(e_0, \dots, e_{u-1})\rangle \approx \langle info_i(s'_0), M_i(e'_0, \dots, e'_{u-1})\rangle$.

The above procedure can be summarised with the slogan:

First temporal then epistemic.

Notably, [BCL09] presents a fully automated method to generate interpreted systems from formal specification of protocols taking many small details into consideration. Other methods to generate IS-like models include process algebra with epistemic annotations, e.g., [DMO07], which makes use of an operational semantics to generate the model from the protocol specified in process algebra terms.

9.2.4 Dynamic Epistemic Logic Approaches

As we have demonstrated in the previous chapters, DEL can be applied in modelling what agents learn through different communication acts according to epistemic reasoning, for example in the Russian Cards scenario discussed in Chapter 3. Thus it looks promising to analyse security protocols by modelling protocols in terms of action models. In [HMV05], [VO07], and [DW07] the first attempts were made towards the security protocol verification by DEL. Note that, security protocols are much more complicated than the epistemic protocols discussed in Chapter 3 and Chapter 4, thus to model such protocols, more general event models of DEL are needed rather than atomic actions or public announcements only. We summarize the modelling steps as follows based on the above attempts:

Step 1. We start with a finite initial static model $\mathcal{M}$ with epistemic relations $\sim_i$ ready. Similar as in the interpreted system approach, a state is associated with a tuple of information sets modelling the messages that agents possess. The epistemic relations can be given similarly according to the equivalence $\approx$ on lists of messages.

Step 2. We need to build an event model $\mathcal{A}$ which captures all the protocol actions with suitable pre- and postconditions similar to what we described at step 2 for ETL approaches. For example, to model the Needham-Schroeder authentication protocol mentioned above, we can build action model $\mathcal{A} = (E, \{\asymp_i\}_{i\in\mathbf{I}}, Pre, Pos)$ such that E includes all instantiated actions of the protocol, for example: event $e = (j \texttt{ send } \epsilon : \{n_j, j\}_{PK_i})$ with $Pre(e) = \overline{has}_j(\{n_j, j\}_{PK_i})$ and $Pos(e)(has_\epsilon(\{n_j, j\}_{PK_i})) = \top$. The epistemic relations $\asymp_i$ between events can be generated by lifting $\approx$ on lists of messages to events, under the constraint that an agent can always distinguish the events that she is involved in from other actions.

Step 3. The update execution computes the result of performing $\mathcal{A}$ on $\mathcal{M}$ iteratively, thereby it essentially builds up all the possible runs of the protocol[11].

[10] This is an example of asynchronous and *forgetful* agents [SG02], other memory conditions can be applied here.

[11] In [DW07], we introduced the iteration operation on event models in a DEL language which is similar to the one we presented in Chapter 3, but with public announcements replaced by event models.

The above procedure can be summarised as the slogan:

First epistemic then temporal.

Although it seems that DEL modelling is very similar to ETL modelling, we will pinpoint the tricky differences between the two approaches in details in Section 9.3.2.

9.2.5 Tools

In the last decade, many tools have been developed to handle formal verification in the setting of ETL or DEL, with potential application in security analysis. For ETL model checking, we have MCK: Model Checking Knowledge [GvdM04, vdMS04] and MCMAS: Model Checker for Multi-Agents Systems [LR06b, LQR09]. [BCL09] recently presented a fully automatic translation from protocol descriptions given in CAPSL (Common Authentication Protocol Specification Language) into the input language for MCMAS, enabling the automated checking of the security protocols from the Clark-Jacobs security protocol library by means of epistemic temporal logic. For DEL model checking, we have DEMO: Dynamic Epistemic MOdelling [vE07] and LYS: a knowledge anaLYSis toolset [Orz05]. Other relevant tool sets include the ETL-model checker MCTK [Su04], the ATL-model checker [AHM+98], and the real-time system model checker [KNN+08].[12]

In the literature, various tools are presented with some case studies demonstrating how the framework can be applied. For these demonstrations, often well-known situations or protocols are chosen which require relatively small models. The *classic* examples in the epistemic verification demonstrations are the *Dining Cryptographers* protocol for anonymous broadcast [Cha88], the *Muddy Children* (see, e.g., [FHMV95]) for demonstrating the effect of (repetition of) public announcements, and *Russian Cards Problem* (see [vD03]) for secure public announcements. Such common examples facilitate comparisons of the modelling and efficiency among different tools based on different frameworks, see, for example [vDvdHvdMR06], which takes the Russian Cards problem as a test case for MCK, MCMAS and DEMO.

9.3 Comparisons

In this section we will compare more technical aspects of the approaches mentioned in the previous section. In the first part, we discuss the different versions of equivalence. In the second part, we compare the epistemic temporal approach with the dynamic epistemic one in the security setting.

9.3.1 On Equivalences

Some well known formal methods have been adapted or designed to include (trace) equivalences to deal with multi-trace security properties (e.g., *applied pi-calculus*

[12] This is definitely not a complete list, see [LP07] for a survey of symbolic model checking for ETL.

[AF01]). In this part, we focus on how the equivalence relations of agents are defined, based on the lists $\langle m_1, \cdots m_n \rangle$ that record the messages that an agent received in order. The rest of this subsection will be devoted to the comparison of the following equivalence relations:

- *simple deduction equivalence* $\approx_d$
- *pattern matching equivalence* $\approx_{pat}$ (in [AT91, AR02] and [BRS07]);
- *static equivalence* $\approx_s$ (defined in [AF01, AC04], and later used in [CDK09b, CDK09a]);
- *permutation equivalence* $\approx_{per}$ (in [CD05a, GHPvR05], and later used in [CD07, JP06]).

We assume there is a fixed equational theory E corresponding to the derivation system on terms of messages. Let $M = \langle m_1, \ldots, m_n \rangle$ and $M' = \langle m'_1, \ldots, m'_n \rangle$[13] then:

- $M \approx_d M'$ iff for all message terms m: $M \vdash m \iff M' \vdash m$.
- $M \approx_{pat} M'$ iff M and M' induce the same recognisable message patterns, i.e. for all j: $pat(m_j, M) = pat(m'_j, M')$, where $pat(m_j, M)$ is roughly the message term in which the unconstructable parts are replaced by an uninterpreted symbol $\Box$. For example:

$$pat(\{m\}_k, M) = \begin{cases} \{pat(m, M)\}_k & \text{if } M \vdash k \\ \Box & \text{otherwise} \end{cases}$$

  For formal details on various cryptographic operations we refer to [AR02, BRS07].

- $M \approx_s M'$ iff M and M' satisfy the same equality tests. Formally, defining $\sigma_M, \sigma_{M'}$ to be the substitutions replacing x_j with m_j and m'_j respectively, then $M \approx_s M'$ iff for any message terms with variables $t(x_1, \ldots, x_n)$ and $t'(x_1, \ldots, x_n)$:

$$\sigma_M(t) =_E \sigma_M(t') \iff \sigma_{M'}(t) =_E \sigma_{M'}(t').$$[14]

- $M \approx_{per} M'$ iff there is a permutation $\pi : M \to M'$ such that for all j: $\pi(m_j) = m'_j$ and $\pi(t(\bar{m})) = t(\overline{\pi(m)})$ for any message term with variables t and any suitable list $\overline{m}$ from $\{m \mid M \vdash m\}$. [CD05a] shows that $\approx_{per}$ is indeed an equivalence relation.

The relation $\approx_d$ is very fine (despite the fact it does not require a one-one correspondence of messages) and thereby assigns strong observational power to the agents:

[13] Note that the equivalences we consider here all respect the number of messages.

[14] Here we leave out the details about protected names in the original *frame* (our σ) in applied-pi calculus.

e.g. $M_1 = \langle \{c\}_k \rangle \not\approx_d M_2 = \langle \{c'\}_k \rangle$. It may only make sense to employ such an equivalence relation for the intruder if we need to guarantee extreme security. On the other hand $\approx_{pat}$ is rather coarse as it treats all the unreadable parts as the same: e.g. $M_3 = \langle \{c\}_k, \{c'\}_k \rangle \approx_{pat} M_4 = \langle \{c\}_k, \{c\}_k \rangle$ since $pat(\{c\}_k, M_3) = pat(\{c'\}_k, M_3) = pat(\{c\}_k, M_4) = \Box$.

Static equivalence is somewhere in between e.g., $M_1 \approx_s M_2$ but $M_3 \not\approx_s M_4$ since $\{c\}_k \neq_E \{c'\}_k$ but $\{c\}_k =_E \{c\}_k$. To relate $\approx_s$ and $\approx_{per}$, Cohen and Dam show that:

9.3.1. T ([CD07]). *For any lists of messages M and M' satisfying $|\{m \mid M \vdash m\}| = |\{m \mid M' \vdash m\}| = \omega$:*

$$M \approx_s M' \iff M \approx_{per} M'$$

where the cardinality condition allows us to permute all the non-derivable messages in M to the non-derivable messages in M'.

[PP07] pleads for a principled approach to model indistinguishability relations that is worth elaborating upon. They define two states to be indistinguishable for an agent if the agent can compute the same observations from both states. There observations can be considered as tests in the spirit of static equivalence. They generate relations on the basis of the computational power of the agents: taking Θ to be a set of observations θ (tests), and A an algorithm returning for each $\theta \in \Theta$ and M the answer "yes", "no" or "unknown" to the question whether θ holds at M, they let $M \approx_{\Theta,A} M'$ iff for all $\theta \in \Theta$ $A(\theta, M) = A(\theta, M')$. For example $\approx_{pat}$ can be reformulated as $\approx_{\Theta,A}$ where θ is built as follows:

$$t ::= x \mid c \mid k \mid \{t\}_k \mid (t, t)$$
$$\theta ::= has(t) \mid \exists x.\theta \text{ where the only free variable in } \theta \text{ is } x.$$

It is easy to see that θ expresses the pattern of a message. The corresponding algorithm A then takes a pattern and then try to match it in M. On the other hand, to have Θ define $\approx_s$, we at least need to introduce equality into the language of Θ. In fact, if we take Θ as a logical language then this proposal is actually asking for logical characterisations of different equivalence relations with corresponding "model checking" algorithms for Θ on M. As another example, a logical characterisation of $\approx_{per}$ is given in [CD07, Theorem 3].

Regarding the complexity of checking such equivalence, we should first note that the decidability of $M \vdash m$ can be encoded by the decidability of $\approx_s$ or $\approx_{pat}$. However, checking $\vdash$ can be undecidable [AC04] depending on the underlying derivation system. [BRS07] shows that when M is finite, a derivation system containing encryption and blind signature can be decided in PTIME . This implies the decidability of $\approx_{pat}$ according to the definition of $\approx_{pat}$ in [BRS07]. More general results in [AC04] show that when E is a *convergent subterm theory* that can cover many important cryptographic operations, both $\approx_s$ and $\vdash$ are decidable in PTIME .

9.3.2 ETL vs. DEL in Modelling

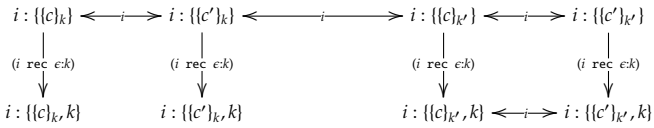
We now compare the epistemic temporal approach with the dynamic epistemic approach in modelling.

Limitations of DEL

As epistemic temporal logic and dynamic epistemic logic are two important methods of describing epistemic interaction over time, technical comparisons have been done to pinpoint the differences between the two. From an abstract point of view, ignoring the structure of the local states, an ETL-model is a tree-like Kripke structure with relations labelled by events and agent names. We get a similar structure, if we start from a static initial Kripke model, performing sequences of DEL-updates, and link each state and its update by the corresponding event ($s \xrightarrow{e} \langle s, e \rangle$).

Van Benthem et al. ([vBGHP09]) characterise the class of ETL tree-like structures that are DEL constructable by a uniform protocol in the above sense, by the notions of *Synchronicity* (agents are always aware if something has happened), *Perfect Recall* (the local history is remembered), *No Miracles*, and *Epistemic Bisimulation Invariance* (see below). This means that standard DEL can only deal with idealised agents who satisfy those properties. If we model intruders with enough observation power for better security, then Synchronicity and Perfect Recall can be intuitively assumed. However, No Miracles and Epistemic Bisimulation Invariance may lead to some drawbacks of DEL approaches in security verification:

No Miracles: An ETL-model $\mathcal{M}$, considered as a Kripke model with temporal action transitions $\xrightarrow{e}$ and epistemic relations $\sim_i$, has the property *No Miracles* if the following holds: *for all states s, s' and events e, e' such that $s \xrightarrow{e} t$ and $s' \xrightarrow{e'} t'$, for some t, t': if $s \sim_i s'$ and there are s'', s''' with $s'' \xrightarrow{e} t''$, $s''' \xrightarrow{e'} t'''$ for some $t'' \sim_i t'''$, then $t \sim_i t'$.* (If two actions lead to indistinguishable states somewhere in the model, then it cannot be the case that performing these actions on indistinguishable states will lead to distinguishable states.)
However consider the following (partial) model where $\sim_i$ denotes an equivalence relation based on $\approx_{pat}$:



where $k' \neq k$ and $c' \neq c$. Clearly, this model violates *No Miracle*, so it is impossible for it to be generated by the standard DEL approach. The problem is rooted in the definition of epistemic relations in the action models. Recall that the epistemic relations in the updated model are defined by the synchronisation of the epistemic relations in the static model and those in the action model. However, in the security protocol setting, the same receive action on indistinguishable states may cause the resulting states to be distinguishable, as the example shows. One way to go around this is to "*split*" each action into multiple *copies* with different preconditions such that different copies of the same action may be distinguished under different preconditions. For example, in the action model, the action (i `rec` $\epsilon : k$) with the

precondition $has_i\{c\}_k$ should be i-distinguishable from the same action with the pre-condition $has_i\{c'\}_k$, if $c \neq c'$. However, this ad-hoc method may introduce infinitely many copies of actions in the action model, which is not allowed by the standard DEL.

Epistemic Bisimulation Invariance requires the same event to happen at the states that are epistemically bisimilar (i.e. bisimulation disregarding the temporal relations). This is because the pre-conditions in the action model are formalised in the dynamic epistemic language. This may cause problems if we want to model protocol actions with temporal preconditions in terms of the past (for example, if i sends k only if j sent k'). The usual solution is to encode the history of actions by new basic propositions.

Limitations of ETL

According to the modelling procedure we described in Section 9.2.3, the epistemic relations are built after the temporal structures. This may prevent us from handling *knowledge-based protocols*, which have preconditions in terms of knowledge, e.g., i sends m only if i knows that j has k. As shown in [HF89, FHMV97], it is possible to construct the unique temporal structure and epistemic relations simultaneously according to a knowledge-based protocol, if the system is synchronous and the epistemic preconditions are not about the *future*.[15] On the other hand, DEL by definition can handle conditions about what *may* happen in the future, since in action models we can have preconditions like $K_i\langle\mathcal{A}, e\rangle\phi$ (i knows that e may happen and in that case ϕ will be true).

In the ETL modelling of security protocols, the initial (global) states represent the initial distribution of names, keys, and other messages. If we focus on a particular distribution, then we can start with a unique initial state. By doing so, we implicitly assume that the distribution of the information, e.g, who has what key, is commonly known [BRS07]. However, what if one agent is uncertain about whether another agent knows that she has a public key? Such higher order uncertainties are not well-handled if we generate epistemic relations between initial states based on matching local states. For example, suppose the only message term is a public key k and agent i has it while agent j does not. To make the formula $\phi = K_i(has_i k \wedge \neg has_j k) \wedge \hat{K}_i K_j has_i k \wedge \hat{K}_i \neg K_j has_i k$ true in an initial model, we need at least two states which represent the same initial distribution of messages, as the following model shows:

$$\begin{array}{ccc} i:\{k\}; j:\{\} & \overset{i}{\longleftrightarrow} & i:\{k\}; j:\{\} \\ \updownarrow j & & \\ i:\{\}; j:\{\} & & \end{array}$$

[15] As argued in [HF89], if a protocol has *"forward-looking"* conditions (like $K_i F\phi$: "i knows that ϕ will hold eventually"), it is circular to define the admissible runs uniquely. Therefore there may be none or several solutions to the fix-point-like definition of the admissible runs.

It is clear that ϕ holds at the upper two states. However, if the epistemic relations are generated by matching local states or other local information, then there should be a j relation linking all the states. But then formula $K_i \neg K_j has_i k$ will be true at the upper worlds, contradictory to our initial intention. In fact, if we want to handle higher order uncertainties by the generated epistemic relations, we need to introduce some extra *tokens* in the local states of j to distinguish the two upper states. Intuitively, a local state of one agent, though called *local*, should also contain information about one's opinion of others, in order to handle higher order uncertainties. However, it is rather ad-hoc to introduce those auxiliary tokens. On the other hand, DEL is more flexible in modelling how agents reason about each other, because the equivalences can be defined by choice. More flexibility is also offered by the possibility of modelling higher order uncertainties in the action models.

To summarize, the distinct features in either the DEL or ETL approaches are usually double-edged swords:

Features	ETL	DEL
Equivalence relations	generated by matching local information	generated by product update or by hand (for initial models)
Pros	flexible and automatic; generated in a distributed fashion	easy to handle higher order uncertainties; update mechanism is formally defined
Cons	inconvenient for higher order uncertainties at initial states	inconvenient in a cryptographic setting
Events	represented by transitions on global states	modelled in action models
Pros	flexible	pre- and postconditions are encoded in the DEL language thus easy to handle epistemic conditions in protocols
Cons	detailed modelling (e.g. pre- and postconditions) is outside the framework	equivalence relations between events are designed by hand

Based on the above observation, we may want to combine the two frameworks, as already attempted in [HY09, vBGHP09, Hos10, WSvE10]. Chapter 5 of this thesis also presents an effort to bring the distributed features of ETL to DEL modelling.

Compared to the ETL approach, the standard DEL approach has limitations in generating suitable equivalence relations in the security setting. On the other hand, as we demonstrated in Part I, DEL seems convenient for epistemic protocols where:

- preconditions are in terms of the knowledge of the agents;
- higher order uncertainties are crucial (e.g., higher order uncertainty about initial distribution of information or observation of actions);

- protocol goals are in terms of the *nested* knowledge form.

Epistemic protocols use epistemic reasoning rather than cryptography to obtain security. Examples of such protocols include e.g., Dining Cryptographers [CCD88] and Card Cryptography [FW96, vD03, vD08]. As we have shown in Chapter 3, to verify such protocols, meta-knowledge of the protocols themselves matters and creates some complications. It is not yet clear whether intruder's knowledge about the goal of a security protocol will also affect the verification result.

9.4 To Know or Not, Towards a Technical Answer

As emphasised in the previous sections, many epistemic approaches are motivated by a common conviction that epistemic logic can express security properties "more naturally". However, in practice, in most of the formal frameworks, security properties are formalised as temporal formulae rather than in terms of knowledge. To really justify the use of epistemics, it is crucial to understand better whether adding epistemics can indeed help to express more security properties, and if so, what the cost is for the improved expressivity.

9.4.1 On Expressivity of ETL

Aiming at a technical basis to answer the above questions, we formally compare the expressivity of epistemic temporal logic (ETL) versus pure temporal logic (TL) in the rest of this section. Here we regard ETL and TL as *classes* of logics: we do not fix the exact logic unless necessary. The comparison will always be between a temporal logic L and an epistemic temporal logic that extends L with epistemic operators.

A logic $\mathtt{L}_1$ is strictly *more expressive* than $\mathtt{L}_2$, if (1) for every formula in $\mathtt{L}_2$ there is a formula in $\mathtt{L}_1$ defining the same class of models (i.e. they have exactly the same models.); but (2) there is a formula in $\mathtt{L}_1$ which does not have a corresponding formula in $\mathtt{L}_2$. Note that the comparison of the expressivity of different logics is usually studied given the condition that the logics concerned are defined on the same type of models. However, in the case of ETL and TL, this condition does not hold: the models of ETL involve epistemic relations, while these are absent in the TL-models. This complicates formal comparisons of the two logics in terms of expressivity. To make the comparison of ETL and TL possible, we need to provide the common playground for these two logics.

A rather straightforward observation is that if we consider the epistemic relations of agent i to be just another kind of transitions, labelled 'i', then ETL can be "reduced" to TL. Let $\mathtt{CTL}^*_{\mathbf{I}}$ and $\mathtt{Mu}_{\mathbf{I}}$ be $\mathtt{CTL}^*$ and modal $\mu-$calculus with extra actions labelled by the names of agents in $\mathbf{I}$ respectively. Let $C^{\mathtt{ETL}}$ and $C^{\mathtt{TL}}$ be the classes of all ETL- and TL-models respectively. Then:

9.4.1. T . *There exists a language translation* $t_{\mathtt{L}} : L_{\mathtt{ETL}} \to L_{\mathtt{TL}}$ *and a model transfor-*

mation $t_{\mathcal{M}} : C^{\mathtt{ETL}} \to C^{\mathtt{TL}}$ *where* $\mathtt{TL} \in \{\mathtt{CTL}^*_{\mathbf{I}}, \mathtt{PDL}, \mathtt{Mu}_{\mathbf{I}}\}$ *such that:*

$$\forall \varphi \in \mathtt{ETL} \forall \mathcal{M} \in C^{\mathtt{ETL}}(\mathcal{M} \models_{\mathtt{ETL}} \varphi \Leftrightarrow t_{\mathcal{M}}(\mathcal{M}) \models_{\mathtt{TL}} t_{\mathrm{L}}(\psi)).$$

P We only discuss the $\mathtt{CTL}^*$ case. Let t_{L} be the translation that, for each formula, 1) replaces each occurrence of K_i by AX_i, 2) recursively replaces each common knowledge operator $C_{\mathbf{I}'}$ (with $\mathbf{I}' \subseteq \mathbf{I}$) by $A(\neg((\bigvee_{i\in\mathbf{I}'} X_i\top)Ut_{\mathrm{L}}(\neg\phi)))$. Let $t_{\mathcal{M}}$ be the transformation which unravels the epistemic relations into labelled temporal relations. ⋈

This observation suggests a way to reduce ETL model checking to TL model checking, with the help of some small model property. However, the unravelling of epistemic relations may introduce an exponential blow-up of the models, see, for instance [AvC07].

On the other hand, the above result is somehow misleading in understanding the expressivity of ETL and TL, since we reinterpret epistemic relations as temporal operators by introducing *new* operators in the temporal language. To address the comparison of expressivity without manipulating the language we can consider the following case:

Suppose that the epistemic relations of the ETL*-models are generated by the temporal structures as explained in Section 9.2.3. We can turn the* ETL*-models into* TL*-models by ignoring the generated epistemic relations. A straightforward question is to ask whether explicit epistemics helps to define more classes of such temporal models, or if the epistemic information can be retrieved from the temporal structure. Formally we need to prove or disprove the following:*

$$\exists \phi \in L_{\mathtt{ETL}}, \forall \psi \in L_{\mathtt{TL}} : t^-_{\mathcal{M}}(C_\phi) \neq C_\psi.$$

where C_ϕ *(*C_ψ*) is the class of* ETL *(*TL*) models which satisfy* ϕ *(*ψ*);* $t^-_{\mathcal{M}}$ *transforms the* ETL *models in* C_ϕ *into corresponding* TL *models by ignoring the generated epistemic relations.*

In case that the epistemic relations are generated respecting synchronicity (i.e. epistemic relations only appear in the same *level* of the tree unravelling of the temporal model), then we have a clear answer to the above question. We can reformulate Theorem 1 of [AvZ06] in spirit as follows:

9.4.2. T . *If we only consider the* ETL *models satisfying synchronicity, then the secrecy flavoured* ETL *formula* $AXAG(\neg K\neg p \wedge \neg Kp)$ *(never be sure about* p *in the future) is not* $t^-_{\mathcal{M}}$*-translatable into* L_μ*.*

The proof is essentially hidden in [Eme87], which shows that the class of the trees that have a level where p is true everywhere, is not recognisable by non-deterministic Muller tree automata. We can employ the pumping-lemma-like argument of [Eme87] to obtain this result.

More generally, it is known that *Monadic Second Order Logic* (MSO) cannot express "x and y are at the same level" on trees [LS87]. Thus, the merit of the above untranslatability result may actually be rooted in the property of synchronicity. Hence, although synchronicity is a commonly accepted idealisation of the agents, we still

want to know whether we can ignore it or replace it by other properties but get a similar untranslatability result. This is still open.

9.4.2 Model Checking ETL

The previous sections gave both the intuitive and technical arguments on the usability and expressivity of the epistemic approaches in protocol verification. However, do we pay any cost in the complexity of model checking? In this section we summarize the important model checking results of the literature. For complexity results regarding the satisfiability problems of the corresponding logics, we refer to [HV86, SG02].

[SG02] shows that on explicit Kripke models the model checking problem of CTL with common knowledge operators (CTL + C) can be done in PTIME and [vdHW02] proved that for *Alternating Time Logic* (ATL) with knowledge, it is PTIME -complete. This looks similar to the logics without knowledge operators. However, due to the construction of epistemic models in the protocol verification setting, we are more interested in the model checking problem on finitely generated infinite epistemic temporal models. Results in [SG02] indicate that on asynchronous generated models with forgetful agents, the complexity of model checking complies to the general case on Kripke structures. However, we are more interested in the finitely generated synchronous system with perfect recall agents as intruders. Here are some results for this situation:

Reference	Logic	Fragment	Complexity
[vdMS99]	LTL + K	full	non-elementary
[vdMS99]	LTL + C	full	undecidable
[vdMS99]	LTL + C	UNTIL-free	PSPACE-complete
[EGvdM07]	LTL + C	single agent	PSPACE-complete
[SG02]	CTL + K	full	non-elementary
[SG02]	CTL + C	full	undecidable
[AvC07]	CTL + K	nesting-free	PSPACE-complete
[SG02]	PDL + C	full	PSPACE-complete
[SG02]	MU + K	full	undecidable
[AvC07]	MU + K	nesting-free	EXPTIME-complete

Putting together the decidability of $\approx$ on messages terms (cf. Section 9.3.1) and the model checking results above, we can obtain decidability results for security verification (e.g. [BRS07]).[16]

The above results suggest that we may need to restrict ourselves to single agent cases or nesting-free ETL formulas due to the computational complexity. This somehow coincides with the disadvantages of ETL modelling we mentioned in Section 9.3: ETL modelling is not very suitable for multi-agent cases with higher order uncertainty.

[16] Important security properties are generally undecidable if there are no restrictions on the number of messages and nonces, for example, cf. [DLMS99] for the undecidability for secrecy. A solution is to focus on decidable subclasses as in e.g. [RS05a].

On the other hand, although multi-agent cases are often undecidable in general, we can still have some hope by restricting ourselves to certain classes where equivalence relations of agents have certain patterns (e.g. [EVDMS02]). Moreover, some model checking techniques such as abstraction and symmetry reduction that are specific to ETL or DEL can be found in [DOW08, CDLR09, CLDQ09].

As a final note, in practice, the performance of an ETL model checking tool kit relies on the particular class of models to be checked and their representations, for example, model checking CTL+K against *"compact models"* is in PSPACE [LR06a].

9.5 Conclusion

In this chapter, we surveyed the epistemic approaches to security protocol verification with the questions: are security protocols essentially about knowledge (what is there *to know*?), how to model the different kinds of knowledge involved (how *to know*?), and does an epistemic approach bring benefits (why *to know*?). We first made the distinctions between different types of knowledge relevant in the security setting and then gave an overview of commonly used techniques in the epistemic approaches. In particular, we compared various equivalence relations defined in the literature that correspond to the semantics of propositional knowledge. We also compared two major epistemic logical approaches proposed to model interaction in multi-agent systems. It turns out that, in a setting of security protocol verification, ETL approaches are more suitable to model message passing over time, based on which appropriate equivalence relations can be generated. On the other hand, the DEL approach offers more freedom to model higher order information and uncertainties in terms of agents' knowledge about each other as we demonstrated in Part I of this thesis. The model checking results of ETL also confirm that it is better to focus on a single agent case: in the security setting, this would be the intruder. Finally, we collected clues for the comparison of the expressivity of ETL and TL, in order to see when an epistemic approach is inevitable. We showed under the assumption of synchronicity, that ETL can define more (security) properties of the temporal structures.

Appendix A

Alloy Code for Russian Cards Problem (3.3.1)

```
 module RCP
//Alloy program for finding a deterministic protocol solution
//to Russian Cards Problem  (3.3.1)
//y.wang@cwi.nl

sig Cards {
}

sig Hands {
    content: set Cards
}

sig Pa {
    member: set Hands
}

fact {all h: Hands | #h.content = 3 }
//every hand contains 3 cards

fact {all p: Pa | #p.member > 1 }
//Any announcement contains at least 2 hands

fact {all h: Hands | some p:Pa | h in p.member }
//Every hand appears at some announcement (executability)

fact {no h: Hands, g: Hands | h != g && h.content = g.content }
//No two hands are the same

fact {no p: Pa, q: Pa |p != q && # p.member & q.member > 0 }
//No two annoucnements share a hand (for determinism)
```

```
fact {no p: Pa |some h: p.member, g: p.member |
h != g && # h.content & g.content > 1 }
//In order to let B know:
//any two hands in one announcement share at most 1 card

fact {no p: Pa| some c: Cards, d: Cards | all h: p.member|
(not c in h.content) => d in h.content }
//In order to let C stay ignorant: if we fix one card in an announcement
//then the hands that do not contain this card don't have a card in common

pred RCP { }

run RCP for exactly 7 Cards , exactly 35 Hands, 7 Pa
//Given 7 cards there are 35 3-hand. At most 7 announcements.
```

Bibliography

[ABV03] R. Accorsi, D. Basin, and L. Vigano. Towards an awareness-based semantics for security protocol analysis. *Electronic Notes in Theoretical Computer Science*, 55(1):5–24, January 2003. Cited on page **143**.

[ABvDS09] T. Ågotnes, P. Balbiani, H. van Ditmarsch, and P. Seban. Group announcement logic. *Journal of Applied Logic*, July 2009. Cited on page **35**.

[AC04] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *Proceedings of ICALP '04*, volume 3142 of *LNCS*, pages 46–58, 2004. Cited on pages **150** and **151**.

[AF01] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of POPL '01*, pages 104–115, 2001. Cited on pages **141** and **150**.

[AHL+08] A. Antonik, M. Huth, K. Larsen, U. Nyman, and A. Wasowski. 20 years of mixed and modal specifications. *Bulletin of the European Association for Theoretical Computer Science*, June 2008. Cited on page **104**.

[AHM+98] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Tasiran. Mocha: Modularity in model checking. In *Proceedings of CAV'98*, pages 521–525, 1998. Cited on page **149**.

[AN95] R. Anderson and R. Needham. Programming satan's computer. In J. Leeuwen, editor, *Computer Science Today*, volume 1000 of *LNCS*, pages 426–440, Berlin/Heidelberg, 1995. Springer-Verlag. Cited on page **140**.

[AR02] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Proceedings of the International Conference IFIP*, 2002. Cited on page **150**.

[AT91] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication (extended abstract). In *Proceedings of PODC '91*, pages 201–216, New York, NY, USA, 1991. ACM. Cited on page **150**.

[Auc09] G. Aucher. BMS revisited. In *Proceedings of TARK '09*, pages 24–33, 2009. Cited on pages **39** and **56**.

[Aum76] R. J. Aumann. Agreeing to disagree. *The Annals of Statistics*, 4(6):1236–1239, 1976. Cited on page **84**.

[Aum89] R. Aumann. Notes on interactive epistemology. 1989. Cited on page 84.

[AvC07] R. Alur, P. Černý, and S. Chaudhuri. Model checking on trees with path equivalences. In *Proceedings of TACAS '07*, pages 664–678, 2007. Cited on pages 156 and 157.

[AvDR09] M. D. Atkinson, H. van Ditmarsch, and S. Roehling. Avoiding bias in cards cryptography. *Australasian Journal of Combinatorics*, 44:3–17, February 2009. Cited on pages 3 and 33.

[AvZ06] R. Alur, P. Černý, and S. Zdancewic. Preserving secrecy under refinement. In *Automata, Languages and Programming*, pages 107–118, 2006. Cited on page 156.

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, 426(1871):233–271, December 1989. Cited on page 142.

[BCG87] M. Browne, E. Clarke, and O. Grümberg. Characterizing kripke structures in temporal logic. In *Proceedings of TAPSOFT '87*, pages 256–270, 1987. Cited on page 83.

[BCL09] I. Boureanu, M. Cohen, and A. Lomuscio. Automatic verification of temporal-epistemic properties of cryptographic protocols. *Journal of Applied Non-Classical Logics*, 19(4):463–487, 2009. Cited on pages 146, 147, 148, and 149.

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992. Cited on page 6.

[BdRV02] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, November 2002. Cited on pages 13, 88, and 132.

[BE09] D. Bonnay and P. Égré. Inexact knowledge with introspection. *Journal of Philosophical Logic*, 38(2):179–227, April 2009. Cited on page 40.

[BG99] G. Bruns and P. Godefroid. Model checking partial state spaces with 3-valued temporal logics. In Nicolas Halbwachs and D. Peled, editors, *Proceedings of CAV '99*, volume 1633 of *LNCS*, page 684, Berlin, Heidelberg, January 1999. Springer Berlin Heidelberg. Cited on page 103.

[BG04] G. Bruns and P. Godefroid. Model checking with multi-valued logics. In *Automata, Languages and Programming*, pages 245–273. 2004. Cited on page 108.

[BHR84] D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, 1984. Cited on pages 60 and 143.

[Bie90] P. Bieber. A logic of communication in hostile environment. In *Proceedings of Computer Security Foundations Workshop III*, pages 14–22, 1990. Cited on page 143.

[BK85] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985. Cited on page 60.

[BKR09] V. Bárány, Łukasz K.ser, and A. Rabinovich. Cardinality quantifiers in MLO over trees. In *Proceedings of CSL 09*, pages 117–131, 2009. Cited on page 85.

[BM96] J. Barwise and L. Moss. *Vicious Circles*. (Center for the Study of Language and Information, August 1996. Cited on page 83.

[BM04] A. Baltag and L. Moss. Logics for epistemic programs. *Synthese*, 139(2):165–224, March 2004. Cited on pages 22, 39, 49, and 117.

[BMS98] A. Baltag, L. Moss, and S Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of TARK '98*, pages 43–56. Morgan Kaufmann Publishers Inc., 1998. Cited on pages 3, 15, and 51.

[BP05] M. Bhargava and C. Palamidessi. Probabilistic anonymity. In *Proceedings of CONCUR'05*, volume 3653 of *LNCS*, pages 171–185, 2005. Cited on page 144.

[BRS07] A. Baskar, R. Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *Proceedings of TARK '07*, pages 62–71, New York, NY, USA, 2007. ACM. Cited on pages 141, 142, 145, 150, 151, 153, and 157.

[Brz64] J. A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, October 1964. Cited on page 40.

[BS97] J. Barwise and J. Seligman. *Information flow: the logic of distributed systems*. Cambridge University Press, New York, NY, USA, 1997. Cited on page 3.

[BS06] J. Bradfield and C. Stirling. Modal μ-calculi. In *Handbook of Modal Logic*, volume 3, pages 722–756. Elsevier Science Inc., New York, NY, USA, 2006. Cited on page 84.

[BS08a] A. Baltag and S. Smets. Probabilistic dynamic belief revision. *Synthese*, 165(2):179–202, 2008. Cited on pages 56 and 143.

[BS08b] Mario Benevides and L. Schechter. A propositional dynamic logic for CCS programs. pages 83–97, 2008. Cited on page 39.

[CBRZ01] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, July 2001. Cited on page 6.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of POPL '77*, pages 238–252, New York, NY, USA, 1977. ACM. Cited on page 6.

[CCD88] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proceedings of STOC '88*, pages 11–19, New York, NY, USA, 1988. ACM. Cited on page 155.

[CD05a] M. Cohen and M. Dam. A completeness result for BAN logic. In *Proceedings of Methods for Modalities '05*, 2005. Cited on page 150.

[CD05b] M. Cohen and M. Dam. Logical omniscience in the semantics of BAN logic. In *Proceedings of FCS '05*, 2005. Cited on page 143.

[CD07] M. Cohen and M. Dam. A complete axiomatization of knowledge and cryptography. In *Proceedings of LiCS '07*, pages 77–88. IEEE Computer Society, 2007. Cited on pages 143, 150, and 151.

[CDGLV02] D. Calvanese, G. De Giacomoa, M. Lenzerinia, and M. Y. Vardi. Rewriting of regular expressions and regular path queries. *Journal of Computer and System Sciences*, 64(3):443–465, May 2002. Cited on pages **121**, **123**, and **125**.

[CDK09a] R. Chadha, S. Delaune, and S. Kremer. Epistemic logic for the applied pi calculus. In *Proceedings of FMOODS '09/FORTE '09*, pages 182–197, Berlin, Heidelberg, 2009. Springer-Verlag. Cited on page **150**.

[CDK09b] S. Ciobâcă, S. Delaune, and S. Kremer. Computing knowledge in security protocols under convergent equational theories. In *Processdings of CADE*, pages 355–370, 2009. Cited on page **150**.

[CDLR09] M. Cohen, M. Dam, A. Lomuscio, and F. Russo:. Abstraction in model checking multi-agent systems. In *Proceedings of AAMAS '09*, 2009. Cited on pages **6**, **104**, and **158**.

[CEFJ96] E. M. Clarke, R. Enders, T. Filkorn, and S. Jha. Exploiting symmetry in temporal logic model checking. *Formal Methods in System Design*, 9(1-2):77–104, 1996. Cited on page **6**.

[CGJ+03] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, 50(5):752–794, September 2003. Cited on pages **6** and **117**.

[CGL94] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, September 1994. Cited on page **6**.

[CGP99] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, January 1999. Cited on pages **5**, **125**, and **143**.

[Cha88] D. Chaum. The dining cryptographers problem: unconditional sender and receiver untraceability. *Journal of Cryptology*, 1:65–75, 1988. Cited on page **149**.

[CJM98] E. M. Clarke, S. Jha, and W. R. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proceedings of PROCOMET '98*, pages 87–106, London, UK, UK, 1998. Chapman & Hall, Ltd. Cited on page **144**.

[CLDQ09] M. Cohen, A. Lomuscio, M. Dam, and H. Qu. A symmetry reduction technique for model checking temporal epistemic logic. In *Proceedings of IJCAI '09*, 2009. Cited on pages **6**, **80**, **104**, and **158**.

[Con71] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, September 1971. Cited on pages **40** and **126**.

[Cre06] C. J. F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006. Cited on page **146**.

[CvdPW08] T. Chen, J. van de Pol, and Y. Wang. Pdl over accelerated labeled transition systems. In *Proceedings of TASE '09*, pages 193–200, Los Alamitos, CA, USA, 2008. IEEE Computer Society. Cited on pages **8** and **136**.

[DETW09] F. Dechesne, D. J. N. Eijck, W. Teepe, and Y. Wang. What is protocol analysis? In D. J. N. van Eijck and R. Verbrugge, editors, *Discourses on Social Software*, volume 5 of *Texts in Logic and Games*. Amsterdam University Press, January 2009. Cited on page **8**.

[dJ09] T. de Jager. *Awareness, Attention, Assumption*. PhD thesis, October 2009. Cited on page 81.

[DKR06] S. Delaune, S. Kremer, and M. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proceedings of CSFW '06*, pages 28–42. IEEE Computer Society, 2006. Cited on page 142.

[DKR07] S. Delaune, S. Kremer, and M. Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 2007. Cited on pages 139 and 142.

[DLMS99] N. A. Durgin, P. D. Lincoln, J. C. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proceedings of the Workshop on Formal Methods and Security Protocols (FMSP)*, 1999. Cited on page 157.

[DMO07] F. Dechesne, M. R. Mousavi, and S. Orzan. Operational and epistemic approaches to protocol analysis: Bridging the gap. In *Proceedings of LPAR '07*, pages 226–241, 2007. Cited on page 148.

[DN05] D. Dams and K. S. Namjoshi. Automata as abstractions. In *Proceedings of VMCAI '05*, pages 216–232, 2005. Cited on page 86.

[DOW08] F. Dechesne, S. Orzan, and Y. Wang. Refinement of kripke models for dynamics. In *Proceedings of ICTAC '08*, pages 111–125, 2008. Cited on pages 6, 8, 117, and 158.

[DvETW09] F. Dechesne, J. van Eijck, W. Teepe, and Y. Wang. Dynamic epistemic logic for protocol analysis. In D. J. N. van Eijck and R. Verbrugge, editors, *Discourses on Social Software*, volume 5 of *Texts in Logic and Games*. Amsterdam University Press, January 2009. Cited on page 8.

[DvEW10] H. Ditmarsch, J. van Eijck, and W. Wu. One hundred prisoners and a lightbulb – logic and computation. In *Proceedings of KR '10*. AAAI, 2010. Cited on page 3.

[DW07] F. Dechesne and Y. Wang. Dynamic epistemic verification of security protocols: framework and case study. In *A Meeting of the minds: Proceedings of LORI-I workshop*, Texts in Computer Science, pages 129–144, 2007. Cited on pages 6, 8, 23, 59, 117, and 148.

[DW10] F. Dechesne and Y. Wang. To know or not to know: Epistemic approaches to security protocol verification. *To appear in Synthese, special section of Knowledge, Rationality and Action*, 2010. Cited on page 8.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. Cited on page 146.

[EGvdM07] K. Engelhardt, P. Gammie, and R. van der Meyden. Model checking knowledge and linear time: Pspace cases. In *Proceedings of LFCS '07*, pages 195–211, 2007. Cited on page 157.

[Eme87] E. Allen Emerson. Uniform inevitability is tree automaton ineffable. *Information Processing Letters.*, 24(2):77–79, 1987. Cited on page 156.

[ES96] E. A. Emerson and A. Prasad Sistla. Symmetry and model checking. *Formal Methods in System Design*, 9(1-2):105–131, 1996. Cited on page 6.

[EvdMM98] K. Engelhardt, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In *Proceedings of TARK '98*, pages 29–41, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. Cited on page 69.

[EVDMS02] K. Engelhardt, R. Van Der Meyden, and K. Su. Modal logics with a linear hierarchy of local propositional quantifiers. In *Proceedings of AiML '02*, volume 9, 2002. Cited on page 158.

[EvdP06] M. Espada and J. van de Pol. Accelerated modal abstractions of labelled transition systems. In M. J.son and Varmo Vene, editors, *Proceedings of AMAST '06*, volume 4019, pages 338–352, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. Cited on pages 120, 121, 128, and 136.

[FGM04] R. Focardi, R. Gorrieri, and F. Martinelli. *Classification of Security Properties (Part II: Network Security)*, volume 2946 of *LNCS*, pages 139–185. Springer Berlin / Heidelberg, 2004. Cited on page 141.

[FHMV95] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about knowledge*. MIT Press, Cambridge, MA, USA, 1995. Cited on pages 2, 4, 15, 59, 117, 143, and 149.

[FHMV97] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, July 1997. Cited on pages 2, 4, 24, and 153.

[Fit91] M. Fitting. Many-valued modal logics. *Fundamenta Informaticae*, 15(3-4):235–254, 1991. Cited on page 104.

[Fit92] M. Fitting. Many-valued modal logics II. *Fundamenta Informaticae*, 17(1-2):55–73, 1992. Cited on page 104.

[FL79] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979. Cited on page 13.

[FW96] M. J. Fischer and Rebecca N. Wright. Bounds on secret key exchange using a random deal of cards. *Journal of Cryptology , Springer Verlag*, 9:71–99, 1996. Cited on page 155.

[Gab02] D. M. Gabbay. A theory of hypermodal logics: Mode shifting in modal logic. *Journal of Philosophical Logic*, 31(3):211–243, June 2002. Cited on pages 39 and 40.

[GG97] J. Gerbrandy and W. Groeneveld. Reasoning about information change. *Journal of Logic, Language and Information*, 6(2):147–169, April 1997. Cited on pages 3, 15, 17, and 105.

[GHJ01] P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based model checking using modal transition systems. In K. G. Larsen and Mogens Nielsen, editors, *Proceedings of CONCUR '01*, volume 2154 of *LNCS*, pages 426–440, Berlin, Heidelberg, August 2001. Springer Berlin Heidelberg. Cited on pages 6 and 103.

[GHPvR05] F. D. Garcia, I. Hasuo, W. Pieters, and P. van R.um. Provable anonymity. In *Proceedings of FMSE '05*, pages 63–72, New York, NY, USA, 2005. ACM. Cited on page 150.

[GJ02] P. Godefroid and R. Jagadeesan. Automatic abstraction using generalized model checking. In *Proceedings of CAV '02:*, pages 137–150, London, UK, 2002. Springer-Verlag. Cited on page **105**.

[GJ03] P. Godefroid and R. Jagadeesan. On the expressiveness of 3-valued models. In *Proceedings of VMCAI '03*, pages 206–222, 2003. Cited on page **107**.

[GK03] E. Grädel and S. Kreutzer. Will deflation lead to depletion? on non-monotone fixed point inductions. In *Proceedings of LiCS '03*, pages 158–178, Washington, DC, USA, 2003. IEEE Computer Society. Cited on page **34**.

[GM99] D. Giammarresi and R Montalbano. Deterministic generalized automata. *Theoretical Computer Science*, 215(1-2):191–208, 1999. Cited on page **121**.

[GNY90] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Research in Security and Privacy*, 1990. Cited on page **143**.

[GO06] V. Goranko and M. Otto. Model theory of modal logic. In *Handbook of Modal Logic*, volume 3, pages 249–329. Elsevier Science Inc., New York, NY, USA, 2006. Cited on page **88**.

[GP94] J. F. Groote and A. Ponse. The syntax and semantics of μCRL. In *Algebra of Communicating Processes*, Workshops in Computing, pages 26–62. 1994. Cited on page **60**.

[GPS96] P. Godefroid, D. Peled, and M. Staskauskas. Using partial-order methods in the formal validation of industrial concurrent programs. In *Proceedings of ISSTA '96*, pages 261–269, New York, NY, USA, 1996. ACM. Cited on page **6**.

[GvdM04] P. Gammie and R. van der Meyden. `MCK`: Model checking the logic of knowledge. In *Proceedings of CAV '04*, pages 256–259, 2004. Cited on page **149**.

[Hal87] J. Y. Halpern. A little knowledge goes a long way: simple knowledge-based derivations and correctness proofs for a family of protocols. In *Proceedings of PODC '87*, pages 269–280, New York, NY, USA, 1987. ACM. Cited on page **3**.

[Hal00] J. Y. Halpern. A note on knowledge-based programs and specifications. *Distributed Computing*, 13(3):145–153, July 2000. Cited on page **3**.

[Har96] S. Hart. "knowing whether", "knowing that", and the cardinality of state spaces. *Journal of Economic Theory*, 70(1):249–256, July 1996. Cited on pages **84** and **85**.

[HD07] A. Hunter and J. P. Delgrande. Belief change and cryptographic protocol verification. In *Proceedings of AAAI '07*, pages 427–433, 2007. Cited on page **143**.

[HF89] J. Y. Halpern and R. Fagin. Modelling knowledge and action in distributed systems. *Distributed Computing*, 3(4):159–177, 1989. Cited on pages **2**, **3**, **37**, and **153**.

[Hin62] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca N.Y., 1962. Cited on page **143**.

[HJS01] M. Huth, R. Jagadeesan, and D. Schmidt. Modal transition systems: A foundation for three-valued program analysis. In D. Sands, editor, *Programming Languages and Systems*, volume 2028, pages 155–169, Berlin, Heidelberg, March 2001. Springer Berlin Heidelberg. Cited on pages **103**, **105**, and **107**.

[HKP82] D. Harel, D. Kozen, and R. Parikh. Process logic: Expressiveness, decidability, completeness. *Journal of Computer and System Sciences*, 25(2):144–170, 1982. Cited on page **39**.

[HKT00] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic (Foundations of Computing)*. The MIT Press, 1st edition, October 2000. Cited on pages **50** and **135**.

[HM90] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. Cited on page **15**.

[HMV94] J. Y. Halpern, Y. Moses, and M. Y. Vardi. Algorithmic knowledge. In *Proceedings of TARK '94*, pages 255–266, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. Cited on page **143**.

[HMV05] A. Hommersom, J.-Jules Meyer, and E. Vink. Update semantics of security protocols. In *Information, Interaction and Agency*, pages 289–327, Berlin/Heidelberg, 2005. Springer-Verlag. Cited on page **148**.

[HO02] J. Halpern and K. O'Neill. Secrecy in multiagent systems. In *Proceedings of CSFW '02*, pages 32–46, 2002. Cited on page **144**.

[HO05] J. Halpern and K. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, 13(3):483–514, May 2005. Cited on pages **142** and **144**.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice/Hall International, April 1985. Cited on page **5**.

[Hos09] T. Hoshi. *Epistemic Dynamics and Protocol Information*. PhD thesis, Stanford, 2009. Cited on pages **3**, **21**, **37**, and **39**.

[Hos10] T. Hoshi. Merging DEL and ETL. *Journal of Logic, Language and Information*, January 2010. Cited on pages **3** and **154**.

[HP03] J. Y. Halpern and R. Pucella. Modeling adversaries in a logic for security protocol analysis. In *Formal Aspects of Security*, pages 87–100, 2003. Cited on pages **141**, **143**, **144**, **145**, and **146**.

[HP10a] J. Y. Halpern and R. Pucella. Dealing with logical omniscience: Expressiveness and pragmatics. *Artificial Intelligence*, April 2010. To appear. Cited on page **143**.

[HP10b] T. Hoshi and E. Pacuit. A dynamic logic of knowledge and access. *Synthese*, 2010. forthcoming. Cited on pages **3** and **37**.

[HV86] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time. In *Proceedings of STOC '86*, pages 304–315, New York, NY, USA, 1986. ACM. Cited on page **157**.

[HW05] Y-S Han and D. Wood. The generalization of generalized automata: Expression automata. In *Implementation and Application of Automata*, pages 156–166, 2005. Cited on page **121**.

[HY09] T. Hoshi and A. Yap. Dynamic epistemic logic with branching temporal structures. *Synthese*, 169(2):259–281, July 2009. Cited on pages **3**, **21**, **37**, **39**, and **154**.

[ID96] C. N. Ip and D. L. Dill. Better verification through symmetry. *Formal Methods in System Design*, 9(1-2):41–75, 1996. Cited on page **6**.

[Jac02] D. Jackson. Alloy: a lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology*, 11(2):256–290, April 2002. Cited on page **32**.

[JdV06] H. L. Jonker and E. P. de Vink. Formalising Receipt-Freeness. In Sokratis K. Katsikas, Javier Lopez, M. Backes, Stefanos Gritzalis, and Bart Preneel, editors, *Information Security*, volume 4176 of *LNCS*, pages 476–488, August 2006. Cited on page **142**.

[JP06] H. Jonker and W. Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *IAVoSS Workshop On Trustworthy Elections*, June 2006. Cited on pages **142** and **150**.

[JW95] D. Janin and I. Walukiewicz. Automata for the modal μ-calculus and related results. In *Proceedings of MFCS '95*, pages 552–562, 1995. Cited on pages **84**, **86**, and **87**.

[Kle50] S. C. Kleene. *Introduction to Metamathematics*. D. Van Nostrand, Princeton, NJ, 1950. Cited on page **107**.

[KNN+08] M. Kacprzak, W. Nabiałek, A. Niewiadomski, W. Penczek, Agata Półrola, Maciej Szreter, Bożena Woźna, and Andrzej Zbrzezny. Verics 2007 - a model checker for knowledge and real-time. *Fundamenta Informaticae*, 85(1):313–328, January 2008. Cited on page **149**.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983. Cited on page **84**.

[Koz91] D. Kozen. A completeness theorem for kleene algebras and the algebra of regular events. In *Proceedings of LiCS '91*, pages 214–225, 1991. Cited on pages **40**, **121**, **131**, and **132**.

[Koz01] D. Kozen. Automata on guarded strings and applications. Technical report, Ithaca, NY, USA, 2001. Cited on pages **43**, **46**, and **47**.

[Kra07] S. Kramer. *Logical concepts in cryptography*. PhD thesis, EPFL, 2007. Cited on pages **140** and **141**.

[KvB04] B. Kooi and J. van Benthem. Reduction axioms for epistemic actions. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Preliminary Proceedings of AiML-2004*, pages 197–211. Department of Computer Science, University of Manchester, 2004. Cited on pages **39**, **49**, **51**, **52**, and **53**.

[Lan06] M. Lange. Model checking propositional dynamic logic with all extras. *Journal of Applied Logic*, 4(1):39–49, March 2006. Cited on pages **120** and **125**.

[Lar90] K. Larsen. Modal specifications. In *Automatic Verification Methods for Finite State Systems*, pages 232–246, 1990. Cited on page **110**.

[LC03] C. H. Lundh and V. Cortier. Security properties: two agents are sufficient. In *Proceedings of ESOP '03*, volume 2618 of *LNCS*, pages 99–113, 2003. Cited on page **146**.

[Liu08] F. Liu. *Changing for the better*. PhD thesis, University of Amsterdam, 2008. Cited on page **56**.

[Lod95] K. Lodaya. A logical study of distributed transition systems. *Information and Computation*, 119(1):91–118, May 1995. Cited on page **121**.

[Low96] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using FDR. In *Proceedings of TACAS '96*, pages 147–166, London, UK, 1996. Springer-Verlag. Cited on pages **140** and **143**.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proceedings of POPL '85*, pages 97–107, New York, NY, USA, 1985. ACM. Cited on page **129**.

[LP07] A. Lomuscio and W. Penczek. Symbolic model checking for temporal-epistemic logics. *SIGACT News*, 38(3):77–99, 2007. Cited on page **149**.

[LQR09] A. Lomuscio, H. Qu, and F. Raimondi. `MCMAS`: A model checker for the verification of multi-agent systems. In *Proceedings of CAV '09*, pages 682–688, 2009. Cited on page **149**.

[LR06a] A. Lomuscio and F. Raimondi. The complexity of model checking concurrent programs against `CTLK` specifications. In *Proceedings of AAMAS '06*, pages 548–550, New York, NY, USA, 2006. ACM. Cited on page **158**.

[LR06b] A. Lomuscio and F. Raimondi. Mcmas: A model checker for multi-agent systems. In *Proceedings of TACAS '06*, volume 3920 of *LNCS*, pages 450–454. Springer, 2006. Cited on page **149**.

[LS87] H. Läuchli and C. Savioz. Monadic second order definable relations on the binary tree. *The Journal of Symbolic Logic*, 52(1):219–226, 1987. Cited on page **156**.

[LS07] M. Leucker and C. Sánchez. Regular linear temporal logic. In Cliff B. J.es, Zhiming Liu, and Jim Woodcock, editors, *Proceedings of ICTAC '07*, volume 4711 of *LNCS*, pages 291–305, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. Cited on page **121**.

[LT88] K. G. Larsen and B. Thomsen. A modal process logic. In *Proceedings of LiCS '88*, pages 203–210, July 1988. Cited on page **104**.

[Lut06] Carsten Lutz. Complexity and succinctness of public announcement logic. In *Proceedings of AAMAS '06*, pages 137–143, New York, NY, USA, 2006. ACM. Cited on page **56**.

[Mat03] R. Mateescu. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. *Science of Computer Programming*, 46(3):255–281, March 2003. Cited on page **121**.

[McM02] K. McMillan. Applying sat methods in unbounded symbolic model checking. In *Proceedings of CAV'02*, pages 250–264, London, UK, 2002. Springer-Verlag. Cited on page **6**.

[MDH86] Y. Moses, D. Dolev, and J. Y. Halpern. Cheating husbands and other stories: A case study of knowledge, action, and communication. *Distributed Computing*, 1(3):167–176, September 1986. Cited on page 3.

[Mey87] J. J. Meyer. A different approach to deontic logic: deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29(1):109–136, 1987. Cited on page 39.

[Mil82] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982. Cited on page 60.

[MM05] J. Miller and L. Moss. The undecidability of iterated modal relativization. *Studia Logica*, 79, April 2005. Cited on page 23.

[Niw91] D. Niwiński. On the cardinality of sets of infinite trees recognizable by finite automata. In *Proceedings of MFCS '91*, pages 367–376, 1991. Cited on pages 84, 85, 89, 95, 96, 98, and 99.

[NS78] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978. Cited on page 140.

[Orz05] S. Orzan. LYS: a knowledge analysis toolset, 2005. Available at `http://www.mobanet.nl/simona/lys/`. Cited on page 149.

[Par78] R. Parikh. The completeness of propositional dynamic logic. In *Proceedings of MFCS '78*, pages 403–415, 1978. Cited on page 14.

[Par02] R. Parikh. Social software. *Synthese*, 132:187–211, 2002. Cited on page 55.

[Par03] R. Parikh. Levels of knowledge, games, and group action. *Research in Economics*, 57(3):267–281, September 2003. Cited on pages 84 and 85.

[Pau97] L. C. Paulson. Proving properties of security protocols by induction. In *Proceedings of CSFW '97*, pages 70–83. IEEE Computer Society Press, 1997. Cited on page 144.

[Pau98] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998. Cited on page 144.

[Pel87] D. Peleg. Concurrent dynamic logic. *Journal of the ACM*, 34(2):450–479, 1987. Cited on page 60.

[Pel93] D. Peled. All from one, one for all: on model checking using representatives. In *Proceedings of CAV '93*, pages 409–423, London, UK, 1993. Springer-Verlag. Cited on page 6.

[PK92] R. Parikh and P. Krasucki. Levels of knowledge in distributed systems. *Sadhana*, 17(1):167–191, March 1992. Cited on pages 84 and 85.

[Pla89] J. A. Plaza. Logics of public communications. In M. L. Emrich, M. S. Pfeifer, M. Hadzikadic, and Z. W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989. Cited on pages 3, 15, 17, and 105.

[PP07] S. Petride and R. Pucella. Perfect cryptography, s5 knowledge, and algorithmic knowledge. In *Proceedings of TARK '07*, pages 239–247, New York, NY, USA, 2007. ACM. Cited on page 151.

[PR85] R. Parikh and R. Ramanujam. Distributed processes and the logic of knowledge. In *Proceedings of Conference on Logic of Programs*, pages 256–268, London, UK, 1985. Springer-Verlag. Cited on pages 3 and 15.

[PR03] R. Parikh and R. Ramanujam. A knowledge based semantics of messages. *Journal of Logic, Language and Information*, 12(4), 2003. Cited on pages 3, 21, 37, 38, and 39.

[Pra76] V. R. Pratt. Semantical considerations on floyd-hoare logic. Technical report, Cambridge, MA, USA, 1976. Cited on page 13.

[Pra79] V. R. Pratt. Process logic. In *Proceedings of POPL '79*, pages 93–100, New York, NY, USA, 1979. ACM. Cited on page 39.

[Pra80] V. R. Pratt. A near-optimal method for reasoning about action. *Journal of Computer and System Sciences*, 20(2):231–254, 1980. Cited on page 5.

[PRS09] S. Paul, R. Ramanujam, and S. Simon. Stability under strategy switching. In Klaus Ambos-Spies, Benedikt Löwe, and Wolfgang Merkle, editors, *Mathematical Theory and Computational Practice*, volume 5635, chapter 40, pages 389–398. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. Cited on page 39.

[PS10] E. Pacuit and S. Simon. Reasoning with protocols under imperfect information. 2010. Extended abstract presented at Advances in Modal Logic 10. Cited on page 38.

[Puc06] R. Pucella. Deductive algorithmic knowledge. *Journal of Logic and Computation*, 16(2):287–309, April 2006. Cited on page 143.

[RR98] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1:66–92, 1998. Cited on page 144.

[RS01] P. Ryan and S. Schneider. *Modelling and analysis of security protocols*. Addison Wesley, 2001. Cited on pages 139 and 141.

[RS05a] R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005. Cited on page 157.

[RS05b] R. Ramanujam and S. P. Suresh. Deciding knowledge properties of security protocols. In *Proceedings of TARK '05*, pages 219–235. Morgan Kaufmann, 2005. Cited on pages 141, 145, 146, and 147.

[San91] Beverly Sanders. A predicate transformer approach to knowledge and knowledge-based protocols (extended abstract). In *Proceedings of PODC '91*, pages 217–230, New York, NY, USA, 1991. ACM. Cited on page 3.

[SE89] R. S. Streett and A. E. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81(3):249–264, June 1989. Cited on page 5.

[Seg82] K. Segerberg. A completeness theorem in the modal logic of programs. In T. Traczyk, editor, *Universal Algebra*, volume 9, pages 31–46. Banach Centre Publications, 1982. Cited on page 14.

[SG02] N. V. Shilov and N. O. Garanina. Model checking knowledge and fixpoints. In Zoltán Ésik, Anna Ingólfsdóttir, Zoltán Ésik, and Anna Ingólfsdóttir, editors, *Proceedings of FICS '02*, volume NS-02-2 of *BRICS Notes Series*, pages 25–39. University of Aarhus, 2002. Cited on pages **104**, **148**, and **157**.

[SG04] A. P. Sistla and P. Godefroid. Symmetry and reduced symmetry in model checking. *ACM Transactions on Programming Languages and Systems*, 26(4):702–734, 2004. Cited on page **6**.

[SG08] S. Shoham and O. Grumberg. 3-valued abstraction: More precision at less cost. *Information and Computation*, 206(11):1313–1333, November 2008. Cited on page **6**.

[Shm04] V. Shmatikov. Probabilistic model checking of an anonymity system. *Journal of Computer Security*, 12(3/4):355–377, 2004. Cited on page **144**.

[SS99] P. F. Syverson and S. G. Stubblebine. Group principals and the formalization of anonymity. In *Proceedings of World Congress on Formal Methods '09*, volume 1708 of *LNCS*, pages 814–833. Springer, 1999. Cited on pages **142** and **144**.

[Su04] K. Su. Model checking temporal logics of knowledge in distributed systems. In Deborah L. Mcguinness, George Ferguson, Deborah L. Mcguinness, and George Ferguson, editors, *Proceedings of AAAI '04*, pages 98–103. AAAI Press / The MIT Press, 2004. Cited on page **149**.

[Syv92] P. F. Syverson. Knowledge, belief, and semantics in the analysis of cryptographic protocols. *Journal of Computer Security*, 1(3-4):317–334, 1992. Cited on page **143**.

[TDW08] M. T. Dashti and Y. Wang. Risk balance in exchange protocols. In *Proceedings of ASIAN '07*, pages 70–77, 2008. Cited on page **8**.

[Tee06] W. Teepe. BAN logic is not 'sound', constructing epistemic logics for security is difficult. In Rineke, editor, *Proceedings of FAMAS'06*, pages 79–91, 2006. Cited on page **143**.

[vB98] J. van Benthem. Dynamic odds and ends. Technical report, ILLC, 1998. Cited on page **83**.

[vB09] J. van Benthem. The great art of modeling. Technical report, ILLC, 2009. Cited on pages **5** and **59**.

[vBGHP09] J. van Benthem, J. Gerbrandy, T. Hoshi, and E Pacuit. Merging frameworks for interaction. *Journal of Philosophical Logic*, 38(5):491–526, October 2009. Cited on pages **3**, **21**, **37**, **39**, **152**, and **154**.

[vBI08] J. van Benthem and D. Ikegami. Modal fixed-point logic and changing models. In *Pillars of Computer Science*, pages 146–165, 2008. Cited on page **84**.

[vBL07] J. van Benthem and F. Liu. Dynamic logic of preference upgrade. *Journal of Applied Non-Classical Logics*, 17(2):157–182, 2007. Cited on page **56**.

[vBvEK06] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, November 2006. Cited on pages **15**, **17**, **34**, and **39**.

[vBVQ09] J. van Benthem and F. Velázques-Quesada. Inference, promotion, and the dynamics of awareness. Technical report, ILLC, Amsterdam, 2009. Cited on page 81.

[vD02] H. van Ditmarsch. Descriptions of game actions. *Journal of Logic Language and Information*, 11(3):349–365, 2002. Cited on pages 5, 22, and 84.

[vD03] H. van Ditmarsch. The Russian Cards Problem. *Studia Logica*, pages 31–62, October 2003. Cited on pages 3, 4, 5, 22, 26, 31, 149, and 155.

[vD08] H. van Ditmarsch. Unconditionally secure protocols with card deals, September 2008. Presented at the Lorentz Center workshop Logic and Information Security, available at http://www.cs.otago.ac.nz/staffpriv/hans/lorentz/niaslorentz.pdf. Cited on pages 3 and 155.

[vDF09] H. van Ditmarsch and T. French. Simulation and information: Quantifying over epistemic events. In J.-Jules C. Meyer and J. Broersen, editors, *Proceedings of KR '09*, volume 5605, pages 51–65, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. Cited on pages 64, 80, and 81.

[vdHW02] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proceedings of AAMAS '02*, pages 1167–1174, New York, NY, USA, 2002. ACM. Cited on page 157.

[vDK06] H. van Ditmarsch and B. Kooi. The secret of my success. *Synthese*, 153(2):339, November 2006. Cited on page 28.

[vDK08] H. van Ditmarsch and B. Kooi. Semantic results for ontic and epistemic change. In G. Bonanno, W. van der Hoek, and M. Wooldridge, editors, *Proceedings of LOFT 7*, pages 87–117, October 2008. Cited on page 34.

[vdMS99] R. van der Meyden and N. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proceedings of FSTTCS*, pages 432–445, 1999. Cited on pages 104 and 157.

[vdMS04] R. van der Meyden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proceedings of CSFW 2004*, pages 280–291. IEEE, 2004. Cited on pages 104, 146, and 149.

[vdMW07] R. van der Meyden and T. Wilke. Preservation of epistemic properties in security protocol implementations. In *Proceedings of TARK '07*, pages 212–221, 2007. Cited on page 143.

[vdPE04] J. van de Pol and M. V. Espada. Modal abstractions in μ-CRL. In *Proceedings of AMAST '04*, pages 61–64, 2004. Cited on pages 104 and 109.

[vDRV05] H. van Ditmarsch, J. Ruan, and L. C. Verbrugge. Model checking sum and product. In Shichao Zhang and R. Jarvis, editors, *Proceedings of AI 2005*, volume 3809 of *LNCS*, pages 790–795, 2005. Cited on page 5.

[vDvdHK03a] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Descriptions of game states*, volume 161 of *CSLI Lecture Notes*. CSLI Publications, 2003. Cited on pages 5 and 84.

[vDvdHK03b] H. van Ditmarsch, W. van der Hoek, B. Kooi, and 105-143. Concurrent dynamic epistemic logic. In V. F. Hendricks, K. F. Jørgensen, and S. A. Pedersen, editors, *Knowledge Contributors*, Synthese Library Series, pages 105–143. Kluwer Academic Publishers, 2003. Cited on page 60.

[vDvdHK03c] H. P. van Ditmarsch, W. van der Hoek, and B. P. Kooi. Concurrent dynamic epistemic logic for mas. In *Proceedings of AAMAS '03*, pages 201–208, New York, NY, USA, 2003. ACM. Cited on pages 22 and 60.

[vDvdHK07] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. (Synthese Library). Springer, 1st edition, November 2007. Cited on pages 3, 4, 21, 22, 23, 60, and 105.

[vDvdHvdMR06] H. van Ditmarsch, W. van der Hoek, R. van der Meyden, and J. Ruan. Model checking Russian cards. *Electronic Notes Theoretical Computer Science*, 149(2):105–123, 2006. Cited on pages 22 and 149.

[vE07] J. van Eijck. DEMO — a demo of epistemic modelling. In J. Benthem, Dov Gabbay, and Benedikt Löwe, editors, *Interactive Logic – Proceedings of the 7th Augustus de Morgan Workshop*, number 1 in Texts in Logic and Games. Amsterdam University Press, 2007. Available at http://www.cwi.nl/~jve/demo/. Cited on pages 5, 59, and 149.

[vEW08] J. van Eijck and Y. Wang. Propositional dynamic logic as a logic of belief revision. In *Proceedings of WOLLIC '08*, pages 136–148, 2008. Cited on page 8.

[vEWS10] J. van Eijck, Y. Wang, and F. Sietsma. Composing models. In *Proceedings of LOFT '10*, 2010. Cited on pages 8 and 80.

[VO07] J. Vaneijck and S. Orzan. Epistemic verification of anonymity. *Electronic Notes in Theoretical Computer Science*, 168:159–174, February 2007. Cited on pages 3 and 148.

[VW51] G. H. Von Wright. *An Essay in Modal Logic*. North Holland, Amsterdam, 1951. Cited on page 143.

[Wal00] I. Walukiewicz. Completeness of kozen's axiomatisation of the propositional -calculus. *Information and Computation*, 157(1-2):142–182, February 2000. Cited on page 84.

[Wan06] Y. Wang. Indexed semantics and its application in modelling interactive unawareness. Master's thesis, University of Amsterdam, 2006. Cited on pages 39 and 40.

[WKvE09] Y. Wang, L. Kuppusamy, and J. van Eijck. Verifying epistemic protocols under common knowledge. In *Proceedings of TARK '09*, pages 257–266, New York, NY, USA, 2009. ACM. Cited on pages 8, 34, and 39.

[WSvE10] Y. Wang, F. Sietsma, and J. van Eijck. Logic of information flow on communication channels (extended abstract). In van der Hoek, Kaminka, Lespérance, Luck, and Sen, editors, *Proceedings of AAMAS '10)*, 2010. Cited on pages 8, 55, and 154.

Abstract

This dissertation presents a logical investigation of epistemic protocols, focussing on protocol-dynamics, epistemic modelling, and epistemic model checking.

In Part I, we introduce logics for specifying epistemic protocols including their goals and their dynamics. Chapter 3 departures from the existing discussions about protocols in the field of Dynamic Epistemic Logic by introducing a logic which can specify both the epistemic protocols and their goals *within the language*. We formalize the verification problem of epistemic protocols under the assumption of meta knowledge about the intended goal. The subtlety of this verification problem is discussed in theory and examples. In Chapter 4, we address the question: "How can people get to know a protocol?" For this, we develop logics which are convenient for reasoning about knowledge change and protocol change. With various protocol-changing operators we can handle the dynamics of protocols and formalize how actions acquire new meanings as a result of protocol change. We show that all the three logics we introduced can be translated back to Propositional Dynamic Logic (PDL) on standard Kripke models, thus the techniques of modelling and model checking we develop in the other parts of the dissertation can be applied to these logics.

In Part II we address the issue of epistemic modelling, in order to study model checking for the logics introduced in Part I. In Chapter 5 we propose new composition operations on static and event models with arbitrary vocabularies, aiming at a compositional method for generating initial epistemic models. We prove decomposition theorems w.r.t. our new operator and demonstrate the use of our methods by various examples. Chapter 6 reports results on counting the number of different models given a finite set of initial assumptions. Restricted to image-finite models, we show that if a modal μ-calculus formula has an infinite model modulo bisimulation then it has $2^{\aleph_0}$ (cardinality of the continuum) different models modulo bisimulation. On the other hand, if it does not have any infinite models modulo bisimulation then all its models can be represented in a normal form.

Part III introduces abstraction techniques that are particularly useful on making

the model checking more efficient. A 3-valued semantics for Public Announcement Logic is defined and studied in Chapter 7 to facilitate abstractions of models. We define a relation with vocabulary and agent mappings between concrete models and their abstractions, thus making it possible to also abstract the signatures of models. We then give a logical characterization of this abstraction relation thus showing it is safe to check properties on the abstract model instead of the original concrete model. Chapter 8 studies the PDL on so-called *accelerated Kripke models* where the transitions in the models are labelled by regular expressions in order to obtain informative abstractions. By making use of a technique of regular expression rewriting, we analyse the complexity of the model checking and satisfiability problems of this logic and give a complete axiomatization.

In Part IV (Chapter 9) we survey the epistemic approaches to security protocol verification. We summarize the most important techniques in the Epistemic Temporal Logic and Dynamic Epistemic Logic approaches to security protocol verification, and compare these two approaches in term of convenience. We argue that some security properties can only be faithfully formalized by temporal logic with knowledge operators, but are not expressible by standard temporal logic. However, we need to pay some cost in model checking complexity, in exchange to the expressiveness we gain.

Samenvatting

Dit proefschrift behelst een logisch onderzoek van kennisgerelateerde protocollen, met aandacht voor protocol-dynamiek, voor epistemisch modelleren en voor het bevragen van epistemische modellen ('model checking').

In Deel I presenteren we logische systemen voor het specificeren van epistemische protocollen, met inbegrip van protocol-doel en protocol-verandering. Hoofdstuk 3 verruimt het perspectief ten opzichte van bestaande behandeling van protocollen in Dynamische Epistemische Logica, door een logica te introduceren die zowel het protocol als het doel van het protocol kan specificeren *in de logische taal zelf*. We formaliseren het verificatieprobleem voor epistemische protocollen onder de aanname van meta-kennis over het beoogde doel van het protocol. De subtiliteit van dit verificatieprobleem wordt geïllustreerd met theorievorming en in praktijkvoorbeelden. In Hoofdstuk 4 snijden we de vraag aan hoe mensen een protocol kunnen leren. Hiervoor worden logische systemen geïntroduceerd die geschikt zijn voor het redeneren over kennisverandering en over protocolverandering. Door gebruik te maken van verschillende operatoren om protocollen te veranderen kunnen we dynamiek van protocollen behandelen en kunnen we formaliseren hoe handelingen nieuwe betekenis krijgen als gevolg van verandering in een protocol. We laten zien dat elk van de drie logische systemen die we introduceren terugvertaald kan worden naar Propositionele Dynamische Logica (PDL) op standaard Kripke modellen. Hiermee is aangetoond dat de technieken die we in andere delen van het proefschrift ontwikkelen van toepassing zijn op de drie nieuwe logische systemen.

In Deel II richten we ons op epistemisch modelleren, met als doel het bestuderen van 'model checking' voor de logische systemen die we in Deel I hebben geïntroduceerd. In Hoofdstuk 5 stellen we nieuwe compositie-operatoren voor op statische modellen en op gebeurtenismodellen met willekeurig vocabulair, met als doel een compositionele methode te ontwikkelen voor het genereren van initiële kennismodellen. We bewijzen een aantal decompositie-stellingen voor de nieuwe operatoren, en we laten aan de hand van voorbeelden zien hoe onze methoden kunnen worden gebruikt. Hoofdstuk 6 rapporteert over resultaten met betrekking tot het aantal verschillende modellen dat kan worden verkregen, gegeven een eindige omschrijving van een begintoestand. Voor 'image-finite models' laten we zien dat als

een formule uit de modale μ-calculus een oneindig model heeft modulo bisimulatie, die formule $2^{\aleph_0}$ verschillende modellen heeft modulo bisimulatie (de cardinaliteit van het continuum). Aan de andere kant is het zo dat als een formule waarmee we beginnen *geen* oneindige modellen heeft modulo bisimulatie, dat wil zeggen als alle bisimulatie-minimale modellen van de formule eindig zijn, alle modellen voor die formule kunnen worden gerepresenteerd in een standaardvorm.

Deel III introduceert abstractie-technieken die van belang zijn om 'model checking' efficiënter te maken. In Hoofdstuk 7 wordt een driewaardige semantiek voor de logica van openbare aankondigingen ('public announcement logic') gedefiniëerd en bestudeerd. Het doel hiervan is om abstractie over modellen te vergemakkelijken. Met behulp van propositionele en agent afbeeldingen definiëren we een relatie tussen concrete modellen en hun abstracties. We laten daarmee zien dat het mogelijk is om te abstraheren van de signatuur van een model. We geven vervolgens een logische karakterisering van de abstractie relatie, en we tonen daarmee aan dat het veilig is om eigenschappen op het abstracte model te checken in plaats van op het originele concrete model. Hoofdstuk 8 bestudeert de PDL op zogenaamde *versnelde Kripke modellen* ('accelerated Kripke models'), waar de toestandsovergangen in de modellen geëtiketteerd zijn met reguliere uitdrukkingen die meer informatie geven dan de enkelvoudige etiketten uit gewone Kripke modellen. Met behulp van een herschrijf-techniek voor reguliere uitdrukkingen analyseren we de complexiteit van het 'model checking' probleem en het vervulbaarheidsprobleem voor deze logica, en geven we een volledige axiomatisering.

In Deel IV (Hoofdstuk 9) geven we een overzicht van de epistemische invalshoeken op het verificatieprobleem voor beveiligingsprotocollen. We vatten de belangrijkste technieken hiervoor uit epistemische temporele logica en uit dynamische epistemische logica samen, en we vergelijken de twee soorten van technieken. We beargumenteren waarom sommige veiligheidseigenschappen betrouwbaar kunnen worden geformaliseerd met temporele logica plus kennisoperatoren, maar niet met standaard temporele logica. De extra expressiviteit heeft echter een prijs: 'model checking' met epistemische temporele logica is complexer dan met standaard temporele logica.

Index