10

MONOGRAPH SERIES: W. JAMROGA - LOGICAL METHODS FOR SPECIFICATION AND VERIFICATION OF MULTI-AGENT SYSTEMS

10

MONOGRAPH SERIES

WOJCIECH JAMROGA

LOGICAL METHODS FOR SPECIFICATION AND VERIFICATION OF MULTI-AGENT SYSTEMS

INSTITUTE OF COMPUTER SCIENCE
POLISH ACADEMY OF SCIENCES

MONOGRAPH SERIES
INFORMATION TECHNOLOGIES: RESEARCH
AND THEIR INTERDISCIPLINARY APPLICATIONS
10

WOJCIECH JAMROGA

LOGICAL METHODS FOR SPECIFICATION AND VERIFICATION OF MULTI-AGENT SYSTEMS

INSTITUTE OF COMPUTER SCIENCE
POLISH ACADEMY OF SCIENCES

Warsaw, 2015

Publication issued as a part of the project:
"Information technologies: research and their interdisciplinary applications",
Objective 4.1 of Human Capital Operational Programme.
Agreement number UDA-POKL.04.01.01-00-051/10-01.

Publication is co-financed by European Union from resources of European Social Fund.

Project leader: Institute of Computer Science, Polish Academy of Sciences

Project partners: System Research Institute, Polish Academy of Sciences, Nałęcz
Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Sciences

Editors-in-chief: Olgierd Hryniewicz
Jan Mielniczuk
Wojciech Penczek
Jacek Waniewski

Reviewers: Wojciech Penczek

Wojciech Jamroga
Institute of Computer Science, Polish Academy of Sciences
w.jamroga@ipipan.waw.pl
http://krak.ipipan.waw.pl/~wjamroga/

Publication is distributed free of charge

ISBN 978-83-63159-25-2

Cover design: Waldemar Słonina

Contents

Chapter 1

Preface

Interacting autonomous agents become ubiquitous in present-day IT environments. More and more systems involve social as much as technological aspects, and even those that focus on technology are often based on distributed components exhibiting self-interested, goal-directed behavior. Moreover, the components (be it software processes, hardware devices, or humans) act in environments characterized by incomplete knowledge and uncertainty. The field of *multi agent systems (MAS)* studies the whole spectrum of phenomena ranging from agent architectures to communication and coordination in agent groups to agent-oriented software engineering. The theoretical foundations are mainly based on game theory and formal logic.

The framework of MAS has been used to formalize various problems in computer science, artificial intelligence, game theory, social choice theory, and other related disciplines. Depending on the context, multi agent systems are presented as a paradigm for computation, programming, and/or design. Perhaps most importantly, however, they can be seen a philosophical metaphor that provides a way of modeling the world, and makes one use specific vocabulary when describing the phenomena we are interested in. Putting it in another way, multi-agent systems form a paradigm for *thinking* and *talking* about the world, and assigning it a specific conceptual structure. Components of such systems are assumed to be autonomous, perhaps intelligent, definitely active or even pro-active... having some goals and beliefs... et cetera. Thus, the metaphor builds on the intuition that humans are agents, and that other entities we study can be just like us to some extent.

Logic-based methods for multi-agent systems have several advantages. Logic provides a vocabulary for naming properties of systems, and the vocabulary is given precise meaning via models and semantic rules. Moreover, logical models provide a conceptual apparatus for reasoning that can be as well used outside mathematical logic. In this book, we focus on modal logics with their clear and intuitively appealing conceptual machinery of *possible world semantics* (also known as *Kripke semantics*). The logics draw from the long tradition of philosophical studies on human behavior and the behavior of the world in general, that yielded epistemic logic, deontic logic, temporal logic etc. In particular, we investigate a branch of modal logics that can be described as *strategic logics*. That is, the generic modal structure is infused with game-theoretical notions of *player*, *coalition* , *choice*, *strategy*, *outcome*, *rationality*, etc. Since all those concepts are highly relevant for interaction between autonomous entities, it seems a perfect starting point for specification and verification of multi-agent systems.

It should be pointed out that modal logics for multi-agent systems (and their models) can be used in at least two ways. First, one may strive to represent an objective observer's view to an agent system with the instruments they provide. This is the viewpoint we usually adopt while talking about "specification", "design", "verification" etc. The observer (e.g., the designer or the administrator of the system) may collect all relevant aspects of the system in a Kripke model, and then derive or check certain properties of this model. Or, the designer can specify some desirable properties of a system, and then try to engineer a model in which those properties hold. On the other hand, a model can be also used to capture the *subjective* view of an agent to the reality he is acting in. In that case, the agent can ask about properties of the world via the properties of the model, or, more importantly, look for a strategy that makes some desirable property true in the model.

This book presents an overview of some important concepts in the area of logic-based specification and verification of multi-agent systems. We introduce modal logics that can be used to specify temporal, epistemic, and strategic properties of MAS. Moreover, we present basic verification algorithms for those logics, and discuss the computational complexity of the corresponding verification problems. Each section ends with a review of existing literature, and pointers to relevant further reading.

Acknowledgements

I would like to thank all the people who generously collaborated with me on various research undertakings over the years: Thomas Ågotnes, Luis Antunes, Marek Bednarczyk, Francesco Belardinelli, Nils Buling, Madalina Croitoru, Mehdi Dastani, Catalin Dima, Hans van Ditmarsch, Jürgen Dix, The Bui Duy, Valentin Goranko, Wiebe van der Hoek, Piotr Kaźmierczyk, Michał Knapik, Beata Konikowska, Mirek Kurkowski, Damian Kurpiewski, Jõao Leite, Sjouke Mauw, Artur Męski, Matthijs Melissen, Nello Murano, Peter Novák, Wiesiek Pawłowski, Wojciech Penczek, Jerzy Pilecki, Matei Popovici, Peter Ryan, Holger Schlingloff, Henning Schnoor, Inanc Seylan, Marija Slavkovik, Maciej Szreter, Masoud Tabatabaei, Leon van der Torre, Nicolas Troquard, Paolo Turrini, and Mike Wooldridge. I wouldn't be where I am now without all the interactions with them. And I am quite content of where I am. 🙂

Moreover, the materials presented in this book include fragments that had been prepared for various survey articles and book chapters together with other researchers, in particular Thomas Ågotnes, Nils Buling, Jürgen Dix, Valentin Goranko, Wojciech Penczek, and Mike Wooldridge. I am sure that some pieces written actually by *them* crept into this text and lent some splendor and quality to my humble scribbles.

Last but not least, I would like to thank my own private multi-agent system: my wife Iwona and our kids Filip, Alicja, Natalia, and Sonia (to say nothing of Ata the dog, Chilli the cat and Farfocel the yorkshire terrier). You put up with irregular working hours, weekends spent on proofchecking incomprehensible articles, holidays wasted on project proposals, and the rollercoaster life of a scientist. I only hope that those are made up by the advantages of researcher's life, namely: irregular working hours, long weekends after a deadline, scientific trips to places like Budapest, Madrid, and Lisbon, and the adventure of life on a rollercoaster. It wouldn't be even half as much fun without you!

The author gratefully acknowledges the financial support of the 7th Framework Programme of the European Union under the Marie Curie IEF project ReVINK (PIEF-GA-2012-626398).

About the author

Wojciech “Wojtek” Jamroga is an associate professor at the Polish Academy of Sciences, working on various aspects of multi-agent systems. His research is mainly focused on modal logics for reasoning about agents, verification of agent systems, and game-theoretic analysis of MAS. He obtained a PhD from University of Twente, the Netherlands in 2004, and completed his habilitation at Clausthal University of Technology, Germany in 2009. Wojtek Jamroga has coauthored over 70 refereed publications, and has been on the program committees of most important conferences and workshops on multi-agent systems. His teaching record includes multiple courses at ESSLLI (European Summer School in Logic, Language and Information) and EASSS (European Agent Systems Summer School), all of them on logical aspects of MAS.

Chapter 2

Introduction to Specification and Verification of Systems

2.1 Expressing properties of MAS

This section serves as an introduction of some fundamental concepts that we are going to use throughout.

2.1.1 Multi-Agent Systems

Multi-agent systems (*MAS*) are systems that involve several autonomous entities acting in the same environment. The entities are called *agents*. What is an agent? Despite numerous attempts to answer the question, there seems to be no conclusive definition. We assume that MAS are, most of all, a philosophical metaphor that induces a specific way of seeing the world. Thus, while some researchers present multi-agent systems as a paradigm for computation or design, we believe that primarily multi-agent systems form a new paradigm for *thinking* and *talking* about the world. As a consequence, the paradigm makes us use agent-oriented vocabulary when describing the phenomena we are interested in, and model them with a specific conceptual structure.

The metaphor of a multi-agent system builds on the intuition that *we* are agents – and that other entities we study or create can be just like us to a large extent.

Example 1 (Robots on a Rescue Mission: Scenario)

A group of k robots operates in a burning house in order to rescue people from the building. There are n people inside, and the house consists of m places. The state of each robot can be characterized by its status (alive or dead), current position, and an indication whether the robot is carrying some person (and, if so, which person). Similarly, a person can be characterized by its current status and position. If it is being carried by a robot, the information will be included in the robot's state. Each place can be burning, damaged, or still in a good shape.

Robots and people that are alive can try to move North, South, East or West. Robots can additionally Pick up a person or Lay it on the ground. Every agent can also decide to do nothing (action Wait).

Example 2 (Voting Scenario)

Citizens of Pneumonia are voting in the presidential election. There are n voters, each of them supposed to enter a voting booth at a polling station, select one of the candidates from the ballot, register their vote, and exit the polling station. There are also k coercers who can attempt to bribe or blackmail the voters into voting for a particular candidate. The coercers are, among other things, capable of intercepting unencrypted messages.

We will use the two scenarios to construct motivating examples throughout the book, and show how various tasks can be achieved by use of formal methods based on mathematical logic.

Features of agents

An agent can possibly be:

- *Autonomous:* operates without direct intervention of others, has some kind of control over its actions and internal state,
- *Reactive:* reacts to changes in the environment,
- *Pro-active:* takes the initiative,
- *Goal-directed:* acts to achieve a goal,
- *Social:* interacts with others (i.e., engages in cooperation, communication, coordination, competition, etc.),
- *Embodied:* has *sensors* and *effectors* to read from and make changes to the environment,
- *Intelligent:* ...whatever it means,
- *Rational:* always does the right thing.

It can be argued, however, that the above properties of agents are secondary: they are results of an introspection rather than primary assumptions we start with. So, is there any essential (and commonly accepted) feature of an agent? Yes. An agent *acts*.

The classical AI approach to intelligent agents describes an agent by defining its set of actions, set of perceptions, and a function

$$act : \text{set of percept sequences} \mapsto \text{set of actions},$$

thus assuming that agents act in a systematic way, according to a clear recipe. Note that, in game theory, such a function is called a *strategy*. In planning, it is called a *conditional plan*.

Logics for MAS

Formal logic, especially the way it is used in philosophy and computer science, can be seen as a framework for thinking and reasoning about systems. Describing a system formally makes one realise the implicit assumptions that inform our understanding of the problem domain. Then, we can investigate the assumptions, and accept or reject them. We can also relax some of the assumptions and still use a part of the formal and conceptual machinery. The logics that we will use in this book are sufficiently expressive for the properties that we need to address. Still, they are much simpler and more rigorous than the full language of mathematics.

We note that this view of logic comes close to the role of multi-agent systems as a metaphor for understanding and describing the world. Logic provides conceptual structures for *modeling* and *reasoning* about the world in a precise manner – and, occasionally, it also provides tools to do it automatically. The following generic ideas provide logical specifications with different computational flavors:

- *Verification*: check specification against implementation;
- Other decision problems: *validity*, *satisfiability*, *realizability*;
- *Executable specifications*;
- *Planning as model checking*.

In the context of MAS, closely related computational problems are: *game solving*, *mechanism design*, and *reasoning about games*. We will show that they all have natural interpretation in decision problems for agent logics. Typically, the starting point is to describe a desirable property in the logical language. Then, we can check if the property holds in the model of a given system, in all possible models of all possible systems, or in at least one possible system (that is hopefully synthesized and returned by the checking process). Or, conversely, one can specify a dangerous property, and check if the danger applies to the system at hand, all possible systems, or at least one vulnerable scenario.

Example 3 (Rescue Robots: Properties)

Below we list some desirable properties that may (or may not) hold in the Rescue Robots scenario:

♣ Each person in the building is safe;

♣ Each person will eventually be safe;

♣ Each person may eventually be safe, provided that they cooperate

♡ People inside the building know that the robots are outside

♡ Each robot knows the position of every person

♡ If the robots shared information, then each of them would know the position of every person

♠ The robots can rescue all the people in the building

♠ The robots can rescue all the people, and they know that they can

♠ The robots can rescue all the people, and they know how to do it

Example 4 (Voting: Properties)

Desirable properties for the Voting scenario:

Privacy: The system cannot reveal how a particular voter voted. Thus, privacy guarantees that the link between a voter and her vote remains secret.

Receipt-freeness: The voter does not gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way.

Coercion-resistance: The voter cannot cooperate with a coercer to prove to him that she voted in a certain way. Coercion resistance requires that the coercer cannot become convinced of how the voter has voted, even if the voter cooperates with him.

2.1.2 Modal Logic

Modal logic is an extension of classical logic with new operators $\Box$ (*necessity*) and $\Diamond$ (*possibility*). Formula $\Box\varphi$ says that statement φ is *necessarily true*, i.e., true in every possible situation. Similarly, $\Diamond\varphi$ means that φ is *possibly true*, that is, true in at least one possible situation. We will soon give examples of what "situations" can mean. However, independently of the precise definition, the following is usually assumed to hold:

$$\Diamond\varphi \leftrightarrow \neg\Box\neg\varphi.$$

Formally, the Kripke semantics of standard modal logic is defined as follows.

Definition 1 (Kripke model) *Let $\mathcal{PV} = p, p', p'', \dots$ be the set of* propositional variables *(also called* atomic propositions*). Models of modal logics are called* Kripke models *or* possible world models*, and include the set of* possible worlds *(or states) St, modal accessibility relation $\mathcal{R} \subseteq St \times St$, and interpretation of the propositions $\mathcal{V} : \mathcal{PV} \to 2^{St}$.*

Let $M = (St, \mathcal{R}, \mathcal{V})$ be a Kripke model, and q be a possible world in M. The truth of formula φ in M, q is given by the semantic relation $\models$, and defined by induction on the structure of φ:

$M, q \models p$ iff $q \in \mathcal{V}(p)$, for $p \in \mathcal{PV}$;

$M, q \models \neg\varphi$ iff not $M, q \models \varphi$ (often written $M, q \not\models \varphi$);

$M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$;

$M, q \models \varphi \vee \psi$ iff $M, q \models \varphi$ or $M, q \models \psi$;

$M, q \models \Box\varphi$ iff for every $q' \in St$ such that $q\mathcal{R}q'$, we have $M, q' \models \varphi$;

$M, q \models \Diamond\varphi$ iff for some $q' \in St$ such that $q\mathcal{R}q'$, we have $M, q' \models \varphi$.

Note that disjunction and modal possibility can be omitted from the primitive constructs of the logic, since we can define them as combinations of the other operators.

Modal logic can be further extended to multi-modal logic, where we deal with several modal operators: $\Box_i$ and $\Diamond_i$, each of them interpreted over the corresponding modal accessibility relation $\mathcal{R}_i \subseteq St \times St$.

What we have presented so far is pretty generic. The actual accessibility relations can capture various dimensions of the reality (and therefore give rise to different kinds of modal logics): knowledge ($\rightsquigarrow$ *epistemic logic*), beliefs ($\rightsquigarrow$ *doxastic logic*), obligations ($\rightsquigarrow$ *deontic logic*), actions ($\rightsquigarrow$ *dynamic logic*), time ($\rightsquigarrow$ *temporal logic*), ability ($\rightsquigarrow$ *strategic logic*) etc. In particular, various aspects of agents and agent systems can be naturally captured within this generic framework.

We will present examples of modal logic specifications in the next subsection.

2.1.3 Reasoning about Knowledge

Epistemic logic deals with specification and reasoning about agents' knowledge. Epistemic properties have simple interpretation in modal logic, and at the same time are very important for reasoning about interaction between agents. We will use epistemic logic here to present example properties, and show how they are evaluated in models of simple multi-agent systems.

Example 5 (Rescue Robots: Properties to express)

Some interesting epistemic properties that can be written for the rescue robots scenario are listed below:

- ♣ People inside the building know that there are some robots outside;
- ♣ Robot i knows that someone is still in the building;
- ♣ The robot knows that the person knows that it knows (so she will wait for rescue);
- ♣ All the robots know the position of every person;
- ♣ If the robots shared information, then each of them would know the position of every person;
- ♣ The robots have distributed knowledge, but not common knowledge (...whatever it means) that person j is in location l.

Example 6 (Rescue Robots: Properties to express)

- ♠ The coercer does not know how a particular voter has voted (so he doesn't have an effective threat);
- ♠ The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail);
- ♠ The coercer knows that the voter knows that the coercer does not know (hence he knows that blackmail doesn't make sense);
- ♠ ...and so on, ad infinitum. ☺

The basic epistemic logic which we consider here involves modal necessity operators for individual knowledge. For historical reasons and greater readability, one uses K_i rather than $\Box_i$ for knowledge modalities. $\mathrm{K}_i\varphi$ is interpreted as "agent i knows that

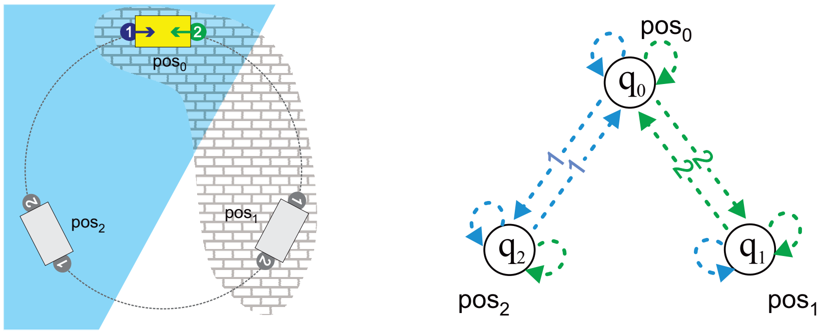


Figure 2.1: Two robots and a carriage: a schematic view (left) and a Kripke model M_{carr1} of the robots' knowledge (right).

φ." Additionally, one can consider modalities for collective knowledge of groups of agents: *mutual knowledge* ($\mathrm{E}_A\varphi$: "everybody in group A knows that φ"), *common knowledge* ($\mathrm{C}_A\varphi$: "all the agents in A know that φ, and they know that they know it, etc."), and *distributed knowledge* ($\mathrm{D}_A\varphi$: "if the agents shared their available information, they would be able to recognize that φ").

The formal semantics of epistemic logic is based on *multi-agent epistemic models* $M = \langle St, \sim_1, ..., \sim_k, \mathcal{V}\rangle$, where St is a set of *epistemic states*, $\mathcal{V}$ is a valuation of propositions, and each $\sim_i \subseteq St \times St$ is an equivalence relation that defines the *indistinguishability of states* for agent i.[1] Operators K_i are provided with the usual Kripke semantics given by the clause:

$$M, q \models \mathrm{K}_i\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim_i q'.$$

We mention in passing that epistemic models are usually constructed from the point of view of an *external omniscient observer*, most typically a system designer or analyst who has a complete view of the entire system.

Example 7 (Robots and Carriage) *Consider the scenario depicted in Figure 2.1. Two robots push a carriage from opposite sides. As a result, the carriage can move clockwise or anticlockwise, or it can remain in the same place – depending on who pushes with more force (and, perhaps, who refrains from pushing). To make our model of the domain discrete, we identify* 3 *different positions of the carriage, and associate them with states* q_0*,* q_1*, and* q_2*. We label the states by propositions* $\mathsf{pos}_0, \mathsf{pos}_1, \mathsf{pos}_2$*, respectively, to allow for referring to the current position of the carriage in the object language.*

Moreover, we assume that robot 1 *is only able to observe the color of the surface on which it is standing, and robot* 2 *perceives only the texture. As a consequence, the first robot can distinguish between position* 0 *and position* 1*, but positions* 0 *and* 2 *look the same to it. Likewise, the second robot can distinguish between positions* 0 *and* 2*, but not* 0 *and* 1*. In the resulting epistemic model, we have for instance that* $M_{carr1}, q_0 \models \neg\mathrm{K}_1\mathsf{pos}_0 \wedge \neg\mathrm{K}_1\mathsf{pos}_2 \wedge \mathrm{K}_1(\mathsf{pos}_0 \vee \mathsf{pos}_2)$*: the first robot knows that the position is either* 0 *or* 2*, but not which of them precisely. Moreover,* $M_{carr1}, q_0 \models \mathrm{K}_1\neg\mathsf{pos}_1$ *(robot* 1 *knows that the current position is not* 1*). The robot also knows that the other agent can*

[1] Symbol $\sim_i$ is used instead of $\mathcal{R}_i$ for historical reasons.

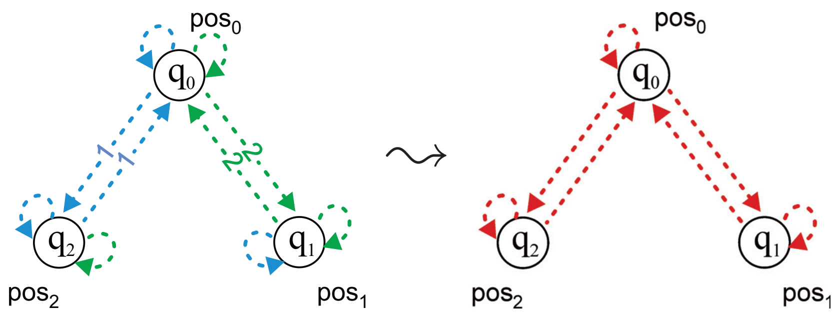


Figure 2.2: Deriving the relation of mutual knowledge for robots 1 and 2 (relation $\sim^{E}_{\{1,2\}}$ on the right hand side)

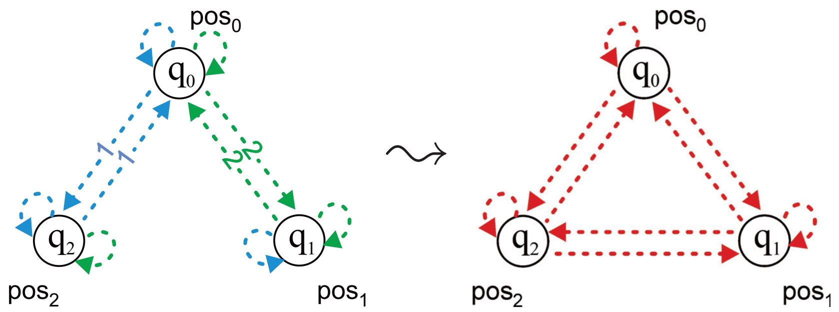


Figure 2.3: Deriving common knowledge ($\sim^{C}_{\{1,2\}}$)

currently distinguish between smooth and rough texture: $M_{carr1}, q_0 \models \mathrm{K}_1\big((\mathsf{pos}_2 \to \mathrm{K}_2\mathsf{pos}_2) \wedge (\neg\mathsf{pos}_2 \to \mathrm{K}_2\neg\mathsf{pos}_2)\big)$. *Finally, the robot knows that the other robot knows that it knows:* $M_{carr1}, q_0 \models \mathrm{K}_1\mathrm{K}_2\mathrm{K}_1\big((\mathsf{pos}_2 \to \mathrm{K}_2\mathsf{pos}_2) \wedge (\neg\mathsf{pos}_2 \to \mathrm{K}_2\neg\mathsf{pos}_2)\big)$.

We leave it to the reader to check that the above formulae indeed hold in the model.

How to interpret collective knowledge operators? The accessibility relation corresponding to E_A is defined as $\sim^{\mathrm{E}}_{A} = \bigcup_{i\in A} \sim_i$, and the semantics of E_A becomes

$$M, q \models \mathrm{E}_A\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim^{\mathrm{E}}_{A} q'.$$

Common knowledge C_A is given semantics in terms of the relation $\sim^{\mathrm{C}}_{A}$ defined as the transitive closure of $\sim^{\mathrm{E}}_{A}$:

$$M, q \models \mathrm{C}_A\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim^{\mathrm{C}}_{A} q'.$$

Finally, distributed knowledge D_A is given semantics in terms of the relation $\sim^{\mathrm{D}}_{A}$ defined as $\bigcap_{i\in A} \sim_i$, following the same pattern:

$$M, q \models \mathrm{D}_A\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for all } q' \text{ such that } q \sim^{\mathrm{D}}_{A} q'.$$

Note that $\mathrm{E}_A\varphi \equiv \bigwedge_{i\in A} \mathrm{K}_i\varphi$, and hence the operators for mutual knowledge can be also omitted from the language.

Example 8 (Robots and Carriage: Collective knowledge) *The mutual, common, and distributed knowledge relations of the two robots in the Robots & Carriage model are shown in Figures 2.2, 2.3, and 2.4, respectively. With the indistinguishability relations for collective knowledge in place, interpreting statements about collective knowledge of the robots is straightforward. For instance, in state* q_2 *(colorful surface, smooth texture), the robots do not have mutual knowledge about the current position of the carriage:* $M_{carr1}, q_2 \models \neg E_{\{1,2\}}\mathsf{pos}_2$. *On the other hand, they both know that the position is not* q_1 *(white surface, rough texture):* $M_{carr1}, q_2 \models E_{\{1,2\}}\neg\mathsf{pos}_1$. *Still, their collective knowledge does not extend to common knowledge:* $M_{carr1}, q_2 \models \neg C_{\{1,2\}}\neg\mathsf{pos}_1$. *Finally, if the robots shared their information, they would know the current position precisely:* $M_{carr1}, q_2 \models D_{\{1,2\}}\mathsf{pos}_2$.

We conclude this part by showing how properties of rescue robots and voting agents can be expressed in epistemic logic.

Example 9 (Voting: Expressing the properties)

♠ The coercer does not know how a particular voter has voted (so he doesn't have an effective threat):

$$\bigwedge_{c\in Candidates} \neg \mathrm{K}_{coerc}\mathsf{voted}_{\mathsf{v,c}}.$$

♠ The voter knows that the coercer does not know (so that she doesn't have to worry about blackmail):

$$\mathrm{K}_v\Big(\bigwedge_{c\in Candidates} \neg \mathrm{K}_{coerc}\mathsf{voted}_{\mathsf{v,c}}\Big).$$

♠ The coercer knows that the voter knows that the coercer does not know (hence he knows that blackmail doesn't make sense):

$$\mathrm{K}_{coerc}\mathrm{K}_v\Big(\bigwedge_{c\in Candidates} \neg \mathrm{K}_{coerc}\mathsf{voted}_{\mathsf{v,c}}\Big).$$

♠ ...and so on, ad infinitum:

$$\mathrm{C}_{coerc,v}\Big(\bigwedge_{c\in Candidates} \neg \mathrm{K}_{coerc}\mathsf{voted}_{\mathsf{v,c}}\Big).$$

Example 10 (Rescue Robots: Expressing the properties)

♣ People inside the building know that there are some robots outside:

$$\bigwedge_{j\in People} \mathrm{K}_j\Big(\bigvee_{i\in Robots} \mathsf{outside}_{\mathsf{i}}\Big).$$

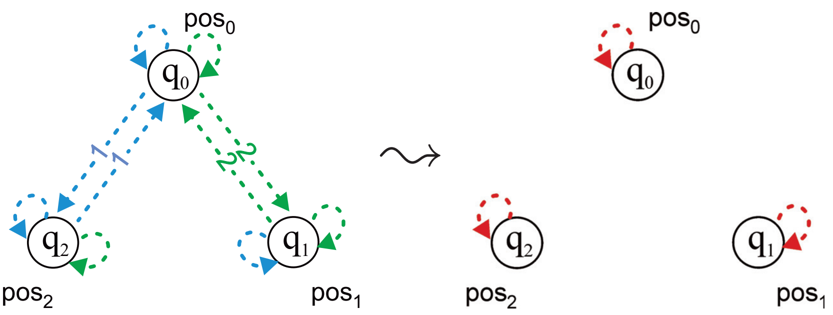


Figure 2.4: Deriving distributed knowledge ($\sim^D_{\{1,2\}}$)

♣ Robot i knows that someone is still in the building:

$$\mathrm{K}_i\Big(\bigvee_{j\in People} \mathsf{inside}_\mathsf{j}\Big).$$

♣ The robot knows that the person knows that it knows (so she will wait for rescue):

$$\mathrm{K}_i\mathrm{K}_j\mathrm{K}_i\ \mathsf{inside}_\mathsf{j}.$$

♣ All the robots know the position of every person:

$$\bigwedge_{j\in People}\ \bigvee_{l\in Locations} \mathrm{E}_{Robots}\ \mathsf{at}_{\mathsf{j,l}}.$$

♣ If the robots shared information, then each of them would know the position of every person:

$$\bigwedge_{j\in People}\ \bigvee_{l\in Locations} \mathrm{D}_{Robots}\ \mathsf{at}_{\mathsf{j,l}}.$$

♣ The robots have distributed knowledge, but not common knowledge, that person j is in location l:

$$\mathrm{D}_{Robots}\ \mathsf{at}_{\mathsf{j,l}} \ \wedge\ \neg\mathrm{C}_{Robots}\ \mathsf{at}_{\mathsf{j,l}}.$$

Why is the last distinction important? This is because only common knowledge guarantees successful coordinated action (cf. the Coordinated Attack Problem [7]). On the other hand, distributed knowledge means that it is possible to obtain knowledge by information exchange within the group. Thus, having distributed but not common knowledge suggests the *need for communication and information sharing* between members of the group.

References and Further Reading. The standard AI account of agents and agent models can be found in [146]. Readers interested in issues related to multi-agent systems are referred to [184, 182, 156]. Logical aspects of MAS have been covered, from various angles in the compendia [50, 166].

The modern treatment of modal logic has been proposed by Kripke in [106]. A gentle introduction to modal logic can be found in [23].

[63] presents what has become the standard treatment of reasoning about knowledge within the computer science community. More lightweight surveys can be found in [75, 169]. We also recommend a recent compendium on various aspects of epistemic reasoning [174].

2.2 Model Checking

There are several ways of applying formal logic to facilitate multi-agent systems. First, they can be used to support analysis of a problem domain and design of a new system, e.g., by providing intuitive conceptual structures and a systematic approach to specify desirable properties of the system. Secondly, one can use logic for verification and exploration of an existing MAS through model checking, theorem proving, or correctness testing. Thirdly, modal logics have been employed for automatic generation of behaviors by programming with executable specifications and automatic planning. The idea of planning as model checking also fall into this category. Finally, formal logic is often used in philosophy of mind and theories of agency to characterize mental attitudes, discuss concepts of rational agents, and test different rationality assumptions.

In this book, we focus especially on the second strand, and in particular on verification by model checking. We begin our exposition of model checking by a brief introduction to logical decision problems and their complexity.

2.2.1 Decision Problems

Theory of computation formalizes tasks like verification through so called *decision problems*. A decision problem asks, given representation of an instance of the problem, whether the instance belongs to the set of "good" instances. For example, the Hamiltonian path problem determines whether a path that visits each vertex exactly once exists in a given graph. Clearly, each decision problem can be seen as a yes/no question. The answer depends on the input, i.e., the actual values of the problem parameters.

Example 11 (Rescue Robots: Tasks to be done)

♣ Check whether robot i knows that someone is still in the building;

♣ Verify that if person j gets outside the building then she will never be in danger anymore;

♣ Check if all the robots in all rescue missions know the positions of every involved person;

♣ Show or disprove that a group of robots sharing information knows at least as much as its members.

Example 12 (Voting: Tasks to be done)

♠ Verify that the coercer does not know how a particular voter has voted;

♠ Design a system that does not issue receipts;

♠ Show (or disprove) that no system will ever reveal how a particular voter voted.

Typical problems for logical specifications are: validity, satisfiability, and model checking:

- *Validity*: given formula φ, determine if φ is valid (i.e., true in every model);
- *Satisfiability*: given formula φ, determine if φ is satisfiable (i.e., true in some model);
- *Model checking*: given formula φ and model M, determine if φ is true in M.

The usual view at decision problems is algorithmic: We want a *machine* (an algorithm) to answer the question captured by the problem. The user gives the input, and the machine returns the answer. Does that work? It depends on how difficult the question is, i.e., on the *computational complexity* of the problem.

2.2.2 Computational Complexity

In this book we consider both algorithms for verifying properties of MAS, and the theoretical complexity of the corresponding problems. Here is a short and rather informal description of the most relevant complexity classes. A formal introduction can be found in any textbook on complexity theory.

- $\mathbf{P}$ (deterministic polynomial time): problems solvable in *polynomially many steps* by a *deterministic* Turing machine;
- $\mathbf{NP}$ (nondeterministic polynomial time): problems solvable in *polynomially many steps* by a *nondeterministic* Turing machine. $\mathbf{NP}$ can be seen as the class of problems that will be solved fast if the algorithm makes shrewd guesses about how to search the space of possible solutions;
- **co-NP**: the complement of $\mathbf{NP}$. The class of problems that can be solved fast if the algorithm makes shrewd guesses about where to look for counterexamples;
- $\mathbf{\Delta_2^P} = \mathbf{P^{NP}}$: problems solvable in polynomial time by a deterministic Turing machine asking adaptive queries to an *oracle* for a problem in $\mathbf{NP}$ (i.e., a “guessing machine”);
- $\mathbf{\Sigma_2^P} = \mathbf{NP^{NP}}$: problems solvable in polynomial time by a nondeterministic Turing machine asking adaptive queries to an oracle for a problem in $\mathbf{NP}$. That is, one guessing machine is allowed to query another guessing machine;

- $\mathbf{\Pi_2^P} = \mathbf{co\text{-}\Sigma_2^P}$: the complement of $\mathbf{\Sigma_2^P}$;
- $\mathbf{\Delta_n^P}$ / $\mathbf{\Sigma_n^P}$ / $\mathbf{\Pi_n^P}$: problems solvable in polynomial time with use of adaptive queries to an n-level heap of oracles;
- $\mathbf{PH} = \bigcup_{n=1}^{\infty} \mathbf{\Sigma_n^P}$ (Polynomial Hierarchy): problems solvable by queries to a heap of oracles with bounded height (for any given bound);
- $\mathbf{PSPACE}$ (polynomial space): Problems solvable by a Turing machine that uses only polynomially many memory cells. Alternatively: problems solvable by queries to a heap of oracles with unbounded height;
- $\mathbf{EXPTIME}$: problems solvable in *exponential time* by a deterministic Turing machine;
- $\mathbf{k\text{-}EXPTIME}$: problems solvable by a deterministic Turing machine in *k-fold exponential time*, i.e., the number of steps is restricted by a function that "stacks" k exponents;
- $\mathbf{ELEMENTARY} = \bigcup_{k=1}^{\infty} \mathbf{k\text{-}EXPTIME}$: elementary problems (solvable in multiple-exponential time, i.e., time restricted by a function that stacks a bounded number of exponents for some given bound);
- $\mathbf{NONELEMENTARY}$: nonelementary problems (may require time expressed by an unbounded stack of exponents).

Of course, theoretical complexity has many deficiencies: it refers only to the worst (hardest) instance in the set, neglects coefficients in the function characterizing the complexity, etc. However, it often gives a good indication of the inherent hardness of the problem in terms of *scalability*. For low complexity classes, scaling up from small instances of the problem to larger instances is relatively easy. For high complexity classes, this is not the case anymore.

Also, the difference between deterministic and nondeterministic complexity classes draws the borderline between problems that can rely on brute force algorithms, and those that should use smart heuristics. In particular, the problems in $\mathbf{P}$ can in principle be solved efficiently by a brute force approach. $\mathbf{NP}$ collects problems that can be solved fast if one comes up with the right heuristics. Problems in $\mathbf{EXPTIME}$ do not scale even with smart heuristics; they are inherently exponential in terms of the time that they demand.

2.2.3 Local and Global Model Checking

There are various ways to verify the correctness of a system. By far, the most popular and successful is *model checking* which answers whether a given formula φ is satisfied in a state q of model M. Formally, *model checking* is the decision problem that determines membership in the set

$$\mathsf{MC}(\mathcal{L}, \mathsf{Struc}, \models) = \{((M,q),\varphi) \in \mathsf{Struc} \times \mathcal{L} \mid M,q \models \varphi\},$$

where $\mathcal{L}$ is a logical language, Struc is a class of (pointed) models for $\mathcal{L}$, and $\models$ is a semantic satisfaction relation compatible with $\mathcal{L}$ and Struc.

Algorithmically, this amounts to implementing the function

$$mcheck(M,q,\varphi) = \begin{cases} \top & \text{if} \quad M,q \models \varphi \\ \bot & \text{else} \end{cases}$$

that answers "true" if and only if formula φ holds in the pointed model (M, q). This variant of the problem is sometimes called *local model checking* to emphasize that the value of φ is computed for a particular state of the system.

It is often useful to compute the set of states in M that satisfy formula φ instead of checking if φ holds in a given state. This variant of the problem is known as *global model checking*. That is, we want to implement the function

$$mcheck(M, \varphi) = \{q \in St \mid M, q \models \varphi\}.$$

For the verification problems that we will consider here, the complexities of local and global model checking coincide, and the algorithms for one variant of model checking can be adapted to the other variant in a simple way. As a consequence, we will use both notions of model checking interchangeably.[2]

2.2.4 Verification of Epistemic Properties

Model checking epistemic logic is rather straightforward. Still, it is instructive to begin our study of verification algorithms with the simple case. Below, we present the recursive algorithm for local model checking of knowledge related properties. It is easy to notice that clauses of the algorithm simply map the respective semantic clauses of epistemic logic to an executable form.

```
function $mcheck(M, q, \varphi)$.
   Recursive model checking for formulae of epistemic logic.
   Returns $\top$ if $M, q \models \varphi$ and $\bot$ otherwise.
case $\varphi \equiv p$ : return ($q \in \mathcal{V}(p)$)
case $\varphi \equiv \neg\psi$ : return (not $mcheck(M, q, \psi)$)
case $\varphi \equiv \psi_1 \wedge \psi_2$ : return ($mcheck(M, q, \psi_1)$ and $mcheck(M, q, \psi_2)$)
case $\varphi \equiv \mathrm{K}_a\psi$ : return ($\forall q'$. if $q \sim_a q'$ then $mcheck(M, q', \psi)$)
end case
```

We omit the clause for disjunction, as it can be defined as a combination of conjunction and negation.

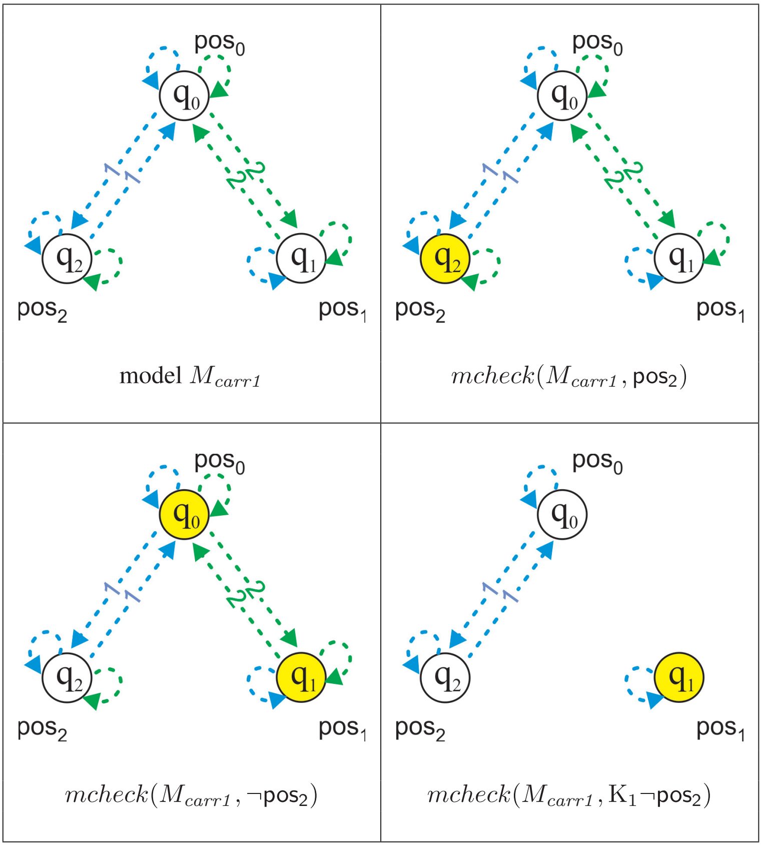
Instead of K_a, it is sometimes convenient to use the *"epistemic possibility"* operator $\overline{\mathrm{K}}_a$ (defined formally as $\overline{\mathrm{K}}_a\varphi \equiv \neg\mathrm{K}_a\neg\varphi$). Local model checking for $\overline{\mathrm{K}}_a\psi$ can be done as follows:

```
case $\varphi \equiv \overline{\mathrm{K}}_a\psi$ : return ($\exists q'$. $q \sim_a q'$ and $mcheck(M, q', \psi)$)
```

Global model checking for knowledge

An algorithm for global model checking takes a model and a formula as input, and returns the exact subset of states in the model that satisfy the formula. It can be obtained by mapping the logical operations from local model checking to their set-theoretic counterparts. That is, negation becomes the complement, conjunction turns into intersection, and so on. Modal operators are mapped to pre-images of the appropriate modal relation. More precisely, necessity operator K_i maps to the universal pre-image

[2] The only logic mentioned in this book, for which the complexities of global and model checking differ, is Constructive Strategic Logic in Chapter 6. However, we will not discuss model checking or CSL.



model M_{carr1}

$mcheck(M_{carr1}, \mathsf{pos}_2)$

$mcheck(M_{carr1}, \neg\mathsf{pos}_2)$

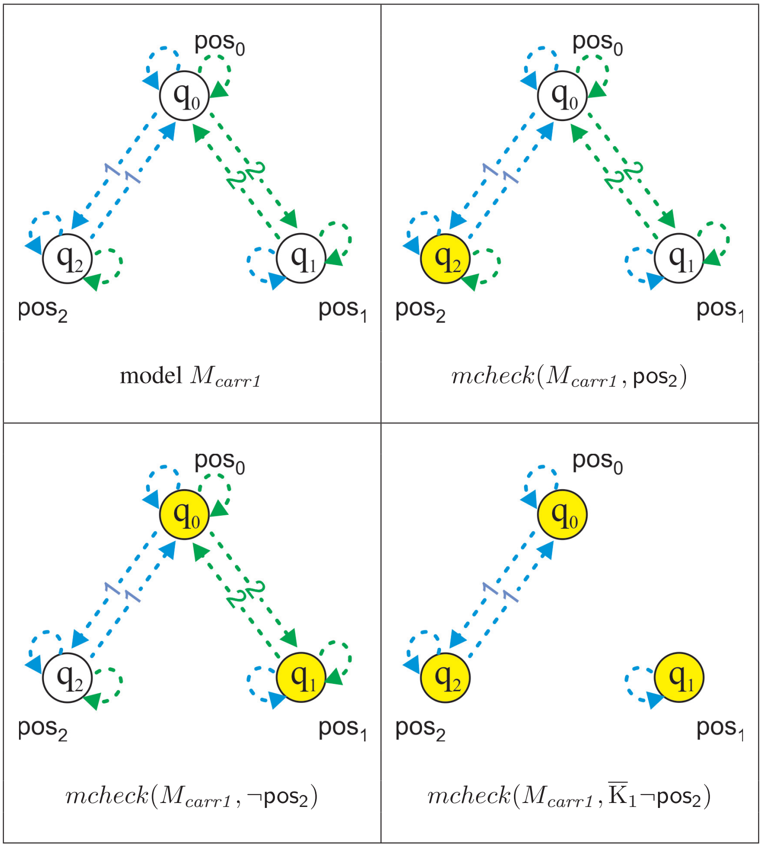
$mcheck(M_{carr1}, \mathrm{K}_1\neg\mathsf{pos}_2)$

Figure 2.5: Global model checking: $mcheck(M_{carr1}, \mathrm{K}_1\neg\mathsf{pos}_2)$

function $pre_\forall$ generated by relation $\sim_i$, and possibility operator $\overline{\mathrm{K}}_i$ maps to the existential pre-image function $pre_\exists$. Thus, the algorithm for model checking epistemic specifications is revised as follows:

```
function $mcheck(M, \varphi)$.
    Global model checking for formulae of epistemic logic.
    Returns the exact subset of states in $M$ for which formula $\varphi$ holds.
case $\varphi \equiv p$ : return $\mathcal{V}(p)$
case $\varphi \equiv \neg\psi$ : return $St \setminus mcheck(M, \psi)$
case $\varphi \equiv \psi_1 \wedge \psi_2$ : return $mcheck(M, \psi_1) \cap mcheck(M, \psi_2)$
case $\varphi \equiv \mathrm{K}_a\psi$ : return $pre_\forall(a, mcheck(M, \psi))$
case $\varphi \equiv \overline{\mathrm{K}}_a\psi$ : return $pre_\exists(a, mcheck(M, \psi))$
end case
```

$$pre_\forall(a, Q) = \{q \mid \forall q' \,.\, q \sim_a q' \Rightarrow q' \in Q\}$$
$$pre_\exists(a, Q) = \{q \mid \exists q' \,.\, q \sim_a q' \& q' \in Q\}$$



model M_{carr1}

$\mathit{mcheck}(M_{\mathit{carr1}}, \mathsf{pos}_2)$

$\mathit{mcheck}(M_{\mathit{carr1}}, \neg\mathsf{pos}_2)$

$\mathit{mcheck}(M_{\mathit{carr1}}, \overline{\mathrm{K}}_1\neg\mathsf{pos}_2)$

Figure 2.6: Model checking of robots & carriage: $\mathit{mcheck}(M_{\mathit{carr1}}, \overline{\mathrm{K}}_1\neg\mathsf{pos}_2)$

The set-theoretic operations of complement, intersection and pre-image will show up throughout the book as the main ingredients of model checking algorithms for more complex logics (temporal, strategic, etc.).

Example 13 (Robots and Carriage: Model checking) *Consider the Robots and Carriage model M_{carr1} in Figure 2.1. The execution of the global model checking algorithm for formula $\mathrm{K}_1\neg\mathsf{pos}_2$ is shown in Figure 2.5. Similarly, Figure 2.6 shows the execution of the algorithm for formula $\overline{\mathrm{K}}_1\neg\mathsf{pos}_2$ in M_{carr1}.*

Complexity of epistemic model checking

It is clear that the above algorithms run in relatively few steps. The following standard result states this in a formal way.

Theorem 1 *Model checking of the multi-agent epistemic logic K_n is* $\mathbf{P}$-complete *with respect to the size of the Kripke model and the length of the formula.*

The result looks appealing, but it is in fact rather imprecise. What kind of polynomial should we expect? And, more importantly, what do we mean by "the size of the

Kripke model" and "the length of the formula"? A more informative phrasing of the same result is given below.

Theorem 2 *Model checking of the multi-agent epistemic logic K_n is* $\mathbf{P}$-complete*, and can be performed in time $O(|M| \cdot |\varphi|)$ where $|M|$ is the number of the vertices and the edges in the Kripke model,*[3] *and $|\varphi|$ is the length of the formula, defined as the number of the subformulae in φ.*[4]

How does it compare to other decision problems for reasoning about knowledge? The following theorem indicates that validity and satisfiability checking are featured much higher in the complexity hierarchy:

Theorem 3 *Validity checking and satisfiability checking for the multi-agent epistemic logic K_n is* $\mathbf{PSPACE}$-complete *with respect to the number of distinct subformulae in the formula.*

Thus, validity and satisfiability checking seem much harder than model checking. Is that really the case? Not necessarily. One must remember that the complexity of a decision problem is always relative to the size of the input, and the three concerned problems do *not* have the same input. So, the only thing we can state for sure is that validity and satisfiability checking are much harder than model checking *for inputs of comparable size*.

We conclude the chapter by showing how the reasoning tasks from Section 2.2.1 for our motivating scenarios can be translated to appropriate decision problems.

Example 14 (Rescue Robots: Implementing the tasks)

♣ Check whether robot i knows that someone is still in the building:

> Model checking of formula $\mathrm{K}_i\big(\bigvee_{j\in People} \mathsf{inside_j}\big)$ in the model of the rescue mission;

♣ Verify that if person j gets outside the building then she will never be in danger anymore:

> Model checking of formula $\mathsf{AG}(\mathsf{outside_j} \to \mathsf{AGsafe_j})$ in the model of the rescue mission;[a]

♣ Check if all the robots in all rescue missions know the positions of every involved person:

> Validity checking of formula $\bigwedge_{j\in People} \bigvee_{l\in Locations} E_{Robots}\, \mathsf{at_{j,l}}$;

♣ Show or disprove that a group of robots sharing information knows at least as much as its members:

> Validity checking of formula $\bigwedge_{i\in A}(\mathrm{K}_i\varphi \to D_A\varphi)$.

[a] "AG" means "for all states that can be possibly reached from the current state." The construction will be formally introduced in Chapter 3.

[3] Note that, since the epistemic relations are reflexive, this can be actually simplified to "the number of the edges."

[4] $|\varphi|$ can be also defined as the number of the symbols in φ without affecting any of the results presented in this book.

Example 15 (Voting: Implementing the tasks)

♠ Verify that the coercer does not know how voter v has voted:

Model checking of formula $\bigwedge_{c \in \mathit{Candidates}} \neg K_{coerc} \mathsf{voted}_{\mathsf{v,c}}$ in the model of the voting protocol;

♠ Design a system that does not issue receipts:

Satisfiability checking of $\mathrm{AG}(\bigwedge_{v \in \mathit{Voters}} \bigwedge_{c \in \mathit{Candidates}} \neg \mathsf{receiptVote}_{\mathsf{v,c}})$;

♠ Show (or disprove) that no system will ever reveal how voter v voted:

Validity checking of $\mathrm{AG}(\bigwedge_{c \in \mathit{Candidates}} \neg \mathsf{revealedVote}_{\mathsf{v,c}})$.

References and Further Reading. Model checking was proposed in early 1980s by Clarke and Emerson [59, 45] and independently by Queille and Sifakis [141]. For a comprehensive treatment of model checking, we refer the interested reader to the textbooks [47, 15]. An introduction to computational complexity can be found e.g. in [126, 14]. A compact introduction to the basics of specification and verification for multi-agent systems was published in [98].

Chapter 3

Specification and Verification of Temporal Properties

3.1 Specification of System Dynamics

In this section, we present a brief overview of logics that refer to dynamics of systems. That is, logics that focus on actions which can be performed by (or in) a system, and which make the system evolve over time. We will first briefly consider the approach that takes actions as first-class citizens of the object language (called dynamic logic). Then, we will abstract from particular actions, and show how to reason about change in general and the evolution of the system over time (temporal logic).

Example 16 (Rescue Robots: Properties to express)

- ♣ Each person in the building is safe;
- ♣ Each person will eventually be safe;
- ♣ Each person may eventually be safe, if everything goes fine;
- ♣ Whenever person i gets in trouble, she will eventually be rescued;
- ♣ If person i gets outside the building, then she will never be in danger anymore;
- ♣ Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter.

Example 17 (Voting: Properties to express)

- ♠ The system will not reveal how a particular voter voted;
- ♠ The system does not issue receipts;
- ♠ The voter can vote, and can refrain from voting;
- ♠ The voter can vote, and can refrain from voting. If she votes, the system

will not reveal afterwards how she voted.

3.1.1 Modal Logics of Time and Action

Propositional Dynamic Logic

Propositional dynamic logic (**PDL**), which was primarily designed to reason about computer programs, is probably the most typical representative of logics with explicit actions. Actions are represented in the language by *action labels* $\alpha_1, \alpha_2, ...$ from a finite set Act. Complex action terms can be also constructed, for example sequential composition ($\alpha_1; \alpha_2$), nondeterministic choice ($\alpha_1 \cup \alpha_2$), finite iteration ($\alpha*$) etc. Now, $[\alpha]\varphi$ expresses the fact that φ is bound to hold after *every* execution of α, and $\langle\alpha\rangle\varphi \equiv \neg[\alpha]\neg\varphi$ says that φ holds after *at least one* possible execution of α. On the semantic side, we have *labeled transition systems* $M = \langle St, \xrightarrow{\alpha_1}, \dots, \xrightarrow{\alpha_k}, \mathcal{V}\rangle$, where actions are modeled as (nondeterministic) state transformations $\xrightarrow{\alpha_i} \subseteq St \times St$. The semantics of dynamic operators is given as follows:

$$M, q \models [\alpha_i]\varphi \quad \text{iff} \quad M, q' \models \varphi \text{ for all } q' \text{ such that } q \xrightarrow{\alpha_i} q'.$$

We mention **PDL** only in passing here, as it will not be used in the rest of the book.

Temporal Logic

Temporal logic leaves actions implicit, and instead focuses on possible patterns of evolution. Typical temporal operators are: X ("next"), G ("always"), F ("sometime"), and U ("strong until") that are used to build more complex formulae according to the following scheme:

$\mathrm{X}\varphi$	φ will be true in the *next* moment in time
$\mathrm{G}\varphi$	φ will be true in *all* future moments, from now on
$\mathrm{F}\varphi$	φ is true now, or will be true in *some* future moment
$\varphi\,\mathrm{U}\,\psi$	ψ will be true in some future moment, and φ will be true *until* the moment before ψ becomes true.

For example, formula $\mathrm{G}((\neg\mathsf{passport} \lor \neg\mathsf{ticket}) \rightarrow \mathrm{X}\neg\mathsf{board\_flight})$ expresses that, whenever a passenger arrives with no passport or no ticket, she will not be allowed to board the flight. Similarly, $\mathsf{send}(\mathsf{msg}, \mathsf{rcvr}) \rightarrow \mathrm{F}\mathsf{receive}(\mathsf{msg}, \mathsf{rcvr})$ says that if the message is sent to the receiver now, it will be eventually received.

Temporal Specification Templates

Temporal logic was originally developed in order to represent tense in natural language. Within computer science, it has achieved significant success in the *formal specification and verification of concurrent and distributed systems*. Much of the popularity was achieved because some useful concepts can be formally and concisely specified using temporal logics, namely:

- safety properties,

- liveness properties, and
- fairness properties.

Safety properties correspond to *maintenance goals*, and refer to a state of affairs that should be preserved (or avoided) throughout the lifespan of the system. That is, they correspond to statements like: “something bad will never happen” or “something good will always hold.” As an example, $\mathrm{G}\neg\mathsf{bankrupt}$ can be used to require that I will never go bankrupt. Similarly, $\mathrm{G}(\mathsf{fuelOK} \vee \mathrm{X}\mathsf{fuelOK})$ expresses that the fuel tank will never be empty for more than one time unit. Thus, the temporal template for safety properties is $\mathrm{G}\varphi$.

Liveness properties correspond to *achievement goals*, and refer to a state of affairs that should be achieved at some point in the future (“something good will happen”). As an example, $\mathrm{F}\mathsf{rich}$ can be used to express that I will eventually be rich. Better still, $\mathrm{FG}\mathsf{rich}$ requires that I will become and stay rich from some moment on. Finally, $\mathsf{requested} \rightarrow \mathrm{F}\mathsf{granted}$ says that if a resource is requested now, it will be granted sooner or later. Thus, the temporal template for liveness properties is $\mathrm{F}\varphi$.

Fairness properties correspond to *service goals*, and refer to qualities that should be provided sufficiently often. Typically, they correspond to statements like: “Whenever something is attempted/requested, it will be successful/granted.” For example, $\mathrm{GF}\mathsf{rich}$ can be used to express that even if I happen to be poor in the future, I will always recover and become rich again. Alternatively, it can be read as “I will be rich infinitely often.” $\mathrm{G}(\mathsf{attempt} \rightarrow \mathrm{F}\mathsf{success})$ says that whenever an action is attempted, it will eventually succeed. A weaker property, $(\mathrm{GF}\mathsf{attempt}) \rightarrow (\mathrm{GF}\mathsf{success})$, expresses that if the action is attempted infinitely often, it will also succeed infinitely often. Thus, the temporal templates for fairness properties are $\mathrm{GF}\varphi$ and $\mathrm{G}(\varphi \rightarrow \mathrm{F}\psi)$. Fairness properties are useful when specifying properties for scheduling processes, responding to messages, etc. Most importantly, they can be used to specify properties of the environment in which agents are embedded.

Models of Time

Models of temporal logic include one transition relation, and come in two versions. *Linear time models* define a total ordering on possible worlds (“time moments”), so that a model can be seen as a single infinite path λ with successive states $\lambda[0], \lambda[1], \ldots$. *Branching-time models*, on the other hand, consist of a tree that encapsulates all possible evolutions of the system. In a way, a linear time model is supposed to capture what *will* happen in the system from now on, whereas a branching time tree captures what *may* possibly happen. Such infinite models of the future are often called *behavioral structures* of temporal evolution.

Obviously, when reasoning about properties of a particular system, a model given in the form of an infinite path or an infinite tree would be rather impractical. Instead, the behavior of the system is usually represented by a Kripke transition model, sometimes also called the *computational structure*.

Definition 2 (Unlabelled transition system, Kripke transition model) *An* unlabelled transition system *is a pair $\langle St, \longrightarrow\rangle$ where St is a non-empty set of states, and $\longrightarrow \subseteq St \times St$ is a transition relation.*

A Kripke transition model *$\langle St, \longrightarrow, \mathcal{V}\rangle$ is an unlabeled transition system augmented by a valuation of atomic propositions.*

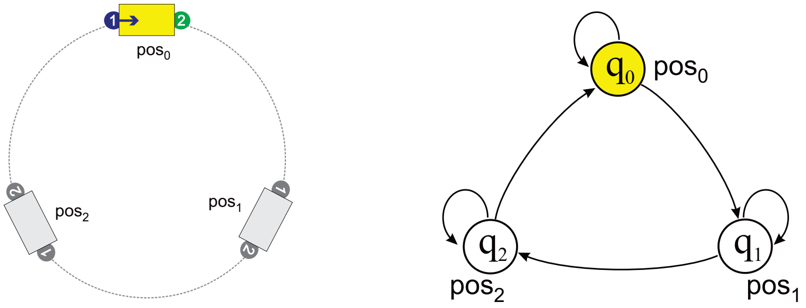


Figure 3.1: Robots and carriage: simple transition model M_{carr2}

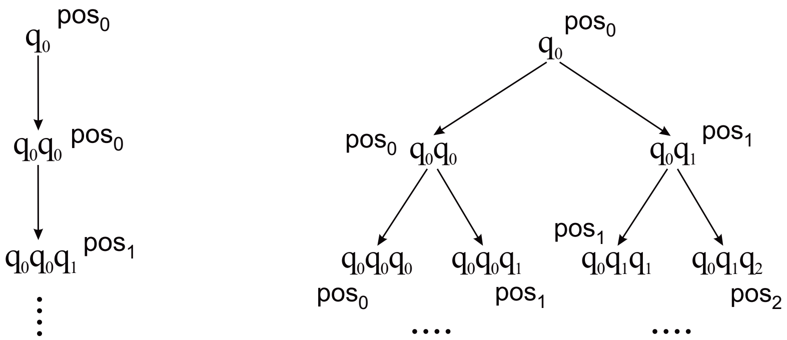


Figure 3.2: Behavioral structures for linear time (left) and branching time (right)

Example 18 (Robots & Carriage: Simple dynamics) *Consider the following variation of the Robots and Carriage scenario, depicted in Figure 3.1. Now, robot 1 can push the carriage so that it moves clockwise. It can also refrain from pushing, in which case the carriage does not move. Robot 2 has no influence on the position of the carriage. The corresponding Kripke transition model is shown on the right hand side of the picture.*

We show how the computational structure gives rise to behavioral models of linear and branching time in Figure 3.2. On the left hand side, the linear time model is presented for the carriage staying in position 0 for two moments, and then moving to position 1. The right hand side depicts the branching time model that assumes q_0 to be the initial state of the system.

Paths

Dynamics of the system for a particular chain of events is captured by a *path*. In case of computational systems, it is often referred to as a *computation path*. Formally, a path is a “complete” sequence of states that can be effected by subsequent transitions.

Definition 3 (Path in a transition system) *A* path λ *is a sequence of states that can* *be effected by subsequent transitions. A path must be* full*, i.e., either infinite or ending in a state with no outgoing transition.*

Usually, we assume that the transition relation is serial (i.e., time flows forever). Then, all paths are infinite.

3.1.2 Linear Temporal Logic

Let $\mathcal{PV}$ be the set of atomic propositions with the typical element p. The syntax of *linear time logic (*LTL*)* is formally defined as follows:

$$\gamma ::= p \mid \neg\gamma \mid \gamma \wedge \gamma \mid \mathrm{X}\gamma \mid \gamma \,\mathrm{U}\, \gamma.$$

Additional boolean operators (disjunction, implication etc.) are defined as usual. Moreover, one can define additional temporal operators for "sometime in the future" ($\mathrm{F}\gamma \equiv \top \,\mathrm{U}\, \gamma$), "always from now on" ($\mathrm{G}\gamma \equiv \neg\mathrm{F}\neg\gamma$), "weak until" ($\gamma_1 \,\mathrm{W}\, \gamma_2 \equiv \gamma_1 \,\mathrm{U}\, \gamma_2 \vee \mathrm{G}\gamma_1$). Another temporal operator for "release" is deifned as $\gamma_1 \,\mathrm{R}\, \gamma_2 \equiv \gamma_2 \,\mathrm{W}\, (\gamma_1 \wedge \gamma_2)$ or, equivalently, $\gamma_1 \,\mathrm{R}\, \gamma_2 \equiv \neg(\neg\gamma_1 \,\mathrm{U}\, \neg\gamma_2)$.

The semantics of **LTL** is typically defined in behavioral models, i.e., infinite paths drawn from executions of a Kripke transition model M. Let λ be a path in M. We define the necessary notation as follows:

- $\lambda[i]$ denotes the ith state on λ (starting from 0),
- $\lambda[i \dots j]$: denotes the part of λ between moments i and j, and
- $\lambda[i \dots \infty]$ is the suffix of λ starting from position i.

The semantic relation for **LTL** is given below:

$\lambda \models p$ iff $\lambda[0] \in \mathcal{V}(p)$ (i.e., p is true at moment $\lambda[0]$);

$\lambda \models \neg\gamma$ iff $\lambda \not\models \gamma$;

$\lambda \models \gamma_1 \wedge \gamma_2$ iff $\lambda \models \gamma_1$ and $\lambda \models \gamma_2$;

$\lambda \models \mathrm{X}\gamma$ iff $\lambda[1..\infty] \models \gamma$;

$\lambda \models \gamma_1 \,\mathrm{U}\, \gamma_2$ iff $\lambda[i..\infty] \models \gamma_2$ for some $i \geq 0$, and $\lambda[j..\infty] \models \gamma_1$ for all $0 \leq j < i$.

Thus, the semantics of the additional temporal operators can be derived as follows:

$\lambda \models \mathrm{F}\gamma$ iff $\lambda[i..\infty] \models \gamma$ for some $i \geq 0$;

$\lambda \models \mathrm{G}\gamma$ iff $\lambda[i..\infty] \models \gamma$ for all $i \geq 0$;

$\lambda \models \gamma_1 \,\mathrm{W}\, \gamma_2$ iff $\lambda[i..\infty] \models \gamma_1$ for all $i = 0, \dots, j-1$ where j is the first moment such that $\lambda[j..\infty] \models \gamma_2$, or $j = \infty$ otherwise;

$\lambda \models \gamma_1 \,\mathrm{R}\, \gamma_2$ iff $\lambda[i..\infty] \models \gamma_2$ for all $i = 0, \dots, j$ where j is the first moment such that $\lambda[j..\infty] \models \gamma_1$, or $j = \infty$ otherwise.

Example 19 (Robots & Carriage: Semantics of LTL) *Consider* $\lambda = (q_0q_1q_2)^\omega = q_0q_1q_2q_0q_1q_2q_0q_1q_2\dots$ *from the Robots and Carriage transition model in Figure 3.1.*

For that path, we have for instance that $\lambda \models \mathrm{F}\mathsf{pos}_2$ *(position 2 will be eventually achieved), and even* $\lambda \models \mathrm{GF}\mathsf{pos}_2$ *(position 2 will be achieved infinitely many times). However, it is not the case that the carriage will stop and stay in position 0 for good:* $\lambda \models \neg\mathrm{FG}\mathsf{pos}_2$.

When **LTL** is used for specifying properties of a particular system, the formulae are usually interpreted over all the infinite paths from a transition model of the system.

Definition 4 (Semantics of LTL in Kripke models) *Let* M *be a Kripke transition model,* q *a state in* M*, and* γ *a formula of* **LTL**. *We say that* γ *holds in* M, q *(written* $M, q \models \gamma$*) iff* $\lambda \models \gamma$ *for every path* λ *in* M *starting from* q.

Example 20 (Robots & Carriage: Semantics of LTL, ctd.) *For the Kripke transition model* M_{carr2} *in Figure 3.1, we have for instance that* $M_{carr2}, q_0 \not\models \mathrm{F}\mathsf{pos}_2$. *Note that this does not imply* $M_{carr2}, q_0 \models \neg\mathrm{F}\mathsf{pos}_2$*; on the contrary, it is also the case that* $M_{carr2}, q_0 \not\models \neg\mathrm{F}\mathsf{pos}_2$. *Similarly,* $M_{carr2}, q_0 \not\models \mathrm{G}\mathsf{pos}_0$ *and* $M_{carr2}, q_0 \not\models \neg\mathrm{G}\mathsf{pos}_0$.

An example formula that does hold in M_{carr2}, q_0 *is* $(\neg\mathrm{G}\mathsf{pos}_0) \rightarrow \mathrm{F}\mathsf{pos}_1$*: if at some point the carriage moves away from position 0, it must achieve position 1.*

Example 21 (Rescue Robots: Expressing the properties)

- Everybody is safe:
$$\bigwedge_{j \in People} \mathsf{safe_j}.$$
- Everybody will eventually be safe:
$$\bigwedge_{j \in People} \mathrm{F}\mathsf{safe_j}.$$
Another interpretation: $\mathrm{F}\big(\bigwedge_{j \in People} \mathsf{safe_j}\big)$.
- Everybody will always be safe, from some moment on:
$$\bigwedge_{j \in People} \mathrm{FG}\mathsf{safe_j}.$$
Equivalently: $\mathrm{FG}\big(\bigwedge_{j \in People} \mathsf{safe_j}\big)$.
- Everybody may eventually be safe, if everything goes fine:
Cannot be expressed in LTL!
- Whenever person j gets in trouble, she will eventually be rescued:
$$\mathrm{G}(\neg\mathsf{safe_j} \rightarrow \mathrm{F}\mathsf{safe_j}).$$
- If person j gets outside, she will never be in danger anymore:
$$\mathrm{G}(\mathsf{outside_j} \rightarrow \mathrm{G}\mathsf{safe_j}).$$
- Person j may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter:
Cannot be expressed in LTL!

Example 22 (Voting: Expressing the properties)

- The system will not reveal how a particular voter voted:

$$\mathrm{G}\big(\bigwedge_{c\in Candidates} \neg\mathsf{revealedVote}_{\mathsf{v},\mathsf{c}}\big).$$

- The system does not issue receipts:

$$\mathrm{G}\big(\bigwedge_{c\in Candidates} \neg\mathsf{receiptVote}_{\mathsf{v},\mathsf{c}}\big).$$

- The voter can vote, and can refrain from voting:
  Cannot be expressed in LTL!

- The voter can vote, and can refrain from voting. If she votes, the system will not reveal afterwards how she voted:
  Cannot be expressed in LTL!

3.1.3 Branching Time Logic CTL*

Linear temporal logic lacks the ability to distinguish between *necessary* and *possible* courses of action. Given a particular infinite path, **LTL** addresses what *will* happen. Given a Kripke transition model, it captures what *must* happen. However, it is sometimes also important to express that something *may* happen on at least one possible path.

Computation tree logic (**CTL***) extends **LTL** with *path quantifiers* E ("there is a path") and A ("for every path"). Formally, the language of **CTL*** is given as the set of all the state formulae φ (interpreted in the states of a model), defined using path formulae γ (interpreted on the paths of a model), by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi\wedge\varphi \mid \mathsf{E}\gamma,$$
$$\gamma ::= \varphi \mid \neg\gamma \mid \gamma\wedge\gamma \mid \mathrm{X}\gamma \mid \gamma\,\mathrm{U}\,\gamma.$$

The derived boolean and temporal operators are defined like in the previous sections. Additionally, we define $\mathsf{A}\gamma \equiv \neg\mathsf{E}\neg\gamma$.

The semantics of **CTL*** is typically defined in computational structures. Let $M = (St, \longrightarrow, \mathcal{V})$ be a Kripke transition model, and q a state in M. The semantics of **CTL*** state formulae is given by the following clauses:

$M, q \models p$ iff $q \in \mathcal{V}(p)$;

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;

$M, q \models \varphi_1 \wedge \varphi_2$ iff $M, q \models \varphi_1$ and $M, q \models \varphi_2$;

$M, q \models \mathsf{E}\gamma$ iff there is a path λ in M, starting from q, for which we have that $M, \lambda \models \gamma$.

The semantics of **CTL*** path formulae essentially copies the semantic clauses of **LTL**:

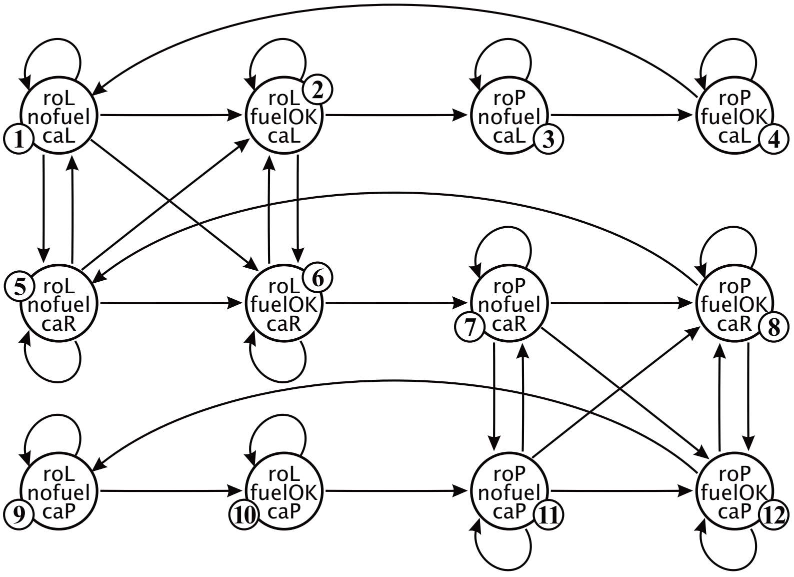


Figure 3.3: Simple Rocket model M_{rockt}

$$\begin{array}{lll}
M, \lambda \models \varphi & \text{iff} & M, \lambda[0] \models \varphi \quad \text{(i.e., } \varphi \text{ holds in state } \lambda[0] \text{ of model } M\text{);} \\
M, \lambda \models \neg\gamma & \text{iff} & M, \lambda \not\models \gamma; \\
M, \lambda \models \gamma_1 \wedge \gamma_2 & \text{iff} & M, \lambda \models \gamma_1 \text{ and } M, \lambda \models \gamma_2; \\
M, \lambda \models \mathrm{X}\gamma & \text{iff} & M, \lambda[1..\infty] \models \gamma; \\
M, \lambda \models \gamma_1 \,\mathrm{U}\, \gamma_2 & \text{iff} & M, \lambda[i..\infty] \models \gamma_2 \text{ for some } i \geq 0 \text{, and } M, \lambda[j..\infty] \models \gamma_1 \\
 & & \text{for all } 0 \leq j < i.
\end{array}$$

It is easy to see that the derived semantics of “for all paths” is as expected:

$$M, q \models \mathsf{A}\gamma \quad \text{iff} \quad \text{for all paths } \lambda \text{ in } M \text{ starting from } q \text{ we have } M, \lambda \models \gamma.$$

Example 23 (Robots & Carriage: Semantics of CTL*) *Recall model M_{carr2} in Figure 3.1. In that model, we have for instance $M_{carr2}, q_0 \models \mathsf{EFpos}_2$: in state q_0, there is a path such that the carriage will reach position 2 sometime in the future. The same is clearly not true for* all *paths, so we also have that $M_{carr2}, q_0 \models \neg\mathsf{AFpos}_2$.*

Example 24 (Simple Rocket Domain) *Figure 3.3 presents a Kripke model M_{rockt} for a simple variant of the* rocket domain *that used to be popular in the STRIPS planning community. A rocket can be either in its base in London (denoted by proposition* roL*) or at the aerodrome in Paris (*roP*). The rocket may also move between the aerodromes if it has fuel (*fuelOK*), but the flight uses up all fuel and empties the fuel tank (*nofuel*) until it is refilled. A piece of cargo can be either at the London base (*caL*), at the Paris base (*caP*), or inside the rocket (*caR*). When outside, it can be loaded into the rocket if they are in the same location. When inside, it can be unloaded.*

The following example formulae hold in every state of M_{rockt}:

- EFcaP*: there is a path such that eventually the cargo will be in Paris,*
- $\mathsf{AG}(\mathsf{roL} \vee \mathsf{roP})$*: in all the reachable states the rocket is in London or in Paris,*
- $\mathsf{roL} \rightarrow \mathsf{AX}(\mathsf{roP} \rightarrow \mathsf{nofuel})$*: if the rocket is in London, then, in all the successor states after the rocket has moved to Paris, the fuel will be used up.*

CTL (called sometimes "vanilla" **CTL** or "**CTL** without star") is a sublanguage of **CTL*** where every occurrence of a path quantifier is immediately followed by exactly one temporal operator:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{EX}\varphi \mid \mathsf{EG}\varphi \mid \mathsf{E}\varphi\,\mathrm{U}\,\varphi.$$

Note that in "vanilla" **CTL** it is not possible anymore to express $\mathrm{G}\varphi$ by a boolean combination of U-formulae. Thus, the case of G must be explicitly added to the syntax. **CTL** is important because it allows to interpret temporal formulae entirely in terms of their satisfaction in the *states*:

$M, q \models p$	iff	$q \in \mathcal{V}(p)$;
$M, q \models \neg\varphi$	iff	$M, q \not\models \varphi$;
$M, q \models \varphi_1 \wedge \varphi_2$	iff	$M, q \models \varphi_1$ and $M, q \models \varphi_2$;
$M, q \models \mathsf{EX}\varphi$	iff	there is a path λ starting from q such that $M, \lambda[1] \models \varphi$;
$M, q \models \mathsf{EG}\varphi$	iff	there is a path λ starting from q such that $\lambda[i] \models \varphi$ for all $i \geq 0$;
$M, q \models \mathsf{E}\varphi_1\varphi_2$	iff	there is a path λ starting from q such that $M, \lambda[i] \models \varphi_2$ for some $i \geq 0$ and $M, \lambda[j] \models \varphi_1$ for all $0 \leq j < i$.

As a consequence, reasoning in "vanilla" **CTL** is usually much easier to automatize than in **CTL***. We will present the standard **CTL** model checking algorithm in Section 3.2.

Example 25 (Rescue Robots: Expressing the properties)

♣ Everybody is safe:

$$\bigwedge_{j \in \mathit{People}} \mathsf{safe_j}.$$

♣ Everybody will eventually be safe:

$$\bigwedge_{j \in \mathit{People}} \mathrm{AF}\mathsf{safe_j}.$$

Another interpretation: $\mathrm{AF}\big(\bigwedge_{j \in \mathit{People}} \mathsf{safe_j}\big)$.

♣ Everybody will always be safe, from some moment on:

$$\bigwedge_{j \in \mathit{People}} \mathrm{AFG}\mathsf{safe_j}.$$

Equivalently: $\mathrm{AFG}\big(\bigwedge_{j \in \mathit{People}} \mathsf{safe_j}\big)$.

- ♣ Everybody may eventually be safe, if everything goes fine:

$$\bigwedge_{j\in People} \mathsf{EF}\mathsf{safe_j}.$$

Another interpretation: $\mathsf{EF}\big(\bigwedge_{j\in People}\mathsf{safe_j}\big)$.

- ♣ Whenever person j gets in trouble, she will eventually be rescued:

$$\mathsf{AG}(\neg\mathsf{safe_j} \to \mathsf{AF}\mathsf{safe_j}).$$

- ♣ If person j gets outside the building, then she will never be in danger anymore:

$$\mathsf{AG}(\mathsf{outside_j} \to \mathsf{AG}\mathsf{safe_j}).$$

- ♣ Person j may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter:

$$\mathsf{E}\big(\mathsf{F}\mathsf{safe_j} \wedge \mathsf{G}(\bigwedge_{i\in Robots}\mathsf{outside_i})\big) \;\wedge\; \neg\mathsf{A}\big(\mathsf{F}\mathsf{safe_j} \wedge \mathsf{G}(\bigwedge_{i\in Robots}\mathsf{outside_i})\big).$$

Alternative formalization:
$\mathsf{E}(\bigwedge_{i\in Robots}\mathsf{outside_i})\,\mathsf{U}\,\mathsf{safe_j} \;\wedge\; \neg\mathsf{A}(\bigwedge_{i\in Robots}\mathsf{outside_i})\,\mathsf{U}\,\mathsf{safe_j}$.

Example 26 (Voting: Expressing the properties)

- ♠ The system will not reveal how a particular voter voted:

$$\mathsf{AG}\big(\bigwedge_{c\in Candidates}\neg\mathsf{revealedVote_{v,c}}\big).$$

- ♠ The system does not issue receipts:

$$\mathsf{AG}\big(\bigwedge_{c\in Candidates}\neg\mathsf{receiptVote_{v,c}}\big).$$

- ♠ The voter can vote, and can refrain from voting:

Interpretation 1 The voter *may* vote, and *may* refrain from voting: $\mathsf{EF}\mathsf{voted_v} \wedge \mathsf{EG}\neg\mathsf{voted_v}$.
More refined specification:
$\mathsf{EF}(\bigvee_{c\in Candidates}\mathsf{voted_{v,c}}) \;\wedge\; \mathsf{EG}(\bigwedge_{c\in Candidates}\neg\mathsf{voted_{v,c}})$.

Interpretation 2 She *is able* to vote, and *able* to refrain from voting:

Cannot be expressed in CTL*!

- ♠ If the voter votes, the system will not reveal afterwards how she voted:

$$\bigwedge_{c\in Candidates}\mathsf{AG}\big(\mathsf{voted_{v,c}} \to \mathsf{AG}(\neg\mathsf{revealedVote_{v,c}})\big).$$

Temporal and dynamic dimensions have been combined with other modalities, e.g., in the well-known *BDI logics* of beliefs, desires, and intentions [48, 144]. We will present simple extension of **CTL** with epistemic modalities, and discuss verification of temporal-epistemic properties in Section 3.3.

3.1.4 Fixpoint Equivalences and Modal μ-Calculus

Fixpoint Characterization of CTL Operators

The following formulae are valid in **CTL**, i.e., true in every state of every model:

$$\begin{aligned}
\mathsf{E}\mathrm{F}\varphi &\leftrightarrow \varphi \vee \mathsf{EX}\,\mathsf{E}\mathrm{F}\varphi,\\
\mathsf{E}\mathrm{G}\varphi &\leftrightarrow \varphi \wedge \mathsf{EX}\,\mathsf{E}\mathrm{G}\varphi,\\
\mathsf{E}\varphi_1\,\mathrm{U}\,\varphi_2 &\leftrightarrow \varphi_2 \vee (\varphi_1 \wedge \mathsf{EX}\,\mathsf{E}\varphi_1\,\mathrm{U}\,\varphi_2),\\
\mathsf{A}\mathrm{F}\varphi &\leftrightarrow \varphi \vee \mathsf{AX}\,\mathsf{A}\mathrm{F}\varphi,\\
\mathsf{A}\mathrm{G}\varphi &\leftrightarrow \varphi \wedge \mathsf{AX}\,\mathsf{A}\mathrm{G}\varphi,\\
\mathsf{A}\varphi_1\,\mathrm{U}\,\varphi_2 &\leftrightarrow \varphi_2 \vee (\varphi_1 \wedge \mathsf{AX}\,\mathsf{A}\varphi_1\,\mathrm{U}\,\varphi_2).
\end{aligned}$$

What is the importance of fixpoint equivalences? Essentially, they say that paths satisfying **CTL** specifications can be constructed incrementally, step by step. Moreover, solutions to the verification problem, as well as satisfiability checking, can be obtained iteratively. The algorithmic view at computing states satisfying a temporal formula can be formalized through *logics of fixpoint computation*, i.e., variants of *modal μ-calculus*.

Modal μ-Calculus

The μ-calculus is an extension of propositional modal logic with the least and greatest fixpoint operators μ, ν. The main idea is to use denotational semantics of modal sentences, in which formulae are assumed to denote sets of states, and (unary) modal operators are interpreted as transformers of such sets. Then, for a transformer τ, formula $\mu Z.\tau(Z)$ refers to the least fixpoint of τ, that is, the smallest set of states Q such that $\tau(Q) = Q$. Analogously, $\nu Z.\tau(Z)$ denotes the greatest fixpoint of τ, i.e., the largest set of states Q such that $\tau(Q) = Q$.

The μ-calculus is usually built on top of **PDL** operators. Here, we present the simpler variant based on the **CTL** "nexttime" operators. Formally, let $\mathcal{PV}$ be a set of atomic propositions with typical element p, and let FV be a set of fixpoint variables with typical element Z. Elements of FV will serve as second-order variables that range over sets of states. The language of μ-calculus is defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{EX}\varphi \mid Z \mid \mu Z.\varphi(Z),$$

where $\varphi(Z)$ is a modal μ-calculus formula syntactically monotone in the fixed-point variable Z, i.e., all the free occurrences of Z in $\varphi(Z)$ fall under an even number of negations. Additionally, we can define $\mathsf{AX}\varphi \equiv \neg\mathsf{EX}\neg\varphi$, and $\nu Z.\varphi(Z) \equiv \neg\mu Z.\neg\varphi(\neg Z)$.

Let $M = (St, \longrightarrow, \mathcal{V})$ be a Kripke transition model. Additionally, let $\mathcal{E} : FV \to 2^{St}$ be the valuation of second-order variables (the "environment"). Notice that the set 2^{St} of all subsets of St forms a lattice under the set inclusion ordering. Each element $Q \subseteq St$ of the lattice can also be thought of as a *predicate* on St. The predicate is viewed as being true for exactly the states in Q. The least element in the lattice is the empty set, which we also refer to as $\bot$, and the greatest element in the lattice is the whole set St, which we sometimes write as $\top$. A function τ mapping 2^{St} to 2^{St} is called a *predicate transformer*. A set $Q \subseteq St$ is a *fixed point* (or *fixpoint*) of τ iff $\tau(Q) = Q$.

Now we can define the semantics of μ-calculus. We write $\llbracket \varphi \rrbracket_{M,\mathcal{E}}$ for the set of states that satisfy φ in model M for environment $\mathcal{E}$.

$$\begin{array}{lcl}
\llbracket p \rrbracket_{M,\mathcal{E}} & = & \mathcal{V}_M(p); \\
\llbracket Z \rrbracket_{M,\mathcal{E}} & = & \mathcal{E}(Z); \\
\llbracket \neg\varphi \rrbracket_{M,\mathcal{E}} & = & St_M \setminus \llbracket \varphi \rrbracket_{M,\mathcal{E}}; \\
\llbracket \varphi \wedge \psi \rrbracket_{M,\mathcal{E}} & = & \llbracket \varphi \rrbracket_{M,\mathcal{E}} \cap \llbracket \varphi \rrbracket_{M,\mathcal{E}}; \\
\llbracket \mathsf{EX}\varphi \rrbracket_{M,\mathcal{E}} & = & \{q \mid \text{ there is } q' \text{ such that } q \longrightarrow q' \text{ and } q' \in \llbracket \varphi \rrbracket_{M,\mathcal{E}}\}; \\
\llbracket \mu Z.\varphi(Z) \rrbracket_{M,\mathcal{E}} & = & \text{the smallest } Q \subseteq St \text{ such that } \llbracket \varphi(Z) \rrbracket_{M,\mathcal{E}[Z \leftarrow Q]} = Q
\end{array}$$

where $\mathcal{E}[Z \leftarrow Q]$ is like $\mathcal{E}$ except that it maps Z to Q. Consequently, the semantics of derived operators is as follows:

$$\begin{array}{lcl}
\llbracket \varphi \vee \psi \rrbracket_{M,\mathcal{E}} & = & \llbracket \varphi \rrbracket_{M,\mathcal{E}} \cup \llbracket \varphi \rrbracket_{M,\mathcal{E}}; \\
\llbracket \mathsf{AX}\varphi \rrbracket_{M,\mathcal{E}} & = & \{q \mid \text{ for all } q' \text{ such that } q \longrightarrow q' \text{ and } q' \in \llbracket \varphi \rrbracket_{M,\mathcal{E}}\}; \\
\llbracket \nu Z.\varphi(Z) \rrbracket_{M,\mathcal{E}} & = & \text{the largest } Q \subseteq St \text{ such that } \llbracket \varphi(Z) \rrbracket_{M,\mathcal{E}[Z \leftarrow Q]} = Q.
\end{array}$$

Finally, $M, q \models \varphi$ iff $q \in \llbracket \varphi \rrbracket_{M,\mathcal{E}}$ for every environment $\mathcal{E}$. Note that if φ contains only bounded second-order variables then $\llbracket \varphi \rrbracket_{M,\mathcal{E}}$ does not depend on $\mathcal{E}$, and we can use the notation $\llbracket \varphi \rrbracket_M$ without loss of generality.

The following results greatly facilitate computing fixed points:

Theorem 4 (Knaster-Tarski fixed point theorem) *If τ is a monotonic*[1] *transformer of state sets, then its least and greatest fixpoints exist, are unique, and can be obtained according to the following equations:*

$$\begin{array}{lcl}
\llbracket \mu Z.\tau(Z) \rrbracket_{M,\mathcal{E}} & = & \bigcap\{Q \subseteq St \mid \llbracket \tau(Z) \rrbracket_{M,\mathcal{E}[Z \leftarrow Q]} \subseteq Q\}; \\
\llbracket \nu Z.\tau(Z) \rrbracket_{M,\mathcal{E}} & = & \bigcup\{Q \subseteq St \mid \llbracket \tau(Z) \rrbracket_{M,\mathcal{E}[Z \leftarrow Q]} \supseteq Q\}.
\end{array}$$

Theorem 5 (Kleene fixed point theorem) *If τ is monotonic and $\bigcup$-continuous*[2] *then its least fixpoint can be obtained as follows:*

$$\mu Z.\tau(Z) \quad = \quad \bigcup_{i \geq 0} \tau^i(\bot).$$

If τ is monotonic and $\bigcap$-continuous[3] *then its greatest fixpoint can be obtained by:*

$$\nu Z.\tau(Z) \quad = \quad \bigcap_{i \geq 0} \tau^i(\top).$$

[1] τ is *monotonic* iff $S_1 \subseteq S_2$ implies $\tau(S_1) \subseteq \tau(S_2)$.

[2] τ is $\bigcup$-*continuous* iff $S_1 \subseteq S_2 \subseteq \ldots$ implies $\tau(\bigcup_{i \geq 0} S_i) = \bigcup_{i \geq 0} \tau(S_i)$.

[3] τ is $\bigcap$-*continuous* iff $S_1 \supseteq S_2 \supseteq \ldots$ implies $\tau(\bigcap_{i \geq 0} S_i) = \bigcap_{i \geq 0} \tau(S_i)$.

What does that mean in practice?

- In order to compute $[\![\mu Z.\varphi(Z)]\!]_M$, it suffices to start with the empty set of states, and keep adding "good" states until the set gets stable.
- In order to compute $[\![\nu Z.\varphi(Z)]\!]_M$, it suffices to start with the set of all states, and keep removing "bad" states until the set gets stable.

Embedding Temporal Logics in μ-Calculus

Modal μ-calculus is strictly more expressive than $\mathbf{PDL}$, $\mathbf{LTL}$, $\mathbf{CTL}$ and $\mathbf{CTL}^\star$. In particular, every formula of $\mathbf{CTL}^\star$ can be translated into μ-calculus. While the translation of full $\mathbf{CTL}^\star$ is rather involved, the basic $\mathbf{CTL}$ modalities can be translated to fixpoint formulae by following a simple scheme:

$$\begin{aligned}
tr(\mathsf{E}\mathrm{F}\varphi) &\equiv \mu Z\,.\,(tr(\varphi) \vee \mathsf{EX}Z);\\
tr(\mathsf{E}\mathrm{G}\varphi) &\equiv \nu Z\,.\,(tr(\varphi) \wedge \mathsf{EX}Z);\\
tr(\mathsf{E}\varphi_1\,\mathrm{U}\,\varphi_2) &\equiv \mu Z\,.\,(tr(\varphi_2) \vee tr(\varphi_1) \wedge \mathsf{EX}Z);\\[1ex]
tr(\mathsf{A}\mathrm{F}\varphi) &\equiv \mu Z\,.\,(tr(\varphi) \vee \mathsf{AX}Z);\\
tr(\mathsf{A}\mathrm{G}\varphi) &\equiv \nu Z\,.\,(tr(\varphi) \wedge \mathsf{AX}Z);\\
tr(\mathsf{A}\varphi_1\,\mathrm{U}\,\varphi_2) &\equiv \mu Z\,.\,(tr(\varphi_2) \vee tr(\varphi_1) \wedge \mathsf{AX}Z).
\end{aligned}$$

Thus, modal μ-calculus can be seen as a kind of "assembly language" for reasoning about time. On one hand, μ-calculus is more expressive and closer to actual algorithms. On the other hand, specifications written in logics such as $\mathbf{CTL}$ and $\mathbf{CTL}^\star$ are often much more readable, and express temporal properties in a more intuitive way.

References and Further Reading. Dynamic logic is treated extensively in [76]. Readers interested in temporal logic are referred to [58, 64] for an in-depth exposition. An introduction to μ-calculus can be found in [160].

Propositional modal μ-calculus was introduced by Kozen in [105]. Fixpoint translations of $\mathbf{PDL}$, $\mathbf{CTL}$ and $\mathbf{CTL}^\star$ can be found in [105, 57]. For the fixed point theorems, see [161].

3.2 Verification of Temporal Properties

In this section, we look at model checking of temporal logic. We start by presenting the standard fixpoint algorithm for global model checking of $\mathbf{CTL}$. Then, we discuss the most important complexity results for verification of temporal properties.

3.2.1 Fixpoint Model Checking for CTL

The standard model checking algorithm for $\mathbf{CTL}$ combines the ideas from verification of epistemic properties (see Section 2.2.4) with fixpoint computation (cf. Section 3.1.4). That is, verification of "one step" modalities proceeds by computing the appropriate *pre-image*, whereas model checking of long-term properties is done by computing the right *fixpoint*. The former applies to the "nexttime" operators EX, AX. The latter forms the backbone of the algorithm for long-term modalities $\mathsf{E}\,\mathrm{U}\,, \mathsf{A}\,\mathrm{U}$ (by

```
function $mcheckctl(M, \varphi)$.
    Global model checking for formulae of $\mathbf{CTL}$.
    Returns the exact subset of states in $M$ for which formula $\varphi$ holds.

case $\varphi \equiv p$ : return $\mathcal{V}(p)$
case $\varphi \equiv \neg\psi$ : return $St \setminus mcheckctl(M, \psi)$
case $\varphi \equiv \psi_1 \wedge \psi_2$ : return $mcheckctl(M, \psi_1) \cap mcheckctl(M, \psi_2)$
case $\varphi \equiv \mathsf{EX}\psi$ : return $pre_\exists(mcheckctl(M, \psi))$
case $\varphi = \mathsf{EG}\psi$ :
    $Q_1 := St;$    $Q_2 := mcheckctl(M, \psi);$
    while $Q_1 \not\subseteq Q_2$
    do $Q_1 := Q_2;$    $Q_2 := pre_\exists(Q_1) \cap Q_1$ od;
    return $Q_1$
case $\varphi = \mathsf{E}\psi_1 \,\mathsf{U}\, \psi_2$ :
    $Q_1 := \varnothing;$    $Q_2 := mcheckctl(M, \psi_1);$
    $Q_3 := mcheckctl(M, \psi_2);$
    while $Q_3 \not\subseteq Q_1$
    do $Q_1 := Q_1 \cup Q_3;$    $Q_3 := pre_\exists(Q_1) \cap Q_2$ od;
    return $Q_1$
end case
```

$$pre_\exists(Q) = \{q \mid \exists q'.q \longrightarrow q' \;\&\; q' \in Q\}$$

Figure 3.4: Fixpoint model checking for $\mathbf{CTL}$

computing the least fixpoint) as well as EG, AG (by computing the greatest fixpoint). The algorithm is shown in Figure 3.4. We also present the additional cases for universal path quantification in Figure 3.5.

Example 27 (Simple Rocket: Model checking) *Consider the simple rocket model M_{rockt} in Figure 3.3. Figure 3.6 shows the execution of the* $\mathbf{CTL}$ *model checking algorithm for formula* EFcaR *by computing the least fixpoint of $Z \vee (\mathsf{EX}Z)$. As it turns out, all the states in M_{rockt} satisfy the formula.*

Similarly, Figure 3.7 shows the computation of the greatest fixpoint for formula $\mathsf{AG}(\mathsf{roL} \vee \mathsf{caL})$ *which turns out to be satisfied in no state of M_{rockt}. Finally, Figure 3.8 presents the least fixpoint computation for formula* $\mathsf{E}(\mathsf{fuelOk}\ \mathsf{U}\ \mathsf{caR})$*, with the set of states $\{q_2, q_5, q_6, q_7, q_8, q_{12}\}$ as the output.*

3.2.2 Complexity Results

Let M be a Kripke transition model and q be a state in the model. Model checking a $\mathbf{CTL}$ formula φ in M, q determines whether $M, q \models \varphi$, i.e., whether φ holds in M, q. The same applies to model checking $\mathbf{CTL}^\star$. For $\mathbf{LTL}$, checking $M, q \models \varphi$ means that we check the validity of φ in the pointed model M, q, i.e., whether φ holds *on all the paths* in M that start from q (equivalent to $\mathbf{CTL}^\star$ model checking of formula $\mathsf{A}\varphi$ in M, q).

It has been known since the 1980s that formulae of $\mathbf{CTL}$ can be model-checked in time linear with respect to the size of the model and the length of the formula. This follows directly from the correctness of the algorithm presented in Figure 3.4. The size

```
case $\varphi \equiv \mathsf{A}\mathrm{X}\psi$ : return $pre_\forall(mcheckctl(M,\psi))$
case $\varphi = \mathsf{A}\mathrm{G}\psi$ :
   $Q_1 := St$;   $Q_2 := mcheckctl(M,\psi)$;
   while $Q_1 \not\subseteq Q_2$
   do $Q_1 := Q_2$;   $Q_2 := pre_\forall(Q_1) \cap Q_1$ od;
   return $Q_1$
case $\varphi = \mathsf{A}\psi_1 \,\mathrm{U}\, \psi_2$ :
   $Q_1 := \varnothing$;   $Q_2 := mcheckctl(M,\psi_1)$;
   $Q_3 := mcheckctl(M,\psi_2)$;
   while $Q_3 \not\subseteq Q_1$
   do $Q_1 := Q_1 \cup Q_3$;   $Q_3 := pre_\forall(Q_1) \cap Q_2$ od;
   return $Q_1$
end case
```

$$pre_\forall(Q) = \{q \mid \forall q'.q \longrightarrow q' \Rightarrow q' \in Q\}$$

Figure 3.5: Fixpoint model checking for $\mathbf{CTL}$: additional cases for operator A

of the model M, denoted by $|M|$, is defined by the sum of the number of its states and its transitions $|St| + |\longrightarrow|$.[4] The length of the formula φ, denoted by $|\varphi|$, is defined by the number of the subformulae in φ.

Moreover, the problem is not easier than $\mathbf{P}$, which can be for instance proven by a reduction of the tiling problem [150].

Theorem 6 *Model checking* $\mathbf{CTL}$ *is* $\mathbf{P}$*-complete, and can be performed in time* $\mathbf{O}(|M| \cdot |\varphi|)$.

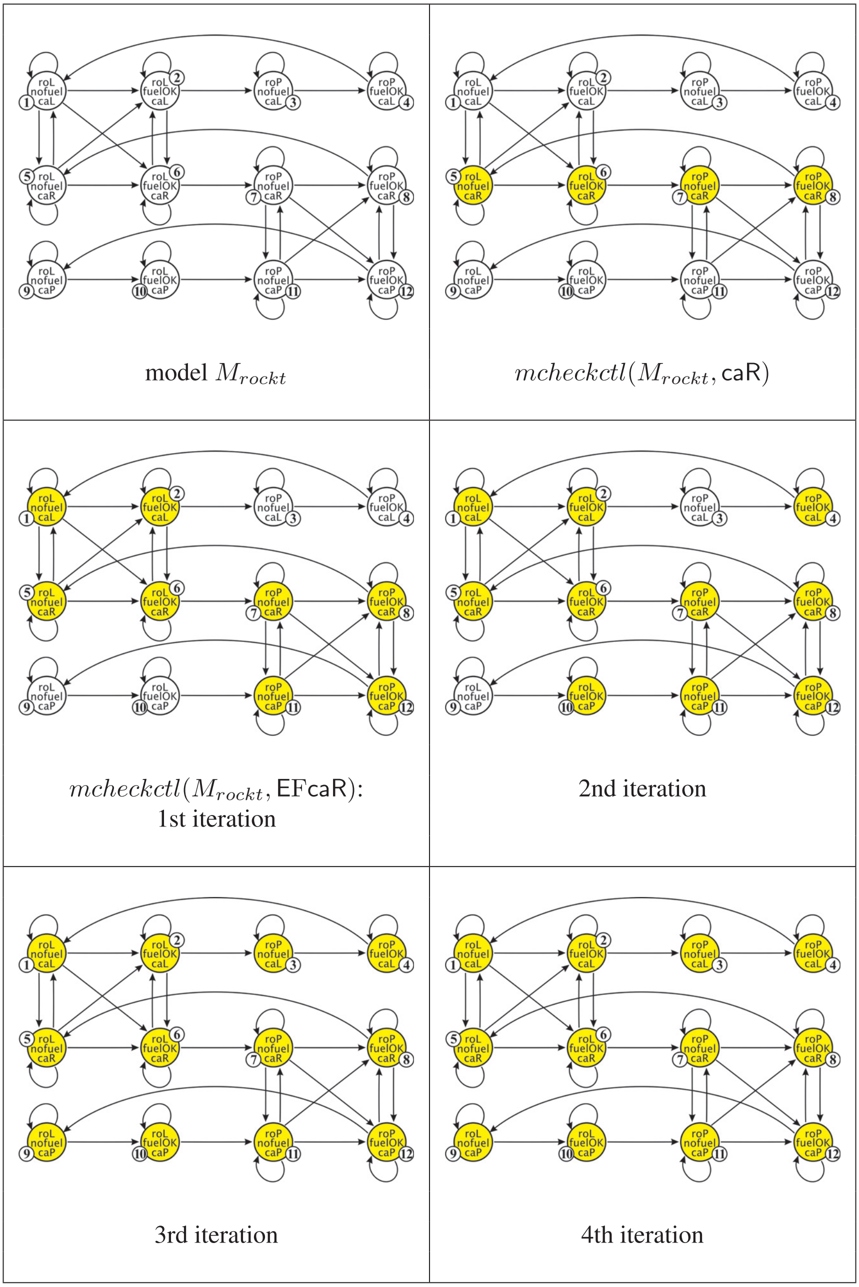
Formulae of $\mathbf{LTL}$ and $\mathbf{CTL}^\star$ are significantly harder to verify.

Theorem 7 *Model checking* $\mathbf{LTL}$ *is* $\mathbf{PSPACE}$*-complete, and can be performed in time* $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|M|)$.

The classical approach to $\mathbf{LTL}$ model checking is based on automata theory. Given an $\mathbf{LTL}$ formula γ, a Büchi automaton $\mathcal{A}_{\neg\gamma}$ of size $2^{\mathbf{O}(|\gamma|)}$ is constructed accepting exactly the paths satisfying $\neg\gamma$. The pointed Kripke model (M, q) is also interpreted as a Büchi automaton $\mathcal{A}_{M,q}$ of size $\mathbf{O}(|M|)$ accepting all possible paths in M starting from q. Then, model checking of γ in (M, q) reduces to the non-emptiness check of $\mathcal{L}(\mathcal{A}_{M,q}) \cap \mathcal{L}(\mathcal{A}_{\neg\gamma})$, which can be performed in time $\mathbf{O}(|M|) \cdot 2^{\mathbf{O}(|\gamma|)}$ by constructing the product automaton of size $\mathbf{O}(|M|) \cdot 2^{\mathbf{O}(|\gamma|)}$, and then checking the non-emptiness in linear time with respect to the size of the product automaton. A $\mathbf{PSPACE}$-hardness proof can be for instance found in [158].

The model checking algorithm for $\mathbf{CTL}^\star$ applies the $\mathbf{CTL}$ and $\mathbf{LTL}$ model checking technique recursively. Consider a $\mathbf{CTL}^\star$ formula φ which contains a state subformula $\mathsf{E}\gamma$, where γ is a pure $\mathbf{LTL}$ formula. We use the $\mathbf{LTL}$ model checking algorithm to determine the states that satisfy $\mathsf{E}\gamma$ (these are all states q in which the $\mathbf{LTL}$ formula $\neg\gamma$ is *not* true). We label them by a fresh propositional symbol, say p, replace $\mathsf{E}\gamma$ in

[4] Since the transition relation is serial, we can equivalently define the size of M by the number of the transitions only.



model M_{rockt}

$mcheckctl(M_{rockt}, \mathsf{caR})$

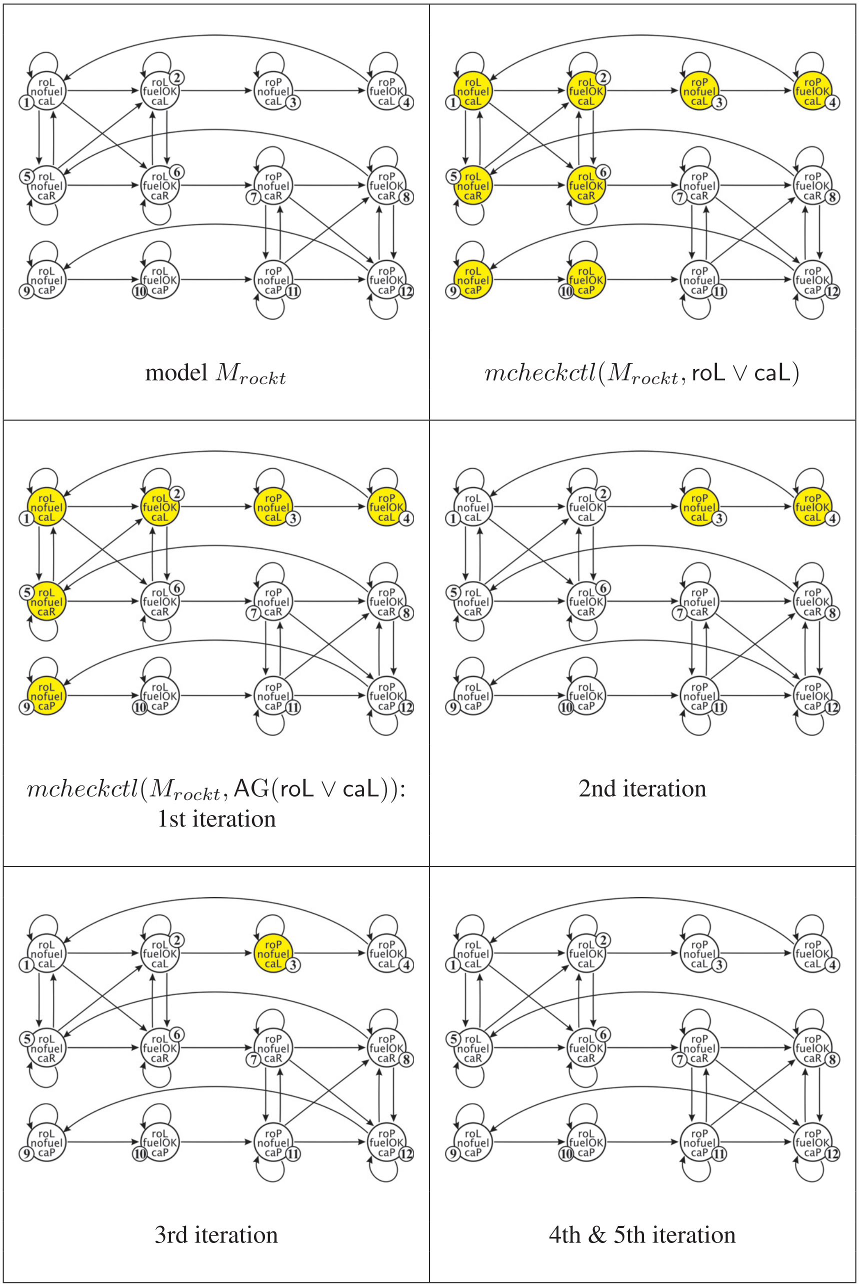
$mcheckctl(M_{rockt}, \mathsf{EFcaR})$:
1st iteration

2nd iteration

3rd iteration

4th iteration

Figure 3.6: Model checking **CTL**: $mcheckctl(M_{rockt}, \mathsf{EFcaR})$



model M_{rockt}

$mcheckctl(M_{rockt}, \mathsf{roL} \vee \mathsf{caL})$

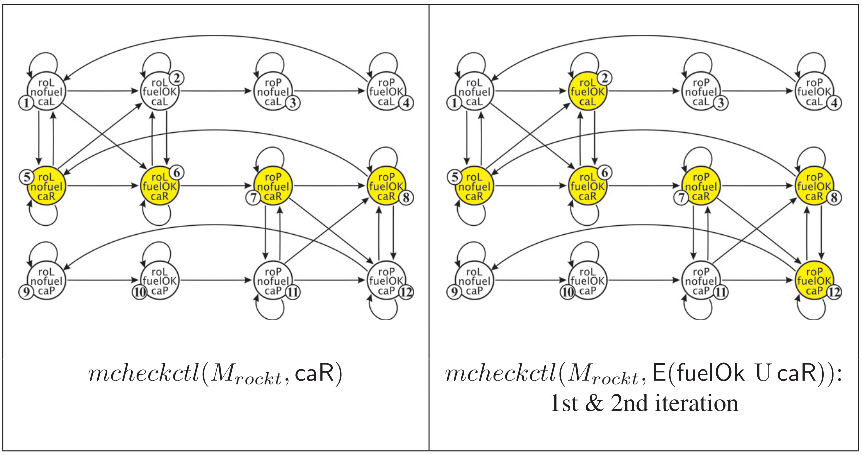
$mcheckctl(M_{rockt}, \mathsf{AG}(\mathsf{roL} \vee \mathsf{caL}))$:
1st iteration

2nd iteration

3rd iteration

4th & 5th iteration

Figure 3.7: Model checking **CTL**: $mcheckctl(M_{rockt}, \mathsf{AG}(\mathsf{roL} \vee \mathsf{caL}))$



$mcheckctl(M_{rockt}, \mathsf{caR})$

$mcheckctl(M_{rockt}, \mathsf{E}(\mathsf{fuelOk}\ \mathsf{U}\,\mathsf{caR}))$: 1st & 2nd iteration

Figure 3.8: Model checking **CTL**: $mcheckctl(M_{rockt}, \mathsf{E}(\mathsf{fuelOk}\ \mathsf{U}\,\mathsf{caR}))$

	$\|M\|, \|\varphi\|$
Epistemic logic	**P**-complete
PDL	**P**-complete
CTL	**P**-complete
LTL	**PSPACE**-complete
CTL*	**PSPACE**-complete

Figure 3.9: Basic complexity results: model checking

φ by p, and model-check the resulting **CTL*** formula. The procedure can be implemented by an oracle machine of type $\mathbf{P}^{\mathbf{PSPACE}} = \mathbf{PSPACE}$, since the **LTL** model checking algorithm is executed polynomially many times. As a consequence, we have the following.

Theorem 8 *Model checking* **CTL*** *is* **PSPACE**-*complete, and can be performed in time* $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|M|)$.

Figure 3.9 presents the basic complexity results for model checking of temporal and epistemic logics. For comparison, we also list the complexities of analogous satisfiability problems in Figure 3.10. The header of each table indicates the relevant complexity parameters. The input of the SAT problem is given by a logical formula, and its size is measured by the length of the formula. The input of the model checking problem is given by the formula and the model; the size of an input instance is measured as $|M| \cdot |\varphi|$, where $|M|$ is the size of the model, and $|\varphi|$ is the length of the formula.

References and Further Reading. Readers interested in the complexity of temporal model checking are referred to the excellent survey [150]. Additional information on verification of temporal logics can be also found in [47, 87, 132].

The idea of model checking was introduced in [59, 45] and independently in [141]. The basic algorithms and results for **CTL** were proposed in [45, 46]. Model checking

	$\lvert\varphi\rvert$
Epistemic logic	**PSPACE**-complete
PDL	**PSPACE**-complete
CTL	**EXPTIME**-complete
LTL	**PSPACE**-complete
$\mathbf{CTL}^\star$	**2EXPTIME**-complete

Figure 3.10: Basic complexity results: satisfiability

of **LTL** (especially automata-based) was studied in [158, 110, 178]. The complexity of model checking $\mathbf{CTL}^\star$ was established in [46, 61].

3.3 Combining Knowledge and Time

We have seen how one can use modal logic to reason about knowledge (Section 2.1.3) and the dynamics of a multi-agent system (Section 3). Similarly, one can address other dimensions of an agent system: beliefs, desires, obligations, and so on. One of the biggest advantages of modal logic is that it allows for *combining various dimensions* in a unified framework. In this section we discuss how the interaction between knowledge and time can be captured in modal logic. To this end, we combine epistemic and temporal dimensions in a multi-modal logic. We consider two variants of temporal-epistemic logics, namely $\mathbf{CTLK}^\star$ and **CTLK** (Section 3.3.1). Then, in Section 3.3.2, we discuss the idea of *interpreted systems* that provide a "grounded" methodology for modeling multi-agent systems, built on the modalities for agents' knowledge and the system's evolution. Finally, the model checking problem for temporal-epistemic logic is addressed in Section 3.3.3.

Example 28 (Rescue Robots: Properties to express)

- Person i may eventually be safe, and she knows about it;
- Person i knows that she may eventually be safe, and when she is, she will know about it;
- Each person will eventually be safe, and they know about it;
- Robot i may eventually know the position of every person;
- The robots may eventually have distributed knowledge that person j is in location l, but they will never obtain common knowledge about that;
- The robots can rescue a given person, and they know that they can;
- The robots can rescue a given person, and they know how to do it.

Example 29 (Voting: Properties to express)

Privacy: The coercer will never know how the voter has voted;

Receipt-freeness: The system does not issue pieces of information which can be used to convince the coercer that the voter voted in a certain way;

Coercion-resistance: The voter cannot cooperate with a coercer to prove to him that she voted in a certain way.

3.3.1 Temporal-Epistemic Logic

How to combine reasoning about the dynamics of the system state with properties of agents' mental states? The simplest, and quite effective, idea is to take a straightforward combination of the "component" logics.

CTLK*. In the most general syntactic variant, the language of temporal-epistemic logic includes all the operators of **CTL*** and standard epistemic logic, with knowledge operators joining the set of state formulae:

$$\varphi ::= p \mid \neg\varphi \mid \varphi\wedge\varphi \mid \mathrm{K}_a\varphi \mid \mathrm{C}_A\varphi \mid \mathrm{E}_A\varphi \mid \mathrm{D}_A\varphi \mid \mathsf{E}\gamma,$$
$$\gamma ::= \varphi \mid \neg\gamma \mid \gamma\wedge\gamma \mid \mathrm{X}\gamma \mid \gamma\,\mathrm{U}\,\gamma.$$

This syntax allows us to specify requirements on what agents will learn in the future ($\mathsf{A}\mathrm{F}\mathrm{K}_{r1}\mathsf{saved}_{\mathsf{pers1}}$), what they know now about things to come ($\mathrm{K}_{r1}\mathsf{A}\mathrm{F}\mathsf{saved}_{\mathsf{pers1}}$), what they know they will learn ($\mathrm{K}_{r1}\mathsf{A}\mathrm{F}\mathrm{K}_{r1}\mathsf{saved}_{\mathsf{pers1}}$), and so on.

The semantics is based on Kripke models which include both temporal and epistemic accessibility relations: $M = \langle St, \longrightarrow, \sim_1, \dots, \sim_k, \mathcal{V}\rangle$.

Formulae are interpreted according to semantic relation $\models$ defined by the union of the semantic clauses for **CTL*** from Section 3.1.3 and those for epistemic logic presented in Section 2.1.3:

$M,q \models p$	iff	$q \in \mathcal{V}(p)$;
$M,q \models \neg\varphi$	iff	$M,q \not\models \varphi$;
$M,q \models \varphi_1\wedge\varphi_2$	iff	$M,q \models \varphi_1$ and $M,q \models \varphi_2$;
$M,q \models \mathsf{E}\gamma$	iff	there is a path λ starting from q such that $M,\lambda \models \gamma$;
$M,q \models \mathrm{K}_a\varphi$	iff	for each $q' \in St$ such that $q \sim_a q'$, we have $M,q' \models \varphi$;
$M,q \models \mathcal{K}_A\varphi$	iff	for every $q' \in St$ such that $q \sim_A^{\mathcal{K}} q'$, we have $M,q' \models \varphi$ (for $\mathcal{K} = \mathrm{C},\mathrm{E},\mathrm{D}$);
$M,\lambda \models \varphi$	iff	$M,\lambda[0] \models \varphi$ (i.e., φ holds in state $\lambda[0]$ of model M);
$M,\lambda \models \neg\gamma$	iff	$M,\lambda \not\models \gamma$;
$M,\lambda \models \gamma_1\wedge\gamma_2$	iff	$M,\lambda \models \gamma_1$ and $M,\lambda \models \gamma_2$;
$M,\lambda \models \mathrm{X}\gamma$	iff	$M,\lambda[1..\infty] \models \gamma$;
$M,\lambda \models \gamma_1\,\mathrm{U}\,\gamma_2$	iff	$M,\lambda[i..\infty] \models \gamma_2$ for some $i \geq 0$, and $M,\lambda[j..\infty] \models \gamma_1$ for all $0 \leq j < i$.

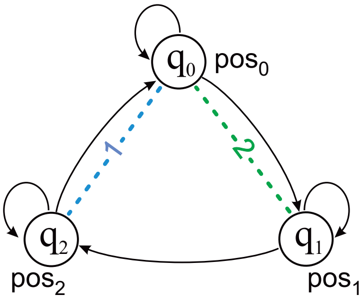


Figure 3.11: Robots and carriage: temporal-epistemic model M_{carr3}. Arrows indicate possible one-step transitions. Dotted lines indicate indistinguishability of states

Thus, formally speaking, the logic **CTLK*** is a *fusion* of **CTL*** and multi-agent epistemic logic.

Example 30 (Robots & Carriage: Temporal-epistemic properties) *Let M_{carr3} be the temporal-epistemic fusion of models M_{carr1} and M_{carr2} from Figures 2.1 and 3.1, respectively. The combined model is presented in Figure 3.11.*

Now, $M_{carr3}, q_0 \models \mathsf{EX} \bigvee_{i=0,1,2}(\mathsf{pos_i} \rightarrow K_1\mathsf{pos_i})$: in state q_0, it is possible that in the next moment robot 1 will know its position precisely. Moreover, $M_{carr3}, q_0 \models K_1\mathsf{EF}\mathsf{pos_2}$: robot 1 knows that the carriage can eventually get to position 2. On the other hand, the same is not true if we require the robot to see *that the system got to position 2, i.e., $M_{carr3}, q_0 \not\models K_1\mathsf{EF}K_1\mathsf{pos_2}$.*

"Vanilla" CTLK. **CTLK** is the syntactic fragment of **CTLK*** where each path quantifier is coupled with exactly one temporal operator. As before, the importance of **CTLK** stems from the fact that its semantics can be given entirely in terms of states, rather than infinite paths in the model. This will again prove advantageous when constructing verification algorithms for **CTLK**, see Section 3.3.3.

Example 31 (Robots & Carriage: CTLK) *Consider again the model in Figure 3.11. The following* **CTLK** *formula expresses that both robots might eventually (simultaneously) know the position of the carriage: $\mathsf{EF}\big((\bigvee_i K_1\mathsf{pos_i}) \wedge (\bigvee_i K_2\mathsf{pos_i})\big)$. Unfortunately, the property does not hold in the Robots & Carriage scenario, i.e.,*

$$M_{carr3}, q_0 \not\models \mathsf{EF}\big((\bigvee_i K_1\mathsf{pos_i}) \wedge (\bigvee_i K_2\mathsf{pos_i})\big).$$

A property that does hold says that both robots might learn the position but not necessarily at the same moment:

$$M_{carr3}, q_0 \models \mathsf{E}\big(\mathrm{F}(\bigvee_i K_1\mathsf{pos_i}) \wedge \mathrm{F}(\bigvee_i K_2\mathsf{pos_i})\big).$$

Notice, however, that the formula does not *belong to "vanilla"* **CTLK**. *Still, there is a* **CTLK** *formula that expresses exactly the same property, and indeed it holds in our scenario:*

$$M_{carr3}, q_0 \models \mathsf{EF}\big((\bigvee_i K_1\mathsf{pos_i}) \wedge \mathsf{EF}(\bigvee_i K_2\mathsf{pos_i}) \;\vee\; (\bigvee_i K_2\mathsf{pos_i}) \wedge \mathsf{EF}(\bigvee_i K_1\mathsf{pos_i})\big).$$

Similarly to $\mathbf{CTLK}$, one can also consider other syntactic restrictions of $\mathbf{CTLK}^\star$, such as $\mathbf{LTLK}$.

3.3.2 Interpreted Systems

The above formulation is pretty abstract. In particular, it does not indicate how the interplay between mental states of agents should be represented in a Kripke model. The best known proposal in this respect is that of *interpreted systems* which build on a notion of *local states*, defined formally as follows.

Let $\mathbb{A}\mathrm{gt} = \{1, \dots, k\}$ be the set of all agents. Each agent $i \in \mathbb{A}\mathrm{gt} = \{1, \dots, k\}$ has a set of its *local states* St_i that represent different "states of mind" of the agent. Additionally, we assume a set of *environment states* St_e that capture different states of the "external reality." Now, a *global state* of the system is represented by a tuple of agents' local states plus a state of the environment: $\langle q_1, \dots, q_k, q_e \rangle$. Thus, the global state space St is a subset of $St_1 \times \cdots \times St_k \times St_e$. It has been frequently argued that interpreted systems provide a more grounded semantics for agents' knowledge than abstract Kripke models, because starting from the local state spaces makes it clearer how the temporal-epistemic model should be made up.

It is usually assumed in interpreted systems that each agent has access only to its own local state, i.e.:

$$\langle q_1, \dots, q_k, q_e \rangle \sim_i \langle q_1', \dots, q_k', q_e' \rangle \text{ iff } q_i = q_i'.$$

The temporal dimension is added by considering *runs*, i.e., sequences of global states:

$$r : \mathbb{N} \to St.$$

A *system* is a set $\mathcal{R}$ of such runs. An *interpreted system* is a system plus a valuation of propositions: $\mathcal{I} = \langle \mathcal{R}, \mathcal{V} \rangle$. Given an interpreted system $\mathcal{I}$, a *point* in $\mathcal{I}$ is a pair $\langle r, m \rangle$ where r is a run and $m \in \mathbb{N}$ refers to a time moment on r. Epistemic equivalence between points is defined as follows:

$$\langle r, m \rangle \sim_i \langle r', m' \rangle \text{ iff } r(m) \sim_i r'(m').$$

Now, all epistemic modalities can be interpreted as before, e.g.:

$$\mathcal{I}, r, m \models \mathrm{K}_i\varphi \quad \text{iff} \quad \mathcal{I}, r', m' \models \varphi \text{ for all } \langle r', m' \rangle \text{ such that } \langle r, m \rangle \sim_i \langle r', m' \rangle.$$

Moreover, the standard temporal operators of $\mathbf{LTL}$ can be interpreted as follows:

$$\mathcal{I}, r, m \models \mathrm{X}\varphi \quad \text{iff} \quad \mathcal{I}, r, m+1 \models \varphi,$$

$$\mathcal{I}, r, m \models \varphi \,\mathrm{U}\, \psi \quad \text{iff} \quad \mathcal{I}, r, m' \models \psi \text{ for some } m' > m \text{ and } \mathcal{I}, r, m'' \models \varphi \text{ for all } m'' \text{ such that } m \le m'' < m'.$$

Finally, the semantics of $\mathbf{CTL}^\star$ path quantifiers can be defined in interpreted systems, too:

$$\mathcal{I}, r, m \models \mathsf{E}\varphi \quad \text{iff} \quad \text{there is } r' \text{ such that } r'[0..m] = r[0..m] \text{ and } \mathcal{I}, r', m \models \varphi.$$

It should be pointed out that the semantics of knowledge presented above is only one of several possibilities. It encodes the assumption that agents are *memoryless* in the sense that they do not have "external" memory of past events; the whole memory of an agent is encapsulated in its local state.

The other extreme is to assume that local states represent the agents' observations rather than knowledge, and that agents have *perfect recall* of everything that has been observed. If we additionally assume the existence of a global universally accessible clock, the semantics of knowledge can be defined as follows:

$\langle r, m\rangle \approx_i \langle r', m'\rangle$ iff $m = m'$ and $r(j) \sim_i r'(j)$ for all $j \leq m$,

$\mathcal{I}, r, m \models \mathrm{K}_i\varphi$ iff $\mathcal{I}, r', m' \models \varphi$ for all $\langle r', m'\rangle$ such that $\langle r, m\rangle \approx_i \langle r', m'\rangle$.

This is sometimes called the *synchronous perfect recall semantics* of temporal-epistemic logic. We will use the subscript "R" (perfect **R**ecall) to indicate when the semantics is used, writing e.g. $\mathbf{CTLK}_{\mathrm{R}}$, $\mathbf{LTL}_{\mathrm{R}}$, and so on.

Interpreted systems have been applied to modeling of distributed systems, knowledge bases, message passing systems, etc. Among other things, they have been used to represent phenomena like different types of recall, synchrony and asynchrony, fault tolerance and influence of norms and obligations on the behavior of a multi-agent system.

3.3.3 Model Checking Temporal-Epistemic Properties

Since $\mathbf{CTLK}$ is a straightforward fusion of $\mathbf{CTL}$ and epistemic logic, verification of $\mathbf{CTLK}$ properties can be obtained by the union of the model checking algorithm for epistemic logic (Section 2.2.4) and the fixpoint $\mathbf{CTL}$ model checking algorithm from Section 3.2.1. This gives us the following complexity result.

Theorem 9 *Model checking* $\mathbf{CTLK}$ *in Kripke models is* $\mathbf{P}$*-complete, and can be performed in time* $\mathbf{O}(|M| \cdot |\varphi|)$*, where* $|M| = |St| + |\longrightarrow| + |\sim_1| + \dots + |\sim_k|$ *is the size of the model, and* $|\varphi|$ *is the number of the subformulae in the formula.*

Similarly, combining techniques for $\mathbf{LTL}$ and $\mathbf{CTL}^\star$ with epistemic model checking leads to the following:

Theorem 10 *Model checking* $\mathbf{LTLK}$ *and* $\mathbf{CTLK}^\star$ *in Kripke models is* $\mathbf{PSPACE}$*-complete.*

Assuming perfect recall changes the situation considerably. We define model checking for $\mathbf{CTLK}^\star_{\mathrm{R}}$ and its subsets in the following way. Let $M = \langle St, \longrightarrow, \sim_1, \dots, \sim_k, \mathcal{V}\rangle$ be a generator of interpreted systems, i.e., a finite Kripke model where $St \subseteq St_1 \times \dots \times St_k \times St_e$, and the indistinguishability relations are $\sim_i$ such that $q \sim_i q'$ iff $q[i] = q'[i]$. By $\mathcal{I}(M)$, we denote the interpreted system $\langle \mathcal{R}, \mathcal{V}\rangle$ with $\mathcal{R}$ being the set of infinite paths in M. Moreover, let q be a state in M, and φ a formula of $\mathbf{CTLK}^\star_{\mathrm{R}}$. Checking φ in M, q asks if $\mathcal{I}(M), r, 0 \models \varphi$ for any run r such that $r(0) = q$.

Theorem 11 *Model checking* $\mathbf{CTLK}_{\mathrm{R}}$ *with common knowledge operator in generators of interpreted systems is undecidable. The same applies to model checking* $\mathbf{LTLK}_{\mathrm{R}}$ *with common knowledge and* $\mathbf{CTLK}^\star_{\mathrm{R}}$ *with common knowledge.*

Theorem 12 *Model checking* $\mathbf{CTLK}_{\mathrm{R}}$*,* $\mathbf{LTLK}_{\mathrm{R}}$*, and* $\mathbf{CTLK}^\star_{\mathrm{R}}$ *without common knowledge operators is decidable with nonelementary upper and lower bounds.*

For formulae of epistemic depth[5] *of at most* k*, the model checking problem is in*

[5] The maximal number of nested epistemic operators.

k-EXPTIME.

3.3.4 Back to Motivating Examples

We conclude the chapter by showing how the interplay between dynamics of the systems and knowledge of the agents can be specified for our motivating examples of rescue robots and voting agents.

Example 32 (Rescue Robots: Expressing the properties)

♣ Person j may eventually be safe, and she knows about it:

$$\mathrm{K}_j\mathsf{EFsafe_j}.$$

♣ Person j knows that she may eventually be safe, and that when she is, she will know about it:

$$\mathrm{K}_j\mathsf{EFsafe_j} \wedge \mathrm{K}_j\mathsf{AG}(\mathsf{safe_j} \rightarrow \mathrm{K}_j\mathsf{safe_j}).$$

♣ Each person will eventually be safe, and they know about it:

$$\bigwedge_{j\in People} \mathrm{K}_j\mathsf{AFsafe_j}.$$

Another interpretation: $\mathrm{E}_{People} \bigwedge_{j\in People} \mathsf{AFsafe_j}$.

Another interpretation: $\mathrm{C}_{People} \bigwedge_{j\in People} \mathsf{AFsafe_j}$.

Yet another interpretation: $\mathrm{D}_{People} \bigwedge_{j\in People} \mathsf{AFsafe_j}$.

Other interpretations: $\mathrm{E}_{People}\mathsf{AF} \bigwedge_{j\in People} \mathsf{safe_j}$, $\mathrm{C}_{People}\mathsf{AF} \bigwedge_{j\in People} \mathsf{safe_j}$, $\mathrm{D}_{People}\mathsf{AF} \bigwedge_{j\in People} \mathsf{safe_j}$.

♣ Robot i may eventually know the position of every person:

$$\mathsf{EF}\Big(\bigwedge_{j\in People} \bigvee_{l\in Locations} \mathrm{K}_i\mathsf{at_{j,l}}\Big).$$

Another interpretation: $\bigwedge_{j\in People} \bigvee_{l\in Locations} \mathsf{EF}\mathrm{K}_i\mathsf{at_{j,l}}$.

♣ The robots may obtain distributed knowledge that person j is in location l, but they will never have common knowledge about that:

$$\mathsf{EF}\mathrm{D}_{Robots}\mathsf{at_{j,l}} \wedge \mathsf{AG}\neg\mathrm{C}_{Robots}\mathsf{at_{j,l}}.$$

♣ The robots can rescue a given person, and they know that they can:

Interpretation 1 The robots *may* rescue the person, and they know that they *may*:
$\mathrm{E}_{Robots}\mathsf{EFsafe_i}$ or $\mathrm{C}_{Robots}\mathsf{EFsafe_i}$ or $\mathrm{D}_{Robots}\mathsf{EFsafe_i}$.

Interpretation 2 They *are able* to do it, and they know that they are:

Cannot be expressed in CTLK* (no notion of ability) !

♣ The robots can rescue a given person, and they know how to do it:

Cannot be expressed in CTLK* (no notion of ability) !

Example 33 (Voting: Expressing the properties)

Privacy. The coercer will never know how the voter has voted:

$$\mathsf{AG}\Big(\bigwedge_{c \in Candidates} \neg \mathrm{K}_{coerc} \mathsf{voted}_{\mathsf{v},\mathsf{c}}\Big).$$

Receipt-freeness. The system does not issue pieces of information which can be used to convince the coercer that the voter voted in a certain way:

Cannot be expressed in CTLK* (no notion of knowledge update) !

Coercion-resistance: The voter cannot cooperate with a coercer to prove to him that she voted in a certain way:

Cannot be expressed in CTLK* (no notions of ability & cooperation) !

References and Further Reading. The standard reference on combination of knowledge and time is [63], including different ways of how the temporal and epistemic dimensions can intertwine. The model checking problem for **CTLK** was treated extensively in [112]. For other non-symbolic approaches to verification of temporal-epistemic properties, see e.g. [170] where a translation of a fragment of **LTLK** to **LTL** was proposed, and the SPIN model checker was used for experiments. The results for (un)decidability and complexity of model checking for temporal-epistemic logics with perfect recall can be found in [173, 155, 154].

Chapter 4

Strategic Ability

So far, we have presented logics that allow to specify how things *must* go, or how they *may* evolve. In multi-agent systems, it is often very important to know *who* can make them evolve in a particular way.

In this chapter, we focus on modal logics that can be used to reason about strategies and abilities of agents in game-like scenarios. We begin with a short exposition of the game-theoretic inspiration. However, it should be pointed out that the current modal logic-based approaches to reasoning about strategic play are very weak in game-theoretic sense. They are based on the worst case analysis ("surely winning") and binary winning conditions, and hence roughly correspond to maxmin analysis in two-player zero-sum games with binary payoffs. Some attempts have been made at incorporating more sophisticated solution concepts like Nash equilibrium, dominance, Pareto-optimality, etc. [78, 77, 167, 181, 40] as well as probabilistic features like chance nodes and mixed strategies [51, 91, 37, 152, 85]. Still, none of them matches the elegance and simplicity of the way models and solution concepts are defined in game theory.

Example 34 (Rescue Robots: Properties to express)

- ♣ The robots can rescue all the people in the building;
- ♣ If person i gets outside the building, then she can stay away from trouble forever;
- ♣ Person i may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter;
- ♣ The robots can rescue all the people, and they know that they can;
- ♣ The robots can rescue all the people, and they know how to do it.

Example 35 (Voting: Properties to express)

Privacy: The system cannot reveal how a particular voter voted;

Receipt-freeness: The voter cannot gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way;

Coercion-resistance: The voter cannot cooperate with the coercer to prove to him that she voted in a certain way.

4.1 Models of Strategic Behavior

Logics of strategic reasoning build upon several fundamental concepts from game theory, the most important being that of a *strategy*. We understand strategies as conditional plans that prescribe what action a given agent (or a coalition of agents) should take in every possible situation that may arise in the game. The notion will be made mathematically more precise, and we will use it to provide formal logical semantics of abilities.

We will use the terms "agent" and "player" interchangeably, and consider an arbitrary nonempty finite set of all agents $\mathbb{A}\mathrm{gt}$. We also fix a nonempty set of atomic propositions $\mathcal{PV}$ that encode basic properties of game states.

4.1.1 Games and Strategies

Interactions between autonomous and rational agents acting strategically have been extensively studied in the field of *game theory*. The models used in game theory can be categorised into two types: *non-cooperative* games, in which the possible actions of individual players are taken as primitives, and *coalitional* (or *cooperative*) games which are based on the possible joint actions of groups of players.

A standard model in non-cooperative game theory is that of a *strategic game* (or *normal form game*). In a strategic game, it is assumed that each agent chooses her future actions (her strategy) once and for all at the beginning of the game, and that all agents do this simultaneously. As a consequence, all players perform a vector of actions that determines the outcome of the game. This does not mean that the game must necessarily describe a one-shot interaction, but even if it doesn't, the interaction is represented as *atomic*.

Definition 5 (Strategic game frame, strategic game) *A* strategic game frame *is a tuple* $\Gamma = (\mathbb{A}\mathrm{gt}, \{Act_a \mid a \in \mathbb{A}\mathrm{gt}\}, \Omega, o)$ *that consists of a nonempty finite set of players* $\mathbb{A}\mathrm{gt}$*, a nonempty set of* actions *(also known as* choices*)* Act_a *for each player* $a \in \mathbb{A}\mathrm{gt}$*, a nonempty set of* outcomes Ω*, and an* outcome function $o : \prod_{a\in\mathbb{A}\mathrm{gt}} Act_a \to \Omega$*, that associates an outcome with every* action profile *– that is, a tuple of actions, one per player.*[1]

A strategic game G *is a strategic game frame endowed with* preference orders $\leq_a$ *on the set of outcomes, one for each player. Players' preferences are often expressed by functions* $u_a : \Omega \to \mathbb{R}$*, representing agent* a*'s* utility *or* payoff *in each possible outcome. Then, the preference relations are implicitly defined as follows:* $o \leq_a o'$ *iff* $u_a(o) \leq u_a(o')$*. Thus, strategic games can be represented either as tuples* $(\mathbb{A}\mathrm{gt}, \{Act_a \mid a \in \mathbb{A}\mathrm{gt}\}, \Omega, o, (\leq_a)_{a\in\mathbb{A}\mathrm{gt}})$ *or* $(\mathbb{A}\mathrm{gt}, \{Act_a \mid a \in \mathbb{A}\mathrm{gt}\}, \Omega, o, (u_a)_{a\in\mathbb{A}\mathrm{gt}})$.

Example 36 (Prisoner's Dilemma) *We illustrate the basic concepts by a variant of the well-known Prisoner's Dilemma game, see Figure 4.1. Each of the two players can choose to "cooperate" with the other one (i.e., play action* $coop$*) or to "defect" his comrade (i.e., play action* $defct$*). Formally, the game is defined as* $G =$

[1] We assume a default ordering on $\mathbb{A}\mathrm{gt}$ which is respected in the definitions of action profiles, collective strategies, etc.

$1\backslash 2$	*coop*	*defct*
coop	(3, 3)	(0, 5)
defct	(5, 0)	(1, 1)

Figure 4.1: Prisoner's Dilemma

$(\{1,2\},\{Act_1,Act_2\},\{o_1,o_2,o_3,o_4\},o,(\leq_1,\leq_2))$ with $Act_1 = Act_2 = \{coop, defct\}$, $o(coop, coop) = o_1$, $o(coop, defct) = o_2$, $o(defct, coop) = o_3$, and $o(defct, defct) = o_4$. Moreover, we define $\leq_1$ and $\leq_2$ as the smallest transitive relations with $o_2 \leq_1 o_4 \leq_1 o_1 \leq_1 o_3$ and $o_3 \leq_2 o_4 \leq_2 o_1 \leq_2 o_2$. In the figure we have shown the players' utilities, given by functions u_1 and u_2 defined as follows: $u_1(o_1) = u_2(o_1) = 3$, $u_1(o_4) = u_2(o_4) = 1$, $u_1(o_2) = u_2(o_3) = 0$, and $u_1(o_3) = u_2(o_2) = 5$.

In this book, we abstract from players' preferences, and focus on their powers to enforce particular *properties* of game outcomes. We formalize this approach through game models, defined as follows.

Definition 6 (Strategic game model) *A* strategic game model *is defined as a tuple $M = (\mathbb{A}\mathrm{gt}, St, \{Act_a \mid a \in \mathbb{A}\mathrm{gt}\}, o, \mathcal{V})$ where $\Gamma = (\mathbb{A}\mathrm{gt}, \{Act_a \mid a \in \mathbb{A}\mathrm{gt}\}, St, o)$ is a strategic game frame with outcomes drawn from a set of states St, and $\mathcal{V} : \mathcal{PV} \to 2^{St}$ is a valuation of atomic propositions in states.*

4.1.2 Coalitional Play and Coalition Effectivity Models

It is important to note that in strategic games none of the players knows in advance the actions chosen by the other players, and therefore has relative little control over the outcome of the game. In *cooperative game theory*, agents can join forces and coordinate their actions towards a chosen goal. We assume a very simple model of cooperation in games. A *coalition* is just a group of agents $A \subseteq \mathbb{A}\mathrm{gt}$. Coalitional actions are tuples of individual actions, one per member of the coalition; formally, $Act_A = \prod_{i \in A} Act_i$.

What power does an individual player or a coalition of players have to influence the outcome in such a game? One way to represent strategic power is by *effectivity functions*, introduced in cooperative game theory by Moulin and Peleg and in social choice theory by Abdou and Keiding.

Definition 7 (Effectivity functions) *Given a set of players $\mathbb{A}\mathrm{gt}$ and a set of outcomes Ω, a* coalitional effectivity function *over $\mathbb{A}\mathrm{gt}$ and Ω is a mapping $\mathsf{E} : 2^{\mathbb{A}\mathrm{gt}} \to 2^{2^{\Omega}}$ that associates a family of sets of outcomes with each coalition of players.*

Given a set of outcomes X, we say that a coalition A is *effective* for X if A can cooperate to ensure that the outcome of the game will be in X. Thus, $\mathsf{E}(A)$ consists of goals that coalition A can successfully pursue.

Intuitively, every element of $\mathsf{E}(A)$ is a set of possible outcomes that can result from a joint action of players in A, depending on how the remaining agents decide to play. In other words, for every set X in $\mathsf{E}(A)$ the coalition A has a collective action that is guaranteed to yield an outcome in X, regardless of the actions taken by the players in $\overline{A} = \mathbb{A}\mathrm{gt} \setminus A$. Effectivity functions induced from game frames in this way are called *α-effectivity functions* in social choice theory.

Definition 8 (Effectivity in strategic games) *For a strategic game frame Γ, the α-*effectivity function $\mathsf{E}^{\alpha}_{\Gamma} : 2^{\mathbb{A}\mathrm{gt}} \to 2^{2^{\Omega}}$ *is defined as follows: $X \in \mathsf{E}^{\alpha}_{\Gamma}(A)$ if and only if there exists a joint action σ_A for A such that for every joint action $\sigma_{\overline{A}}$ of $\overline{A}$ we have $o(\sigma_A, \sigma_{\overline{A}}) \in X$.*

Example 37 (Coalitional effectivity for prisoners) *The Prisoner's Dilemma frame of Example 36 can be represented by the following effectivity function over $\mathbb{A}\mathrm{gt} = \{1, 2\}$ and $\Omega = \{o_1, \dots, o_4\}$:*

$$\begin{array}{rcl} \mathsf{E}(\emptyset) & = & \{\Omega\}, \\ \mathsf{E}(\{1\}) & = & \{X \subseteq \Omega \mid \{o_1, o_2\} \subseteq X \textit{ or } \{o_3, o_4\} \subseteq X\}, \\ \mathsf{E}(\{2\}) & = & \{X \subseteq \Omega \mid \{o_1, o_3\} \subseteq X \textit{ or } \{o_2, o_4\} \subseteq X\}, \textit{ and} \\ \mathsf{E}(\{1, 2\}) & = & 2^{\Omega} \setminus \{\emptyset\}. \end{array}$$

Effectivity models define the powers of coalitions in every possible state of the system, and add an interpretation of atomic statements.

Definition 9 (Effectivity models) *A* coalitional effectivity model *is defined as a tuple $\mathcal{E} = (St, \mathsf{E}, \mathcal{V})$ where St is a set of states, $\mathsf{E} : St \to (2^{\mathbb{A}\mathrm{gt}} \to 2^{2^{St}})$ is a* global effectivity function *that maps each state to an effectivity function drawing outcomes from St, and $\mathcal{V} : \mathcal{PV} \to 2^{St}$ is a valuation of atomic propositions.*

Example 38 (Prisoner's Dilemma as effectivity model) *Take $St = \Omega$, and assume $\mathsf{E}(q)$ to be the effectivity function of Example 37 for every $q \in St$. This corresponds to* repeated Prisoner's Dilemma with infinite horizon*, i.e., the game is going to be played in subsequent rounds, infinitely many times.*

Additionally, let us adopt atomic propositions $\{\mathsf{p}^{\mathsf{j}}_{\mathsf{a}} \mid a \in \mathbb{A}\mathrm{gt}, j \in \{0, 1, 3, 5\}\}$ representing the utility values for each agent, and label the outcomes appropriately. Then, for example, we have $\mathcal{V}(o_1) = \{\mathsf{u}^3_1, \mathsf{u}^3_2\}$ and $\mathcal{V}(o_2) = \{\mathsf{u}^0_1, \mathsf{u}^5_2\}$.

Characterization of Effectivity in Strategic Games

Clearly, every strategic game induces an effectivity function, but not every abstract effectivity function must correspond to a strategic game frame. The following notion captures the properties required for such correspondence.

Definition 10 (True playability) *An effectivity function $\mathsf{E} : 2^{\mathbb{A}\mathrm{gt}} \to 2^{2^{\Omega}}$ is* truly playable *iff the following conditions hold:*

Outcome monotonicity: *$X \in \mathsf{E}(A)$ and $X \subseteq Y$ implies $Y \in \mathsf{E}(A)$;*

Safety: *$\mathsf{E}(A) \neq \emptyset$;*

Liveness: *$\emptyset \notin \mathsf{E}(A)$;*

Superadditivity: *If $A_1 \cap A_2 = \emptyset$, $X \in \mathsf{E}(A_1)$ and $Y \in \mathsf{E}(A_2)$, then $X \cap Y \in \mathsf{E}(A_1 \cup A_2)$;*

$\mathbb{A}\mathrm{gt}$***-maximality:*** *$\overline{X} \notin \mathsf{E}(\emptyset)$ implies $X \in \mathsf{E}(\mathbb{A}\mathrm{gt})$;*

Determinacy: *If $X \in \mathsf{E}(\mathbb{A}\mathrm{gt})$, then $\{x\} \in \mathsf{E}(\mathbb{A}\mathrm{gt})$ for some $x \in X$.*

It is easy to see that every α-effectivity function of a strategic game is truly playable. The converse holds, too.

Theorem 13 (Representation Theorem for effectivity in strategic games) *An effectivity function* E *for* $(\mathbb{A}\mathrm{gt}, \Omega)$ *is truly playable if and only if there exists a strategic game frame* $\Gamma = (\mathbb{A}\mathrm{gt}, \{Act_i \mid i \in \mathbb{A}\mathrm{gt}\}, \Omega, o)$ *such that* $\mathsf{E}^{\alpha}_{\Gamma} = \mathsf{E}$ *.*

References and Further Reading. [125] is a standard textbook in game theory; for alternative expositions see [80, 109]. In [156], game-theoretical concepts are approached from the multi-agent systems perspective. Effectivity functions were introduced in [124, 1], and studied e.g. in [129, 24, 70], including correspondence results between concrete and abstract models of strategic power in games.

4.2 Models of Multi-Step Interactions

Typical interaction between agents does not correspond to a simultaneous atomic activity. Thus, strategic games are often considered in a repeated setting: game G is played a number of times, and the payoffs from all rounds are aggregated. This kind of models is especially popular in evolutionary game theory. Another class of models for multi-step scenarios are *games in extensive form* that model the interaction with a finite tree whose nodes stand for game states and edges represent moves available to players in the game. Extensive form games are turn-based, i.e., every non-terminal node is controlled by exactly one player. The payoffs are distributed at the end of the game, that is, in the leaves of the tree.

Concurrent game structures, also known as multi-player game frames, generalize the setting of repeated games by allowing *different* strategic games to be played at different stages. This way we can model multi-step interactions defined on some state space in which every state is connected to a strategic game with outcomes being states again. The resulting multi-step game consists of playing one-shot strategic games in successive rounds. The outcome of every round determines the successor state, and therefore the strategic game to be played in the next round. Alternatively, one can see concurrent game structures as a generalization of extensive form games where simultaneous moves of different players are allowed, as well as loops to previously visited states.

4.2.1 Concurrent Game Models

Models of concurrent multi-step interaction can be defined as follows.

Definition 11 (Concurrent game structures and models) *A* concurrent game structure *(CGS) is a tuple* $S = (\mathbb{A}\mathrm{gt}, St, Act, d, o)$ *which consists of a non-empty finite set of players* $\mathbb{A}\mathrm{gt} = \{1, \dots, \mathsf{k}\}$*, a non-empty*[2] *set of states* St*, a non-empty set of atomic actions* Act*, a function* $d : \mathbb{A}\mathrm{gt} \times St \to 2^{Act} \setminus \{\emptyset\}$ *that defines the set of actions available to each player at each state, and a (deterministic) transition function* o *that assigns a unique* successor state $o(q, \alpha_1, \dots, \alpha_{\mathsf{k}})$ *to each state* q *and each tuple of actions* $\langle \alpha_1, \dots, \alpha_{\mathsf{k}} \rangle$ *that can be executed in* q*, i.e., such that* $\alpha_a \in d(a, q)$ *for every* $a \in \mathbb{A}\mathrm{gt}$*. We will sometimes write* $d_a(q)$ *instead of* $d(a, q)$*, and use* $d_A(q) = \prod_{a \in A} d_a(q)$ *to denote the set of joint actions of coalition* A *in state* q*.*

A concurrent game model *(CGM)* $M = (\mathbb{A}\mathrm{gt}, St, Act, d, o, \mathcal{V})$ *over a set of atomic propositions* $\mathcal{PV}$ *is a concurrent game structure extended with a valuation of atomic propositions* $\mathcal{V} : \mathcal{PV} \to 2^{St}$*.*

[2] The set of states is sometimes assumed finite but that restriction is not necessary for our purposes.

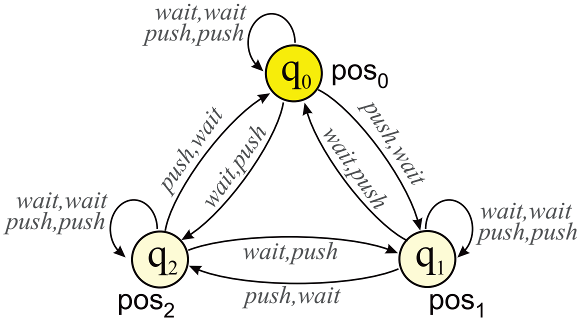


Figure 4.2: The robots and the carriage: concurrent game model M_{carr4}

Thus, all players in a CGS execute their actions synchronously, and the combination of the actions together with the current state determines the transition to a successor state in the CGS.

Example 39 (Robots and Carriage: Modeling the interaction) *Models and languages in Chapter 3 enabled studying evolution of the robots and carriage as a whole. However, they did not allow for representing* who *can achieve* what, *and how possible actions of the agents interfere. Concurrent game model M_{carr4}, presented in Figure 4.2, fills the gap. We assume that each robot can either push (action $push$) or refrain from pushing (action $wait$). Moreover, both robots use the same force when pushing. Thus, if they push simultaneously or wait simultaneously, the carriage does not move. When only one of the robots is pushing, the carriage moves accordingly: clockwise if pushed by robot 1, and anticlockwise when robot 2 is pushing.*

Atomic propositions $\mathsf{pos}_0, \mathsf{pos}_1, \mathsf{pos}_2$ *will later enable us to refer to various positions of the carriage in logical formulae.*

4.2.2 Strategies in Concurrent Game Models

Similarly to temporal models, we define a *path* in a CGM to be an infinite sequence of states $\lambda \in St^\omega$ that can result from subsequent transitions. Moreover, we define a *history* $h \in St^+$ as a finite sequence of states that can occur in the system. A *strategy* of a player a in a CGM M is a conditional plan that specifies what a should do in each possible situation. Depending on the type of memory that we assume for the players, a strategy can be *memoryless* (alias *positional*), formally represented by a function $s_a : St \to Act$ such that $s_a(q) \in d_a(q)$, or *memory-based* (alias *perfect recall*), represented by a function $s_a : St^+ \to Act$ such that $s_a(\langle \dots, q \rangle) \in d_a(q)$. The latter corresponds to players with perfect memory of the past states; the former corresponds to players whose memory, if any, is entirely encoded in the current state of the system. Intermediate types of strategies for agents with finite memory have been studied by Ågothes and Walther [6] as well as Vester [179].

Remark 1 *Note that concurrent game structures include no semantic means for representing agents' uncertainty. In particular, strategies can freely assign arbitrary choices in states (resp. histories). In consequence, CGM's can be only used to model agents that always have* perfect information *about the current global state of the system.*

A *collective strategy* of coalition $A = \{a_1, ..., a_r\}$ is simply a tuple of strategies $s_A = \langle s_{a_1}, ..., s_{a_r}\rangle$, one per player from A. We denote agent a's component of the collective strategy s_A by $s_A[a]$. Then, in the case of a positional collective strategy s_A, the action that $s_A[a]$ prescribes to player a at state q is $s_A[a](q)$. Analogously, $s_A[a](h)$ is the action that a memory-based collective strategy s_A prescribes to a after history h. We will denote the set of memoryless strategies of team A by Σ_A^{Ir} (Ir standing for *perfect **I**nformation* and *imperfect **r**ecall*). Similarly, the set of A's memory-based strategies A will be denoted by Σ_A^{IR} (*perfect **I**nformation* and *perfect **R**ecall*).

Example 40 (Robots and Carriage: Strategies) *An example memoryless strategy for robot 1 is $s_1(q_0) = s_1(q_1) = s_1(q_2) = wait$, i.e., robot 1 will execute $wait$ no matter what happens. A slightly more sophisticated strategy is $s'_1(q_0) = s'_1(q_1) = wait, s'_1(q_2) = push$ (wait unless the carriage is in the position 2; in that case, push).*

An example memory-based strategy is "$s''_1(h) = push$ if h is of even length and ends with q_0, otherwise $wait$." Obviously, strategies s_1 and s'_1 also belong to perfect recall strategies of robot 1.

Function $out(q, s_A)$ returns the set of all paths $\lambda \in St^\omega$ that may occur when agents A execute strategy s_A from state q onward. For a memoryless strategy the set is given as follows:

$out(q, s_A) = \{\lambda = q_0, q_1, q_2 \ldots \mid q_0 = q$ and for each $i = 0, 1, \ldots$ there exists $\langle \alpha^i_{a_1}, \ldots, \alpha^i_{a_k}\rangle$ such that $\alpha^i_a \in d_a(q_i)$ for every $a \in \mathbb{A}\mathrm{gt}$, and $\alpha^i_a = s_A[a](q_i)$ for every $a \in A$, and $q_{i+1} = o(q_i, \alpha^i_{a_1}, \ldots, \alpha^i_{a_k})\}$.

For a memory-based strategy s_A, the outcome set is defined analogously:

$out(q, s_A) = \{\lambda = q_0, q_1, q_2 \ldots \mid q_0 = q$ and for each $i = 0, 1, \ldots$ there exists $\langle \alpha^i_{a_1}, \ldots, \alpha^i_{a_k}\rangle$ st. $\alpha^i_a \in d_a(q_i)$ for every $a \in \mathbb{A}\mathrm{gt}$, and $\alpha^i_a = s_A[a](\langle q_0 \ldots, q_i\rangle)$ for every $a \in A$, and $q_{i+1} = o(q_i, \alpha^i_{a_1}, \ldots, \alpha^i_{a_k})\}$.

Example 41 (Robots and Carriage: Outcomes of strategies) *Consider the strategies presented in Example 40. The outcome set of the memoryless strategy s_1 from state q_0 is as follows:*

$$\begin{aligned} out(s_1, q_0) \quad = \quad & \{\lambda \in \{q_0, q_1, q_2\}^\omega \mid \lambda[0] = q_0\ \& \\ & \qquad \forall i\ .\ (\lambda[i] = q_j) \Rightarrow (\lambda[i+1] = q_j\ \mathit{or}\ \lambda[i+1] = q_{(j-1) mod\ 3})\}. \end{aligned}$$

Similarly, the outcome set of the perfect recall strategy s''_1 from state q_0 is:

$$\begin{aligned} out(s''_1, q_0) \quad = \quad & \{\lambda \in \{q_0, q_1, q_2\}^\omega \mid \lambda[0] = q_0\ \& \\ & \mathit{if}\ (|\lambda[0..i]| \in 2N\ \&\ \lambda[i] = q_0) \\ & \mathit{then}\ (\lambda[i+1] = q_0\ \mathit{or}\ \lambda[i+1] = q_1) \\ & \mathit{else}\ \forall j\ .\ (\lambda[i] = q_j) \Rightarrow (\lambda[i+1] = q_j\ \mathit{or}\ \lambda[i+1] = q_{(j-1) mod\ 3}))\}. \end{aligned}$$

Given a concurrent game model, an interesting question is: *what can a particular player or a coalition achieve in the game*? So far, the objectives of players and coalitions are not formally specified, but a typical objective would be to reach a state satisfying a given property, e.g. a winning state. Generally, an objective is a property of paths resulting from playing the strategy. For instance, one can mark some paths as winning, and other ones as losing for the given coalition. More precisely, we say that

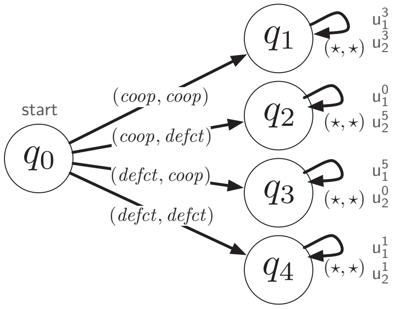

start
q_0
q_1
q_2
q_3
q_4
$(coop, coop)$
$(coop, defct)$
$(defct, coop)$
$(defct, defct)$
$(\star,\star)$
u_1^3 u_2^3
u_1^0 u_2^5
u_1^5 u_2^0
u_1^1 u_2^1

Figure 4.3: Prisoner's Dilemma modelled as CGM M_{pris}

coalition A can enforce objective γ from state q if there is a collective strategy s_A for A such that every play in $out(q, s_A)$ satisfies γ. The central issue that we will discuss in the rest of this chapter is how to use modal logic to formally specify strategic objectives of players and coalitions, and how to formally determine their abilities with respect to such objectives.

4.2.3 Representing Games as Concurrent Games Models

Concurrent game models have a close relationship to strategic, repeated, and extensive form games with perfect information. The major difference is that CGM's lack the notion of payoffs (or, more generally, players' preferences over possible game outcomes). However, those can be embedded in CGM's in a natural way, for example by adding auxiliary propositions in the leaf nodes of the CGM. Formally, consider any strategic or extensive game G, and let U be the set of all possible payoff/utility values in it. For each value $v \in U$ and agent $a \in \mathbb{A}\mathrm{gt}$, we introduce a proposition $\mathsf{u}_\mathsf{a}^\mathsf{v}$ and fix $\mathsf{u}_\mathsf{a}^\mathsf{v} \in \mathcal{V}(q)$ iff a gets a payoff v in state q.

We illustrate the correspondence with two examples of how strategic and extensive games can be represented as concurrent game models.

Example 42 (Prisoner's Dilemma as CGM) *The Prisoner's Dilemma game of Example 36 can be represented by the following concurrent game model:*

$$M_{pris} = (\{1,2\}, \{q_0, \dots, q_4\}, \{defct, coop\}, d, o, \mathcal{V})$$

with $d_a(q) = \{defct, coop\}$ for all players a and states q, $o(q_0, coop, coop) = q_1$, $o(q_0, coop, defct) = q_2$, $o(q_0, defct, coop) = q_3$, $o(q_0, defct, defct) = q_4$, and $o(q_i, a_1, a_2) = q_i$ for $i = 1, \dots, 4$ and $a_1, a_2 \in \{defct, coop\}$. The CGM is shown in Figure 4.3. The labeling function $\mathcal{V}$ is defined over atomic propositions $\mathcal{PV} = \{\mathsf{start}\} \cup \{\mathsf{u}_\mathsf{a}^\mathsf{v} \mid a \in \mathbb{A}\mathrm{gt}, v \in \{0,1,3,5\}\}$. As shown in the figure, we have e.g. $\mathcal{V}(q_2) = \{\mathsf{u}_1^0, \mathsf{u}_2^5\}$ representing that players 1 and 2 receive utility of 0 and 5, respectively, if strategy profile $(coop, defct)$ is played.

For embeddings of extensive form games in CGM's, we additionally assume that the resulting concurrent game structure is tree-like. Moreover, each non-leaf state provides an appropriate set of available actions to the player whose turn it is to move from the state. All the other players are only allowed a single action "pass." In leaf states,

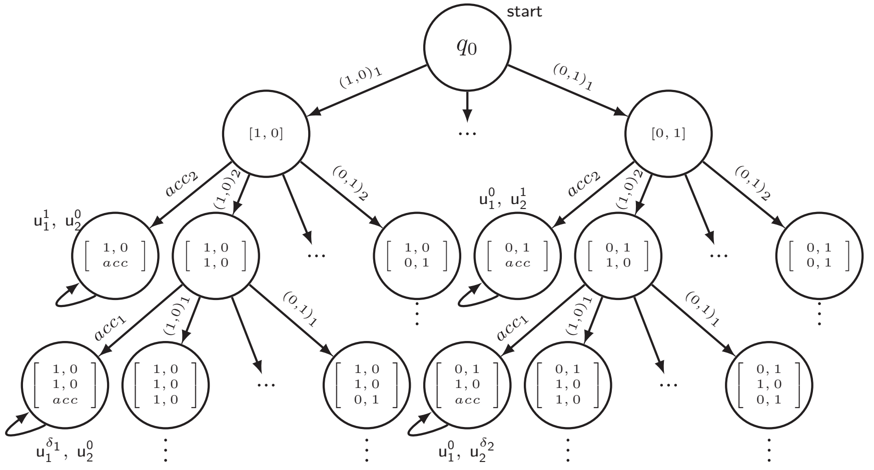


Figure 4.4: CGM M_{barg} modeling the bargaining game

all the players are only allowed to "pass," and the result of such collective "do nothing" action is a transition to the same state, thus looping there forever.

Example 43 (Bargaining) *The following example shows that CGM's are rich enough to model (possibly infinite) extensive form games. Consider bargaining with time discount. Two players,* 1 *and* 2*, bargain over how to split goods worth initially* $w_0 = 1k$ *euro. After each round without agreement, the subjective worth of the goods reduces by* discount rates δ_1 *(for player* 1*) and* δ_2 *(for player* 2*). So, after* t *rounds the goods are worth* $\langle \delta_1^t, \delta_2^t \rangle$*, respectively. Subsequently,* 1 *(if* t *is even) or* 2 *(if* t *is odd) makes an offer to split the goods in proportions* $\langle x, 1-x \rangle$*, and the other player accepts or rejects it. If the offer is accepted, then* 1 *takes* $x\delta_1^t$*, and* 2 *gets* $(1-x)\delta_2^t$*; otherwise the game continues.*

The CGM corresponding to this extensive form game is shown in Figure 4.4. Note that the model has a tree-like structure with infinite depth and an infinite branching factor. Nodes represent various states of the negotiation process, and arcs show how agents' moves change the state of the game. A node label refers to the history of the game for better readability. For instance, $\left[\begin{smallmatrix} 0,1 \\ 1,0 \\ acc \end{smallmatrix}\right]$ *has the meaning that in the first round player* 1 *offered to split the goods* $\langle 0, 1 \rangle$ *which was rejected by agent* 2*. In the next round,* 2*'s offer* $\langle 1, 0 \rangle$ *was accepted by* 1*, and the game has ended.*

References and Further Reading. Concurrent game models a.k.a. multi-player game models were introduced and studied independently in [129, 12]. Effectivity functions for multi-step multi-player games were investigated in [68]. Other interesting models of interaction include alternating transition systems [10, 11], together with appropriate correspondence results [66, 67]. Various embeddings of players' preferences in concurrent game models were considered in [79, 17, 165, 177, 100].

4.3 Logics for Strategic Ability

Logics for strategic ability are an important field of study in multi-agent systems. They build on game models and the concept of strategic play, but the main inspiration came from artificial intelligence rather than game theory. AI attempts to develop a logical formalization of ability date back at least to the 1969 paper *Some Philosophical Problems from the Standpoint of Artificial Intelligence* by McCarthy and Hayes, which offers the following observation:

> We want a computer program that decides what to do by inferring in a formal language [i.e., a logic] that a certain strategy will achieve a certain goal. This requires formalizing concepts of causality, ability, and knowledge.[3]

From the late 1980s on, philosophy and artificial intelligence have witnessed the emergence of several influential logical treatments of strategic ability. In philosophical logic, this goes back to the work of Brown [30] and especially Belnap and Perloff [20] who proposed their logic of "seeing to it that" ($\mathtt{stit}$). More computation-friendly approaches began with Parikh [127], and culminated in the work of Alur, Henzinger and Kupferman on alternating-time temporal logic [10, 12]. Independently, Pauly proposed his coalition logic in [128, 129]. In this book, we focus on the latter kind of approaches.

4.3.1 Coalition Logic

Modal logics of strategic ability form one of the fields where logic and game theory can successfully meet. *Coalition logic* ($\mathbf{CL}$) formalises reasoning about the abilities of coalitions in one-shot games. The language of $\mathbf{CL}$ extends propositional logic with modalities $[A]$ for each coalition A. The intended meaning of $[A]\varphi$ is that A can make the outcome of the game satisfy φ.

The semantics of $\mathbf{CL}$, interpreted over concurrent game models, is given by the following clauses:

$M, q \models p$ iff $q \in \mathcal{V}(p)$;

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;

$M, q \models \varphi_1 \wedge \varphi_2$ iff $M, q \models \varphi_1$ and $M, q \models \varphi_2$;

$M, q \models [A]\varphi$ iff there is a collective action $\alpha_A \in d_A(q)$ such that, for every response $\alpha_{\mathbb{A}\mathrm{gt}\setminus A} \in d_{\mathbb{A}\mathrm{gt}\setminus A}(q)$, we have that $M, o(q, \alpha_A, \alpha_{\mathbb{A}\mathrm{gt}\setminus A}) \models \varphi$.

Example 44 (Robots and Carriage: One-step abilities) *The following* $\mathbf{CL}$ *formula holds in the Robots and Carriage scenario:* $M_{carr4}, q_0 \models \neg[1]\mathsf{pos}_0 \wedge \neg[2]\mathsf{pos}_0 \wedge [1,2]\mathsf{pos}_0$, *expressing that neither robot can keep the carriage in its current position singlehandedly, but they can enforce that if they cooperate.*

Alternatively, $\mathbf{CL}$ can be interpreted over coalition effectivity models by means of the following, extremely simple semantic clause:

$$\mathcal{E}, q \models [A]\varphi \quad \text{iff} \quad \varphi^{\mathcal{E}} \in \mathsf{E}(q)(A)$$

[3] Quoted after [115].

where E is the global effectivity function in model $\mathcal{E}$, and $\varphi^{\mathcal{E}} = \{q \in St : \mathcal{E}, q \models \varphi\}$. We recall the intuition that coalition effectivity models are usually based on α-effectivity functions of some strategic game model. Indeed, the two semantics of coalition logic are equivalent in the following sense:

Theorem 14 *Given a concurrent game model M and a state q in M, we have that $M, q \models \varphi$ iff $\mathcal{E}^{\alpha}_{M}, q \models \varphi$.*

Axiomatization of CL

Pauly has shown that the conditions of liveness, safety, superadditivity, and $\mathbb{A}\mathrm{gt}$-maximality in Definition 10 can be captured by a few simple axiom schemes presented below, where $A, A_1, A_2 \subseteq \mathbb{A}\mathrm{gt}$ are arbitrary coalitions of players:

1. Complete set of axioms for classical propositional logic,
2. $[\mathbb{A}\mathrm{gt}]\top$,
3. $\neg[A]\bot$,
4. $\neg[\emptyset]\varphi \rightarrow [\mathbb{A}\mathrm{gt}]\neg\varphi$, and
5. $[A_1]\varphi \wedge [A_2]\psi \rightarrow [A_1 \cup A_2](\varphi \wedge \psi)$ for any disjoint $A_1, A_2 \subseteq \mathbb{A}\mathrm{gt}$.

These, together with the standard inference rule of *Modus Ponens* and a new rule of *Monotonicity*:

$$\frac{\varphi \rightarrow \psi}{[A]\varphi \rightarrow [A]\psi}$$

provide a sound and complete axiomatization of the valid formulae of $\mathbf{CL}$.

4.3.2 Alternating-Time Temporal Logic

Coalition logic allows to reason about agents' outcomes in strategic games. What if the interaction consists of multiple steps, like in the case of extensive form games? In this section, we focus on *alternating-time temporal logic (*$\mathbf{ATL}^*$*)*, one of the most important logics of time and strategies that have emerged in recent years. $\mathbf{ATL}^*$ can be seen as a generalization of the branching time temporal logic $\mathbf{CTL}^\star$, in which path quantifiers (E: "there is a path", A: "for every path") are replaced with *strategic quantifiers* $\langle\!\langle A \rangle\!\rangle$. The formula $\langle\!\langle A \rangle\!\rangle\gamma$ expresses that coalition A has a collective strategy to enforce the temporal property γ throughout the interaction. $\mathbf{ATL}^*$ formulae include the usual temporal operators: X ("in the next state"), G ("always from now on"), F ("now or sometime in the future"), and U ("until").

Syntax and Semantics

Formally, the language of the full variant $\mathbf{ATL}^*$ is defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A \rangle\!\rangle\gamma,$$
$$\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \mathrm{X}\gamma \mid \gamma\,\mathrm{U}\,\gamma.$$

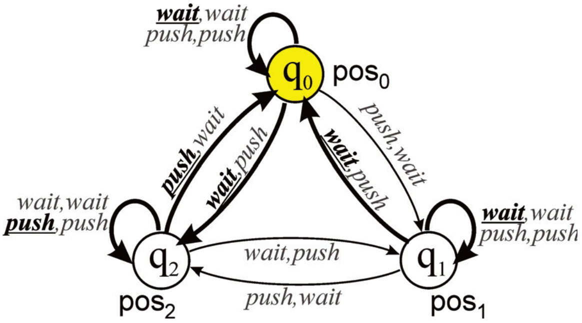


Figure 4.5: The robots and the carriage: playing strategically

where $A \subseteq \mathbb{A}\mathrm{gt}$ and p $\in \mathcal{PV}$. The derived temporal operators can be defined as usual, i.e., $\mathrm{F}\gamma \equiv \top \,\mathrm{U}\, \gamma$ and $\mathrm{G}\gamma \equiv \neg\mathrm{F}\neg\gamma$.

Several semantics have been introduced for **ATL***, of which the one based on concurrent game models is the most popular, given by the following semantic clauses:

$M, q \models p$ iff $q \in \mathcal{V}(p)$;

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;

$M, q \models \varphi_1 \wedge \varphi_2$ iff $M, q \models \varphi_1$ and $M, q \models \varphi_2$;

$M, q \models \langle\langle A \rangle\rangle\gamma$ iff there is a collective strategy $s_A \in \Sigma_A^{\mathrm{IR}}$, such that $M, \lambda \models \gamma$ for every $\lambda \in out(q, s_A)$.

Notably, the clause for $\langle\langle A \rangle\rangle$ asks about existence of a suitable *perfect recall* strategy. We will later see that, for a relevant part of the language, the semantics can be equivalently defined in terms of memoryless plans.

Again, the interpretation of path formulae essentially copies the semantics of **LTL**:

$M, \lambda \models \varphi$ iff $M, \lambda[0] \models \varphi$ (i.e., φ holds in state $\lambda[0]$ of model M);

$M, \lambda \models \neg\gamma$ iff $M, \lambda \not\models \gamma$;

$M, \lambda \models \gamma_1 \wedge \gamma_2$ iff $M, \lambda \models \gamma_1$ and $M, \lambda \models \gamma_2$;

$M, \lambda \models \mathrm{X}\gamma$ iff $M, \lambda[1..\infty] \models \gamma$;

$M, \lambda \models \gamma_1 \,\mathrm{U}\, \gamma_2$ iff $M, \lambda[i..\infty] \models \gamma_2$ for some $i \geq 0$, and $M, \lambda[j..\infty] \models \gamma_1$ for all $0 \leq j < i$.

As we have already observed in Section 4.2.2, concurrent game structures do not allow for representation of agents' uncertainty. In consequence, ATL can be seen as a logic for reasoning about agents with perfect information. That is, it is implicitly assumed that each agent always precisely knows the current state of the game. The notions of perfect vs. imperfect information will be address more formally in Chapter 6.

Example 45 (Robots and Carriage: Long-term abilities) *Consider again the concurrent game model M_{carr4} in Figure 4.2. The immediate outcome of each robot's action depends on the concurrent action of the other robot, and in consequence no agent* *can make sure that the carriage moves to any particular position. So, we have for example that* $M_{carr4}, q_0 \models \neg\langle\!\langle 1\rangle\!\rangle \mathrm{X}\mathsf{pos}_0 \wedge \neg\langle\!\langle 2\rangle\!\rangle \mathrm{X}\mathsf{pos}_0$*, and similarly for propositions* $\mathsf{pos}_1, \mathsf{pos}_2$*. The same applies to long-term abilities of the robots, for example* $M_{carr4}, q_0 \models \neg\langle\!\langle 1\rangle\!\rangle \mathrm{F}\mathsf{pos}_0$ *as well as* $M_{carr4}, q_0 \models \neg\langle\!\langle 1\rangle\!\rangle \mathrm{G}\mathsf{pos}_0$.

On the other hand, robot 1 can at least make sure that the carriage will avoid *particular positions. For instance, it holds that* $M_{carr4}, q_0 \models \langle\!\langle 1\rangle\!\rangle \mathrm{G}\neg\mathsf{pos}_1$*, the right strategy being* $s_1(\dots q_0) = wait$ *and* $s_1(\dots q_2) = push$*. The actions to be executed after histories ending with* q_1 *are irrelevant, so we can choose arbitrarily, e.g.,* $s_1(\dots q_1) = wait$*. Note also that the same behavior of the system can be obtained by the following memoryless strategy of robot 1:* $s'_1(q_0) = wait, s'_1(q_2) = push, s'_1(q_1) = wait$*. The transitions enabled by both strategies are depicted in Figure 4.5; it is easy to see that the system will never reach* q_1*.*

Moreover, the robots together can move the carriage to any position they like: $M_{carr4}, q_i \models \langle\!\langle 1,2\rangle\!\rangle \mathrm{F}\mathsf{pos}_0 \wedge \langle\!\langle 1,2\rangle\!\rangle \mathrm{F}\mathsf{pos}_1 \wedge \langle\!\langle 1,2\rangle\!\rangle \mathrm{F}\mathsf{pos}_2$ *for every* $i = 0, 1, 2$*. They can even achieve this by a single strategy:* $M_{carr4}, q_i \models \langle\!\langle 1,2\rangle\!\rangle (\mathrm{F}\mathsf{pos}_0 \wedge \mathrm{F}\mathsf{pos}_1 \wedge \mathrm{F}\mathsf{pos}_2)$*, an example strategy being* $push$ *for robot 1 and* $wait$ *for robot 2 in any situation that can arise.*

"Vanilla" ATL

The above clauses define the semantics of the full version of alternating-time logic, called $\mathbf{ATL}^*$ for historical reasons. It must be noted, however, that the typical variant of alternating-time logic is restricted to formulae in which strategic and temporal operators are coupled into compound modalities:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A\rangle\!\rangle \mathrm{X}\varphi \mid \langle\!\langle A\rangle\!\rangle \mathrm{G}\varphi \mid \langle\!\langle A\rangle\!\rangle \varphi \,\mathrm{U}\, \varphi.$$

We will refer to the restricted variant as "vanilla" $\mathbf{ATL}$, or simply $\mathbf{ATL}$. $\mathbf{ATL}$ allows to interpret formulae entirely in states of the system, without explicit reference to paths in the sematic relation:

$M, q \models p$ iff $q \in \mathcal{V}(p)$;

$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;

$M, q \models \varphi_1 \wedge \varphi_2$ iff $M, q \models \varphi_1$ and $M, q \models \varphi_2$;

$M, q \models \langle\!\langle A\rangle\!\rangle \mathrm{X}\varphi$ iff there is a collective strategy $s_A \in \Sigma_A^{\mathrm{IR}}$ such that, for every path $\lambda \in out(q, s_A)$, we have that $M, \lambda[1] \models \varphi$;

$M, q \models \langle\!\langle A\rangle\!\rangle \mathrm{G}\varphi$ iff there is a collective strategy $s_A \in \Sigma_A^{\mathrm{IR}}$ such that, for every path $\lambda \in out(q, s_A)$, we have that $\lambda[i] \models \varphi$ for all $i \geq 0$;

$M, q \models \langle\!\langle A\rangle\!\rangle \varphi_1 \,\mathrm{U}\, \varphi_2$ iff there is a collective strategy $s_A \in \Sigma_A^{\mathrm{IR}}$ such that, for every path $\lambda \in out(q, s_A)$, we have that $M, \lambda[i] \models \varphi_2$ for some $i \geq 0$ and $M, \lambda[j] \models \varphi_1$ for all $0 \leq j < i$.

Example 46 (Robots and Carriage: State-based specifications) *All but the last formula of Example 45 are in fact formulae of* $\mathbf{ATL}$*. The last formula,* $\langle\!\langle 1,2\rangle\!\rangle (\mathrm{F}\mathsf{pos}_0 \wedge$

$\mathrm{F}\mathsf{pos}_1 \wedge \mathrm{F}\mathsf{pos}_2$)*, is a formula of* $\mathbf{ATL}^*$ *but not "vanilla"* $\mathbf{ATL}$. *Still, it can be translated to an equivalent* $\mathbf{ATL}$ *formula:*

$$\begin{aligned}\langle\!\langle 1,2\rangle\!\rangle\mathrm{F}\ \big(\ &\mathsf{pos}_0 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}(\mathsf{pos}_1 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}\mathsf{pos}_2 \vee \mathsf{pos}_2 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}\mathsf{pos}_1) \vee \\ &\mathsf{pos}_1 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}(\mathsf{pos}_0 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}\mathsf{pos}_2 \vee \mathsf{pos}_2 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}\mathsf{pos}_0) \vee \\ &\mathsf{pos}_2 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}(\mathsf{pos}_0 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}\mathsf{pos}_1 \vee \mathsf{pos}_1 \wedge \langle\!\langle 1,2\rangle\!\rangle\mathrm{F}\mathsf{pos}_0)\ \big).\end{aligned}$$

As an aside, we notice two things. First, the translation exponentially increases the number of the subformulae in the formula. Secondly, it only works in the perfect recall semantics of $\mathbf{ATL}^*$. *With memoryless strategies, the two properties are not equivalent anymore.*

Embedding Simpler Logics in $\mathbf{ATL}^*$

We observe that $\mathbf{ATL}^*$ syntactically embeds two important logics. First, the branching time logic $\mathbf{CTL}^\star$ can be seen as a subset of $\mathbf{ATL}^*$ with very limited strategic quantification. This is because the path quantifiers of $\mathbf{CTL}^\star$ can be expressed in the standard semantics of $\mathbf{ATL}^*$ as follows: $\mathsf{A}\gamma \equiv \langle\!\langle\emptyset\rangle\!\rangle\gamma$ and $\mathsf{E}\gamma \equiv \langle\!\langle\mathbb{A}\mathrm{gt}\rangle\!\rangle\gamma$. We point out that the above translation of E does *not* work for several extensions of $\mathbf{ATL}^*$, e.g., with imperfect information, nondeterministic strategies, and irrevocable strategies. On the other hand, the translation of A into $\langle\!\langle\emptyset\rangle\!\rangle$ does work for all the semantic variants of $\mathbf{ATL}^*$ considered in this book. Thanks to that, we can define a translation $atl(\varphi)$ from $\mathbf{CTL}^\star$ to $\mathbf{ATL}^*$ as follows. First, we convert φ so that it only includes universal path quantifiers, and then replace every occurrence of A with $\langle\!\langle\emptyset\rangle\!\rangle$. For example, $atl(\mathsf{EG}(\mathsf{p}_1 \wedge \mathsf{AF}\mathsf{p}_2)) = \neg\langle\!\langle\emptyset\rangle\!\rangle\mathrm{F}(\neg\mathsf{p}_1 \vee \neg\langle\!\langle\emptyset\rangle\!\rangle\mathrm{F}\mathsf{p}_2)$.

Secondly, coalition logic can be seen as the fragment of $\mathbf{ATL}$ that uses only the "nexttime" temporal operators X, with the following embedding of the central modality of $\mathbf{CL}$: $[A]\varphi \equiv \langle\!\langle A\rangle\!\rangle\mathrm{X}\varphi$.

We also point out that, semantically, $\mathbf{CTL}^\star$ can be seen as the single-agent fragment of $\mathbf{ATL}^*$ where formulae are interpreted over structures that include only one player ("the system").

4.3.3 Properties of $\mathbf{ATL}^*$ and $\mathbf{ATL}$

Fixpoint Characterization of $\mathbf{ATL}$ Operators

The following formulae are valid in $\mathbf{ATL}$, i.e., true in every state of every model:

$$\begin{aligned}\langle\!\langle A\rangle\!\rangle\mathrm{F}\varphi &\quad\leftrightarrow\quad \varphi \vee \langle\!\langle A\rangle\!\rangle\mathrm{X}\,\langle\!\langle A\rangle\!\rangle\mathrm{F}\varphi,\\ \langle\!\langle A\rangle\!\rangle\mathrm{G}\varphi &\quad\leftrightarrow\quad \varphi \wedge \langle\!\langle A\rangle\!\rangle\mathrm{X}\,\langle\!\langle A\rangle\!\rangle\mathrm{G}\varphi,\\ \langle\!\langle A\rangle\!\rangle\varphi_1\,\mathrm{U}\,\varphi_2 &\quad\leftrightarrow\quad \varphi_2 \vee (\varphi_1 \wedge \langle\!\langle A\rangle\!\rangle\mathrm{X}\,\langle\!\langle A\rangle\!\rangle\varphi_1\,\mathrm{U}\,\varphi_2).\end{aligned}$$

Note that the validities are straightforward adaptations of the $\mathbf{CTL}$ fixpoint equivalences from Section 3.1.4.

Alternating μ-Calculus

Similarly to temporal logics, one can define a version of modal μ-calculus for expressing strategic properties in game-like scenarios. To do this, we observe that the language of μ-calculus consists of generic constructs – namely boolean connectives and fixpoint operators – plus one or more *basic modality* which is specific to the phenomena we want to reason about. For example, the classical version of modal μ-calculus is built upon $\mathbf{PDL}$-style action operators $\langle\alpha\rangle$. In case of branching-time logic, the basic modality is EX allowing for statements about immediate successors of the current state. In general, μ-calculus can be seen as a generic template, parameterized by a set of basic modal constructs.

Alternating μ-calculus ($\mathbf{AMC}$) is the variant of μ-calculus with strategic-nexttime operators $\langle\!\langle A\rangle\!\rangle\mathrm{X}$ given as basic modalities. Equivalently, one can see $\mathbf{AMC}$ as the fixpoint extension of coalition logic. The denotational semantics of $\mathbf{AMC}$ is obtained by replacing the clause for EX from Section 3.1.4 with the following clause:

$$[\![\langle\!\langle A\rangle\!\rangle\mathrm{X}\varphi]\!]_{M,\mathcal{E}} \quad = \quad \{q \mid \text{there is } \alpha_A \in d_A(q) \text{ such that for all } \alpha_{\mathbb{A}\mathrm{gt}\setminus A} \in d_{\mathbb{A}\mathrm{gt}\setminus A}(q) \text{ we have } o(q,\alpha_A,\alpha_{\mathbb{A}\mathrm{gt}\setminus A}) \in [\![\varphi]\!]_{M,\mathcal{E}}\}.$$

Alternating μ-calculus is strictly more expressive than $\mathbf{ATL}^*$, i.e., every formula of $\mathbf{ATL}^*$ can be translated into an equivalent formula of $\mathbf{AMC}$. Like for $\mathbf{CTL}^\star$, the translation of full $\mathbf{ATL}^*$ is technically involved, but the strategic-temporal modalities of "vanilla" $\mathbf{ATL}$ can be translated to fixpoint formulae according to a simple scheme:

$$\begin{aligned}
tr(\langle\!\langle A\rangle\!\rangle\mathrm{F}\varphi) &\equiv \mu Z\,.\,(tr(\varphi) \vee \langle\!\langle A\rangle\!\rangle\mathrm{X}Z);\\
tr(\langle\!\langle A\rangle\!\rangle\mathrm{G}\varphi) &\equiv \nu Z\,.\,(tr(\varphi) \wedge \langle\!\langle A\rangle\!\rangle\mathrm{X}Z);\\
tr(\langle\!\langle A\rangle\!\rangle\varphi_1\,\mathrm{U}\,\varphi_2) &\equiv \mu Z\,.\,(tr(\varphi_2) \vee tr(\varphi_1) \wedge \langle\!\langle A\rangle\!\rangle\mathrm{X}Z).
\end{aligned}$$

Thus, alternating μ-calculus can serve as a logical "assembly language" for reasoning about outcomes of perfect information strategies.

Axiomatic System for ATL

$\mathbf{ATL}$ extends coalition logic with long-term modalities G, U. Thus, in order to obtain an axiomatization of $\mathbf{ATL}$, we first need to rephrase the inference system for $\mathbf{CL}$, obtaining the following axioms:

1. Complete set of axioms for classical propositional logic,
2. $\langle\!\langle \mathbb{A}\mathrm{gt}\rangle\!\rangle\mathrm{X}\top$,
3. $\neg\langle\!\langle A\rangle\!\rangle\mathrm{X}\bot$,
4. $\neg\langle\!\langle\emptyset\rangle\!\rangle\mathrm{X}\varphi \to \langle\!\langle\mathbb{A}\mathrm{gt}\rangle\!\rangle\mathrm{X}\neg\varphi$,
5. $\langle\!\langle A_1\rangle\!\rangle\mathrm{X}\varphi \wedge \langle\!\langle A_2\rangle\!\rangle\mathrm{X}\psi \to \langle\!\langle A_1 \cup A_2\rangle\!\rangle\mathrm{X}(\varphi \wedge \psi)$ for any disjoint $A_1, A_2 \subseteq \mathbb{A}\mathrm{gt}$,

together with the inference rules *Modus Ponens* and *Monotonicity*:

$$\frac{\varphi \to \psi}{\langle\!\langle A\rangle\!\rangle\mathrm{X}\varphi \to \langle\!\langle A\rangle\!\rangle\mathrm{X}\psi}\,.$$

Modalities $\langle\!\langle A\rangle\!\rangle\mathrm{G}$ and $\langle\!\langle A\rangle\!\rangle\,\mathrm{U}$ satisfy the following axioms that define them recursively as fixpoints of certain monotone operators:

(FP$_{\mathrm{G}}$) $\langle\!\langle A\rangle\!\rangle\mathrm{G}\varphi \leftrightarrow \varphi \wedge \langle\!\langle A\rangle\!\rangle\mathrm{X}\langle\!\langle A\rangle\!\rangle\mathrm{G}\varphi$,

(GFP$_{\mathrm{G}}$) $\langle\!\langle\emptyset\rangle\!\rangle\mathrm{G}(\theta \to (\varphi \wedge \langle\!\langle A\rangle\!\rangle\mathrm{X}\theta)) \to \langle\!\langle\emptyset\rangle\!\rangle\mathrm{G}(\theta \to \langle\!\langle A\rangle\!\rangle\mathrm{G}\varphi)$,

(FP$_{\mathcal{U}}$) $\langle\langle A\rangle\rangle\psi \,\mathrm{U}\, \varphi \leftrightarrow \varphi \vee (\psi \wedge \langle\langle A\rangle\rangle \mathrm{X} \langle\langle A\rangle\rangle\psi \,\mathrm{U}\, \varphi)$,

(LFP$_{\mathcal{U}}$) $\langle\langle\emptyset\rangle\rangle \mathrm{G}((\varphi \vee (\psi \wedge \langle\langle A\rangle\rangle \mathrm{X}\theta)) \to \theta) \to \langle\langle\emptyset\rangle\rangle \mathrm{G}(\langle\langle A\rangle\rangle\psi \,\mathrm{U}\, \varphi \to \theta)$.

Adding yet another inference rule for $\langle\langle\emptyset\rangle\rangle$G-*Necessitation*:

$$\frac{\varphi}{\langle\langle\emptyset\rangle\rangle \mathrm{G}\varphi}$$

we obtain a sound and complete axiomatization for "vanilla" $\mathbf{ATL}$.

No explicit axiomatizations of $\mathbf{ATL}^*$ are known yet.

Bisimulation for ATL

The concept of *bisimulation* is central in formal analysis of discrete processes. In process-algebraic approaches, bisimulation allows to compare seemingly different process descriptions, and prove them equivalent. This in turn provides a powerful machinery for checking correctness of an implementation with respect to specification, both given by the same kind of mathematical structures.

In modal logic, bisimulation is typically understood as a relationship between semantic structures, i.e., models. It is often studied in the context of expressiveness, more specifically: the *distinguishing power* of a given logic. That is, a suitable bisimulation for logic L is such that two bisimilar structures satisfy exactly the same formulae of L. An adaption of the standard concept of bisimulation to $\mathbf{ATL}^*$ and concurrent game models is presented below.

Let $D(q, A) = \prod_{a \in A} d_a(q)$ denote the set of *joint actions* of coalition A in state q. For $\vec{a}_A \in D(q, A)$, we use

$$next_M(q, \vec{a}_A) = \{o(q, \alpha) : \text{ there is } \alpha \in D(q, \mathbb{A}\mathrm{gt}) \text{ such that } \vec{a}_a = \alpha_a \text{ for all } a \in A\}$$

to denote the set of possible successor states in M when coalition A choose actions $\vec{a}_A$.

Given two concurrent game models $M_1 = (\mathbb{A}\mathrm{gt}, St_1, Act_1, d_1, o_1, \mathcal{V}_1)$ and $M_2 = (\mathbb{A}\mathrm{gt}, St_2, Act_2, d_2, o_2, \mathcal{V}_2)$, and a set of agents $A \subseteq \mathbb{A}\mathrm{gt}$, a relation $\beta \subseteq St_1 \times St_2$ is an A-*bisimulation* between M_1 and M_2, denoted $M_1 \rightleftarrows^A_\beta M_2$, iff it satisfies the following conditions for each pair of states $q_1 \in St_1, q_2 \in St_2$ such that $q_1 \beta q_2$:

Local harmony $\mathcal{V}_1(q_1) = \mathcal{V}_2(q_2)$;

Forth For every joint action $\vec{a}^1_A \in D_1(q_1, A)$ for A, there exists a joint action $\vec{a}^2_A \in D_2(q_2, A)$ for A such that for all states $q'_2 \in next_{M_2}(q_2, \vec{a}^2_A)$, there exists a state $q'_1 \in next_{M_1}(q_1, \vec{a}^1_A)$ such that $q'_1 \beta q'_2$;

Back Likewise, for 1 and 2 swapped.

Relation β is a *strategic bisimulation* (also called *alternating bisimulation*) between M_1 and M_2 iff $M_1 \rightleftarrows^A_\beta M_2$ for every $A \subseteq \mathbb{A}\mathrm{gt}$. We denote this by $M_1 \rightleftarrows_\beta M_2$.

Theorem 15 $\mathbf{ATL}^*$ *is invariant under bisimulation. That is, if* $M_1 \rightleftarrows_\beta M_2$ *and* $q_1 \beta q_2$*, then for every* $\mathbf{ATL}^*$ *formula* φ *we have that:*

$$M_1, q_1 \models \varphi \quad \textit{iff} \quad M_2, q_2 \models \varphi.$$

Note that the above theorem is correct only for the perfect recall semantics of $\mathbf{ATL}^*$, based on IR strategies. $\mathbf{ATL}^*$ with memoryless strategies is *not* invariant under bisimulation, as we will soon demonstrate.

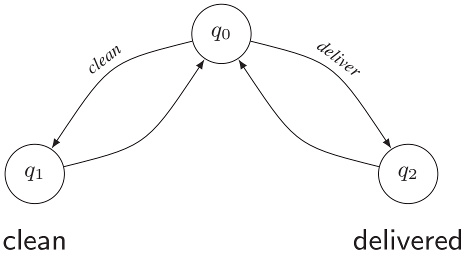


Figure 4.6: Single-agent model M_{clean}: robot with multiple tasks

The Role of Memory

In Section 4.2.2, two types of strategies were introduced. Memoryless strategies (or Ir strategies) assign actions to states of the system, whereas perfect recall strategies (or IR strategies) are mappings from possible histories of the game to actions. The semantics of $\langle\!\langle A \rangle\!\rangle$ can be defined in both ways, by restricting the quantification in the semantic clause ("there is a strategy") to either Σ_A^{Ir} or Σ_A^{IR}. To make the distinction formal, we will refer to the two resulting semantics of $\mathbf{ATL}^*$ as $\models_{\mathrm{Ir}}$ and $\models_{\mathrm{IR}}$. The corresponding logical systems will be denoted by $\mathbf{ATL}^*_{\mathrm{Ir}}$ and $\mathbf{ATL}^*_{\mathrm{IR}}$, respectively.

Note that, up to now, we have used $\models_{\mathrm{IR}}$ as the semantic relation.

We can now show that, for "vanilla" $\mathbf{ATL}$, memory does not matter: the interpretation of a formula is exactly the same when agents use perfect recall strategies and when they use only memoryless strategies. On the other hand, memory matters for the full language of $\mathbf{ATL}^*$: then, the truth of a formula depends on the type of recall characterizing agents in the coalition. The proof relies on the following, rather technical result:

Theorem 16 $\mathbf{ATL}_{\mathrm{Ir}}$ *is invariant under bisimulation. That is, if* $M_1 \rightleftarrows_\beta M_2$ *and* $q_1 \beta q_2$*, then for every* $\mathbf{ATL}$ *formula* φ *we have that:* $M_1, q_1 \models_{\mathrm{Ir}} \varphi$ *iff* $M_2, q_2 \models_{\mathrm{Ir}} \varphi$.

Now, we obtain the following.

Theorem 17 *For "vanilla"* $\mathbf{ATL}$*, memory does not matter, i.e.,*

$$M, q \models_{\mathrm{Ir}} \varphi \quad \textit{iff} \quad M, q \models_{\mathrm{IR}} \varphi.$$

We sketch the proof as the reasoning is illustrative of how strategic bisimulation can be used to prove meta-properties of $\mathbf{ATL}$.

Proof. Let $hist_M(q)$ denote the set of finite prefixes of computations starting in q, and let $last(q_0 \dots q_k) = q_k$. Given a CGM $M = (\mathbb{A}\mathrm{gt}, St, Act, d, o, \mathcal{V})$ and $q \in St$, the tree unfolding $T(M, q)$ of M from q is defined as follows:

$$T(M, q) = (\mathbb{A}\mathrm{gt}, St^*, Act, d^*, o^*, \mathcal{V}^*),$$

where $St^* = hist_M(q)$, $\mathcal{V}^*(h) = \mathcal{V}(last(h))$, $d_i^*(h) = d_i(last(h))$, and $o^*(h, \alpha) = h \cdot o(last(h), \alpha)$.

First, we observe that the tree unfolding includes exactly the same possibilities of action as the original model. Formally, for any (M, q), we have that $T(M, q) \rightleftarrows_\beta M$ with $\beta = \{(h, last(h)) \mid h \in hist_M(q)\}$. Secondly, perfect recall strategies in

M correspond exactly to memoryless strategies in the tree unfolding of M. Thus, $T(M,q),q \models_{\mathrm{Ir}} \varphi$ iff $M,q \models_{\mathrm{IR}} \varphi$ for every q and φ. Finally, by invariance under bisimulation, we obtain that $M,q \models_{\mathrm{Ir}} \varphi$ iff $T(M,q),q \models_{\mathrm{Ir}} \varphi$ iff $M,q \models_{\mathrm{IR}} \varphi$. ■

On the other hand, memory matters for $\mathbf{ATL}^*$.

Theorem 18 *There is a pointed model* (M,q) *and a formula* φ *of* $\mathbf{ATL}^*$ *such that the truth of* φ *in the model is different in the* Ir *and* IR *semantics.*

Proof. Consider the single-agent CGM M_{clean} given in Figure 4.6 and the $\mathbf{ATL}^*$ formula $\varphi \equiv \langle\!\langle a \rangle\!\rangle(\mathsf{F}\mathsf{clean} \wedge \mathsf{F}\mathsf{delivered})$. It is easy to see that $M_{clean}, q_0 \models_{\mathrm{IR}} \varphi$ but $M_{clean}, q_0 \not\models_{\mathrm{Ir}} \varphi$. ■

Corollary 1 $\mathbf{ATL}^*_{\mathrm{Ir}}$ *is not invariant under bisimulation.*

4.3.4 Back to Motivating Examples

We wrap up the section by showing how the strategic properties of rescue robots and voting systems can be expressed in $\mathbf{ATL}^*$.

Example 47 (Rescue Robots: Expressing the properties)

♣ The robots can rescue all the people in the building:

$$\bigwedge_{j \in People} \langle\!\langle Robots \rangle\!\rangle \mathrm{F}\mathsf{safe_j}.$$

Another interpretation: $\langle\!\langle Robots \rangle\!\rangle \mathrm{F}\big(\bigwedge_{j \in People} \mathsf{safe_j}\big)$.

♣ If person j gets outside the building, then she can stay away from trouble forever:

$$\langle\!\langle \emptyset \rangle\!\rangle \mathrm{G}(\mathsf{outside_j} \rightarrow \langle\!\langle j \rangle\!\rangle \mathrm{G}\mathsf{safe_j}).$$

♣ Person j may be rescued without any robot ever entering the building, but guaranteed rescue requires some robots to enter *(new interpretation)*:

$$\langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \big(\mathrm{F}\mathsf{safe_j} \wedge \mathrm{G}(\bigwedge_{i \in Robots} \mathsf{outside_i})\big)$$
$$\wedge \quad \neg\langle\!\langle Robots \rangle\!\rangle \big(\mathrm{F}\mathsf{safe_j} \wedge \mathrm{G}(\bigwedge_{i \in Robots} \mathsf{outside_i})\big).$$

♣ The robots can rescue all the people, and they know that they can:
Cannot be expressed in ATL*!

♣ The robots can rescue all the people, and they know how to do it:
Cannot be expressed in ATL*!

Example 48 (Voting: Expressing the properties)

♠ The system cannot reveal how a particular voter voted:

$$\neg\langle\langle system\rangle\rangle \mathrm{F}\Big(\bigvee_{c\in Candidates} \mathsf{revealedVote}_{\mathsf{v,c}}\Big).$$

♠ The voter can gain no receipt which can be used to prove that she voted in a certain way:

$$\neg\langle\langle v\rangle\rangle \mathrm{F}\Big(\bigvee_{c\in Candidates} \mathsf{receiptVote}_{\mathsf{v,c}}\Big).$$

♠ The voter cannot cooperate with the coercer to prove to him that she voted in a certain way:

Cannot be expressed in ATL*!

(we need a notion of knowledge for the coercer)

References and Further Reading. A survey of logical formalisms for strategic reasoning can be found in the handbook chapter [36].

Coalition logic was introduced and studied in [129, 131]. Alternating-time temporal logic was originally proposed in [10], and later substantially revised in [12]. Our presentation of **ATL*** and its properties follows the latter paper. Different semantics of **ATL***, based on concurrent game models, alternating transition systems, and effectivity functions were compared in [67, 68]. Meta-properties of **ATL** that were studied include expressivity [12, 108], axiomatization [72], and model equivalence [13, 4].

4.4 Other Logical Approaches to Ability

It is rather surprising that until the 1980s formal logic had been seldom employed to either analyze or facilitate strategic reasoning. However, with the ongoing invasion of logic into multi-agent systems over the past 20 years, its role in strategic reasoning has become increasingly more instrumental and recognized. The logical systems presented and discussed in this chapter focus mainly on reasoning about objective strategic abilities of players and coalitions pursuing a specific goal in competitive concurrent multi-player games where the remaining players are regarded as adversaries. We conclude the chapter by presenting an overview of several related logic-based approaches to strategic reasoning.

Logics for Compositional Reasoning about Strategies

Modern approaches at game logics have been initiated by Parikh in 1985 [127]. In his view, strategies are treated as first-class citizens to which an endogenous, structural view is applied, and "the study of rationality in extensive form games largely takes a functional view of strategies." In a way, Parikh's approach pre-dates **ATL**-style reasoning about strategies, and combines it with dynamic logic-style compositional reasoning about complex programs.

A similar approach was later adopted by Harrenstein et al. in [77], with the main goal being a logical characterization of game-theoretic solution concepts, such as Nash equilibrium.

Philosophical Logics of Agency

Philosophical attempts to come up with a logic of agency and ability began with early works of von Wright and Kanger. More recently, Brown [30] proposed in 1988 a modal logic formalising the idea that the modality for ability should have a more complex, existential-universal meaning (the agent has *some* action or choice, such that *every* outcome potentially resulting from executing that action achieves the aim).

At about the same time, Belnap and Perloff developed in [20] the basics of their theory of *seeing to it that*, usually abbreviated to $\mathtt{stit}$. The $\mathtt{stit}$ family of logics were formulated in an attempt to define a formal semantics of the concept of agency, where an agent is loosely understood as an entity that makes deliberate purposeful actions. Although accounts differ on details, the general aim is to make sense of statements such as "the agent a purposefully sees to it that φ" denoted by the logical formula $\mathtt{stit}_a\varphi$. The semantics of Belnap and Perloff's operator (known as *achievement* $\mathtt{stit}$) is presented below. We base our exposition of the main concepts on [84, 28]. For a broader discussion and extensions of $\mathtt{stit}$, we refer the reader to [20, 21, 83, 28, 82, 26].

Models of "seeing to it that" take branching time structures as the starting point, and enhance them to give account of how agents can influence the dynamics of the system. Formally, a $\mathtt{stit}$ *frame* is a tuple $(St, <, \mathbb{A}\mathrm{gt}, \mathit{Choice})$ where:

- $(St, <)$ is a branching-time structure;
- $\mathbb{A}\mathrm{gt}$ is a finite set of agents;
- $\mathit{Choice} : \mathbb{A}\mathrm{gt} \times St \to 2^{2^{\mathit{Paths}}}$ assigns agents with their choices. Paths is the set of all maximal linearly ordered sequences of points in $(St, <)$.[4] Moreover, for every $a \in \mathbb{A}\mathrm{gt}$ and $q \in St$, we requite that $\mathit{Choice}(a, q)$ is a partition of the set $\mathit{Paths}(q)$ of all paths passing through q. The partition represents the available choices for a at q, similarly to effectivity functions.

A $\mathtt{stit}$ *model* extends a $\mathtt{stit}$ frame with a valuation of atomic propositions into sets of paths.

Note that, since $(St, <)$ is a tree, we can see the elements of St as both states and (finite) histories of interaction. To avoid confusion, they will be referred to in the remainder of this section as *positions*. Moreover, for $\mathtt{stit}$ models, the concepts of memoryless and perfect recall play coincide.

Collective choices – when considered – are usually assumed to independently influence the resulting evolution of the system. Thus, the outcome of a collective choice can be seen as the intersection of the individual choices that it combines. This can be formally modeled by extending the function Choice to type $2^{\mathbb{A}\mathrm{gt}} \times St \to 2^{2^{\mathit{Paths}}}$ as follows. First, for each $q \in St$, a *choice selection function at* q is a function $s_q : \mathbb{A}\mathrm{gt} \to 2^{\mathit{Paths}(q)}$, such that $s_q(a) \in \mathit{Choice}(a, q)$ for each $a \in \mathbb{A}\mathrm{gt}$. The set of all selection functions s_q for a given q is denoted by Select_q. Now, for any $A \subseteq \mathbb{A}\mathrm{gt}$ and $q \in St$, we define

$$\mathit{Choice}(A, q) = \big\{ \textstyle\bigcap_{a \in A} s_q(a) \mid s_q \in \mathit{Select}_q \big\}.$$

[4] In the literature on $\mathtt{stit}$, such sequences are called *histories*, and their set is denoted by H. We use the term *paths* here to be consistent with the terminology used throughout the book.

It is easy to see that $Choice(A, q)$ forms a partition of $Paths(q)$ refining each of the individual partitions $Choice(a, q)$ for $a \in A$, and representing the possible collective choices of A.

The following condition of *Independence of agents' choices* must hold for $Choice$:

$$\emptyset \notin Choice(\mathbb{A}\mathrm{gt}, q) \text{ for all } q \in St.$$

Roughly speaking, we say that $\mathbf{stit}_a\varphi$ holds in a world/moment pair if there was some earlier moment in the history at which point the agent a made a choice such that:

- φ is true in all histories consistent with that choice, and
- at the point where the choice was made, the status of φ was not settled, i.e., there were possible histories in which φ was false.

Expressed differently, $\mathbf{stit}_a\varphi$ means that the agent a made a choice such that φ was a necessary consequence of this choice, while φ would not necessarily have been true had the agent not made the choice.

Troquard [163] presented a survey of $\mathbf{stit}$ variants, in which he discussed a number of possible axioms characterising $\mathbf{stit}$ statements. Some relevant candidate axioms and deduction rules for a logic of agency are:

(M) $\mathbf{stit}_a(\varphi \wedge \psi) \rightarrow (\mathbf{stit}_a\varphi \wedge \mathbf{stit}_a\psi)$

(C) $(\mathbf{stit}_a\varphi \wedge \mathbf{stit}_a\psi) \rightarrow \mathbf{stit}_a(\varphi \wedge \psi)$

(N) $\mathbf{stit}_a\top$

(No) $\neg\mathbf{stit}_a\top$

(T) $\mathbf{stit}_a\varphi \rightarrow \varphi$

(RE) If $\varphi \leftrightarrow \psi$, then $\mathbf{stit}_a\varphi \rightarrow \mathbf{stit}_a\psi$.

Taken together, axioms (M) and (C) essentially state that *seeing to it that* is compositional with respect to conjunction: a sees to it that $\varphi \wedge \psi$ iff a sees to φ and ψ. Axioms (N) and (No) are clearly contradictory, and one can accept at most one of them. Axiom (N) says that agent a sees to all the inevitabilities, while (No) says that a cannot bring about things that are inevitable anyway. Most treatments of $\mathbf{stit}$ reject axiom (N), preferring instead the axiom (No). Axiom (T) essentially states that agents are successful: if a brings it about that φ, then φ becomes indeed the case.

$\mathbf{stit}$ logics are connected to alternating-time temporal logic through a number of formal results. In particular, Broersen, Herzig, and Troquard have shown how $\mathbf{stit}$ logics can be used to extend and embed $\mathbf{ATL}^*$ [29, 28, 26]. The price to pay for that expressiveness is intractability and – usually – undecidability of reasoning with $\mathbf{stit}$.

ATL* with Persistent Strategies

The introduction of $\mathbf{ATL}^*$ triggered much activity related to logical foundations of multi-agent systems. Many extensions have been proposed, most notably allowing for persistent strategy commitment and explicit references to concrete strategies in the logical language.

The meaning of the $\mathbf{ATL}^*$ formula $\langle\!\langle A \rangle\!\rangle\gamma$ is that coalition A has a collective strategy, say s_A, to bring about the truth of γ if the agents in A follow that strategy. However, the evaluation of γ in the possible paths of the system enabled by s_A does not take that strategy into account anymore. That is, if γ contains a subformula $\langle\!\langle B\rangle\!\rangle\gamma'$, then in the evaluation of $\langle\!\langle B\rangle\!\rangle\gamma'$ the agents in $A \cap B$ are free to choose *any* other strategy as part of their collective play within B. This is in agreement with the semantics of path quantifiers in $\mathbf{CTL}^\star$, where it is natural to read claims like $\mathsf{E}\mathsf{G}\mathsf{E}\gamma$ as "there is a path such that the system can always deviate to another path satisfying γ." One may argue, however, that it disagrees with the game-theoretic view of a strategy as a full conditional plan that *completely specifies the agent's future behavior*. The issue has been independently addressed in different ways in [4, 138, 25], where various proposals were made in order to incorporate strategic commitment and persistent strategies in the syntax and semantics of $\mathbf{ATL}^*$.

$\mathbf{ATL}^*$ *with irrevocable strategies (*$\mathbf{IATL}^*$*)* [4] redefines the semantics of strategic modalities by assuming that agents' strategic choices persist forever. This is implemented by pruning all the transitions that are not compliant with the selected strategies. Formally, let M be a CGM, A a coalition, and s_A a strategy for A. The *update of* M *by* s_A, denoted $M \dagger s_A$, is the model M where the choices of every agent $a \in A$ are permanently fixed by the strategy $s_A[a]$; that is, $d_a(q) = \{s_a(q)\}$ for each state q. The semantics of strategic play in $\mathbf{IATL}^*$ is now defined as follows:

$M, q \models \langle\!\langle A\rangle\!\rangle\gamma$ iff there is a collective strategy s_A such that for every path $\lambda \in out(q, s_A)$ we have that $M \dagger s_A, \lambda \models \gamma$.

A somewhat different and more flexible approach has been proposed by Brihaye et al. [25]. Instead of a "hard" model update that transforms the CGM according to the chosen strategy, the model is kept intact and the strategy is only added to the *strategy context*. The context collects strategies being currently executed, and hence influences the outcome paths that can occur. On the other hand, since the model itself does not change, each strategy can be revoked – in particular when an agent chooses another strategy in a nested cooperation modality. Formally, let s_A be a joint strategy of agents A (the current strategy context), and let t_B be a new joint strategy of agents B. We define the *context update* $s_A \circ t_B$ as the joint strategy f for agents in $A \cup B$ such that $f[i] = t_B[i]$ for $i \in B$ and $f[i] = s_A[i]$ for $i \in A \setminus B$. That is, the new strategies from t_B are added to the context, possibly replacing some of the previous ones. The semantic rule for strategic modalities becomes:

$M, q, f \models \langle\!\langle A\rangle\!\rangle\gamma$ iff there is a joint strategy s_A for the agents in A such that for every path $\lambda \in out(q, f \circ s_A)$ we have that $M, \lambda, f \circ s_A \models \gamma$.

Additionally, $M, q \models \varphi$ iff $M, q, f_\emptyset \models \varphi$, where $f_\emptyset$ is the only joint strategy of the empty coalition (i.e., the empty strategy).

For more details and a thorough analysis of the model checking problem for $\mathbf{ATL}^*$ with strategy contexts, we refer the reader to [25]. The satisfiability problem was investigated in [164], and proved undecidable even for "vanilla" $\mathbf{ATL}$.

Reasoning about Particular Strategies

Counterfactual $\mathbf{ATL}$ *(*$\mathbf{CATL}$*)*, proposed by van der Hoek et al. [167], extends $\mathbf{ATL}$ with operators of "counterfactual commitment" $\mathcal{C}_a(\sigma, \varphi)$ where a is an agent, σ is a strategic term, and φ is a formula. The informal reading of $\mathcal{C}_a(\sigma, \varphi)$ is: *"if it were the case that agent a committed to strategy σ, then φ would hold."* The semantics is based on model updates:

$$M, q \models \mathcal{C}_a(\sigma, \varphi) \quad \text{iff} \quad M \dagger [\![\sigma]\!]_a, q \models \varphi$$

where $[\![\sigma]\!]_a$ is the strategy of agent a denoted by σ.

$\mathbf{ATL}$ *with Intentions* ($\mathbf{ATLI}$), proposed by Jamroga et al. [100], is similar to $\mathbf{CATL}$, but its counterfactual operators have a different flavour: $(\mathbf{str}_a\,\sigma)\varphi$ reads as *"suppose that agent a intends to play strategy σ, then φ holds."* An intention is a kind of commitment – it persists – but it can be revoked by switching to another intention. Semantically, this is done by an additional "marking" of the intended actions in the concurrent game model. Moreover, strategies can be nondeterministic, which provides semantic tools for e.g. partial strategies as well as explicit release of commitments. Thus, Jamroga et al. provide in fact the semantics of $\mathbf{ATL}$ based on strategy contexts (here called intentions), that was later explored in more detail by Brihaye and colleagues [25]. However, $\mathbf{ATLI}$ does not allow to *quantify* over intentions, and hence allows only for limited context change. $\mathbf{ATLI}$ and its richer cousin $\mathbf{ATLP}$ ("$\mathbf{ATL}$ with Plausibility" [40]) have been used to e.g. characterize game-theoretic solution concepts and outcomes that can be obtained by rational agents.

Alternating-time temporal logic with explicit strategies ($\mathbf{ATLES}$*)*, see [181], is a revised version of $\mathbf{CATL}$ which dispenses with the counterfactual operators. Instead, strategic modalities are subscripted by *commitment assignments* which are partial functions of the form $\rho = \{a_1 \mapsto \sigma_1, \dots, a_l \mapsto \sigma_l\}$ where each a_j is an agent and σ_j is a strategy term. The meaning of formula $\langle\!\langle A \rangle\!\rangle_\rho \mathrm{G}\varphi$ is that there exists a strategy for $A \cup \{a_1, \dots, a_l\}$, where each a_j is required to play $[\![\sigma_j]\!]$, such that φ will hold. Note that the semantics of $\mathbf{ATLES}$ involves limited strategic commitment. Consider, for instance, formula $\langle\!\langle A \rangle\!\rangle_\rho \mathrm{G} \langle\!\langle A \rangle\!\rangle_\rho \mathrm{F}\varphi$. If A is a subset of the domain of ρ, then in the evaluation of the subformula $\langle\!\langle A \rangle\!\rangle_\rho \mathrm{F}\varphi$, the agents in A are bound to play the same joint strategy they selected for the outer modality $\langle\!\langle A \rangle\!\rangle_\rho \mathrm{G}$.

Alternating-time temporal epistemic logic with actions ($\mathbf{ATELA}$*)*, proposed by Ågotnes [3], enables reasoning about the interplay between explicit strategies of bounded length and agents' knowledge. The issue is very important, and we will focus on it specifically in Chapter 6.

Explicit Quantification on Strategies

Strategy Logic, introduced by Chatterjee et al. [44] treats strategies in two-player turn-based games as first-order variables that can be subject to explicit existential and universal quantification. This enables specification of important properties for non-zero-sum games in a simple and natural way. In particular, the one-alternation fragment of strategy logic subsumes $\mathbf{ATL}^*$ and is expressive enough to characterize the existence of Nash equilibria and secure equilibria.

The idea of explicit quantification over strategies has been subsequently followed in a series of papers by Mogavero, Murano and colleagues [118, 116, 117], where Strategy Logic was extended and generalized to k-player concurrent games. This degree of expressivity yields undecidability of all the important decision problems (satisfiability, validity, model checking). However, a decidable fragment has been identified, which is strictly more expressive than $\mathbf{ATL}^*$, and yet does not exceed $\mathbf{ATL}^*$ in the complexity of related computational problems.

References and Further Reading. In the quest for the "right" logical toolbox to reason about strategic play, many variants of strategic logics have been proposed, mainly extending and/or revising the framework of alternating-time temporal logic. Some approaches are mentioned above. Other interesting extensions include rationality assumptions and solution concepts [177, 40], agents with with bounded memory [6], bounded resources [8, 34, 9, 35], coalition formation and negotiation [32], opponent modeling and action in stochastic environments [91, 37, 152, 151] and reasoning about irrevocable plans and interplay between strategies of different agents [4, 25].

Also, a large number of works have considered the issue of modeling and reasoning about abilities under imperfect information. We will discuss the topic in detail in Chapter 6.

Chapter 5

Verification of Strategic Ability

5.1 Model Checking Strategic Ability

In this section, we look at model checking of alternating-time temporal logic. We start by presenting the standard fixpoint algorithm for global model checking of $\mathbf{ATL}$, shown in Figure 5.1. The main idea is as follows: to verify a formula of type $\langle\langle A\rangle\rangle\gamma$, the algorithm tries to construct a winning strategy for A, i.e., one that guarantees the truth of γ no matter what the other agents do. To achieve that, the procedure starts with the appropriate candidate set of states ($\emptyset$ for U and the whole set St for G), and iterates backwards over A's one-step abilities until the set gets stable.

The correctness of the algorithm follows immediately from the fixpoint characterizations of strategic-temporal modalities, presented already in Section 4.3.3. Note that it does not matter whether perfect recall or memoryless strategies are used: the algorithm is correct for the IR semantics, but it always finds an Ir-strategy.

It is worth pointing out that the algorithm can be easily adapted for the, seemingly much more difficult, task of *strategy synthesis* for temporal goals.

Example 49 (Simple Rocket: Model checking strategic abilities) *To demonstrate how the algorithm in Figure 5.1 works, we revisit the simple rocket scenario of Example 24. The story is extended by assuming that the rocket is operated by 3 workers (called simply* $1, 2,$ *and* 3*) with the following capabilities:*

- *Agent 1 can: try to load the cargo, try to unload the cargo, initiate the flight, or do nothing (action nop);*
- *Agent 2 can do unload or nop;*
- *Agent 3 can do load, refill the fuel tank (action fuel), or do nothing (nop).*

The actual transition depends on the combination of actions that the workers try to simultaneously execute. We assume the following interaction between choices:

- *Flying has highest priority: if agent 1 initiates the flight, current actions of the other agents have no effect;*
- *If loading is attempted when the cargo is not around, nothing happens;*
- *Same for unloading when the cargo is not in the rocket, and refilling a full tank;*

```
**function** $mcheckatl(M, \varphi)$.
$\mathbf{ATL}$ model checking. Returns the set of states in model $M = \langle \mathbb{A}\mathrm{gt}, St, \mathcal{V}, o \rangle$ for
which formula $\varphi$ holds.
**case** $\varphi \in \mathcal{PV}$ **:** return $\mathcal{V}(p)$
**case** $\varphi = \neg\psi$ **:** return $St \setminus mcheckatl(M, \psi)$
**case** $\varphi = \psi_1 \vee \psi_2$ **:** return $mcheckatl(M, \psi_1) \cup mcheckatl(M, \psi_2)$
**case** $\varphi = \langle\langle A \rangle\rangle \mathrm{X}\psi$ **:** return $pre(M, A, mcheckatl(M, \psi))$
**case** $\varphi = \langle\langle A \rangle\rangle \mathrm{G}\psi$ **:**
    $Q_1 := St; \quad Q_2 := mcheckatl(M, \psi); \quad Q_3 := Q_2;$
    **while** $Q_1 \not\subseteq Q_2$
    **do** $Q_1 := Q_2; \quad Q_2 := pre(M, A, Q_1) \cap Q_3$ **od**;
    return $Q_1$
**case** $\varphi = \langle\langle A \rangle\rangle \psi_1 \,\mathrm{U}\, \psi_2$ **:**
    $Q_1 := \varnothing; \quad Q_2 := mcheckatl(M, \psi_1);$
    $Q_3 := mcheckatl(M, \psi_2);$
    **while** $Q_3 \not\subseteq Q_1$
    **do** $Q_1 := Q_1 \cup Q_3; \quad Q_3 := pre(M, A, Q_1) \cap Q_2$ **od**;
    return $Q_1$
**end case**
```

```
**function** $pre(M, A, Q)$.
Auxiliary function; returns the exact set of states $Q'$ such that, when the system is
in a state $q \in Q'$, agents $A$ can cooperate and enforce the next state to be in $Q$.
return $\{q \mid \exists\alpha_A \forall\alpha_{\mathbb{A}\mathrm{gt}\setminus A}\ o(q, \alpha_A, \alpha_{\mathbb{A}\mathrm{gt}\setminus A}) \in Q\}$
```

Figure 5.1: $\mathbf{ATL}$ model checking in explicit models

- *If different agents try to load and unload at the same time, then the majority prevails;*

- *Refilling fuel can be done in parallel with loading/unloading.*

The resulting CGM M_{rockt2} is depicted in Figure 5.2. We only show labels of transitions outgoing from state q_1 to keep the picture readable.

Suppose that we want to find the set of states from which the coalition of agents 1 and 3 can move the cargo to any given location. This corresponds to global model checking of formula $\langle\langle 1,3 \rangle\rangle \mathrm{F}\mathsf{caP} \wedge \langle\langle 1,3 \rangle\rangle \mathrm{F}\mathsf{caL}$ in M_{rockt2}. To save space, we only focus on the first part of the conjunction: Figure 5.3 shows the execution of the $\mathbf{ATL}$ model checking algorithm for formula $\langle\langle 1,3 \rangle\rangle \mathrm{F}\mathsf{caP}$ by computing the appropriate least fixpoint. As it turns out, all the states in M_{rockt2} satisfy the specification.

Does anything change when the team consists of agents 1 and 2? This can be answered by model checking formula $\langle\langle 1,2 \rangle\rangle \mathrm{F}\mathsf{caP}$; the computation is shown in Figure 5.4. Indeed, coalition $\{1,2\}$ can guarantee that the cargo gets to Paris only if the initial state of the system is $q_2, q_6, q_7, q_8, q_9, q_{10}, q_{11}$ or q_{12}.

Finally, we can use model checking to find out whether one of the workers, say 3, can keep the cargo forever at the Paris airport. To this end, Figure 5.5 shows the computation of the greatest fixpoint for formula $\langle\langle 3 \rangle\rangle \mathrm{G}\mathsf{caP}$. It turns out that the formula is satisfied only in state q_9, i.e., 3 can guarantee the property only if the cargo is already in Paris, the rocket is in London, and it has empty fuel tank.

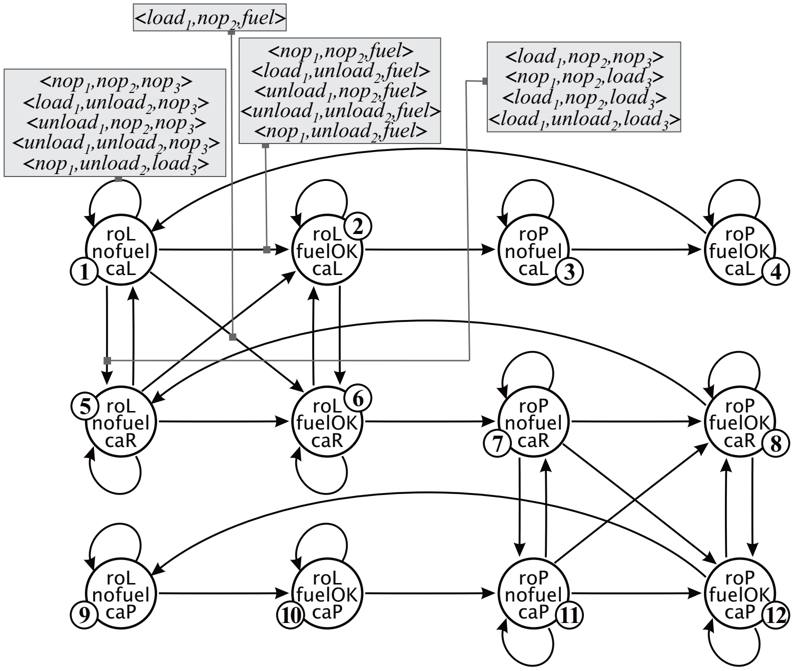


Figure 5.2: Concurrent game model for Simple Rocket M_{rockt2}

References and Further Reading. The standard fixpoint algorithm for model checking **ATL** was presented and discussed extensively in [12]. More advanced techniques were relatively scarce. Unbounded Model Checking was studied in [102], and MCMAS includes an implementation of **ATL** model checking based on Ordered Binary Decision Diagrams [114, 142, 113]. Abstraction techniques for **ATL** and related logics were proposed in [16, 104, 111].

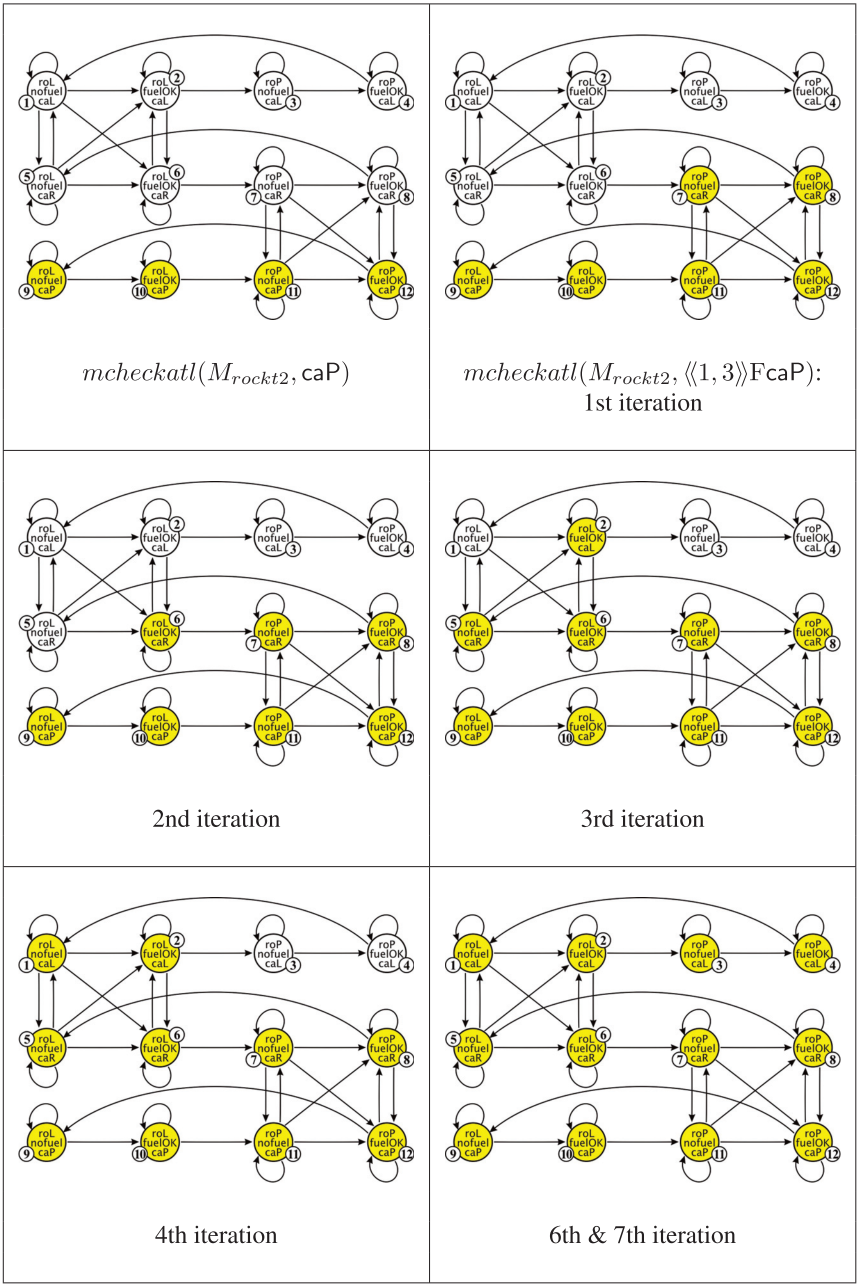
Studies on other decision problems than model checking were much less frequent, though satisfiability of the basic variant of **ATL** has been investigated in [72, 180, 149, 71].

5.2 Complexity of Verification

We will now discuss the most important complexity results for model checking of **ATL*** and its subsets.

5.2.1 Model Checking ATL and CL

One of the main results concerning **ATL** says that it can be model-checked in deterministic linear time, analogously to **CTL**. It is easy to see that the algorithm from Section 5.1 needs to traverse each transition at most once per state from St and subformula of φ. Formally, function $mcheckatl$ is called at most $\mathbf{O}(|\varphi|)$ times, and each call terminates after $\mathbf{O}(|St| \cdot |o|)$ steps. The latter can be improved by translating the model



$mcheckatl(M_{rockt2}, \mathsf{caP})$

$mcheckatl(M_{rockt2}, \langle\langle 1,3 \rangle\rangle \mathrm{F}\mathsf{caP})$: 1st iteration

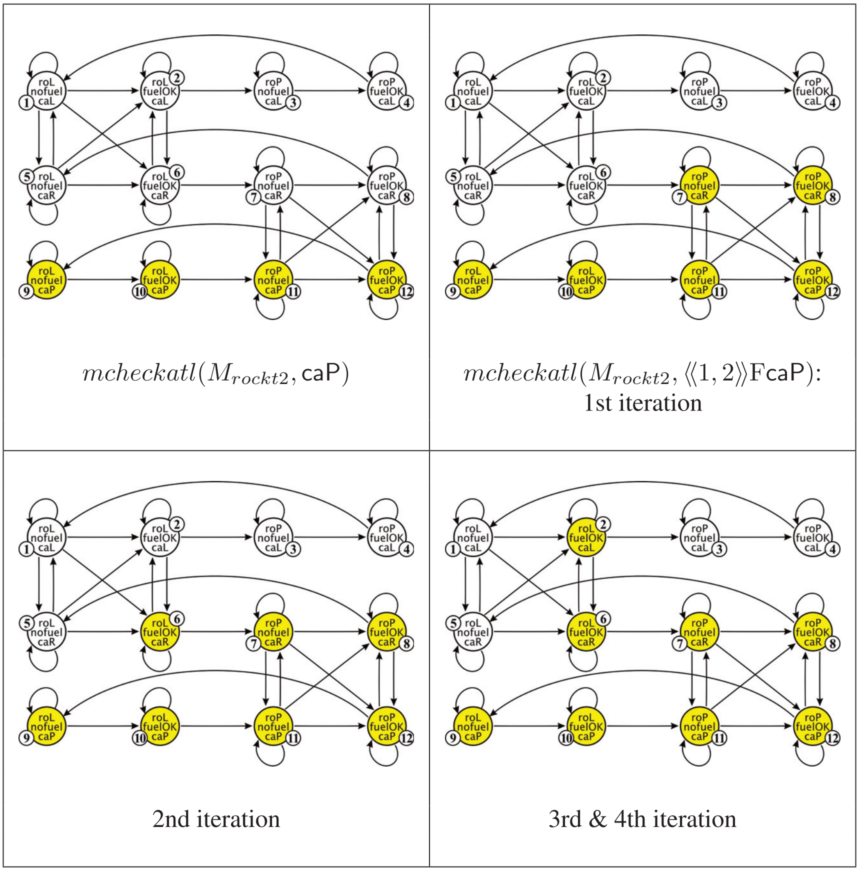
2nd iteration

3rd iteration

4th iteration

6th & 7th iteration

Figure 5.3: Model checking **ATL**: $mcheckatl(M_{rockt2}, \langle\langle 1,3 \rangle\rangle \mathrm{F}\mathsf{caP})$



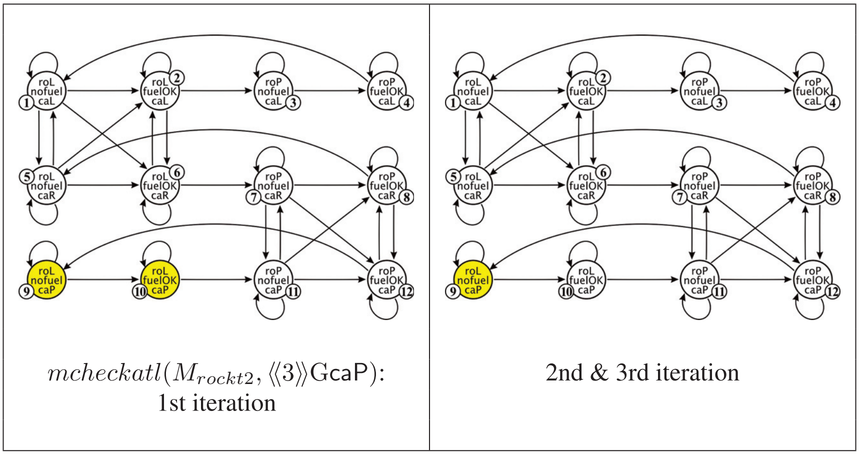
$mcheckatl(M_{rockt2}, \mathsf{caP})$

$mcheckatl(M_{rockt2}, \langle\langle 1,2 \rangle\rangle \mathrm{F}\mathsf{caP})$: 1st iteration

2nd iteration

3rd & 4th iteration

Figure 5.4: Model checking **ATL**: $mcheckatl(M_{rockt2}, \langle\langle 1,2 \rangle\rangle \mathrm{F}\mathsf{caP})$



$mcheckatl(M_{rockt2}, \langle\langle 3 \rangle\rangle \mathrm{G}\mathsf{caP})$: 1st iteration

2nd & 3rd iteration

Figure 5.5: Model checking **ATL**: $mcheckatl(M_{rockt2}, \langle\langle 3 \rangle\rangle \mathrm{G}\mathsf{caP})$

to a two-player game, and then solving the "invariance game" [18] on it in polynomial time. $\mathbf{P}$-hardness can be shown by a reduction of reachability in And-Or-Graphs, which is $\mathbf{P}$-complete [88], to model checking the constant size $\mathcal{L}_{ATL}$-formula $\langle\langle 1\rangle\rangle \mathrm{F}\mathsf{p}$ in a two-player game. Each Or-state is controlled by player 1, and each And-state is "owned" by player 2. In consequence, we get the following.

Theorem 19 *Model checking* $\mathbf{ATL}$ *is* $\mathbf{P}$*-complete, and can be performed in time* $\mathbf{O}(|M| \cdot |\varphi|)$*, where* $|M|$ *is the number of the transitions in* M*, and* $|\varphi|$ *is the number of the subformulae in* φ*.*

So, it seems that $\mathbf{ATL}$ is strictly more expressive than $\mathbf{CTL}$ with no computational price to pay. It is important to emphasize, however, that the result is relative to the number of the transitions in the model. We will come back to this issue in Section 5.3.

The next theorem states that model checking is equally hard even when the language is restricted to "nexttime" modalities. Putting it in another way, model checking coalition logic is as hard as for $\mathbf{ATL}$.

Theorem 20 *Model checking* $\mathbf{CL}$ *is* $\mathbf{P}$*-complete, and can be performed in time* $\mathbf{O}(|M| \cdot |\varphi|)$.

The upper bound follows from Theorem 19. $\mathbf{P}$-hardness can be shown by the following adaption of the reduction of And-Or-Graph reachability. First, we observe that if state y is reachable from state x in graph G then it is also reachable via a path whose length is bounded by the number n of states in the graph. Like in the proof of Theorem 19, we take G to be a turn-based CGM in which player 1 "owns" all the Or-states and player 2 controls all the And-states. We also label node y with a special proposition y, and replace all the transitions outgoing from y with a deterministic loop. Now, we have that y is reachable from x in G iff $G, x \models \big(\langle\langle 1\rangle\rangle \mathrm{X}\big)^n \mathsf{y}$, where $\mathbf{op}^n$ denotes the n-fold repetition of operator $\mathbf{op}$. The reduction uses only logarithmic space.

On the other hand, checking strategic properties in strictly one-step games is somewhat easier. Let us call a formula *flat* if it contains no nested cooperation modalities. Moreover, a formula is *simple* if it is flat and does not include Boolean connectives. In particular, the language of "Simple $\mathbf{CL}$" consists only of formulae p and $\langle\langle A\rangle\rangle \mathrm{X} p$, for $p \in \mathcal{PV}$ and $A \subseteq \mathbb{A}\mathrm{gt}$.

Theorem 21 *Model checking "Simple* $\mathbf{CL}$*" is in* $\mathbf{AC^0}$ *with respect to the number of the transitions in the model and the length of the formula.*[1]

The above results apply to the semantic variants based on memoryless as well as perfect recall strategies, i.e., to both $\mathbf{ATL}_{\mathrm{Ir}}$ and $\mathbf{ATL}_{\mathrm{IR}}$.

5.2.2 Model Checking ATL*

We now turn to the complexity of model checking for the full language of $\mathbf{ATL}^*$. Since the IR and Ir semantics of $\mathbf{ATL}^*$ do not coincide, both variants must be studied separately.

Theorem 22 *Model checking* $\mathbf{ATL}^*_{\mathrm{IR}}$ *is* $\mathbf{2EXPTIME}$*-complete in the number of the transitions in the model and the length of the formula.*

[1] $\mathbf{AC^0}$ is the complexity class corresponding to constant-depth, unbounded-fanin, polynomial-size Boolean circuits with AND, OR, and NOT gates [65].

	Ir	IR
$\mathbf{CTL}$	$\mathbf{P}$-complete	
Simple $\mathbf{CL}$	in $\mathbf{AC^0}$	in $\mathbf{AC^0}$
$\mathbf{CL}$	$\mathbf{P}$-complete	$\mathbf{P}$-complete
$\mathbf{ATL}$	$\mathbf{P}$-complete	$\mathbf{P}$-complete
$\mathbf{ATL}^*$	$\mathbf{PSPACE}$-complete	$\mathbf{2EXPTIME}$-complete

Figure 5.6: Overview of the model checking complexity for concurrent game models. Rows indicate syntactic variants, columns indicate the selected semantics.

The upper bound can be obtained by the following construction. Let M be a CGM, and $\langle\langle A\rangle\rangle\gamma$ be an $\mathbf{ATL}^*$ formula, where γ is a formula of $\mathbf{LTL}$. Given a strategy s_A of A and a state q in M, the model can be unfolded into a q-rooted tree representing all possible behaviors with agents A following their strategy s_A. This structure can be seen as the tree induced by $out(q, s_A)$ and we will refer to it as a (q, A)*-execution tree*. Note that every collective strategy for A may result in a different execution tree. Now, a Büchi tree automaton $\mathcal{A}_{M,q,A}$ can be constructed that accepts exactly (q, A)-execution trees. Moreover, one can construct a Rabin tree automaton which accepts all trees that satisfy the $\mathbf{CTL}^\star$ formula $\mathsf{A}\gamma$ [62]. The $\mathbf{ATL}^*$ formula $\langle\langle A\rangle\rangle\gamma$ holds in M, q iff there is a tree accepted by $\mathcal{A}_{M,q,A}$ and by $\mathcal{A}_\psi$.

The lower bound can be shown by a reduction of $\mathbf{LTL}$ realizability [139, 145].

What about verification of $\mathbf{ATL}^*$ with memoryless strategies? It turns out to be much easier than the perfect recall case. Consider the following nondeterministic algorithm for model checking $\langle\langle A\rangle\rangle\gamma$ where γ is an $\mathbf{LTL}$ formula. First, a memoryless strategy s_A is guessed and the model is "trimmed" according to the strategy, i.e., all transitions which cannot occur by following s_A are removed. Note that a memoryless strategy can be guessed in polynomially many steps with respect to the size of the model, and hence also using only polynomially many memory cells. Then, we model-check the $\mathbf{CTL}^\star$ formula $\mathsf{A}\psi$ in the new model (doable in deterministic polynomial time), and return the result. The procedure runs in $\mathbf{NPSPACE} = \mathbf{PSPACE}$. Moreover, $\mathbf{PSPACE}$-hardness follows from the fact that $\mathbf{ATL}^*_{\mathrm{Ir}}$ embeds $\mathbf{LTL}$ (every $\mathbf{LTL}$ formula φ is equivalent to the $\mathbf{ATL}^*$ formula $\langle\langle\emptyset\rangle\rangle\varphi$), which renders the following result.

Theorem 23 *Model checking $\mathbf{ATL}^*_{\mathrm{Ir}}$ is $\mathbf{PSPACE}$-complete in the number of the transitions in the model and the length of the formula.*

Thus, model checking $\mathbf{ATL}^*$ with memoryless strategies is no more complex than model checking $\mathbf{LTL}$ and $\mathbf{CTL}^\star$, at least in terms of complexity classes. Figure 5.6 presents an overview of the model checking complexity results for strategic logics in concurrent game models.

References and Further Reading. We refer the reader to [33] for an overview of complexity results for model checking strategic logics. Most technical results presented in this section were obtained in [12].

5.3 A Closer Look

The above results are clearly optimistic: model checking $\mathbf{ATL}$ has the same complexity as that of $\mathbf{CTL}$, and verification of $\mathbf{ATL}^*$ with memoryless strategies is no more complex than model checking $\mathbf{CTL}^\star$ and $\mathbf{LTL}$. Thus, apparently, we get extra expressivity for free. Is it really the case? Not quite. The picture changes when we measure the size of models with the number of the states, agents, and actions, rather than the number of the transitions.

We begin by observing that, for CGM's, the number of the transitions can be exponential in the number of the states, actions, and agents. In this sense, the standard fixpoint algorithm for model checking $\mathbf{ATL}$ provides only an exponential-time upper bound for the problem.

Theorem 24 *Let $|St|$ be the number of the states in a concurrent game model M, $|\mathbb{A}\mathrm{gt}|$ denote the number of the agents, and $|Act|$ the maximal number of available decisions (actions) per agent and state. Then, the number of the transitions $|o| = \mathbf{O}(|St| \cdot |Act|^{|\mathbb{A}\mathrm{gt}|})$. Thus, the $\mathbf{ATL}$ model checking algorithm in Figure 5.1 runs in time $\mathbf{O}(|St| \cdot |Act|^{(|\mathbb{A}\mathrm{gt}| \cdot |\varphi|)})$ where $|\varphi|$ is the length of the formula, and hence its complexity is exponential if the number of the agents is a parameter of the problem.*

In comparison, for an unlabeled transition system with $|St|$ states and $|\longrightarrow|$ transitions, we have that $|\longrightarrow| = \mathbf{O}(|St|^2)$. This means that $\mathbf{CTL}$ model checking is in $\mathbf{P}$ also with respect to the number of the states in the model and the length of the formula. The following theorem is an immediate corollary of the fact (and Theorem 6).

Theorem 25 $\mathbf{CTL}$ *model checking over unlabeled transition systems is $\mathbf{P}$-complete in the number of the states and the length of the formula, and can be performed in time $\mathbf{O}(|St|^{(2|\varphi|)})$.*

For $\mathbf{ATL}$ and concurrent game models, however, the situation is different.

5.3.1 Model Checking for Compact Representation of Transitions

In this section we consider the complexity of the model checking problem *with respect to the number of the states, agents, and an implicitly encoded transition function*, rather than the explicit number of the transitions.

Implicit concurrent game models are defined similarly to CGM's but the transition function is encoded in a more compact way by a sequence

$$((\varphi_0^r, q_0^r), \ldots, (\varphi_{t_r}^r, q_{t_r}^r))_{r=1,\ldots,|St|}$$

where $t_r \in \mathbb{N}_0$, $q_i^r \in St$ and each φ_i^r is a Boolean combination of propositions $\mathsf{exec}_\alpha^{\mathsf{j}}$ where $j \in \mathbb{A}\mathrm{gt}$, $\alpha \in Act$, $i = 1, \ldots, t$ and $r = 1, \ldots, |St|$. It is required that $\varphi_{t_r}^r = \top$. Symbol $\mathsf{exec}_\alpha^{\mathsf{j}}$ stands for "agent j executes action α". We use $\varphi[\alpha_1, \ldots, \alpha_k]$ to refer to the Boolean formula over $\{\top, \bot\}$ obtained by replacing $\mathsf{exec}_\alpha^{\mathsf{a_j}}$ with $\top$ (resp. $\bot$) if $\alpha_j = \alpha$ (resp. $\alpha_j \neq \alpha$). The encoding defines the transition function $\hat{o}$ as follows:

$$\hat{o}(q_i, \alpha_1, \ldots, \alpha_k) = q_j^i \text{ where } j = \min\{\kappa \mid \varphi_\kappa^i[\alpha_1, \ldots, \alpha_k] \equiv \top\}$$

That is, $\hat{o}(q_i, \alpha_1, \ldots, \alpha_k)$ returns the state belonging to the formula φ_κ^i (associated with state q_i) with the minimal index κ that evaluates to "true" given the actions $\alpha_1, \ldots, \alpha_k$. Note that the function is well defined as the last formula in each sequence is given by $\top$: no deadlock can occur. The size of $\hat{o}$ is defined as $|\hat{o}| = \sum_{r=1,\dots,|St|}\sum_{j=1,\dots,t_r}|\varphi_j^r|$, that is, the sum of the sizes of all formulae. Hence, the size of an implicit CGM is given by $|St|+|\mathbb{A}\mathrm{gt}|+|\hat{o}|$. Recall that the size of an explicit CGM is $|St|+|\mathbb{A}\mathrm{gt}|+|o|$, where $|o|$ is the number of the transitions.

Theorem 26 *Model checking* $\mathbf{ATL}_{\mathrm{IR}}$ *and* $\mathbf{ATL}_{\mathrm{Ir}}$ *over implicit CGM's is* $\mathbf{\Delta_3^P}$*-complete with respect to the number of the states, agents, and implicit transitions in the model, and the length of the formula.*

The idea of the lower bound proof is clear if we reformulate the model checking of $M, q \models \langle\!\langle a_1,\dots,a_r\rangle\!\rangle \mathrm{X}\varphi$ as

$$\exists(\alpha_1,\dots,\alpha_r)\forall(\alpha_{r+1},\dots,\alpha_k)\; M, o(q,\alpha_1,\dots,\alpha_k)\models\varphi,$$

which closely resembles QSAT$_2$, a typical $\mathbf{\Sigma_2^P}$-complete problem. A reduction of this problem to our model checking problem is straightforward. For each instance of QSAT$_2$, we create a model where the values of propositional variables $p_1,\dots,p_r$ are "declared" by agents A and the values of $p_{r+1},\dots,p_k$ by $\mathbb{A}\mathrm{gt}\setminus A$. The subsequent transition leads to a state labeled by proposition yes iff the given Boolean formula holds for the underlying valuation of $p_1,\dots,p_k$. Then, QSAT$_2$ reduces to model checking formula $\langle\!\langle a_1,\dots,a_r\rangle\!\rangle \mathrm{X}\mathsf{yes}$. In order to obtain $\mathbf{\Delta_3^P}$-hardness, the above schema is combined with nested cooperation modalities, which yields a rather technical reduction of the SNSAT$_3$ problem that can be found in [108].

For the upper bound, consider the following algorithm for checking $M, q \models \langle\!\langle A\rangle\!\rangle\gamma$ with no nested cooperation modalities. First, we guess a strategy s_A of the proponents and fix A's actions to the ones described by s_A. Then we check if $\mathsf{A}\gamma$ is true in state q of the resulting model by asking an oracle about the existence of a counterstrategy $s_{\bar{A}}$ for $\mathbb{A}\mathrm{gt}\setminus A$ that falsifies γ and reverting the oracle's answer. The evaluation takes place by calculating $\hat{o}$ (which takes polynomially many steps) regarding the actions prescribed by $(s_A, s_{\bar{A}})$ at most $|St|$ times. For nested cooperation modalities, we proceed recursively (bottom-up).

Model Checking $\mathbf{ATL}^*$ over Implicit CGM's

Theorem 27 *Model checking* $\mathbf{ATL}^*_{\mathrm{Ir}}$ *over implicit CGM's is* $\mathbf{PSPACE}$*-complete with respect to the number of the states, agents, and implicit transitions in the model, and the length of the formula.*

We observe that every explicit CGM can be encoded as an implicit CGM with no blowup in size. In consequence, the lower bound follows from Theorem 23.

For the upper bound, we model-check $M, q \models \langle\!\langle A\rangle\!\rangle\gamma$ by guessing a memoryless strategy s_A for coalition A. Then we guess a perfect information memoryless counterstrategy $s_{\bar{A}}$ of the opponents. Having a complete strategy profile, we proceed as in the proof of Theorem 29 and check the $\mathbf{LTL}$ path formula γ on the resulting (polynomial model) M' which can be performed in polynomial space (Theorem 23). For nested cooperation modalities, we proceed recursively.

Theorem 28 *Model checking* $\mathbf{ATL}^*_{\mathrm{IR}}$ *over implicit CGM's is* $\mathbf{2EXPTIME}$*-complete with respect to the number of the states, agents, and implicit transitions in the model, and the length of the formula.*

Again, the lower bound follows from Theorem 22. For the upper bound, we have to modify the algorithm given in the proof of Theorem 22 so that it is capable of dealing with implicit models. More precisely, we need to modify the construction of the Büchi automaton $\mathcal{A}_{M,q,A}$ that is used to accept the (q, A)-execution trees. Before, we simply checked all the moves of A in polynomial time and calculated the set of states A is effective for (as the moves are bounded by the number of the transitions). Here, we have to incrementally generate all these moves from A using $\hat{o}$. This may take exponential time (as there can be exponentially many moves in terms of the number of the states and agents). However, as this can be done independently of the non-emptiness check, the overall runtime of the algorithm is still double exponential.

$\mathbf{CTL}$ Revisited

It seemed so far that the complexity of $\mathbf{CTL}$ model checking is not affected when we measure the size of the model in terms of states rather than transitions. Is it really the case? That depends on the representation of transitions. For an *unlabeled* transition system, with the transition relation represented explicitly by pairs of states, the problem stays in deterministic polynomial time, cf. Theorem 25. For transitions labeled by combinations of agents' actions, represented in a compressed way, the problem becomes distinctly harder.

Theorem 29 *Model checking* $\mathbf{CTL}$ *over implicit CGM's is* $\mathbf{\Delta_2^P}$*-complete with respect to the number of the states, agents, and implicit transitions in the model, and the length of the formula.*

To see the upper bound, observe that $M, q \models_{\mathbf{CTL}} \mathsf{E}\gamma$ iff $M, q \models_{\mathbf{ATL_{IR}}} \langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle\gamma$ which is in turn equivalent to $M, q \models_{\mathbf{ATL_{Ir}}} \langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle\gamma$. In other words, $\mathsf{E}\gamma$ holds iff the grand coalition has a *memoryless* strategy to achieve γ. Thus, we can verify $M, q \models_{\mathbf{CTL}} \mathsf{E}\gamma$ (with no nested path quantifiers) as follows: we guess a strategy $s_{\mathbb{A}\mathrm{gt}}$ for $\mathbb{A}\mathrm{gt}$ (in polynomially many steps), then we construct the resulting model M' by asking $\hat{o}$ which transitions are enabled by following the strategy s_A, check if $M', q \models_{\mathbf{CTL}} \mathsf{E}\gamma$, and return the answer. Note that M' is an *unlabeled transition system*, so constructing M' and checking $M', q \models_{\mathbf{CTL}} \mathsf{E}\gamma$ can be done in polynomial time. For nested modalities, we proceed recursively.

The lower bound is obtained by a reduction of the canonical $\mathbf{NP}$-complete problem of boolean satisfiability ($\mathbf{SAT}$), defined formally as follows:

Definition 12 ($\mathbf{SAT}$)

Input: *A boolean formula Φ in Conjunctive Normal Form over propositional variables $x_1, \ldots, x_k$.*

Output: *$\top$ if there exists a valuation ϑ of $x_1, \ldots, x_k$ such that ϑ makes Φ true; otherwise $\bot$.*

Thus, $\mathbf{SAT}$ decides satisfiability of a formula $\Phi \equiv C_1 \wedge \cdots \wedge C_n$ involving k propositional variables from set $X = \{x_1, ..., x_k\}$.

Below we sketch the reduction of $\mathbf{SAT}$ to model checking $\mathbf{CTL}$ formulae with only one path quantifier. For propositional variables $p_1, \ldots, p_k$ and boolean formula φ, we construct an implicit CGM where the values of $p_1, \ldots, p_k$ are "declared" by agents $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_k\}$ (in parallel). The subsequent transition leads to a state labeled by proposition yes iff φ holds for the underlying valuation of $p_1, \ldots, p_k$. Then,

	Ir	IR
CTL	$\mathbf{\Delta_2^P}$-complete	
Simple **CL**	$\mathbf{\Sigma_2^P}$-complete	$\mathbf{\Sigma_2^P}$-complete
CL	$\mathbf{\Delta_3^P}$-complete	$\mathbf{\Delta_3^P}$-complete
ATL	$\mathbf{\Delta_3^P}$-complete	$\mathbf{\Delta_3^P}$-complete
ATL*	**PSPACE**-complete	**2EXPTIME**-complete

Figure 5.7: Overview of the model checking complexity for implicit CGM's

SAT reduces to model checking formula $\langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \mathrm{X}$yes. The reduction of $\mathbf{SNSAT_2}$ (a canonical $\mathbf{\Delta_2^P}$-complete problem) to model checking of **CTL** formulae with nested path quantifiers is an extension of the above **SAT** reduction, analogous to the one presented in Section 7.1.1.

A summary of complexity results for the alternative representation of transitions is presented in Figure 5.7.

5.3.2 Higher-Order Representations of Models

Explicit models of realistic systems are prohibitively large, both in the size of their state spaces and the number of the transitions. Thus, for practical verification, systems must be represented in a more compact way, for instance by generating the state space as valuations of some discrete-valued attributes, and defining transitions through boolean pre– and postconditions. Such high-level representations of multi-agent systems include, e.g., concurrent programs, reactive modules, ISPL specifications, modular interpreted systems, etc. It is easy to see that unfolding a compact representation to an explicit model involves usually an exponential blowup in its size. Consider, for example, a system whose state space is defined by r boolean variables (binary attributes). Obviously, the number of global states in the system is $n = \mathbf{O}(2^r)$.

Below, we quote the main results for model checking temporal and strategic logics in compact representations of states and transitions.

Theorem 30 *Model checking* **CTL**, **LTL**, *and* **CTL*** *over concurrent programs is* **PSPACE**-*complete with respect to the number of local states and agents, and the length of the formula.*

Theorem 31 *Model checking* **ATL** *over simple reactive modules and modular interpreted systems is* **EXPTIME**-*complete with respect to the number of local states and agents, and the length of the formula.*

Thus, while **ATL** appears to have the same model checking complexity as **CTL** at the first glance, it turns out distinctly harder when compact representations of states and/or transitions are considered.

References and Further Reading. Again, we refer the reader to the survey chapter [33] for an overview of relevant results. The technical results presented in this section have been obtained in [107, 95, 168, 92, 96, 97, 108]. Implicit CGM's were first called this way in [108], but had been already present in the ISPL modeling language of the MCMAS model checker [143, 142].

Chapter 6

Imperfect Information

6.1 Knowledge and Ability

The community of artificial intelligence recognized the importance of the concept of *ability* for reasoning about machines and computational systems already in the 1960s. Since the beginning of the discourse, ability and knowledge were seen as intimately connected. We would like to begin this chapter by recalling the quote from McCarthy and Hayes, already presented in Section 4.3:

> We want a computer program that decides what to do by inferring in a formal language that a certain strategy will achieve a certain goal. This requires formalizing concepts of causality, *ability, and knowledge*. ([115], emphasis added)

Although McCarthy and Hayes did not present a formalisation of ability, they did speculate on what such a formalism might look like. They suggested three possible interpretations of what it means for a computer program π to be able to achieve a state of affairs φ (again quoting after [115]):

1. There is a sub-program σ and room for it in memory which would achieve φ if it were in memory, and control were transferred to π. No assertion is made that π knows σ or even knows that σ exists.
2. σ exists as above and that σ will achieve φ follows from information in memory according to a proof that π is capable of checking.
3. π's standard problem-solving procedure will find σ if achieving φ is ever accepted as a subgoal.

The three cases address three viable interpretations of ability for a software agent, that can be in fact generalized to all kinds of autonomous agents. In case (1), ability is viewed from the standpoint of an omniscient external observer who can see that there is some action or procedure such that, if agent π executes the action or followed the procedure, then the achievement of φ will result. Thus, it corresponds to *objective* ability of the agent to bring about φ. Case (2) implies knowledge of the fact on the part of the agent (at least in the theoretical sense of modal epistemic logic). We will later refer to this kind of ability as *subjective* ability. Interpretation (3) refers to *practical* ability: not only does the possibility for the agent to achieve φ exist, but the agent is capable of computing and executing an appropriate strategy σ. Thus, the last interpretation is algorithmic, and may be formalized through the computational problem of *strategy synthesis*.

After the seminal work of McCarthy and Hayes, probably the best-known and most influential study of ability in AI was Moore's analysis of the relationship between knowledge and ability in a dynamic variant of first order epistemic logic. The basic components of the framework are a set of possible worlds (essentially, system states) and set of actions. To keep things simple, we will assume that there is just one agent in the system. The modal operator $(\mathsf{Res}\ \alpha\ \varphi)$ expresses that *after action α is performed, φ will be true*, similarly to the dynamic logic expression $[\alpha]\varphi$. Quantification plays an important role in Moore's logic of ability. Consider the distinction between the following quantified epistemic formulae, where $\mathit{Murderer}(x)$ means that the individual denoted by x is a murderer:

$$K(\exists x : \mathit{Murderer}(x))$$
$$\exists x : (K\ \mathit{Murderer}(x))$$

The first formula is called a *de dicto* formula, while the second is a *de re* formula. The first property asserts that, in every world consistent with the agent's knowledge, $\exists x : \mathit{Murderer}(x)$ is true. Thus, it expresses that the agent knows that somebody is a murderer, but it does *not* imply that the agent knows the *identity* of the individual in question. In contrast, the *de re* formula asserts something stronger. It says that there is some individual x such that in all epistemic alternatives for the agent, x is a murderer. This time, the value of x is fixed across all epistemic alternatives for the agent, and hence the agent *knows the identity of the murderer*.

Remark 2 *The distinction between knowledge* de dicto *and knowledge* de re *can be traced back to earlier philosophical discourse on the nature of knowledge. In particular, the works of Ryle on the concept of mind included an important distinction between* knowing that *and* knowing how. *Roughly speaking, "knowing that" is the standard concept of knowledge as a relation between agents and true propositions, expressed e.g. in epistemic logic. The concept of "knowing how" seems related, but is clearly different: it is concerned with the knowledge of how to achieve things.*

Researchers in AI have largely adopted the view that know-how can be reduced to know-that. However, with a sufficiently rich logical formalism, the two notions turn out distinct again, as we will argue in Section 6.4.

With the concepts of *de dicto* and *de re* in place, we can turn to Moore's formalisation of ability. He was concerned with developing a theory of ability that would capture the following two aspects of the interaction between knowledge and action:

1. As a result of performing an action, an agent can gain knowledge, and in particular, agents can perform "test" actions, in order to find things out.

2. In order to perform some actions, an agent needs knowledge: these are *knowledge preconditions*. For example, in order to open a safe, it is necessary to know the combination.

The ultimate aim is to define a unary operator $(\mathsf{Can}\ \varphi)$, intended to mean that the agent has the ability to achieve φ. A naive attempt to define Can may be as follows:

$$(\mathsf{Can}\ \varphi) \quad \leftrightarrow \quad (\exists \alpha : K(\mathsf{Res}\ \alpha\ \varphi))$$

This definition says that the agent can bring about φ if there exists some action α for which the agent knows that φ will result from the performance of α. Notice that the variable α denoting the relevant action is quantified *de re*, and so this definition implies that the agent *knows the identity of the action*. Moore pointed out, however, that the definition suffers from one major drawback: the agent is required to know in advance the identity of the whole of the action required to achieve the desired outcome. In everyday usage, this seems much too strong a requirement. For example, I might truthfully assure the publisher that "I can finish writing this book before the end of the month" without knowing up front the exact text I am going to type. Moore proposed the following adaptation of the definition:

$$\begin{aligned}&(\mathsf{Can}\ \varphi) \leftrightarrow\\ &\qquad \exists\alpha.K(\mathsf{Res}\ \alpha\ \varphi) \vee\\ &\qquad \exists\alpha.K(\mathsf{Res}\ \alpha\ (\mathsf{Can}\ \varphi))\end{aligned}$$

Thus, an agent has the ability to achieve φ if either:

1. She knows the identity of an action for which she knows that after this action is performed, φ will hold; or

2. She knows the identity of an action for which she knows that after this action is performed, $(\mathsf{Can}\ \varphi)$ will hold.

The second case allows for the possibility of performing an action that will provide the agent with the capability to achieve φ. Note the distinctly fixpoint flavor of the above definition. We will come back to it in Section 7.3.

Moore's formalism was enormously influential in the AI community. Many researchers used ideas from his work to define and characterize ability. The idea of an agent requiring *de re* knowledge of an action has been of particular lasting significance. In this chapter, we focus on a somewhat more complex notion of ability, that looks at *strategies*, i.e., conditional plans rather than simple actions. Still, the issue to what extent an agent (or a group of agents) knows the right strategy, and the difference between knowing a suitable strategy *de re* vs. *de dicto* is a central one. We will discuss it extensively in Section 6.4.

References and Further Reading. For a deeper overiew of philosophical and AI approaches to knowledge and its influence on abilities, we refer the reader to the handbook [174], and especially the chapter [5].

The famous desideratum of McCarthy and Hayes can be found in [115]. For Moore's formalization of ability, see [119, 120]; the list of later works inspired by Moore's idea includes at least [121, 122, 123, 157, 183, 99, 93]. Ryle's discourse on "knowing that" vs. "knowing how" was proposed in [147], cf. also [159] for additional discussion.

6.2 Abilities under Imperfect Information

Agents usually have incomplete knowledge about the environment where they act, as well as about the current course of affairs, including the current mental states of the other agents, the actions they took in the past, etc. That significantly affects their abilities to achieve individual and collective objectives. In this section, we combine concurrent game models and epistemic models in order to give semantics to a language expressing strategic ability under imperfect and/or incomplete information. Except for a brief spell in Section 6.2.2, we do not involve epistemic operators in the object language yet (this will be done more seriously in Section 6.4). Thus, knowledge is reflected only in the models, through agents' epistemic relations, but not explicitly referred to in the formulae. As we will show, taking knowledge into account on purely semantic level is already sufficient to make a strong impact on the meaning of strategic operators and the patterns of ability that emerge.

In game theory, two different terms are traditionally used to indicate lack of information: *incomplete* and *imperfect* information. The former refers to uncertainties about the game structure, whereas the latter refers to uncertainties about the current state of the game while it is played. The models that we use allow for representing both types of uncertainty in a uniform way. Thus, we take the liberty to use the two terms interchangeably to indicate any possible relevant lack of information.

Example 50 (Rescue Robots: Properties to express)

♣ The robots can rescue person j;

♣ The robots can rescue person j, and they know that they can;

♣ The robots can rescue person j, and they know how to do it.

Example 51 (Voting: Properties to express)

Privacy: The system cannot reveal how a particular voter voted;

Receipt-freeness: The voter cannot gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way;

Coercion-resistance: The voter cannot cooperate with the coercer to prove to him that she voted in a certain way.

6.2.1 Bringing Strategies and Uncertainty Together

The decision making and abilities of strategically reasoning players are strongly influenced by the knowledge they have about the world, other players, past actions, and so on. In Chapters 4 and 5, we considered games of perfect information in the sense that players were completely aware of the structure of the system as well as the current state of the play, and the only information they lacked was the choices of the other players at the current state. In reality, this is seldom the case: usually players have only partial information, both about the setup in general and about the specific play. In the following, we are concerned with the question: What can players achieve in such scenarios?

We represent players' incomplete information by *indistinguishability relations* $\sim_a \subseteq St \times St$ on the state space, similarly to models of epistemic logic in Section 2.1.3. The relations are assumed to be equivalences. We note that in game theory clusters of indistinguishable states are called *information sets* of player a.

Formally, concurrent multi-player games with incomplete information can be modelled by *concurrent epistemic game structures* (CEGS), defined as a tuples

$$S = (\mathbb{A}\mathrm{gt}, St, \{\sim_a \mid a \in \mathbb{A}\mathrm{gt}\}, Act, d, out),$$

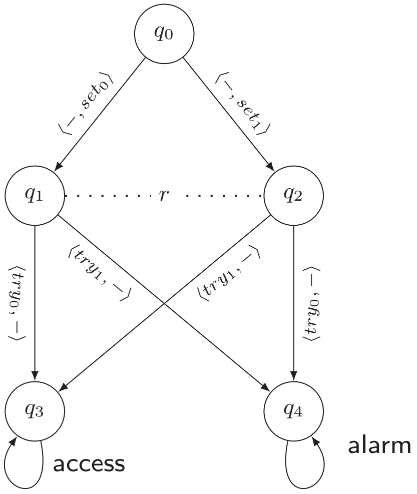

q_0
$\langle -, set_0 \rangle$
$\langle -, set_1 \rangle$
q_1
r
q_2
$\langle try_0, - \rangle$
$\langle try_1, - \rangle$
$\langle try_1, - \rangle$
$\langle try_0, - \rangle$
q_3
q_4
access
alarm

Figure 6.1: Robber in a bank (M_{robb}). The dotted line indicates states indistinguishable to the robber.

where $(\mathbb{A}\mathrm{gt}, St, Act, d, out)$ is a concurrent game structure, and $\sim_a$ are indistinguishability relations over St, one per agent in $\mathbb{A}\mathrm{gt}$. A standard assumption in the case of incomplete information is that players have the same available choices in indistinguishable states, for otherwise they would have a way to discern between these states. That is, we require that $q \sim_a q'$ implies $d_a(q) = d_a(q')$. Such structures are sometimes called *uniform CEGS*. A *concurrent epistemic game model (CEGM)* extends a CEGS with an interpretation of atomic propositions $\mathcal{V} : \mathcal{PV} \to 2^{St}$. Below we present two examples that will be used to test our intuitions with respect to the semantics of ability under imperfect information.

Example 52 (Schobbens' Robber) *Figure 6.1 models the following scenario. A vault containing a large sum of money is protected by a binary code. Realistically, the code should be ca. 20 digits long, but in order to simplify the graph, we assume that it consists of only 1 digit, either* 0 *or* 1. *If someone enters the right code, the vault opens and gives access to what is stored there. If an incorrect code is entered, the alarm system switches on. The code is set anew every morning by the guard agent g. Some time later during the day, the robber (r) enters the bank and tries to open the vault. However, he doesn't know the current code, which is indicated by indistinguishability of states q_1 and q_2.*

Can we say that the rober has the ability to open the vault and get access to the money? Intuitively not. He has all the necessary physical capabilities, but at the same time lacks an important "soft" resource: knowledge which action should be used to achieve the goal.

Example 53 (Poor Duck) *Consider model M_{duck} in Figure 6.2, with the following story. A man wants to shoot down a yellow rubber duck in a shooting gallery. The man knows that the duck is in one of the two cells in front of him, but he does not*

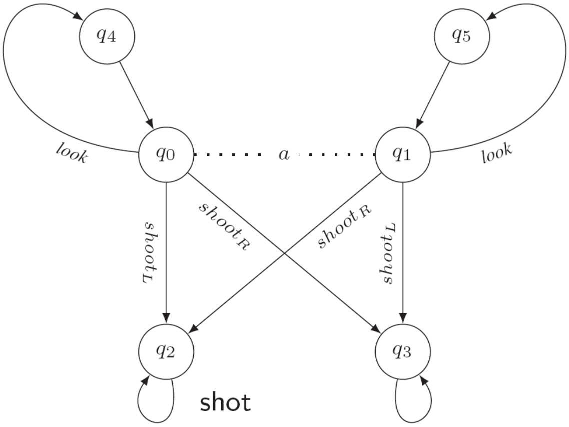


Figure 6.2: Poor Duck model M_{duck} with one player (a) and transitions labeled with a's actions. Automatic transitions (i.e., such that there is only one possible transition from the starting state) are left unlabeled.

know in which one. He can either decide to shoot to the left (action $shoot_L$), or to the right (action $shoot_R$), or reach out to the cells and look what is in (action $look$). Note that only one of the shooting actions can be successful – which one it depends on the starting state of the game.[1]

Intuitively, the man does not have a strategy to ensure shooting down the duck in one step, at least not from his subjective point of view. On the other hand, he should be able to ensure it in multiple steps if he has a perfect recall of his observations. Finally, getting the duck shot in one step can be guaranteed if the man is told the right strategy before the start of the game. For instance, if the starting state is q_0, then the right strategy is "shoot to the left no matter what you see." The man is perfectly capable of executing the strategy (and making sure that the duck gets shot), though he would not be able to come up with it on his own.

We will formalize the intuitions in the rest of the chapter.

6.2.2 Alternating-Time Temporal Epistemic Logic

The standard semantics of **ATL** does not take into account the epistemic limitations of agents. Thus, it assumes implicitly that every agent has complete information about the current global state of the system. We have already seen that actions, transitions, and imperfect information can be put together in logical structures by a straightforward fusion of concurrent game models and epistemic Kripke models. *Alternating-time temporal epistemic logic* (**ATEL**) extends this idea to the level of logical formulae. That is, **ATEL** is a straightforward combination of the multi-agent epistemic logic and **ATL**. Formally, the syntax of **ATEL** is defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A\rangle\!\rangle \mathrm{X}\varphi \mid \langle\!\langle A\rangle\!\rangle \mathrm{G}\varphi \mid \langle\!\langle A\rangle\!\rangle \varphi \,\mathrm{U}\, \varphi \mid \mathrm{K}_a\varphi.$$

[1] For more seriously minded readers, we propose an alternative story: agent a is a doctor who can apply two different treatments to a patient with symptoms that fit two different diseases. Additionally, the doctor can order a blood test to identify the disease precisely.

The semantics of $\mathbf{ATEL}$ is based on concurrent epistemic game models, and defined by the union of semantic clauses for $\mathbf{ATL}$ and epistemic logic, that we presented in detail in Chapters 2 and 4.

The logic enables specification of various modes and nuances of interaction between knowledge and strategic abilities, e.g.:

- $\langle\langle A\rangle\rangle\gamma \rightarrow \mathrm{E}_A\langle\langle A\rangle\rangle\gamma$: if group A can bring about γ, then everybody in A knows that they can;
- $\mathrm{E}_A\langle\langle A\rangle\rangle\gamma \wedge \neg\mathrm{C}_A\langle\langle A\rangle\rangle\gamma$: the agents in A have mutual knowledge but not common knowledge that they can enforce γ;
- $\langle\langle a\rangle\rangle\gamma \rightarrow \mathrm{K}_a\neg\langle\langle \mathbb{A}\mathrm{gt} \setminus \{a\}\rangle\rangle\neg\gamma$: if a can bring about γ, then she knows that the rest of agents cannot prevent it, etc.

While $\mathbf{ATEL}$ indeed extends both $\mathbf{ATL}$ and epistemic logic, it also raises a number of conceptual problems. Most importantly, one would expect that an agent's ability to bring about property γ should at least imply that the agent has enough control and knowledge to *execute* a strategy that enforces γ. Unfortunately, this is not the case.

Example 54 (Schobbens' Robber: Counterintuitive abilities) *Consider again model M_{robb} in Figure 6.1. According the semantics of* $\mathbf{ATEL}$*, we have $M_{robb}, q_0 \models \langle\langle r\rangle\rangle \mathsf{F}\mathsf{open}$, the right strategy being $s_r(q_0) = -$, $s_r(q_1) = try_0$, $s_r(q_2) = try_1$. Note that nothing in the semantics of* $\mathbf{ATL}$ *and* $\mathbf{ATEL}$ *exclude this strategy. Moreover, combining strategic and epistemic operators does not help, as $M_{robb}, q_0 \models \mathrm{K}_r\langle\langle r\rangle\rangle \mathsf{F}\mathsf{open}$ (in q_0, the robber knows precisely the state of the system). So, using* $\mathbf{ATEL}$ *makes us conclude that the robber can enforce opening the safe, and knows about it, while none of those should intuitively be the case.*

A number of approaches have been proposed to overcome this problem. Most of the solutions agree that only so called *uniform strategies* are really executable. That is, agents cannot plan different actions for states that they are not able to distinguish. Moreover, it is often intuitive to assume that ability with respect to γ entails also the agents' capacity to *identify* the right strategy for achieving γ. Note that, in order to identify a successful strategy, the agents must consider not only the courses of action starting from the current state of the system, but also from states that are indistinguishable from the current one. We will discuss this in more detail in Section 6.4.1.

6.2.3 Uniform Strategies

In case of standard $\mathbf{ATL}$, abilities were derived from strategies defined on states or their sequences, i.e., memoryless or perfect recall strategies. For incomplete information, the picture is similar. However, the two notions of a strategy must take into account some constraints due to uncertainty of the players. That is, an executable strategy has to assign the same choices to indistinguishable situations. Such strategies are called *uniform*.

Definition 13 (Uniform strategy) *A memoryless strategy s_a is* uniform *if the following condition is satisfied:*

$$\textit{for all states } q, q' \in St, \textit{ if } q \sim_a q', \textit{ then } s_a(q) = s_a(q').$$

Alternatively, a uniform memoryless strategy can be seen as a mapping from information sets of a player to his actions (in that case, no additional constraints are needed).

For perfect recall strategies, we first lift the indistinguishability between states to indistinguishability between sequences of states. Two histories $h = q_0 q_1 \dots q_n$ *and* $h' = q'_0 q'_1 \dots q'_{n'}$ *are* indistinguishable *for agent* a*, denoted by* $h \approx_a h'$*, if and only if* $n = n'$ *and* $q_i \sim_a q'_i$ *for* $i = 1, \dots, n$.[2] *Now, a perfect recall strategy* s_a *is* uniform *if the following condition holds:*

$$\textit{for all histories } h, h' \in St^+, \textit{ if } h \approx_a h', \textit{ then } s_a(h) = s_a(h').$$

Uniform collective strategies are defined as tuples of uniform individual strategies. Note that the constraints in collective strategies refer to individual choices and individual relations $\sim_a$ *(resp.* $\approx_a$*), and not to collective choices and any derived relations (e.g.,* $\sim_A^E$*,* $\approx_A^E$*,* $\sim_A^D$*, etc.). In particular, no communication is presumed between the agents that would reduce their uncertainties.*

Example 55 (Uniform strategies for robbers and duck hunters) *It is easy to see that the strategy* s_r *of Example 54 is not uniform, as it specifies different actions in states* q_1, q_2 *that look exactly the same to the robber. Since* $q_0 q_1 \approx_r q_0 q_2$*, the same applies if we consider the analogous perfect recall strategy, i.e.,* $s'_r(q_0) = -$, $s'_r(q_0 q_1) = try_0$, $s'_r(q_0 q_2) = try_1$. *In fact, there is no uniform strategy for* r *in* M_{robb} *– either memoryless or memory-based – which would guarantee in* q_0 *that the system will eventually reach* q_3.

The situation in model M_{duck} *is more subtle. Consider* q_0 *as the starting state. Contrary to what one might expect, there* is *a uniform memoryless strategy for agent* a *which guarantees that the system will reach* shot*, namely* $s_a(q_0) = s_a(q_1) = shoot_L$. *When executed in* q_0*, it brings the system to state* q_2 *in one step. However, the agent does not know that he* s_a *is surely winning, since to his knowledge the initial state of the game can be as well* q_1*, and in* q_1 *the strategy does not succeed. Finally, the perfect recall strategy* $s'_a(q_0) = s'_a(q_1) = look$, $s'_a(q_0 q_4) = s'_a(q_1 q_5) = -$, $s'_a(q_0 q_4 q_0) = shoot_L$, $s'_a(q_1 q_5 q_5) = shoot_R$ *is uniform and guarantees success from both* q_0 *and* q_1.

We will soon present two variants of alternating-time logic that take into account the above considerations.

6.2.4 Reasoning about Abilities under Uncertainty

Agents' incomplete information and use of memory can be incorporated into **ATL*** in different ways. One possible approach is not to change the logical language but to consider variations of the semantics by suitably varying the notion of strategy employed in the truth definition of the strategic operators. Combining the dimensions of memory and information gives rise to four natural semantic relations for **ATL***:

$\models_{IR}$**:** perfect **I**nformation and perfect **R**ecall strategies;

$\models_{Ir}$**:** perfect **I**nformation and imperfect **r**ecall strategies;

$\models_{iR}$**:** imperfect **i**nformation and perfect **R**ecall strategies;

$\models_{ir}$**:** imperfect **i**nformation and imperfect **r**ecall strategies.

[2] We note that this corresponds to the notion of synchronous perfect recall according to [63].

In this approach, the semantics of $\mathbf{ATL}^*$ is parameterized with the type of strategies – yielding four different semantic variants of the logic, that we label accordingly $\mathbf{ATL}^*_{\text{IR}}$, $\mathbf{ATL}^*_{\text{Ir}}$, $\mathbf{ATL}^*_{\text{iR}}$, and $\mathbf{ATL}_{\text{ir}}$. The following types of strategies are used in the respective semantic variants:

- Ir: $s_a : St \to Act$ such that $s_a(q) \in d(a, q)$ for all q;
- IR: $s_a : St^+ \to Act$ such that $s_a(q_0 \dots q_n) \in d(a, q_n)$ for all $q_0, \dots, q_n$;
- ir: like Ir, with the additional constraint that $q \sim_a q'$ implies $s_a(q) = s_a(q')$;
- iR: like IR, with the additional constraint that $h \approx_a h'$ implies $s_a(h) = s_a(h')$.

The four semantic variants are obtained by different updating of the main semantic clause from Section 4.3.2. Let M be a CEGM, and let $\sim_A^E = \bigcup_{a \in A} \sim_a$ be the epistemic relation corresponding to the mutual knowledge of coalition A (i.e., what everybody in A knows). Moreover, let $[q]_a = \{q' \mid q \sim_a q'\}$ denote the information set of agent a, and analogously for coalitions A. The truth definitions for $\models_{xy}$, where $x \in \{\text{I}, \text{i}\}, y \in \{\text{R}, \text{r}\}$, read as follows:

$M, q \models_{\text{I}y} \langle\langle A \rangle\rangle \gamma$ iff there is a collective Iy strategy s_A for A such that $M, \lambda \models \gamma$ for every play $\lambda \in out(q, s_A)$;

$M, q \models_{\text{i}y} \langle\langle A \rangle\rangle \gamma$ iff there is a collective iy strategy s_A for A such that $M, \lambda \models \gamma$ for every play $\lambda \in \bigcup_{q' \in [q]_A} out(q', s_A)$.

It is easy to see that the semantic relation $\models$ for standard $\mathbf{ATL}^*$ corresponds to $\models_{\text{IR}}$.

Example 56 (Shooting the duck in variants of $\mathbf{ATL}^*$) *Consider the Poor Duck model M_{duck} of Example 53. There is no good strategy for the man to shoot down the duck in one step regardless of the kind of memory that the man possesses – formally, $M_{duck}, q_0 \models_{\text{iR}} \neg\langle\langle a \rangle\rangle \text{X}\mathsf{shot}$ and $M_{duck}, q_0 \models_{\text{ir}} \neg\langle\langle a \rangle\rangle \text{X}\mathsf{shot}$. However, he should be able to achieve it in multiple steps if he can remember and use his observations, i.e., $M_{duck}, q_0 \models_{\text{iR}} \langle\langle a \rangle\rangle \text{F}\mathsf{shot}$.*

On the other hand, suppose that it has been a long party, and the man is very tired, so he is only capable of using memoryless strategies at the moment. Does he have a memoryless strategy which he knows will achieve the goal? No. For each of the three available strategies (shoot left whatever happens, shoot right whatever happens, look whatever happens), the man has to take into account the possibility of failure. In consequence, $M_{duck}, q_0 \models_{\text{ir}} \neg\langle\langle a \rangle\rangle \text{F}\mathsf{shot}$. However, interestingly enough, the man can *identify an opening strategy that will guarantee his knowing how to shoot the duck in the next moment: $M_{duck}, q_0 \models_{\text{ir}} \langle\langle a \rangle\rangle \text{X} \langle\langle a \rangle\rangle \text{F}\mathsf{shot}$. The opening strategy is to look; if the system proceeds to q_4, then the second strategy is to shoot to the left, otherwise the second strategy is to shoot to the right.*

Example 57 *Let us consider the concurrent epistemic game structure M_{carr5} in Figure 6.3 that combines the strategic structure from model M_{carr4} (Figure 4.2) with the epistemic relations from M_{carr1} (Figure 7). Now, no agent knows how to make the carriage reach or avoid any selected state singlehandedly from q_0, i.e., $M_{carr5}, q_0 \models_{iy} \neg\langle\langle i \rangle\rangle \text{F}\mathsf{pos_j}$ and $M_{carr5}, q_0 \models_{iy} \neg\langle\langle i \rangle\rangle \text{G}\neg\mathsf{pos_j}$ for all $y \in \{\text{r}, \text{R}\}, i \in \{1, 2\}, j \in \{1, 2, 3\}$. Note in particular that the strategy of Example 45 cannot be used here because it is not uniform. The robots cannot even identify the right strategy together, e.g.,*

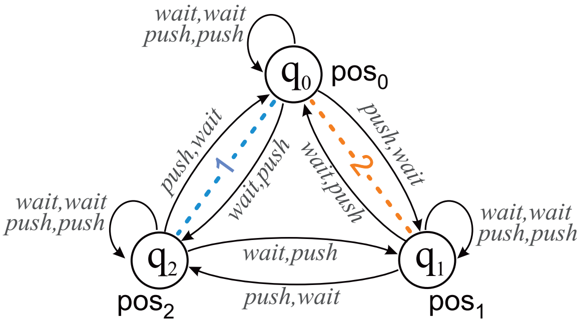


Figure 6.3: Two robots and a carriage: concurrent epistemic game model M_{carr5}. Dashed lines represent indistinguishability relations between states.

$M_{carr5}, q_0 \models_{iy} \neg\langle\langle 1,2\rangle\rangle \mathrm{G}\neg \mathsf{pos}_1$ *(when in q_0, robot 2 considers it possible that the system is already in the "bad" state q_1). So, do the robots know how to play to achieve anything? Yes, for example they know how to make the carriage reach a particular state eventually:* $M_{carr5}, q_0 \models_{iy} \langle\langle 1,2\rangle\rangle \mathrm{F}\mathsf{pos}_1$ *etc. – it suffices that one of the robots pushes all the time and the other waits all the time.*

For the above properties the type of robots' recall does not matter (they hold in both memoryless and perfect recall strategies). $\langle\langle 1,2\rangle\rangle \mathrm{FG}\mathsf{pos}_1$ *is an example* $\mathbf{ATL}^*$ *formula that distinguishes between the two sets of strategies.*

Objective Ability under Imperfect Information

Ability under incomplete information can be additionally classified into *subjective* and *objective*. The latter refers to existence of a strategy that is guaranteed to succeed from the perspective of an external observer with complete information, while the former requires the strategy to guarantee success from the perspective of the player/coalition executing it. The semantic definitions above are based on the subjective view. As it requires that the executing players must be able to identify the right strategy on their own (within the limits of their incomplete information), it imposes stronger requirements on the strategy than the objective ability. Technically, this is because in the evaluation of a state formula $\langle\langle A\rangle\rangle\gamma$ in state q, when judging the suitability of the selected uniform strategy for A in terms of the coalition's subjective ability all epistemic alternatives of q with respect to $\sim_A$ are to be taken into account, whereas objectively it suffices to check that the strategy only succeeds from q. Whenever confusion can arise, we will refer to subjective and objective abilities by $\mathrm{i_s}$ and $\mathrm{i_o}$, respectively.

The semantics of objective ability under imperfect information can be formally defined as follows:

$$M, q \models_{\mathrm{i_o}y} \langle\langle A\rangle\rangle\gamma \quad \text{iff} \quad \text{there is a collective } \mathrm{i}y\text{-strategy } s_A \text{ for } A \text{ such that } M, \lambda \models \gamma \text{ for every play } \lambda \in out(q, s_A).$$

Example 58 (Poor Duck: Subjective vs. objective ability) *We have seen that* $M_{duck}, q_0 \not\models_{\mathrm{ir}} \neg\langle\langle a\rangle\rangle \mathrm{X}\mathsf{shot}$ *because the agent cannot identify the right strategy to ensure* Xshot. *On the other hand, the strategy* $s_a(q_0) = s_a(q_1) = shoot_L$ *does objectively work from q_0. In consequence, we have that* $M_{duck}, q_0 \models_{\mathrm{i_or}} \langle\langle a\rangle\rangle \mathrm{X}\mathsf{shot}$, *and* *hence also* $M_{duck}, q_0 \models_{\mathrm{i_oR}} \langle\!\langle a \rangle\!\rangle \mathsf{X}\mathsf{shot}$.

Strategies Based on Aggregate Coalitional Uncertainty

Most modal logics of strategic ability agree that *executable* strategies are exactly the ones that obey the uniformity constraints. For a joint strategy, this means that each member of the coalition must be able to carry out his part of the joint plan individually. An alternative approach is to redefine the notion of uniform coalitional strategy, based on a suitable *group indistinguishability relation* for the coalition. The most interesting case is when the "distributed knowledge" relation is used. Conceptually, this amounts to assuming that members of the coalition have unlimited communicating capabilities, and freely share relevant information *during the execution of the strategy*.

It can be shown that $\mathbf{ATL}^*$ with the alternative semantics can be embedded in the syntactic restriction that talks only about abilities of individual agents. Formally and more precisely: there exists a truth-preserving translation of models and formulae to the fragment of $\mathbf{ATL}^*$ that allows only for singleton coalitions. Thus, in a way, coalitions whose members can fully coordinate their actions and share their knowledge can be seen as single agents in disguise.

References and Further Reading. For an overview of the issues that arise when strategic ability is combined with incomplete information, we refer the reader to the handbook chapters [5, 36].

Semantic variants for $\mathbf{ATL}$ with imperfect information have been studied in numerous papers. We recommend especially [153, 99] as the starting point. The former clarifies the basic conceptual structure of what it means to have strategic ability with respect to some temporal property. The latter gives a more involved treatment of the interplay between the epistemic and the strategic dimensions. Concurrent epistemic game models, as well as $\mathbf{ATEL}$, were proposed in [171, 172]. Other works on the logical semantics of imperfect information strategies include e.g. [12, 89, 101, 176, 41]. The distinction between objective and subjective abilities was investigated in [39]. Alternative semantics with coalitional strategies based on aggregate uncertainty were proposed and studied in [73, 53, 74, 52], cf. also [103] for additional results.

6.3 Comparing Semantics of Strategic Ability

Semantic variants of $\mathbf{ATL}$ are derived from different assumptions about agents' capabilities. Can the agents "see" the current state of the system, or only a part of it? Can they memorize the whole history of observations in the game? Different answers to these questions induce different semantics of strategic ability, and they provide different ways of analysing interaction models. However, it is not entirely clear to what extent they give rise to different *logics*. One natural question that arises is whether the semantic variants generate different sets of valid (and, dually, satisfiable) sentences. In this section, we show a comparison of the validity sets for $\mathbf{ATL}^*$ with respect to the four semantic variants presented in the previous section.

The comparison of the validity sets is important for at least two reasons. Firstly, many logicians identify a logic with the set of sentences that are valid in the logic. Thus, by comparing validity sets we compare the respective logics in the traditional sense. Perhaps more importantly, the validities of $\mathbf{ATL}^*$ capture general properties of games under consideration. If two variants of $\mathbf{ATL}^*$ generate the same valid sentences, then the underlying notions of ability induce the same kind of games. Conversely, if they generate different sets of validities, then they capture two different classes of games, despite using the same class of models. All the variants studied here are defined over concurrent epistemic game models. Hence, the difference between games induced by different semantics lies in available strategies and the type of winning conditions, encoded in semantic clauses.

We recall that we use the "star"/"no star" notation to identify the syntactic variant of $\mathbf{ATL}$, and subscripts to denote the semantic variant being used. For example, $\mathbf{ATL}^*_{\mathrm{ir}}$ denotes the full language of $\mathbf{ATL}^*$ interpreted with the semantic relation $\models_{ir}$, that is, the one which assumes incomplete information and memoryless strategies. Moreover, we will use $\mathit{Valid}(L)$ to denote the set of validities of logic L, and $\mathit{Sat}(L)$ to denote the set of satisfiable formulae in L.

6.3.1 Perfect vs. Imperfect Information

We begin by comparing properties of games with limited information to those where players can always recognize the current state of the world. Firstly, we recall that complete information can be seen as a special case of incomplete information: each CGM can be seen as a CEGM in which each indistinguishability relation is taken as the smallest reflexive relation. Hence, every valid formula of $\mathbf{ATL}^*_{\mathrm{ir}}$ is also a validity of $\mathbf{ATL}^*_{\mathrm{Ir}}$: if there were a CEGM M with $M \not\models_{Ir} \varphi$, then also $M \not\models_{ir} \varphi$ would be the case. On the other hand, the formula

$$\langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi \leftrightarrow \varphi \vee \langle\!\langle A\rangle\!\rangle \mathrm{X}\langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi$$

is a validity of $\mathbf{ATL}_{\mathrm{Ir}}$ but not of $\mathbf{ATL}_{\mathrm{ir}}$, which shows that the containment is strict even in the limited syntactic fragment of $\mathbf{ATL}$.

Remark 3 *The equivalence between $\langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi$ and $\varphi \vee \langle\!\langle A\rangle\!\rangle \mathrm{X}\langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi$ is very important since it provides a fixpoint characterization of $\langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi$. The fact that $\langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi \leftrightarrow \varphi \vee \langle\!\langle A\rangle\!\rangle \mathrm{X}\langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi$ is not valid under incomplete information is one of the main reasons why constructing verification and satisfiability checking algorithms is so difficult for incomplete information strategies.*

The argument for $\mathbf{ATL}_{\mathrm{iR}}$ vs. $\mathbf{ATL}_{\mathrm{IR}}$ is analogous. Thus, we get that $\mathit{Valid}(\mathbf{ATL}_{\mathrm{ir}}) \subsetneq \mathit{Valid}(\mathbf{ATL}_{\mathrm{Ir}})$ and $\mathit{Valid}(\mathbf{ATL}_{\mathrm{iR}}) \subsetneq \mathit{Valid}(\mathbf{ATL}_{\mathrm{IR}})$, and the same for the full language of $\mathbf{ATL}^*$. Since the IR and Ir semantics coincide for $\mathbf{ATL}$, the second result implies also that $\mathit{Valid}(\mathbf{ATL}_{\mathrm{iR}}) \subsetneq \mathit{Valid}(\mathbf{ATL}_{\mathrm{Ir}})$. For the broader language, the sets $\mathit{Valid}(\mathbf{ATL}^*_{\mathrm{iR}})$ and $\mathit{Valid}(\mathbf{ATL}_{\mathrm{Ir}})$ turn out incomparable. The argument is rather technical, and we omit it here.

6.3.2 Memory-Based vs. Memoryless Strategies

The comparison of memory-based and memoryless strategies is technically more involved. Firstly, we observe that for any *tree-like* CGM M the sets of memory-based and memoryless strategies coincide. Secondly, one can show that every CGM M and state q in M can be unfolded into an equivalent (more precisely, bisimilar) tree-like CGM $T(M,q)$. Thus, $M,q \not\models_{IR} \varphi$ implies $T(M,q),q \not\models_{IR} \varphi$ (by the latter observation) and in consequence also $T(M,q),q \not\models_{Ir} \varphi$ (by the first observation). Hence, we get that $\mathbf{ATL}^*_{\mathrm{Ir}} \subseteq \mathbf{ATL}^*_{\mathrm{IR}}$. Moreover, the formula

$$\varphi \equiv \langle\!\langle A\rangle\!\rangle(\mathrm{F}\varphi_1 \wedge \mathrm{F}\varphi_2) \leftrightarrow \langle\!\langle A\rangle\!\rangle \mathrm{F}((\varphi_1 \wedge \langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi_2) \vee (\varphi_2 \wedge \langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi_1))$$

is a validity of $\mathbf{ATL}^*_{\mathrm{IR}}$ but not of $\mathbf{ATL}^*_{\mathrm{Ir}}$, which shows that the inclusion is strict. The formula expresses decomposability of conjunctive goals: being able to achieve $\varphi_1 \wedge \varphi_2$ must be equivalent to having a strategy that achieves first φ_1 and φ_2, or vice versa. It is easy to see that the requirement holds for agents with perfect memory, but not for ones bound to use memoryless strategies (and hence to play the same action whenever the game comes back to a previously visited state).

Note, however, that φ is *not* a formula of $\mathbf{ATL}$. Indeed, it is well known that the semantics given by $\models_{IR}$ and $\models_{Ir}$ coincide in $\mathbf{ATL}$. As a consequence, $\mathit{Valid}(\mathbf{ATL}^*_{\mathrm{Ir}}) \subsetneq \mathit{Valid}(\mathbf{ATL}^*_{\mathrm{IR}})$ and $\mathit{Valid}(\mathbf{ATL}_{\mathrm{Ir}}) = \mathit{Valid}(\mathbf{ATL}_{\mathrm{IR}})$. On the other hand, strict subsumption holds already for the language of $\mathbf{ATL}^+$ which allows cooperation modalities to be followed by a Boolean combination of simple path formulae.

Finally, we consider the effect of memory in the incomplete information setting. The idea is the same as for perfect information, but the unfolding of a CEGM into an equivalent tree-like CEGM is technically more complex, as one has to take into account the indistinguishability relations. To show that the inclusion is strict, we use

$$\langle\!\langle A\rangle\!\rangle \mathrm{X} \langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi \rightarrow \langle\!\langle A\rangle\!\rangle \mathrm{F}\varphi$$

which is valid in $\mathbf{ATL}_{\mathrm{iR}}$ but not in $\mathbf{ATL}_{\mathrm{ir}}$. The formula states that, if A has an opening move and a follow-up strategy to achieve eventually φ, then both strategies can be combined into a single strategy enforcing eventually φ already from the initial state. Thus, we get that $\mathit{Valid}(\mathbf{ATL}_{\mathrm{ir}}) \subsetneq \mathit{Valid}(\mathbf{ATL}_{\mathrm{iR}})$, and analogously for the broader language of $\mathbf{ATL}^*$.

6.3.3 Summary

We have obtained above the following hierarchy of logics:

$$\begin{gathered}
\mathit{Valid}(\mathbf{ATL}^*_{\mathrm{ir}}) \subsetneq \mathit{Valid}(\mathbf{ATL}^*_{\mathrm{iR}}) \subsetneq \mathit{Valid}(\mathbf{ATL}^*_{\mathrm{IR}}),\\
\mathit{Valid}(\mathbf{ATL}^*_{\mathrm{ir}}) \subsetneq \mathit{Valid}(\mathbf{ATL}^*_{\mathrm{Ir}}) \subsetneq \mathit{Valid}(\mathbf{ATL}^*_{\mathrm{IR}}),\\
\text{and } \mathit{Valid}(\mathbf{ATL}_{\mathrm{ir}}) \subsetneq \mathit{Valid}(\mathbf{ATL}_{\mathrm{iR}}) \subsetneq \mathit{Valid}(\mathbf{ATL}_{\mathrm{Ir}}) = \mathit{Valid}(\mathbf{ATL}_{\mathrm{IR}}).
\end{gathered}$$

Equivalently, we can observe the following pattern in the sets of *satisfiable* sentences:

$$\begin{gathered}
\mathit{Sat}(\mathbf{ATL}^*_{\mathrm{IR}}) \subsetneq \mathit{Sat}(\mathbf{ATL}^*_{\mathrm{Ir}}) \subsetneq \mathit{Sat}(\mathbf{ATL}^*_{\mathrm{ir}}),\\
\mathit{Sat}(\mathbf{ATL}^*_{\mathrm{IR}}) \subsetneq \mathit{Sat}(\mathbf{ATL}^*_{\mathrm{iR}}) \subsetneq \mathit{Sat}(\mathbf{ATL}^*_{\mathrm{ir}}),\\
\text{and } \mathit{Sat}(\mathbf{ATL}_{\mathrm{IR}}) = \mathit{Sat}(\mathbf{ATL}_{\mathrm{Ir}}) \subsetneq \mathit{Sat}(\mathbf{ATL}_{\mathrm{iR}}) \subsetneq \mathit{Sat}(\mathbf{ATL}_{\mathrm{ir}}).
\end{gathered}$$

The first, and most important, conclusion is that all the semantic variants of ability, considered so far, are different with respect to the properties of games they induce. Moreover, the results capture formally the usual intuition: complete information is a particular case of incomplete information, memory-based games are special cases of memoryless games, and information is a more distinguishing factor than memory. Figure 6.4 presents a graphical summary of the relationships. In fact, the figure displays *six* rather than four semantic variants, including the distinction between subjective and objective ability under imperfect information. The subscript $\mathrm{i_s}$ refers to subjective ability, that is, the standard "i" semantics. Subscript $\mathrm{i_o}$ refers to the "objective" ability, where one only looks at paths starting from the actual initial state of the game, cf. Section 6.2.4. As it turns out, general properties of agents' objective and subjective abilities are incomparable.

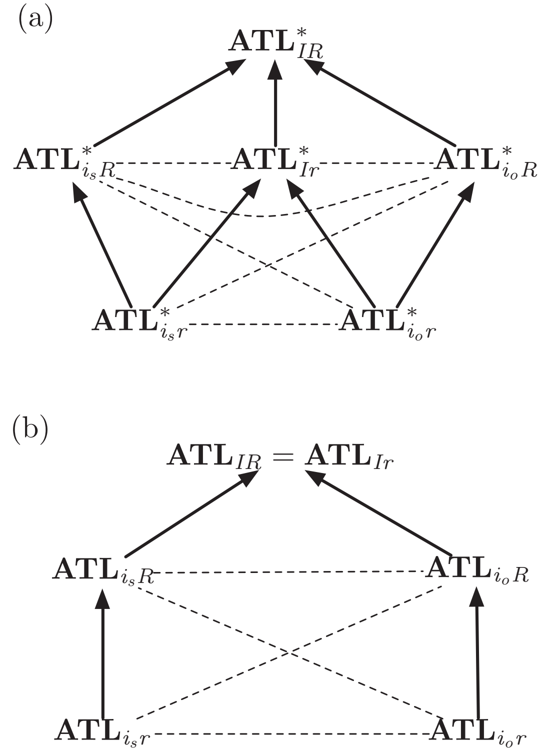


Figure 6.4: Comparison of the sets of validities induced by various semantics of (a) $\mathbf{ATL}^*$, and (b) $\mathbf{ATL}$. The arrows depict strict subsumption of the sets of validities, e.g., “$\mathbf{ATL}^*{}_{\mathrm{Ir}} \to \mathbf{ATL}^*{}_{\mathrm{IR}}$” means that $\mathit{Validities}(\mathbf{ATL}^*{}_{\mathrm{Ir}}) \subsetneq \mathit{Validities}(\mathbf{ATL}^*{}_{\mathrm{IR}})$. The dotted lines connect semantic variants with incomparable sets of validities. We do not include links that follow from transitivity of the subsumption relation.

This chapter is concerned with the influence of incomplete information on the abilities of agents. So, what is the impact? The above results show that assuming imperfect information changes the set of properties that are universally true, i.e., ones that hold in every model and every state. Moreover, the change is essential in the sense that some fundamental validities of standard $\mathbf{ATL}$ (with perfect information) do not hold anymore under imperfect information. We give three examples of such formulae here:

$$\langle\!\langle A\rangle\!\rangle \mathrm{G}\varphi \quad\leftrightarrow\quad \varphi \wedge \langle\!\langle A\rangle\!\rangle \mathrm{X} \langle\!\langle A\rangle\!\rangle \mathrm{G}\varphi \tag{6.1}$$

$$\langle\!\langle A\rangle\!\rangle \varphi_1 \,\mathrm{U}\, \varphi_2 \quad\leftrightarrow\quad \varphi_2 \vee \varphi_1 \wedge \langle\!\langle A\rangle\!\rangle \mathrm{X} \langle\!\langle A\rangle\!\rangle \varphi_1 \,\mathrm{U}\, \varphi_2 \tag{6.2}$$

$$\langle\!\langle \mathbb{A}\mathrm{gt}\rangle\!\rangle \mathrm{F}\varphi \quad\leftrightarrow\quad \neg\langle\!\langle \emptyset\rangle\!\rangle \mathrm{G}\neg\varphi \tag{6.3}$$

Formulae (6.1) and (6.2) provide fixpoint characterizations of strategic-temporal modalities. Formula (6.3) addresses the duality between necessary and obtainable outcomes in a game. All three sentences are validities of $\mathbf{ATL}_{\mathrm{IR}}$ and $\mathbf{ATL}_{\mathrm{Ir}}$, but they are *not* valid in $\mathbf{ATL}_{\mathrm{iR}}$ and $\mathbf{ATL}_{\mathrm{ir}}$.[3] The invalidity of fixpoint equivalences is especially important for practical purposes, since they normally provide the basis for model checking and satisfiability checking algorithms.

On a more general level, the results demonstrate that what agents can achieve is more sensitive to the strategic model of an agent (and a precise notion of achievement) than it was generally realized. Last but not least, they show that the language of $\mathbf{ATL}^*$ is sufficiently expressive to distinguish the main notions of ability.

References and Further Reading. The detailed analysis and technical results can be found in [39]. Analogous results that compare more sophisticated semantic variants of $\mathbf{ATL}^*$ to a standard variant include [90] for undominated play, [4] for irrevocable strategies, [6] for agents with bounded memory, [38] for recomputable strategies, [42] for truly perfect recall, and [103] for coalitional uniformity based on aggregate uncertainty of coalitions.

6.4 Constructive Knowledge and Levels of Ability

Knowledge and abilities are not independent: the more one knows, the more one can achieve by choosing a better suited strategy. We have seen that limited information influences the range of available strategies that agents and coalitions can choose. Respectively, it also affects the semantics of claims about an agent (or a coalition) being able to enforce a given outcome of the game.

So far, we have taken agents' knowledge into account only semantically, by introducing epistemic indistinguishability relations in the models, and defining their influence on the set of possible behaviors. In this section, we show what can be specified by adding a suitable variant of epistemic language. The aim is to better capture the nuances of the interplay between knowledge and ability.

6.4.1 Epistemic Levels of Strategic Ability

There are several possible interpretations of A's ability to bring about property γ, formalized by formula $\langle\!\langle A\rangle\!\rangle\gamma$, under imperfect information:

1. There exists a specification σ_A (not necessarily executable!) of A's behavior such that, for every execution of σ_A, γ holds.

2. There is a uniform strategy s_A such that, for every execution of s_A, γ holds (i.e., *A have the objective ability to enforce γ*).

3. Agents A know (in one sense or another, see below) that there is a uniform s_A such that, for every execution of s_A, γ holds (i.e., *A have a strategy "de dicto" to enforce γ*).

4. There is a uniform strategy s_A such that A know that, for every execution of s_A, γ holds (i.e., *A have a strategy "de re" to enforce γ*).

The above interpretations form a sequence of increasingly stronger levels of ability – each next one implies the previous ones. Case (1) corresponds to formula $\langle\!\langle A\rangle\!\rangle\gamma$ interpreted in the original semantics of $\mathbf{ATL}^*$. Cases (2)–(4), however, are not expressible

[3] Interestingly, (6.3) becomes valid again with the "objective" interpretation of ability under imperfect information.

in the perfect information semantics of $\mathbf{ATL}^*$, nor in straightforward combinations of $\mathbf{ATL}^*$ and epistemic logic such as $\mathbf{ATEL}$. The impossibility holds even for the nexttime fragment of $\mathbf{ATL}$ (i.e., coalition logic) and even when only a single agent is considered. We will show soon how those cases can be formally characterized with a suitable combination of strategic and epistemic modalities.

We note that cases (2), (3), and (4) come close to various philosophical notions of ability that we discussed in Section 6.1. Cases (4) and (3) clearly resemble Ryle's distinction between "knowing how" and "knowing that". In (4), *A know how* to enforce γ. In (3), they *know that* they can somehow enforce it. Moreover, the two cases link the notion of ability to an appropriate type of Moore's knowledge: *de re* (of the right strategy to play) or *de dicto* (that such a strategy exists).

The distinction between objective and subjective ability is closely related to the first two interpretations of program ability by McCarthy and Hayes. Case (2) corresponds to their first level of ability – i.e., the objective existence of a strategy (a "subprogram" σ in McCarthy and Hayes' terminology) that, if executed, will guarantee γ. Case (4) is analogous to the second level – the decision maker (the "main program" π for McCarthy and Hayes) has enough information to verify that σ enforces γ. We note that their third, strongest level of program ability cannot be expressed in the logics presented in this chapter, as they embed no notion of a problem-solving procedure *inside* their semantics.

Out of the four levels of ability, case (4) is arguably most interesting, as it formalizes the notion of agents in A *knowing how to play*. However, the statement "A know that every execution of s_A satisfies γ" is precise only if A is a singleton $\{a\}$. Then, we take into account the paths starting from the states indistinguishable from the current one according to a, i.e., $\bigcup_{q' \in \mathrm{img}(q,\sim_a)} out(q', s_a)$. In case of proper teams, there are several different "modes" in which the members can know the right strategy. That is, given strategy s_A, coalition A can have:

- *Common knowledge* that s_A enforces γ. This requires the least amount of additional communication when coordinating a joint strategy (it suffices that agents from A agree upon a total order over their collective strategies at the beginning of the game and that they will always choose the maximal successful strategy with respect to this order).
- *Mutual knowledge* that s_A enforces γ: everybody in A knows that s_A brings about γ.
- *Distributed knowledge* that s_A enforces γ: if the agents share their knowledge at the current state, they can identify the strategy as successful.
- *"Leader"*: the strategy can be identified by an agent $a \in A$;
- *"Headquarters committee"*: s_A can be identified by a subgroup $A' \subseteq A$.
- *"Consulting company"*: s_A can be identified by another group B.
- ...Other variations are possible, too.

We note that the semantics of coalitional ability in $\mathbf{ATL}^*_{\mathrm{ir}}$ as well as $\mathbf{ATL}^*_{\mathrm{iR}}$, presented in Section 6.2.4, implements the *mutual knowledge* mode of *knowing how to play*.

6.4.2 Constructive Strategic Logic

The issue of expressing various knowledge-related levels of ability through a suitable combination of strategic and epistemic logics has attracted significant attention. Most extensions (or refinements) of $\mathbf{ATL}$, proposed as solutions, cover only some of the possibilities, albeit in an elegant way. Others offer a more general treatment of the problem at the expense of an unnecessarily complex logical language. One of the main problems is how to capture the interplay between epistemic, strategic, and temporal aspects of play with the kind of quantifiers offered by $\mathbf{ATL}$ on one hand, and epistemic logic on the other. For example, "agents A have distributed knowledge about how to play to enforce γ" can be rephrased as "there is a strategy for A such that, for every state that is indistinguishable from the current one for all agents in A, and for every path from that state, possibly resulting from execution of the strategy, γ must hold on the path." This, however, cannot be directly expressed in $\mathbf{ATL}$ which combines quantification over strategies and paths within a single operator $\langle\!\langle A\rangle\!\rangle$.

One way out is to use separate modal operators that quantify over strategies and paths, the way it is done e.g. in Strategic $\mathtt{stit}$ and Strategy Logic. Indeed, "knowing how to play" in the simpler case of one-step games has been expressed using a combination of Chellas $\mathtt{stit}$ and epistemic logic. One can speculate that a straightforward combination of epistemic operators with Strategic $\mathtt{stit}$ or Strategy Logic should work equally well for long-term abilities. Such combinations have been already hinted in the literature, but not properly investigated yet.

Another solution is to define the success of a strategy *from a set of states*, instead of a single global state. This idea was used in *Constructive Strategic Logic* ($\mathbf{CSL}$) which extends $\mathbf{ATL}$ with so called *constructive knowledge* operators. In $\mathbf{CSL}$, each formula is interpreted in a set of states, rather than a single state. We write $M, Q \models \langle\!\langle A\rangle\!\rangle\varphi$ to express the fact that A must have a strategy which is successful for all the states from $Q \subseteq St$. The new epistemic operators $\mathbb{K}_i, \mathbb{E}_A, \mathbb{C}_A, \mathbb{D}_A$ for "practical" or "constructive" knowledge yield the set of states for which a single evidence (i.e., a successful strategy) should be presented (instead of checking if the required property holds in each of the states separately, like standard epistemic operators do).

Formally, let $[q]_{\mathcal{R}} = \{q' \mid q\mathcal{R}q'\}$ denote the image of state q with respect to relation $\mathcal{R}$. We also extend the notation to images of sets of states: $[Q]_{\mathcal{R}} = \bigcup_{q\in Q}[q]_{\mathcal{R}}$. The semantics of $\mathbf{CSL}$ over concurrent epistemic game models is defined by the following clauses:

$M, Q \models p$ iff $p \in \mathcal{V}(q)$ for every $q \in Q$;

$M, Q \models \neg\varphi$ iff $M, Q \not\models \varphi$;

$M, Q \models \varphi \wedge \psi$ iff $M, Q \models \varphi$ and $M, Q \models \psi$;

$M, Q \models \langle\!\langle A\rangle\!\rangle\varphi$ iff there is a uniform strategy s_A such that $M, \lambda \models \varphi$ for every $\lambda \in \cup_{q\in Q}\, out(q, s_A)$;

$M, Q \models \mathbb{K}_i\varphi$ iff $M, [Q]_{\sim_i} \models \varphi$;

$M, Q \models \mathbb{C}_A\varphi$ iff $M, [Q]_{\sim_A^C} \models \varphi$;

$M, Q \models \mathbb{E}_A\varphi$ iff $M, [Q]_{\sim_A^E} \models \varphi$;

$M, Q \models \mathbb{D}_A\varphi$ iff $M, [Q]_{\sim_A^D} \models \varphi$.

The semantic clauses for temporal operators are exactly as in $\mathbf{ATL}^*$. Additionally, we

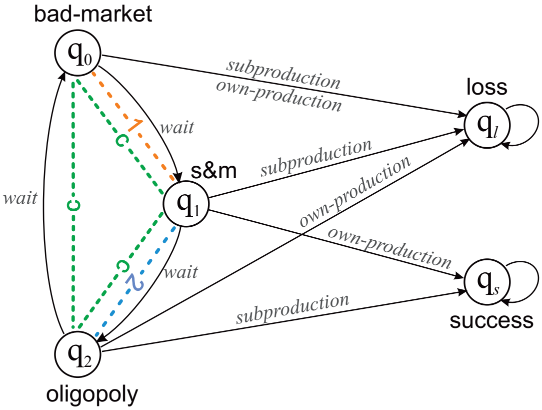


Figure 6.5: Simple market: model M_{markt}

define that $M, q \models \varphi$ iff $M, \{q\} \models \varphi$.

Example 59 (Market scenario) *Consider an industrial company that wants to start production, and looks for a good strategy when and how to do it. The market model is depicted in Figure 6.5. The economy is assumed to run in simple cycles: after the moment of bad economy (*bad-market*), there is always a good time for small and medium enterprises (*s&m*), after which the market tightens and an oligopoly emerges. At the end, the market gets stale, and we have stagnation and bad economy again.*

The company c is the only agent whose actions are represented in the model. The company can wait (action wait) or decide to start production: either on its own (own-production), or as a subcontractor of a major company (subproduction). Both decisions can lead to either loss or success, depending on the current market conditions. However, the company management cannot recognize the market conditions: bad market, time for small and medium enterprises, and oligopoly market look the same to them, as the epistemic links for c indicate.

The company can call the services of two marketing experts. Expert 1 specializes in oligopolies, and can recognize oligopoly conditions (although she cannot distinguish between bad economy and s&m market). Expert 2 can recognize bad economy, but he cannot distinguish between other types of market. The experts' actions have no influence on the actual transitions in the model, and are omitted from the graph in Figure 6.5. It is easy to see that the company cannot identify a successful strategy on its own: for instance, for the small and medium enterprises period, we have that $M_{markt}, q_1 \models \neg\mathbb{K}_c\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$. *It is also not enough to call the help of a single expert:* $M_{markt}, q_1 \models \neg\mathbb{K}_1\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success} \wedge \neg\mathbb{K}_2\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$, *or to ask the experts to independently work out a common strategy:* $M_{markt}, q_1 \models \neg\mathbb{E}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$. *Still, the experts can propose the right strategy if they join forces and cooperate to find the solution:* $M_{markt}, q_1 \models \mathbb{D}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$.

This is not true anymore for bad market: $M_{markt}, q_0 \models \neg\mathbb{D}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$, *as c is a memoryless agent, and it has no uniform strategy to enforce* success *from* q_0 *at all. However, the experts can suggest a more complex scheme that involves consulting them once again in the future:* $M_{markt}, q_0 \models \mathbb{D}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{X}\,\mathbb{D}_{\{1,2\}}\langle\langle c\rangle\rangle\mathrm{F}\mathsf{success}$.

6.4.3 Closer Look at Constructive Knowledge

In order to "constructively know" that φ, agents A must be able to find (or "construct") a mathematical object that supports φ. This is relevant when $\varphi \equiv \langle\langle B\rangle\rangle\gamma$; in that case, the mathematical object in question is a strategy for B which guarantees achieving γ. The semantic role of *constructive knowledge operators* is to produce sets of states that will appear on the left hand side of the satisfaction relation. In a way, these modalities "aggregate" states into sets, and sets into bigger sets.

Note that in $\mathbf{CSL}$ we can use two different notions of validity. We say that a formula is *weakly valid* (or simply *valid*) if it is satisfied individually by *each state* in every model, i.e.,, if $M, q \models \varphi$ for all models M and states q in M. It is *strongly valid* if it is satisfied by all non-empty *sets* in all models; i.e., if for each M and every non-empty set of states Q it is the case that $M, Q \models \varphi$. We are ultimately interested in the former. The importance of strong validity, however, lies in the fact that strong validity of $\varphi \leftrightarrow \psi$ makes φ and ψ completely interchangeable. That is, if $\varphi_1 \leftrightarrow \varphi_2$ is strongly valid, and ψ' is obtained from ψ through replacing an occurrence of φ_1 by φ_2, then $M, Q \models \psi$ iff $M, Q \models \psi'$. It is not difficult to see that the same is not true for weak validity.

Clearly, strong validity implies validity, but not vice versa.

Defining Standard Knowledge from Constructive Knowledge

In the semantics of $\mathbf{CSL}$, formulae are interpreted in sets of states; in order for φ to hold in M, Q, the formula must be "globally" satisfied in all states from Q at once (i.e., with a single strategy). Notice, however, that $M, Q \models \langle\langle\emptyset\rangle\rangle\varphi \,\mathrm{U}\, \varphi$ iff $M, q \models \varphi$ for every $q \in Q$. This can be used as a technical trick to evaluate φ "locally" (i.e., in every state of Q separately). In particular, we can use it to define standard knowledge from constructive knowledge as:

$$K_a\varphi \;\equiv\; \mathbb{K}_a\langle\langle\emptyset\rangle\rangle\varphi \,\mathrm{U}\, \varphi,$$

and analogously for group knowledge operators. It is not difficult to see that $M, q \models \mathbb{K}_a\langle\langle\emptyset\rangle\rangle\varphi \,\mathrm{U}\, \varphi$ iff $M, q' \models \varphi$ for every q' such that $q \sim_a q'$. More generally, the following formula of $\mathbf{CSL}$ is strongly valid:

$$\mathcal{K}_A\varphi \;\leftrightarrow\; \hat{\mathcal{K}}_A\langle\langle\emptyset\rangle\rangle\varphi \,\mathrm{U}\, \varphi,$$

where $\mathcal{K} = C, E, D$ and $\hat{\mathcal{K}} = \mathbb{C}, \mathbb{E}, \mathbb{D}$. In consequence, we obtain that *standard knowledge can be seen as a special case of constructive knowledge*.

Properties of Constructive Knowledge

Operators $\mathbb{C}_A$, $\mathbb{E}_A$, $\mathbb{D}_A$, and $\mathbb{K}_a$ are supposed to capture a special kind of agents' knowledge. An interesting question is: do these notions have the properties usually associated with knowledge? In particular, do postulates $\mathbf{K}, \mathbf{D}, \mathbf{T}, \mathbf{4}, \mathbf{5}$ of epistemic logic hold for constructive knowledge? Below, we list the constructive knowledge versions of the S5 axioms of individual knowledge. "Yes" means that the schema is strongly valid; "No" means that it is not even weakly valid (keep in mind that strong validity implies validity). Incidentally, none of the properties turns out to be weakly but not strongly valid.

K	$\mathbb{K}_a(\varphi \to \psi) \to (\mathbb{K}_a\varphi \to \mathbb{K}_a\psi)$	Yes
D	$\neg\mathbb{K}_a\bot$	Yes
T	$\mathbb{K}_a\varphi \to \varphi$	No
4	$\mathbb{K}_a\varphi \to \mathbb{K}_a\mathbb{K}_a\varphi$	Yes
5	$\neg\mathbb{K}_a\varphi \to \mathbb{K}_a\neg\mathbb{K}_a\varphi$	Yes

Thus, in general, the answer is *no*; particularly, axiom $\mathbf{T}$ does not hold. Note, however, that the axiom does hold in the restricted case when constructive knowledge is applied to positive strategic formulae, i.e., $\mathbb{K}_a\langle\!\langle B\rangle\!\rangle\gamma \to \langle\!\langle B\rangle\!\rangle\gamma$ *is* strongly valid in $\mathbf{CSL}$. Moreover, the above results show that $\mathbb{K}_a$ satisfies the standard postulates for beliefs: logical closure, consistency, and positive as well as negative introspection. This suggests that "knowing how to play" sits somewhere in between the realms of doxastic and epistemic logic: it is stronger than belief but not quite as strong as standard knowledge.

6.4.4 Expressing Epistemic Levels of Ability in CSL

A nice feature of $\mathbf{CSL}$ is that standard knowledge operators can be defined using constructive knowledge. Thus, one can use formulae of $\mathbf{CSL}$ to express the following:

1. $\langle\!\langle a\rangle\!\rangle\gamma$ expresses that agent a has a uniform strategy to enforce γ *from the current state*, i.e., a has the objective ability with respect to γ (but may not know about it);

2. $K_a\langle\!\langle a\rangle\!\rangle\gamma$ refers to agent a having a strategy *"de dicto"* to enforce γ (i.e. knowing that *some* successful uniform strategy is available);

3. $\mathbb{K}_a\langle\!\langle a\rangle\!\rangle\gamma$ refers to agent a having a strategy *"de re"* to enforce γ (i.e. having a successful uniform strategy and knowing the strategy);

It is interesting to see that $\mathbb{K}_a\langle\!\langle a\rangle\!\rangle\gamma$ captures the notion of a's knowing *how to play* to achieve γ, while $K_a\langle\!\langle a\rangle\!\rangle\gamma$ refers to knowing only *that a successful strategy is possible*. This extends naturally to abilities of coalitions. Again, that connects neatly to the fundamental discourse on ability that we mentioned in Section 6.1. It can be argued that standard knowledge operators K_a capture Ryle's notion of know-that, whereas constructive knowledge operators $\mathbb{K}_a$ refer to the notion of know-how. Also, formalisations (1) and (3) above roughly correspond to McCarthy and Hayes's levels (1) and (2) of program ability. Finally, formulae $\mathbb{K}_a\langle\!\langle a\rangle\!\rangle\gamma$ and $K_a\langle\!\langle a\rangle\!\rangle\gamma$ capture formally Moore's distinction between ability *de re* and *de dicto* for long-term strategies. This extends naturally to abilities of coalitions, with $\mathbb{C}_A\langle\!\langle A\rangle\!\rangle\gamma, \mathbb{E}_A\langle\!\langle A\rangle\!\rangle\gamma, \mathbb{D}_A\langle\!\langle A\rangle\!\rangle\gamma$ formalizing common, mutual, and distributed knowledge of how to play, $\mathbb{K}_a\langle\!\langle A\rangle\!\rangle\gamma$ capturing the "leader" scenario, and so on. Different levels of knowledge "de dicto" are captured analogously. We conclude the topic with the following examples.

Example 60 (Onion Soup Robbery) *A virtual safe contains the recipe for the best onion soup in the world. The safe can only be opened by a k-digit binary code, where each digit c_i is sent from a prescribed location i ($1 \le i \le k$). To open the safe and download the recipe it is enough that at least $n \le k$ correct digits are sent at the same moment. However, if a* wrong *value is sent from one of the locations, or if an insufficient number (i.e., between* 1 *and* $n-1$*) of digits is submitted, then the safe locks up and activates an alarm.*

k agents are connected at the right locations; each of them can send 0, send 1, or do nothing (nop). Moreover, individual agents have only partial information about the code. To make the example more concrete, we assume that agent i (connected to location i) knows the values of $c_{i-1}\,\mathrm{XOR}\,c_i$ and $c_i\,\mathrm{XOR}\,c_{i+1}$ (we take $c_0 = c_{k+1} = 0$). This implies that only agents 1 and k know the values of "their" digits. Still, every agent knows whether his neighbors' digits are the same as his.[4]

Formally, the concurrent epistemic game model $Attack_k^n$ is constructed as follows:

- $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$;
- $St = St_1 \cup St_2$, *where states in* $St_1 = \{0,1\}^k$ *identify possible codes for the (closed) safe, and states in* $St_2 = \{open, alarm\}$ *represent the situations when the safe has been opened, or when the alarm has been activated;*
- $\mathcal{PV} = \{\mathsf{open}\}$; $\quad \mathcal{V}(\mathsf{open}) = \{open\}$; $\quad Act = \{0, 1, nop\}$;
- $d(i,q) = \{0,1,nop\}$ *for* $q \in St_1$, *and* $d(i,q) = \{nop\}$ *for* $q \in St_2$;
- *For all* $q \in St$: $o(q, nop, \ldots, nop) = q$. *For* $q \in St_1$, *and at least one* $\alpha_i \neq nop$: $o(q, \alpha_1, \ldots, \alpha_k) = open$ *if* $\alpha_j = c_j$ *for at least* n *agents* j *and* $\alpha_i \notin \{c_i, nop\}$ *for no* i; *else,* $o(q, \alpha_1, \ldots, \alpha_k) = alarm$.
- $q \sim_i q'$ *iff* $q[i-1]\,\mathrm{XOR}\,q[i] = q'[i-1]\,\mathrm{XOR}\,q'[i]$ *and* $q[i]\,\mathrm{XOR}\,q[i+1] = q'[i]\,\mathrm{XOR}\,q'[i+1]$.

The following **CSL** *formulae hold in every state* $q \in St_1$ *of model* $Attack_k^n$, *assuming that* $k \geq 3$:

- $\langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \mathrm{F}\mathsf{open} \wedge \neg \mathbb{E}_{\mathbb{A}\mathrm{gt}} \langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \mathrm{F}\mathsf{open}$: *there is an executable strategy for the agents, which guarantees a win, but not all of them can identify it (in fact, none of them can in this case);*
- $\mathbb{D}_{\mathbb{A}\mathrm{gt}} \langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \mathrm{F}\mathsf{open}$: *if the agents share information they can recognize who should send what;*
- $\mathbb{D}_{\{1,\ldots,n-1\}} \langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \mathrm{F}\mathsf{open}$: *it is enough that the first* $n-1$ *agents devise the strategy. Note that the same holds for the last* $n-1$ *agents, i.e., the subteam* $\{k-n+2, \ldots, k\}$;
- *Still,* $\neg \mathbb{D}_{\{1,\ldots,n-1\}} \langle\!\langle 1, \ldots, n-1 \rangle\!\rangle \mathrm{F}\mathsf{open}$: *all agents are necessary to* execute *the strategy.*

We observe that constructive knowledge operators allow to approximate the amount of communication that is needed to establish a winning strategy in scenarios where explicit modeling of communication is impossible or too expensive. For instance, formula $\mathbb{D}_{\mathbb{A}\mathrm{gt}} \langle\!\langle \mathbb{A}\mathrm{gt} \rangle\!\rangle \mathrm{F}\mathsf{open}$ *says that if the agents in* $\mathbb{A}\mathrm{gt}$ *share their information they will be able to determine a strategy that opens the safe. Of course, the model does not include a possibility of such "sharing", at least not explicitly. That is, there is no transition that leads to a state in which the epistemic relations of agents have been combined via intersection. Still,* $D_A\varphi$ *indicates that there is* epistemic potential *for agents in* A *to realize/infer* φ; *what might be missing is means of exploiting the potential (e.g., by communication). In the same way,* $\mathbb{D}_A \langle\!\langle A \rangle\!\rangle \mathrm{F}\varphi$ *says that the epistemic potential for* A *to determine the right strategy for* $\mathrm{F}\varphi$ *is there, too. So, it might be profitable to design efficient communication mechanisms that make the most of it.*

[4] For the more seriously minded readers, we observe that the story is just a variant of coordinated attack.

Example 61 (Rescue Robots: Expressing the properties)

♣ The robots can rescue person j:

$$\langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\mathsf{safe}_{\mathsf{j}}.$$

Note: this looks like the specification in the previous chapter, but a *uniform strategy* is required for the robots now!

♣ The robots can rescue person j, and they know that they can:

$$\mathrm{E}_{\mathit{Robots}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\mathsf{safe}_{\mathsf{j}}.$$

Or: $\mathrm{C}_{\mathit{Robots}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\mathsf{safe}_{\mathsf{j}}$, $\mathrm{D}_{\mathit{Robots}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\mathsf{safe}_{\mathsf{j}}$, etc.

♣ The robots can rescue person j, and they know how to do it:

$$\mathbb{E}_{\mathit{Robots}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\mathsf{safe}_{\mathsf{j}}.$$

Or: $\mathbb{D}_{\mathit{Robots}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\mathsf{safe}_{\mathsf{j}}$, $\mathbb{C}_{\mathit{Robots}} \langle\langle \mathit{Robots} \rangle\rangle \mathrm{F}\mathsf{safe}_{\mathsf{j}}$, etc.

Example 62 (Voting: Expressing the properties)

♠ The system cannot reveal how a particular voter voted:

$$\neg\langle\langle system \rangle\rangle \mathrm{F}\big(\bigvee_{c \in \mathit{Candidates}} \mathsf{revealedVote}_{\mathsf{v,c}}\big).$$

Or: $\neg\mathbb{K}_{system} \langle\langle system \rangle\rangle \mathrm{F}(\bigvee_{c \in \mathit{Candidates}} \mathsf{revealedVote}_{\mathsf{v,c}})$.

A more refined specification: $\neg\langle\langle system \rangle\rangle \mathrm{F}\mathrm{K}_{coerc}(\bigvee_{c \in \mathit{Candidates}} \mathsf{voted}_{\mathsf{v,c}})$.

What if the system revealed the information but the coercer were too dumb to listen? $\rightsquigarrow$ our final refinement of the specification:

$$\neg\langle\langle system \rangle\rangle \mathrm{F} \Big(\mathbb{K}_{coerc} \langle\langle coerc \rangle\rangle \mathrm{F}\ \mathrm{K}_{coerc}\big(\bigvee_{c \in \mathit{Candidates}} \mathsf{voted}_{\mathsf{v,c}}\big)\Big).$$

♠ The voter cannot cooperate with the coercer to prove to him that she voted in a certain way:

$$\neg\langle\langle v, coerc \rangle\rangle \mathrm{F}\mathrm{K}_{coerc}\big(\bigvee_{c \in \mathit{Candidates}} \mathsf{voted}_{\mathsf{v,c}}\big).$$

More liberal interpretation:

$$\neg\mathbb{D}_{\{v,coerc\}} \langle\langle v, coerc \rangle\rangle \mathrm{F}\mathrm{K}_{coerc}(\textstyle\bigvee_{c \in \mathit{Candidates}} \mathsf{voted}_{\mathsf{v,c}}).$$

♠ The voter cannot gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way:

Cannot be expressed in CSL!

(we need a notion of information update)

References and Further Reading. Various combinations of alternating-time logic and epistemic operators have been considered in [172, 99, 3, 93]. An overview of related issues can be found in [5]. For the discussion on ability *de re* vs. *de dicto* and various epistemic modes of collective ability, we refer the reader to [99, 101]. A **stit** approach to combining knowledge and one-step ability was presented in [82]; a similar approach for long-term abilities has been suggested in [27, 26].

Chapter 7

Model Checking Uncertain Agents

7.1 Verification of Abilities in Imperfect Information Scenarios

In contrast to the perfect information setting, analogous fixpoint characterizations need *not* hold for the incomplete information semantics of $\mathbf{ATL}$. This is because the choice of a particular action at a state q has non-local consequences: it automatically fixes agent i's choices at all states q' indistinguishable from q for i. Note that, for two different members of coalition A, uniformity of their parts of the coalitional strategy imposes different constraints on their choices if their epistemic relations are not exactly the same. Moreover, the agents' ability to *identify* a strategy as winning also varies throughout the game in an arbitrary way (agents can learn as well as forget). This suggests that winning strategies cannot be synthesized incrementally. The observation is supported by the formal results that we will present in this section.

7.1.1 Model Checking $\mathbf{ATL}_{\text{ir}}$

Schobbens was the first to show that model checking strategic ability for agents with imperfect information is not doable anymore in deterministic polynomial time. His argument follows by a reduction of the boolean satisfiability problem ($\mathbf{SAT}$) already presented in Section 5.3.1. We recall that $\mathbf{SAT}$ decides satisfiability of formulae $\Phi \equiv C_1 \wedge \cdots \wedge C_n$ involving k propositional variables from set $X = \{x_1, ..., x_k\}$. Each clause C_i can be written as $C_i \equiv x_1^{s_{i,1}} \vee \cdots \vee x_k^{s_{i,k}}$ where $s_{i,j} \in \{+, -, 0\}$; x_j^+ denotes a positive occurrence of x_j in C_i, x_j^- denotes an occurrence of $\neg x_j$ in C_i, and x_j^0 indicates that x_j does not occur in C_i. The problem asks if $\exists X.\Phi$, that is, if there is a valuation of $x_1, ..., x_k$ such that Φ becomes true.

Schobbens' reduction is as follows: we construct a CEGM $M_{SAT(\Phi)}$ with the states grouped in n rows corresponding to subsequent clauses. Within row i, we put a sequence of states $q_{i,j}$ corresponding to literals $x_j^{s_{i,j}}$ in clause C_i, such that $s_{i,j} \neq 0$. There are also two additional states: the "winning" state $q_\top$, labeled by proposition win, and the "failure" state $q_\perp$. The system includes only one player $\mathbf{v}$ (the "verifier"). At state $q_{i,j}$, the verifier can declare the underlying variable x_j true or false (actions

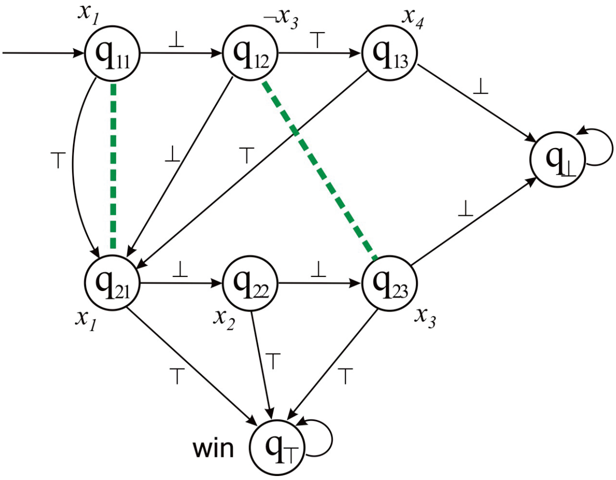


Figure 7.1: Single-agent model $M_{SAT(\Phi)}$ for boolean formula $\Phi \equiv C_1 \wedge C_2$ with $C_1 \equiv x_1 \vee \neg x_3 \vee x_4$ and $C_2 \equiv x_1 \vee x_2 \vee x_4$

$\top$ and $\bot$, respectively). If she "misses" (i.e., the declared value makes the literal $x_j^{s_{i,j}}$ false), then the system proceeds to the next state in the row; for the last state in the row, it proceeds to the failure state $q_\bot$. If she makes a hit (i.e., the declaration makes the $x_j^{s_{i,j}}$ true), then the system proceeds to the initial state for the next clause. If C_i was already the last clause (i.e., $i = n$), then hitting the right value makes the system proceed to the winning state $q_\top$. Finally, the states corresponding to the same variable x_j are indistinguishable to agent $\mathbf{v}$ so that she has to declare the same value for x_j within a uniform strategy. An example of the construction for $\Phi \equiv (x_1 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee x_4)$ is shown in Figure 7.1.

Now, we have that $\mathbf{SAT}(\Phi)$ iff $M_{SAT(\Phi)}, q_{1,1} \models_{\text{ir}} \langle\langle \mathbf{v} \rangle\rangle \mathrm{F}\mathsf{win}$. Both the number of the states and transitions in $M_{SAT(\Phi)}$ are linear in the length of Φ, and the construction of M requires linearly many steps. Thus, the model checking problem for $\mathbf{ATL}_{\text{ir}}$ is $\mathbf{NP}$-hard even for extremely simple instances.

Theorem 32 *Model checking* $\mathbf{ATL}_{\text{ir}}$ *is* $\mathbf{NP}$*-hard. It is* $\mathbf{NP}$*-hard even for 1-player models and formulae of type* $\langle\langle a \rangle\rangle \mathrm{F}\mathsf{p}$.

The upper bound is obtained as follows. First, the algorithm in Figure 7.2 can be used to nondeterministically check if $M, q \models_{\text{ir}} \langle\langle A \rangle\rangle \gamma$ for flat formulae, i.e., such that γ includes no nested cooperation modalities. For nested cooperation modalities, we proceed recursively (bottom up), replacing subformulae by freshly created atomic propositions that hold in exactly the same set of states. Since model checking $\mathbf{CTL}$ can be performed in polynomial deterministic time, we get that model checking $\mathbf{ATL}_{\text{ir}}$ is in $\mathbf{\Delta_2^P} = \mathbf{P^{NP}}$. In other words, it can be done in polynomially many steps by a deterministic Turing machine making adaptive calls to an $\mathbf{NP}$ oracle. Note also that for formulae of "Positive $\mathbf{ATL}$" (where negation is allowed only at the level of literals), the strategies for all the cooperation modalities in φ can be guessed in advance, at the beginning of the algorithm. Thus, we get the following.

Theorem 33 *Model checking* Positive $\mathbf{ATL}_{\text{ir}}$ *is* $\mathbf{NP}$*-complete with respect to the number of the transitions in the model and the length of the formula.*

```
function $mcheckatlir(M, q, \langle\!\langle A\rangle\!\rangle\gamma)$.
    Model checking simple formulae of $\mathbf{ATL}_{\mathrm{ir}}$.
    Returns $\top$ if $M, q \models_{\mathrm{ir}} \varphi$ and $\bot$ otherwise.

    ■ Guess a uniform strategy $s_A$;
    ■ Remove from $M$ all the transitions that are not going to be executed according to $s_A$;
    ■ Model-check CTL formula $\mathsf{A}\gamma$ in the resulting model, and return the outcome.
```

Figure 7.2: Nondeterministic model checking for simple formulae of $\mathbf{ATL}_{\mathrm{ir}}$

The above bounds are not tight for model checking arbitrary $\mathbf{ATL}_{\mathrm{ir}}$ formulae, and the actual complexity can be anywhere between $\mathbf{NP}$ and $\mathbf{\Delta_2^P}$. As it turns out, the more pessimistic estimate is the right one: the problem is in general $\mathbf{\Delta_2^P}$-complete. We will show this by a reduction of a suitable $\mathbf{\Delta_2^P}$-complete problem. To this end, we first present an alternative $\mathbf{SAT}$ reduction which will be subsequently "lifted" to a more complex decision problem.

Given a boolean formula Φ in CNF, we construct the new CEGM $M_{SAT2(\Phi)}$ as follows. There are two players now: the "verifier" $\mathbf{v}$ and the "refuter" $\mathbf{r}$. The refuter decides at the beginning of the game which clause C_i will have to be satisfied: this is done by proceeding from the initial state q_0 to a "clause" state q_i. At q_i, the verifier decides (by proceeding to a "literal" state $q_{i,j}$) which of the literals $x_j^{s_{i,j}}$ from C_i will be attempted. Finally, at $q_{i,j}$, the verifier attempts to prove C_i by declaring the underlying propositional variable x_j true (action $\top$) or false (action $\bot$). If she succeeds (i.e., if she executes $\top$ for x_j^+, or executes $\bot$ for x_j^-), then the system proceeds to the "winning" state $q_\top$. Otherwise, the system stays in $q_{i,j}$. Additionally, "literal" states referring to the same variable are indistinguishable for the verifier, so that she has to declare the same value of x_j in all of them within a uniform strategy. A sole proposition yes holds only in the winning state $q_\top$. Obviously, states corresponding to literals x_j^0 can be omitted from the model.

Speaking more formally, $M_{SAT2(\Phi)} = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, Act, d, o, \sim_1, ..., \sim_k\rangle$, where:

- $\mathbb{A}\mathrm{gt} = \{\mathbf{v}, \mathbf{r}\}$,
- $St = \{q_0\} \cup St_{cl} \cup St_{lit} \cup \{q_\top\}$, where $St_{cl} = \{q_1, \dots, q_n\}$, and $St_{lit} = \{q_{1,1}, \dots, q_{1,k}, \dots, q_{n,1}, \dots, q_{n,k}\}$;
- $\Pi = \{\mathsf{yes}\}, \qquad \pi(\mathsf{yes}) = \{q_\top\}$,
- $Act = \{1, ..., \max(k, n), \top, \bot\}$,
- $d(\mathbf{v}, q_0) = d(\mathbf{v}, q_\top) = \{1\}, \qquad d(\mathbf{v}, q_i) = \{1, ..., k\}$,
  $d(\mathbf{v}, q_{i,j}) = \{\top, \bot\}$,
  $d(\mathbf{r}, q) = \{1, ..., n\}$ for $q = q_0$, and $d(\mathbf{r}, q) = \{1\}$ otherwise;
- $o(q_0, 1, i) = q_i, \qquad o(q_i, j, 1) = q_{i,j}$,
  $o(q_{i,j}, \top, 1) = q_\top$ if $s_{i,j} = +$, and $q_{i,j}$ otherwise,
  $o(q_{i,j}, \bot, 1) = q_\top$ if $s_{i,j} = -$, and $q_{i,j}$ otherwise;

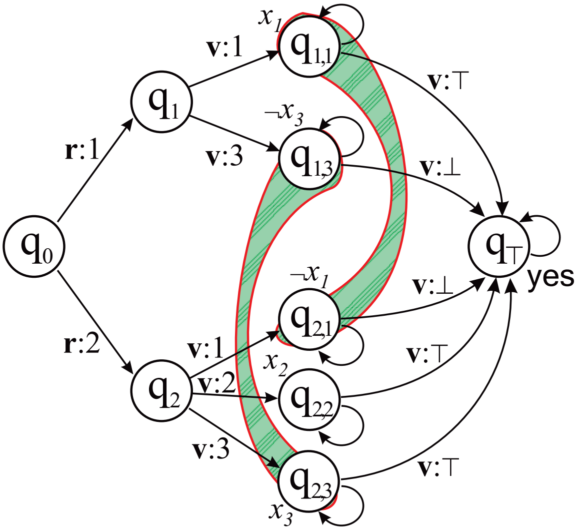


Figure 7.3: Alternative reduction of **SAT** for $\Phi \equiv (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$

- $q_0 \sim_{\mathbf{v}} q$ iff $q = q_0$, $\qquad q_i \sim_{\mathbf{v}} q$ iff $q = q_i$, $\qquad q_{i,j} \sim_{\mathbf{v}} q$ iff $q = q_{i',j}$.

As an example, the model $M_{SAT2(\Phi)}$ for $\Phi \equiv (x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$ is presented in Figure 7.3.

Now, we have $\mathbf{SAT}(\Phi)$ iff $M_{SAT2(\Phi)}, q_0 \models_{\mathrm{ir}} \langle\!\langle \mathbf{v} \rangle\!\rangle \mathrm{F} \mathsf{yes}$. We proceed to lift the reduction so that it works for the *sequential boolean satisfiability problem* $\mathbf{SNSAT_2}$, complete for the class $\mathbf{\Delta_2^P}$.

Definition 14 ($\mathbf{SNSAT_2}$)

Input: *k sets of propositional variables $X_r = \{x_{1,r}, ..., x_{n,r}\}$, k propositional variables z_r, and k boolean formulae Φ_r in CNF, where each Φ_r involves only variables from $X_r \cup \{z_1, ..., z_{r-1}\}$, with the following requirement:*

z_r is true iff there exists an assignment of variables in X_r which makes Φ_r true.

By a slight abuse of notation, this can be expressed as $z_r \equiv \exists X_r\, \Phi_r(z_1, ..., z_{r-1}, X_r)$.
Output: *The truth value of z_k.*

Let m be the maximal number of clauses in any $\Phi_1, ..., \Phi_k$ from the given input. Now, each Φ_r can be written as:

$$\Phi_r \equiv C_1^r \wedge \cdots \wedge C_m^r, \text{ and } C_i^r \equiv x_{1,r}^{s^r_{i,1}} \vee \cdots \vee x_{k,r}^{s^r_{i,k}} \vee z_1^{s^r_{i,k+1}} \vee \ldots z_{r-1}^{s^r_{i,k+r-1}}.$$

Again, $s^r_{i,j} \in \{+, -, 0\}$; x^+ denotes a positive occurrence of x, x^- denotes an occurrence of $\neg x$, and x^0 indicates that x does not occur in the clause. Similarly, $s^r_{i,k+j}$ defines the "sign" of z_j in clause C_i^r. Given such an instance of $\mathbf{SNSAT_2}$, we construct a sequence of concurrent game structures M_r for $r = 1, ..., k$ in a similar way to M_{SAT2}. That is, clauses and variables $x_{i,r}$ are handled in exactly the same way as before. Moreover, if z_i occurs as a positive literal in Φ_r, we embed M_i in M_r, and add a transition to the initial state q_0^i of M_i. If $\neg z_i$ occurs in Φ_r, we do almost the same: the only difference is that we split the transition into two steps, with a state neg_i^r (labeled with an $\mathbf{ATL}_{\text{ir}}$ proposition neg) added in between.

More formally, $M_r = \langle \mathbb{A}\text{gt}, St^r, \Pi, \pi^r, Act^r, d^r, o^r, \sim_1^r, ..., \sim_k^r \rangle$, where:

- $\mathbb{A}\text{gt} = \{\mathbf{v}, \mathbf{r}\}$,
- $St^r = \{q_0^r, q_1^r, \dots, q_m^r, q_{1,1}^r, \dots, q_{m,k}^r, neg_1^r, \dots, neg_{r-1}^r, q_\top\} \cup St^{r-1}$,
- $\Pi = \{\mathsf{yes}, \mathsf{neg}\}$, $\quad \pi^r(\mathsf{yes}) = \{q_\top\}$, $\quad \pi^r(\mathsf{neg}) = \{neg_i^j \mid i, j = 1, ..., r\}$,
- $Act^r = \{1, ..., \max(k + r - 1, m), \top, \bot\}$,
- $d^r(\mathbf{v}, q_0^r) = d^r(\mathbf{v}, neg_i^r) = d^r(\mathbf{v}, q_\top) = \{1\}$, $\quad d^r(\mathbf{v}, q_i^r) = \{1, ..., k + r - 1\}$,
  $d^r(\mathbf{v}, q_{i,j}^r) = \{\top, \bot\}$,
  $d^r(\mathbf{r}, q) = \{1, ..., m\}$ for $q = q_0^r$ and $\{1\}$ for the other $q \in St^r$.
  For $q \in St^{r-1}$, we simply include the function from M_{r-1}: $d^r(a, q) = d^{r-1}(a, q)$;
- $o^r(q_0^r, 1, i) = q_i^r$, $\quad o^r(q_i^r, j, 1) = q_{i,j}^r$ for $j \le k$,
  $o^r(q_i^r, k + j, 1) = q_0^j$ if $s_{i,k+j}^r = +$, and $o^r(q_i^r, k + j, 1) = neg_j^r$ if $s_{i,k+j}^r = -$,
  $o^r(neg_j^r, 1, 1) = q_0^j$,
  $o^r(q_{i,j}^r, \top, 1) = q_\top$ if $s_{i,j}^r = +$, and $q_{i,j}^r$ otherwise,
  $o^r(q_{i,j}^r, \bot, 1) = q_\top$ if $s_{i,j}^r = -$, and $q_{i,j}^r$ otherwise.
  For $q \in St^{r-1}$, we include the transitions from M_{r-1}: $o^r(q, \alpha) = o^{r-1}(q, \alpha)$;
- $q_0^r \sim_{\mathbf{v}} q$ iff $q = q_0^r$, $\quad q_i^r \sim_{\mathbf{v}} q$ iff $q = q_i^r$, $\quad q_{i,j}^r \sim_{\mathbf{v}} q$ iff $q = q_{i',j}^r$.
  For $q, q' \in St^{r-1}$, we include the tuples from M_{r-1}: $q \sim_{\mathbf{v}}^r q'$ iff $q \sim_{\mathbf{v}}^{r-1} q'$.

As an example, model M_3 for $\Phi_3 \equiv (x_3 \vee \neg z_2) \wedge (\neg x_3 \vee \neg z_1)$, $\Phi_2 \equiv z_1 \wedge \neg z_1$, $\Phi_1 \equiv (x_1 \vee x_2) \wedge \neg x_1$, is presented in Figure 7.4. Now, let

$$\begin{aligned} \varphi_1 &\equiv \langle\langle \mathbf{v} \rangle\rangle_{ir} (\neg \mathsf{neg})\, \mathrm{U}\, \mathsf{yes}, \\ \varphi_i &\equiv \langle\langle \mathbf{v} \rangle\rangle_{ir} (\neg \mathsf{neg})\, \mathrm{U}\, (\mathsf{yes} \vee (\mathsf{neg} \wedge \mathsf{AX} \neg \varphi_{i-1})). \end{aligned}$$

We have that, for all r, z_r is true iff $M_r, q_0^r \models \varphi_r$. Thus, for $r = k$, we get a reduction of $\mathbf{SNSAT_2}$. The following is a straightforward corollary.

Theorem 34 *Model checking $\mathbf{ATL}_{\text{ir}}$ is $\mathbf{\Delta_2^P}$-complete with respect to the number of the transitions in the model and the length of the formula.*

7.1.2 Model Checking Nexttime Formulae

Model checking strategic ability under imperfect information turns out distinctly harder than for perfect information. However, all the proofs in Section 7.1.1 used long-term abilities to construct appropriate reductions. What about short-term abilities? That is, what happens to the complexity if we restrict the language to formulae $\langle\langle A \rangle\rangle \mathrm{X} \varphi$ of $\mathbf{ATL}$, or alternatively formulae $[A]\varphi$ of Coalition Logic? It is easy to see that the iR- and ir-semantics are equivalent for $\mathbf{CL}$ since X is the only temporal operator, and thus only the first action in a strategy matters. As a consequence, whenever there is a successful iR-strategy for agents A to enforce $\mathrm{X}\varphi$, then there is also an ir-strategy for A to obtain the same. Perfect recall of the past does not matter in one-step games. Surprisingly, what matters is the size of teams that are allowed to cooperate.

Theorem 35 *Model checking $\mathbf{CL}_{\text{ir}}$ and $\mathbf{CL}_{\text{iR}}$ for formulae that include only strategic operators $\langle\langle A \rangle\rangle$ with $|A| \le 2$ is $\mathbf{P}$-complete with respect to the number of the transitions in the model and the length of the formula, and can be performed in time $\mathbf{O}(|M| \cdot |\varphi|)$.*

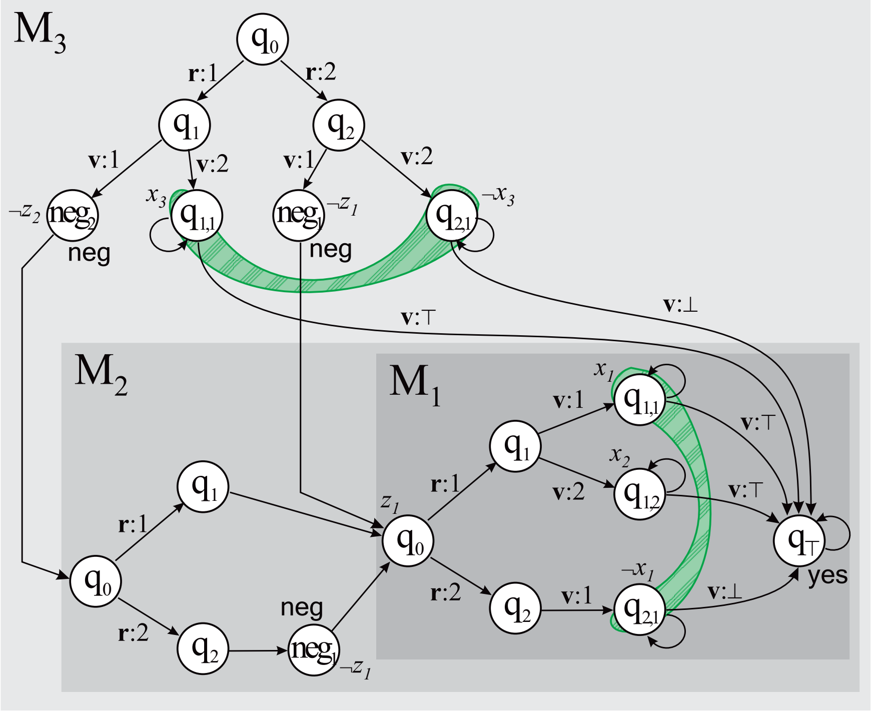


Figure 7.4: CEGM for the reduction of $\mathbf{SNSAT_2}$. The superscripts in state labels are omitted since they can be deduced from the sub-machine in which the state resides.

The $\mathbf{P}$-hardness follows from Theorem 20, since perfect information CGM's can be seen as a special kind of CEGM where indistinguishability relations contain only the reflexive loops. To obtain the upper bound, we use the algorithm in Figure 7.5 to model check formulae of type $\langle\!\langle A \rangle\!\rangle \mathrm{X}\mathsf{p}$ where $A \subseteq \{a_1, a_2\}$. It is easy to see that the algorithm never processes the same transition twice. For $\langle\!\langle A \rangle\!\rangle \mathrm{X}\varphi$ with nested cooperation modalities, we proceed recursively (bottom up).

What about larger coalitions? Surprisingly, having more than two agents on board makes the problem $\mathbf{NP}$-hard again:

Theorem 36 *Model checking Simple* $\mathbf{CL}_{\mathrm{ir}}$ *and Simple* $\mathbf{CL}_{\mathrm{iR}}$ *for* $|A| \geq 3$ *is* $\mathbf{NP}$-*complete with respect to the number of the transitions in the model and the length of the formula.*

The upper bound is obtained by the following algorithm. To check $M, q \models \langle\!\langle A \rangle\!\rangle \mathrm{X}\mathsf{p}$, we guess a one-step strategy of A, remove from M the irrelevant transitions and states outside $[q]_{\sim_A}$, and check if p holds for all the remaining "next" states.

For the lower bound, we use yet another reduction of $\mathbf{SAT}$. Given a boolean formula Φ in CNF, we construct a 3-agent CEGM $M_{SAT3(\Phi)}$ as follows. Each literal l from clause ψ in Φ is associated with a state q_l^{ψ}. At state q_l^{ψ}, player 1 indicates a literal from ψ, and player 2 decides on the valuation of the underlying Boolean variable. If 1 indicated a "wrong" literal $l' \neq l$, then the system proceeds to state $q_\top$ where proposition yes holds. The same happens if 1 indicated the "right" literal (l) and 2

```
function $mcheckclir(M, q, \langle\langle a_1, a_2 \rangle\rangle \mathrm{X}\mathsf{p})$.
Model checking CL formulae with small coalitions and no nested modalities.
Let $Q = [q]_{\sim_A}$ and $D = d_{a_1}(q) \times d_{a_2}(q)$;
while there is still a collective action $(\alpha_1, \alpha_2)$ in $D$ do:

    ■ Fix $\alpha_1$ for $a_1$ in $[q]_{\sim_{a_1}}$ and $\alpha_2$ for $a_2$ in $[q]_{\sim_{a_2}}$. Remove irrelevant transitions;

    ■ For every state in $[q]_{\sim_A}$ there is at most one agent in $A$ for whom the action
      has not been fixed. If $a_i$'s action is not fixed for $q', q''$ such that $q' \sim_{a_i} q''$
      then collapse $q', q''$ into a single state (taking the union of the outgoing
      transitions). Repeat iteratively;

    ■ If in the resulting perfect information CEGM $A$ have a one-step strategy
      to enforce $\mathsf{p}$ in the next state from all states in $[q]_{\sim_A}$ then return true else
      remove $(\alpha_1, \alpha_2)$ from $D$ and revert to the original model $M$;

  od

Return false (since the loop ended with no success, there are no more available
actions);
```

Figure 7.5: Model checking Coalition Logic for small teams and imperfect information

selected the valuation that makes l true. Otherwise the system proceeds to the "failure" state $q_\perp$. Player 1 must select literals uniformly within clauses, so $q_l^\psi \sim_1 q_{l'}^{\psi'}$ iff $\psi = \psi'$. Player 2 is to select uniform valuations of variables, i.e., $q_l^\psi \sim_2 q_{l'}^{\psi'}$ iff $var(l) = var(l')$ where $var(l)$ is the variable contained in l. Finally, all states except $q_\top, q_\perp$ are indistinguishable for 3. An example of the construction is presented in Figure 7.6.

For models constructed this way, Φ is satisfiable iff $M_{SAT3(\Phi)}, q \models \langle\langle 1, 2, 3 \rangle\rangle \mathrm{X}\mathsf{yes}$, where q is an arbitrary "literal" state.

7.1.3 Bad News for Agents with Perfect Recall

The really bad news for model checking **ATL** with imperfect information await when we concern memory-based strategies.

Theorem 37 *Model checking* $\mathbf{ATL}_{\mathrm{iR}}$ *is undecidable.*

Proof idea. The proof follows by a reduction of the non-halting problem for non-deterministic Turing machines. Given a machine T, we construct a CEGM M such that the evolution of the configuration of T is simulated by the tree of computations in M. That is, subsequent configurations of T are represented by subsequent levels in the tree unfolding of M. There are 3 agents in M: the verifiers (agents 1 and 2) who "move" the head of the tape in T, and the refuter (agent 3) who takes care of splitting the branches (i.e., adding new symbols to the tape whenever they are needed). The indistinguishability relations for 1 and 2 are constructed so that moving the head (and possibly changing the current state of T) proceeds uniformly; 1 takes care of the left-hand side of the head, and 2 of the right-hand side. Special proposition ok labels

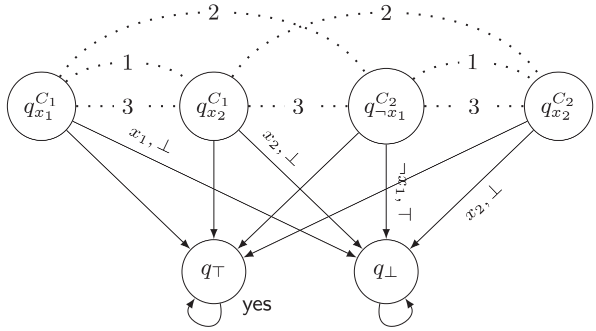


Figure 7.6: Model $M_{SAT3(\Phi)}$ for $\Phi \equiv C_1 \wedge C_2$, where $C_1 \equiv x_1 \vee x_2$ and $C_2 \equiv \neg x_1 \vee x_2$. Only transitions leading to $q_\bot$ are labeled; the other combinations of actions lead to $q_\top$.

elements of configurations from which T has to proceed further. Thus, we get that T does not terminate iff $M, q_0 \models \langle\langle 1, 2\rangle\rangle \mathrm{G}\mathsf{ok}$.

An example unfolding is shown in Figure 7.7. For more details, we refer to the technical construction in [55]. ■

It is worth pointing out that for single-agent coalitions the model checking problem is decidable – more precisely, $\mathbf{EXPTIME}$-complete.

Remark 4 *Theorem 37 is well aligned with the tradition of previous undecidability results [134, 133, 140, 173, 155, 154] for various logical combinations of the temporal and epistemic dimensions. At the first glance, one may even think that it follows from those previously existing results. This impression is misleading. To give a better view of the complicated landscape on the borderline of decidability for modal logics of time, knowledge, and strategies, we comment on it below in more detail.*

Classical results in [134, 133] show that solving games with imperfect information, perfect recall and multiple proponents is undecidable. However, the games in [134, 133] are defined by Turing machines. In case of $\mathbf{ATL}_{\mathrm{iR}}$ *models can be seen as Büchi automata with simple acceptance conditions (reachability or safety). We note that, for games defined by Turing machines, more sophisticated "winning conditions" can be specified, e.g., we can imagine a game in which the protagonist agents must count specific actions of the opponent in order to win. As a more precise example, imagine a game in which team* $T_1 = \{a_1, a_2\}$ *is playing against the "environment" agent* a_0 *in the following way: first, the opponent (*a_0*) throws in any finite number of* $\bigcirc$ *symbols onto the tape, and then agents* a_1, a_2 *can write a number of* $\#$*'s, before they decide to "call". After that,* a_0 *checks if the amounts of* $\bigcirc$*'s and* $\#$*'s are the same: if so, team* T_1 *wins, otherwise it loses the game. Clearly, such a winning condition cannot be specified through a CEGM with a subset of states marked as "winning" states, because the language* $\{\bigcirc^n \#^n \epsilon^\omega \mid n \in \mathbb{N}\}$ *is not* ω*-regular. Moreover, using a Turing Machine allows to define more sophisticated rules of how the game proceeds, e.g., the protagonists can be allowed to put another symbol* $\square$ *on the tape only when they have already put as many* $\#$*'s as there are* $\bigcirc$*'s left by the opposition.*

The paper [140] shows that that the $\mathbf{LTL}$ *realizability problem for distributed systems is undecidable. This implies that model checking of* $\mathbf{ATL}^*_{\mathrm{iR}}$ *is undecidable, but*

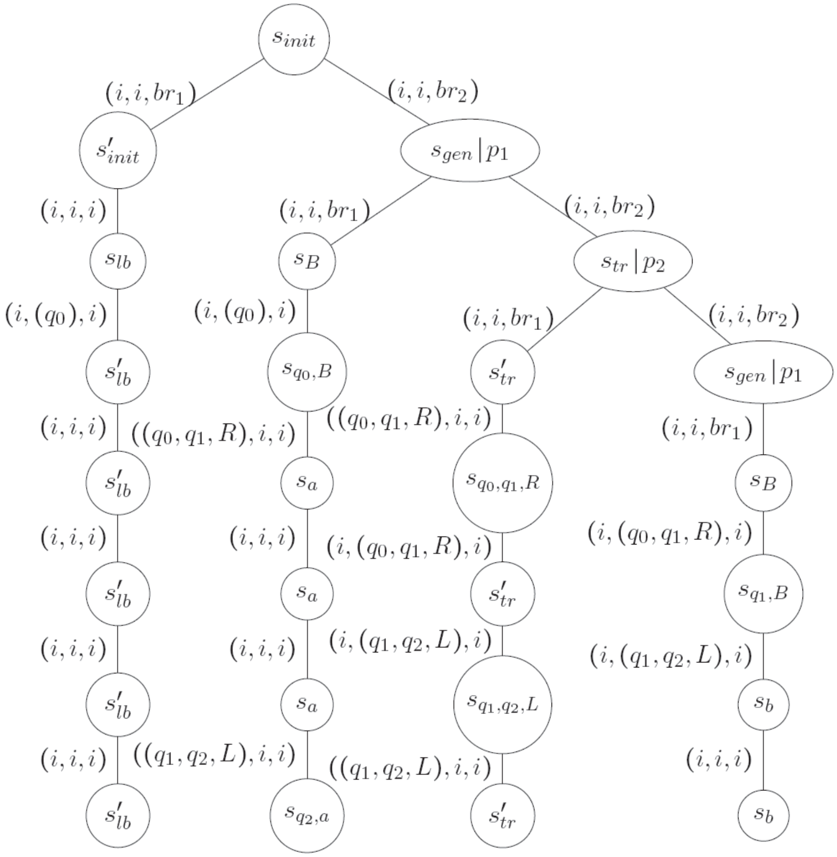


Figure 7.7: Simulation of computation $q_0B \Rightarrow aq_1B \Rightarrow q_2ab$ of the Turing machine

it does not carry over to "vanilla" $\mathbf{ATL}_{\text{iR}}$*. That is because "winning conditions" in [140] are defined through* $\mathbf{LTL}$ *specifications, so we have winning* paths *rather than states. Moreover, the reduction of the halting problem to the realizability problem employs* $\mathbf{LTL}$ *formulae that are expressible in neither* $\mathbf{CTL}$ *nor* $\mathbf{ATL}$ *(cf. [60]).*

Finally, [173] shows undecidability of model checking for $\mathbf{LTL}$ *with perfect recall and common knowledge, and [155, 154] gives analogous results for* $\mathbf{CTL}$*. However,* $\mathbf{ATL}_{\text{iR}}$ *does not allow for expressing common knowledge within teams of agents with perfect recall.*

7.1.4 Model Checking ATL*

After all the bad news in preceding subsections, it comes as a bit of relief that $\mathbf{ATL}^*$ specifications do not make things worse more than necessary. Of course, model checking $\mathbf{ATL}^*_{\text{iR}}$ must be undecidable by Theorem 37. On the other hand, model checking $\mathbf{ATL}^*$ with memoryless strategies is no more complex than for $\mathbf{LTL}$, by the same argument as for Theorem 23.

Theorem 38 *Model checking* $\mathbf{ATL}^*_{\text{ir}}$ *is* $\mathbf{PSPACE}$*-complete with respect to the num-*

	Ir	IR	ir	iR
$\mathbf{CL}$, $\lvert A\rvert \leq 2$	$\mathbf{P}$	$\mathbf{P}$	$\mathbf{P}$	$\mathbf{P}$
$\mathbf{CL}$, $\lvert A\rvert > 2$	$\mathbf{P}$	$\mathbf{P}$	between $\mathbf{NP}$ and $\mathbf{\Delta_2^P}$	
$\mathbf{ATL}$	$\mathbf{P}$	$\mathbf{P}$	$\mathbf{\Delta_2^P}$	Undecidable
$\mathbf{ATL^*}$	$\mathbf{PSPACE}$	$\mathbf{2EXPTIME}$	$\mathbf{PSPACE}$	Undecidable

Figure 7.8: Overview of the model checking complexity results for explicit models. All results are completeness results except where indicated. Rows indicate syntactic restrictions, columns indicate semantic variants.

ber of the transitions in the model and the length of the formula.

Figure 7.8 presents an overview of the model checking complexity results for explicit models.

References and Further Reading. A survey of complexity results for model checking in variants of strategic logics has been published in [33].

$\mathbf{ATL}$ with imperfect information and perfect recall was proved undecidable in [55]. $\mathbf{EXPTIME}$-completeness of solving 2-player adversarial games with perfect recall has been obtained in [56]. It also follows from the proof of a more general decidability result in [74] for model checking abilities of coalitions with unrestricted communication. Model checking complexity for one-step abilities has been established in [38].

7.2 Complexity Results for Compact Representations

Compact Representations of Transitions

For compressed representations of the transitions in the model, model checking of abilities under imperfect information turns out no harder than in the perfect information case.

Theorem 39 *Model checking* $\mathbf{ATL}_{\mathrm{ir}}$ *over implicit CEGM's is* $\mathbf{\Delta_3^P}$*-complete with respect to the number of the states, agents, and implicit transitions in the model, and the length of the formula.*

Thus, the complexity of model checking for $\mathbf{ATL}_{\mathrm{ir}}$ is the same as for $\mathbf{ATL}_{\mathrm{Ir}}$ and $\mathbf{ATL}_{\mathrm{IR}}$. For the upper bound, we can use a straightforward adaptation of the algorithm for $\mathbf{ATL}_{\mathrm{Ir}}$. For the lower bound, we observe that $\mathbf{ATL}_{\mathrm{Ir}}$ can be embedded in $\mathbf{ATL}_{\mathrm{ir}}$ by explicitly assuming perfect information of agents (through the minimal reflexive indistinguishability relations). Moreover, the hardness proof of Theorem 26 can be reconstructed so that only the "nexttime" sublanguage of $\mathbf{ATL}$ is used. Thus, we obtain the following.

Theorem 40 *Model checking* $\mathbf{CL}_{\mathrm{ir}}$*, and* $\mathbf{CL}_{\mathrm{iR}}$ *over implicit CGM's is* $\mathbf{\Delta_3^P}$*-complete with respect to the number of the states, agents, and implicit transitions in the model, and the length of the formula. The same applies to* $\mathbf{CL}_{\mathrm{IR}}$ *and* $\mathbf{CL}_{\mathrm{Ir}}$*.*

It is worth mentioning that model checking *Positive* $\mathbf{ATL}$ (i.e., the fragment of $\mathbf{ATL}$ where negation is allowed only on the level of literals) is $\mathbf{\Sigma_2^P}$-complete with respect to the size of implicit CGM's, and the length of formulae for the IR, Ir, and

	Ir	IR	ir	iR
CL	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$
ATL	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	Undecidable
$\mathbf{ATL}^*$	**PSPACE**	**2EXPTIME**	**PSPACE**	Undecidable

Figure 7.9: Overview of the model checking complexity for implicit CEGM's. All results are completeness results.

ir semantics. We also observe that the proof of Theorem 27 can be directly applied to uniform memoryless strategies.

Theorem 41 *Model checking* $\mathbf{ATL}^*_{\mathrm{ir}}$ *over implicit CGM's is* $\mathbf{PSPACE}$*-complete with respect to the number of the states, agents, and implicit transitions in the model, and the length of the formula.*

Notice that the *finer-grained analysis* puts verification of strategic abilities for memoryless agents to the same complexity class regardless of whether they have perfect or imperfect information. It is somewhat unexpected, as the former case appears to be strictly harder than the latter when we approach it from a more "distant" perspective (i.e., when the input parameters are less detailed). What is different then, that makes model checking of $\mathbf{ATL}_{\mathrm{ir}}$ harder than standard $\mathbf{ATL}$ in relation to the number of the transitions? Definitely not the size of the transition function itself, because CGM's can be seen as a special case of CEGM's. A comparison of model checking complexity results for turn-based structures gives us a hint in this respect. For such turn-based models, the number of the transitions is bounded by $|St| \cdot |Act|$, and in consequence standard $\mathbf{ATL}$ can be model-checked in deterministic polynomial time by means of the standard fixpoint algorithm in Figure 5.1. The same is not possible for $\mathbf{ATL}_{\mathrm{ir}}$.

Theorem 42 *Model checking* $\mathbf{ATL}_{\mathrm{Ir}}$ *and* $\mathbf{ATL}_{\mathrm{IR}}$ *over turn-based implicit CEGM's is* $\mathbf{P}$*-complete and can be performed deterministically in time* $\mathbf{O}(|St| \cdot |Act| \cdot |\varphi|)$*. Since* $|Act| \leq |St|$ *for turn-based structures, the bound can be replaced by* $\mathbf{O}(|St|^2 \cdot |\varphi|)$*.*

On the other hand, model checking $\mathbf{ATL}_{\mathrm{ir}}$ *over turn-based implicit CEGM's is* $\mathbf{\Delta_2^P}$*-complete.*

Moreover, the exhaustive model checking of arbitrary formulae of $\mathbf{ATL}$ with perfect information is performed in time $\mathbf{O}(|St| \cdot |Act|^{|\mathbb{A}\mathrm{gt}|} \cdot |\varphi|)$, while for $\mathbf{ATL}_{\mathrm{ir}}$ it can be done in $\mathbf{O}(|St| \cdot |Act|^{|\mathbb{A}\mathrm{gt}| \cdot |St|} \cdot |\varphi|)$ steps. This is due to the fact that successful $\mathbf{ATL}_{\mathrm{Ir}}$/$\mathbf{ATL}_{\mathrm{IR}}$ strategies can be computed incrementally, state by state. By contrast, uniform strategies must be considered *as a whole*, which requires much more backtracking if we check the possibilities exhaustively.

We present a summary of the complexity results in Figure 7.9.

Local Representations of State Spaces

Another surprise comes to light when we study the verification complexity for compact representations of state spaces.

Theorem 43 *Model checking* $\mathbf{ATL}_{\mathrm{ir}}$ *over simple reactive modules and modular interpreted systems is* $\mathbf{PSPACE}$*-complete with respect to the number of local states and agents, and the length of the formula.*

	Ir	IR	ir	iR
CL	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$
ATL	**EXPTIME**	**EXPTIME**	**PSPACE**	Undecidable
ATL*	**EXPTIME**-hard	**2EXPTIME**-hard	**PSPACE**	Undecidable

Figure 7.10: Overview of the model checking complexity for local representations of state spaces. All the results are completeness results except where indicated.

Thus, model checking over local representations (such as modular interpreted systems) seems to be easier for imperfect rather than perfect information strategies. On the other hand, the results for explicit models were exactly the reverse. There are two possible reasons. The more immediate is that agents with limited information have *fewer* available strategies than if they had perfect information about the current (global) state of the game. Generally, the difference is exponential in the number of the agents. More precisely, the number of perfect information strategies is *double exponential* with respect to the number of the agents and their local states, while there are "only" exponentially many uniform strategies – and that settles the results in favor of imperfect information.

The other reason is more methodological. While model checking imperfect information is easier when we are given a particular modular interpreted system, MIS's may represent systems in a more compact way for agents that have perfect information by definition. In particular, the most compact MIS representation of a given CEGM M can be exponentially larger than the most compact MIS representation of M with the epistemic relations removed. This is because a modular interpreted system encoding a CEGM must represent the epistemic relations explicitly (like in standard interpreted systems where epistemic relations are generated by local state spaces, cf. Section 3.3). In contrast, a modular interpreted system encoding a perfect information model ignores the epistemic aspect, which gives some extra room for a more compact representation of the transition relation.

On the other hand, it should be noted that for systems of agents with "reasonably imperfect information," i.e., ones where the number of each agent's local states is logarithmic in the number of global states of the system, the optimal MIS encodings for perfect and imperfect information are the same. Still, model checking $\mathbf{ATL}_{\rm IR}$ is **EXPTIME**-complete and model checking $\mathbf{ATL}_{\rm ir}$ is **PSPACE**-complete, which suggests that imperfect information can offer some advantage in practical verification.

Finally, we report two results that are straightforward extensions of Theorem 40 and Theorem 43, respectively. Figure 7.10 collects the known results for model checking strategic logics over local representations.

Theorem 44 *Model checking* $\mathbf{CL}_{\rm IR}$, $\mathbf{CL}_{\rm Ir}$, $\mathbf{CL}_{\rm ir}$, *and* $\mathbf{CL}_{\rm iR}$ *is* $\mathbf{\Delta_3^P}$*-complete with respect to the number of local states and agents in the local representation, and the length of the formula.*

Theorem 45 *Model checking* $\mathbf{ATL}^*_{\rm ir}$ *over modular interpreted systems is* **PSPACE**-*complete with respect to the number of local states and agents, and the length of the formula.*

Logic \ Input	Explicit CEGM	Implicit CEGM	Local represent.
$\mathbf{CTL}$	$\mathbf{P}$	$\mathbf{\Delta_2^P}$	$\mathbf{PSPACE}$
$\mathbf{CL}_{\mathrm{Ir,IR}}$	$\mathbf{P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$
$\mathbf{CL}_{\mathrm{ir,iR}}$	between $\mathbf{P}$ and $\mathbf{\Delta_2^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{\Delta_3^P}$
$\mathbf{ATL}_{\mathrm{Ir,IR}}$	$\mathbf{P}$	$\mathbf{\Delta_3^P}$	$\mathbf{EXPTIME}$
$\mathbf{ATL}_{\mathrm{ir}}$	$\mathbf{\Delta_2^P}$	$\mathbf{\Delta_3^P}$	$\mathbf{PSPACE}$
$\mathbf{ATL}_{\mathrm{iR}}$	Undecidable	Undecidable	Undecidable
$\mathbf{ATL}^*_{\mathrm{Ir}}$	$\mathbf{PSPACE}$	$\mathbf{PSPACE}$	$\mathbf{EXPTIME}$-hard
$\mathbf{ATL}^*_{\mathrm{ir}}$	$\mathbf{PSPACE}$	$\mathbf{PSPACE}$	$\mathbf{PSPACE}$
$\mathbf{ATL}^*_{\mathrm{IR}}$	$\mathbf{2EXPTIME}$	$\mathbf{2EXPTIME}$	$\mathbf{2EXPTIME}$-hard
$\mathbf{ATL}^*_{\mathrm{iR}}$	Undecidable	Undecidable	Undecidable

Figure 7.11: Overview of the model checking complexity for different syntactic and semantic variants, and different representations of systems. All results are completeness results except where indicated.

Summary

The most important complexity results obtained in Chapter 5 and this chapter are presented in Figure 7.11. An important outcome of theoretical research on verification is to determine the precise boundary between variants that are decidable and those that are not. As the results show, decidability depends on the specification language, the amount of available information, the type of memory available to agents. Moreover, even if the problem is decidable, the precise complexity of most variants ranges from $\mathbf{\Delta_2^P}$ to $\mathbf{2EXPTIME}$. Thus, Figure 7.11 is filled mostly with complexity classes that are generally considered intractable.

But what does a hardness result really tell us? It shows that there is no general algorithm, efficiently solving the problem at hand. Yet one is often not interested in verification of all possible specifications in all possible models. This raises the question: *Which interesting fragments of the model checking problem have manageable complexity?* Answering the question seems crucial for the future of verification for multi-agent systems.

References and Further Reading. Our exposition of the verification complexity for different variants of ATL follows mostly [33]. Details of the technical results can be found in [108, 97, 31, 92].

7.3 Taming the Complexity by Reinstating Fixpoints

An important feature that distinguishes variants of $\mathbf{ATL}$ for imperfect information scenarios is that the standard fixpoint characterizations of temporal modalities do not hold anymore. In this section, we show that adding explicit fixpoint operators to the "nexttime" fragment of $\mathbf{ATL}$ allows to capture abilities that cannot be expressed in the variants of $\mathbf{ATL}$ that have been considered so far. The new language enables specification of an important kind of abilities, namely ones where agents can always recompute their strategy while executing it. Thus, the agents are not assumed to remember their strategy by definition, like in the previous variants. Even more importantly, verification of such abilities can be cheaper than for all the other variants of "**ATL** with imperfect information."

7.3.1 Alternating Epistemic μ-Calculus

One of the features that distinguish abilities under imperfect information from the perfect information setting is that the fixpoint characterizations of strategic/temporal modalities do not hold. On the conceptual level, this means that having a strategy to achieve γ does not mean that the agents will be able to recompute the strategy when they are executing it. Thus, the semantics includes an implicit assumption that the agents *can remember the strategy which they are executing* even if they forget everything else.

This can be a good thing or a bad thing, depending on the notion of ability that one wants to formalize. Nevertheless, the other kind of ability (enforcing γ without resorting to additional memory of the strategy) is equally important. It captures the idea of agents "*persistently knowing how to play to enforce* γ", i.e., so that they can come up with the right strategy not only at the beginning of the game; they will *know* how to recreate the strategy also in any future moment of the play. Moreover, this kind of ability has a minimalistic flavor regarding epistemic prerequisites: agents are supposed to achieve γ while resorting *only* to observations that they can make along the way.

How can we reason about such recomputable strategies? Instead of considering yet another semantics of strategic modalities, it suffices to add explicit fixpoint operators to the "nexttime" fragment of **ATL**. As it turns out, the class of abilities obtained this way cannot be expressed in the previously presented variants of **ATL**. Moreover, recomputable strategies can be (by definition) synthesized incrementally, which makes the synthesis tractable–at least for small coalitions.

Syntax and Semantics

Consider alternating μ-calculus, introduced in Section 4.3.3 for reasoning about fixpoint properties of agents with perfect information. The variant of **AMC** for imperfect information can be defined in a straightforward way. The main change concerns the semantic clause for operator $\langle\!\langle A\rangle\!\rangle \mathrm{X}$. Now, $\langle\!\langle A\rangle\!\rangle \mathrm{X}\varphi$ is only true if agents A have a uniform one-step strategy which enforces φ from all the states that are indistinguishable from the current one. We also add epistemic operators K_a to the language, since they cannot be expressed in the "nexttime" fragment of $\mathbf{ATL}_{\mathrm{ir}}$. This way, we obtain the *alternation-free alternating epistemic μ-calculus* (**AEMC** in short), given formally as follows:

$$\varphi ::= \mathsf{p} \mid Z \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A\rangle\!\rangle \mathrm{X}\varphi \mid \mu Z(\varphi) \mid \mathrm{K}_a\varphi$$

where $A \subseteq \mathbb{A}\mathrm{gt}$, $a \in \mathbb{A}\mathrm{gt}$, $\mathsf{p} \in \mathcal{PV}$, $Z \in FV$ and each free occurrence of Z in φ is under the scope of an even number of negations in φ. The greatest fixpoint operator $\nu X(\varphi)$ is defined as before. In fact, we will only look at the *alternation-free* part of the language , i.e., the formulae which, in their negation normal form, do not contain both μ and ν on any syntactic path from $\mu/\nu X$ to a bound occurrence of X.

The denotational semantics of **AEMC** takes the semantics of alternating μ-calculus, and updates it as follows:

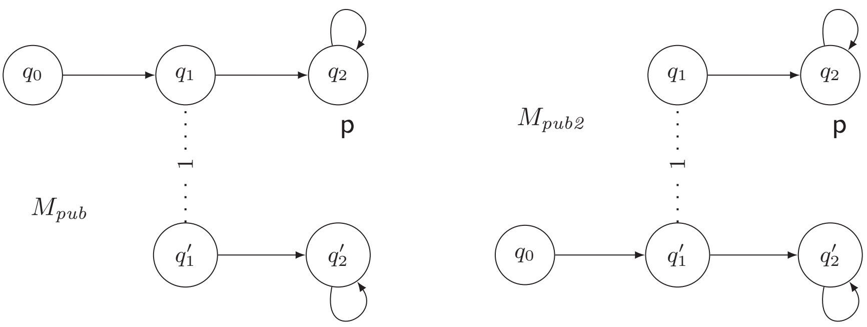


Figure 7.12: Models M_{pub} and M_{pub2}, each with a single agent 1

$$\begin{aligned}
[\![\langle\!\langle A\rangle\!\rangle \mathrm{X}\varphi]\!]_{M,\mathcal{E}} &= \{q \mid \exists s_A \in \Sigma_A^{ir}\ \forall q' \in [q]_{\sim_A^E}\ \forall\lambda \in out(q', s_A)\ . \\
&\qquad \lambda[1] \in [\![\varphi]\!]_{M,\mathcal{E}}\}, \\
[\![\mathrm{K}_a\varphi]\!]_{M,\mathcal{E}} &= \{q \mid \forall q'\ .\ q \sim_a q' \text{ implies } q' \in [\![\varphi]\!]_{M,\mathcal{E}}\}.
\end{aligned}$$

In other words, $q \in [\![\langle\!\langle A\rangle\!\rangle \mathrm{X}\varphi]\!]_{M,\mathcal{E}}$ iff agents A have a collective uniform strategy such that everybody in the group knows that it enforces φ in the next step. We write $M, q \models_i \varphi$ iff $q \in [\![\varphi]\!]_{M,\mathcal{E}}$ for all valuations $\mathcal{E}$ of fixpoint variables in FV. Notice the subscript i indicating that the semantics of $\langle\!\langle A\rangle\!\rangle \mathrm{X}$ is based on imperfect information strategies.

Example 63 *Consider the concurrent epistemic game model M_{pub} in Figure 7.12. The story is as follows: A married man is sitting in a pub drinking with his friends (q_0). In order to get back to his wife (q_2), he needs to finish his drinking session first (q_1), but that will result in a temporary lapse of memory. In particular, the man will have a nagging feeling that something might be wrong with his marriage, e.g., his wife could have left him because of his drinking habits (q_1'), in which case he can only come back to an empty house (q_2').*[1]

Obviously, we have that $M_{pub}, q_0 \models_{\mathrm{ir}} \langle\!\langle 1\rangle\!\rangle \mathrm{F}\mathsf{p}$, so the man should rest assured. However, is it really the property he is after? He knows now *that everything will be fine, but he also knows that he will get confused on the way, which may prevent him from reaching his goal. The stronger, more persistent kind of ability is captured by the* **AEMC** *formula $\mu Z(\mathsf{p} \vee \langle\!\langle 1\rangle\!\rangle \mathrm{X}Z)$, which does not hold in M_{pub}, q_0.*

The right hand side of Figure 7.12 presents a slightly different model M_{pub2} in which the man's marriage is indeed collapsing. Like in M_{pub}, we have $M_{pub2}, q_0 \not\models_i \mu Z(\mathsf{p} \vee \langle\!\langle 1\rangle\!\rangle \mathrm{X}Z)$, but this time also $M_{pub2}, q_0 \not\models_{\mathrm{ir}} \langle\!\langle 1\rangle\!\rangle \mathrm{F}\mathsf{p}$.

The example suggests that $\mathbf{ATL}_{\mathrm{ir}}$ and **AEMC** capture *different types* of strategic ability. We analyse the new kind of ability in the next subsection.

7.3.2 Specification of Fixpoint Abilities

Alternating epistemic μ-calculus can be useful for specification and verification of persistent strategic abilities. In the subsequent paragraphs, we show fixpoint properties

[1] For similar stories in serious literature on interpretation of game models, see [136, 135].

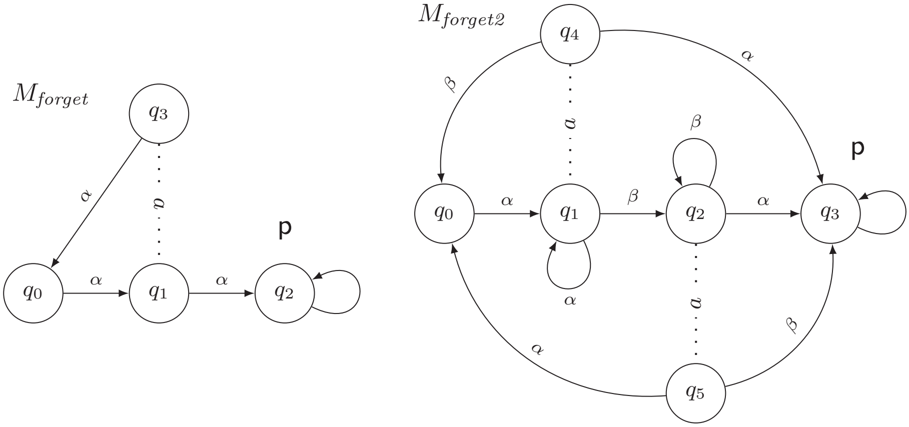


Figure 7.13: Models M_{forget} and $M_{forget2}$: inconveniences of forgetting

that formally capture the informal intuition.

Strategic Fixpoints: Achievement

Formulae of **AEMC** enable expressing that agents A have a strategy to enforce a temporal property γ while knowing how to play *all along the game*. For achievement properties, we have typically $\gamma \equiv \mathrm{F}\varphi$. We note that the ability to eventually achieve φ while knowing how to play all along can have at least two meaningful interpretations:

1. Agents A have a strategy which they know to achieve φ, can be recomputed along the execution, and guarantees that they will know when φ has been achieved;

2. The agents have a recomputable strategy that they know to achieve φ, but they will not necessarily know when φ is achieved.

For a single agent the first achievement ability corresponds closely to the **AEMC**-formula:

$$\mu Z(\mathrm{K}_a\varphi \vee \langle\langle a\rangle\rangle \mathrm{X}Z).$$

Analogously, we can describe the second interpretation by:

$$\mu Z(\varphi \vee \langle\langle a\rangle\rangle \mathrm{X}Z).$$

We observe that these formulae (and formulae of **AEMC** in general) have a strong "constructive" flavor. In particular, they require not only that a has a uniform strategy to eventually obtain φ, but also that:

- she can *recompute* the strategy at *any* future state resulting from executing it,

- she *knows* about that, and furthermore

- with every step, a knows that the goal is closer, and hence the part of the strategy that she needs to recompute becomes smaller.

This is a stronger kind of ability than those typically expressed in $\mathbf{ATL}_{\mathrm{ir}}$. For illustration, consider model M_{forget} in Figure 7.13. Agent a has a uniform strategy to achieve p eventually: $M_{forget}, q_0 \models_{\mathrm{ir}} \langle\langle a\rangle\rangle \mathrm{F}\mathsf{p}$ (the strategy is to play α everywhere). It is even the case that a can recompute the successful strategy on the way: $M_{forget}, q_0 \models_{\mathrm{ir}} \langle\langle a\rangle\rangle\big(\langle\langle a\rangle\rangle \mathrm{F}\mathsf{p}\big) \,\mathrm{U}\, \mathsf{p}$, which is trivial as only one action is available throughout the model. However, the corresponding $\mathbf{AEMC}$ specifications do not hold: $M_{forget}, q_0 \not\models_i \mu Z(\mathsf{p} \lor \langle\langle a\rangle\rangle \mathrm{X}Z)$ and $M_{forget}, q_0 \not\models_i \mu Z(\mathrm{K}_a\mathsf{p} \lor \langle\langle a\rangle\rangle \mathrm{X}Z)$. This is because a is never certain that the goal state is really approaching. More precisely, suppose that a executes action α in state q_0. In the next state, q_1, she executes α again. However, as she considers q_3 possible when in q_1, the resulting state might – from her perspective – also be q_0. Applying this reasoning iteratively, we get $q_0q_3(q_0q_3)^\omega$ as one of possible execution paths, and it never satisfies p. Thus, the agent does not know that she has a recomputable strategy to achieve p.

Strategic Fixpoints: Maintenance

Now we turn to maintenance goals, typically expressed by $\gamma \equiv \mathrm{G}\varphi$. Like for achievement properties, the ability to maintain φ while knowing how to play all along can have at least two interpretations:

1. Agents A have a strategy to maintain both the truth of φ and the knowledge of φ, and the strategy can be recomputed at every step.
2. The agents have a recomputable strategy to maintain the truth of φ (but they may become unsure if φ is true at some steps).

Similarly to achievement properties, we can capture the first interpretation by

$$\nu Z(\mathrm{K}_a\varphi \land \langle\langle a\rangle\rangle \mathrm{X}Z).$$

The second interpretation can be characterized analogously by formula:

$$\nu Z(\varphi \land \langle\langle a\rangle\rangle \mathrm{X}Z).$$

For maintenance goals, the difference between $\mathbf{ATL}_{\mathrm{ir}}$ and $\mathbf{AEMC}$ specifications is even clearer. Consider model $M_{forget2}$ in Figure 7.13. We have $M_{forget2}, q_0 \models_{\mathrm{ir}} \langle\langle a\rangle\rangle \mathrm{G}\neg\mathsf{p}$ which expresses that a can avoid p. For example, take the strategy in which α is played everywhere. We even have $M_{forget2}, q_0 \models_{\mathrm{ir}} \langle\langle a\rangle\rangle \mathrm{G}\big(\neg\mathsf{p} \land \langle\langle a\rangle\rangle \mathrm{G}\neg\mathsf{p}\big)$: while avoiding p, the agent can recompute a successful strategy. However, at q_1, she can only come up with a strategy that is different from the original strategy (i.e., play β everywhere), and indeed $M_{forget2}, q_0 \not\models_i \nu Z(\neg\mathsf{p} \land \langle\langle a\rangle\rangle \mathrm{X}Z)$ as well as $M_{forget2}, q_0 \not\models_i \nu Z(\mathrm{K}_a\neg\mathsf{p} \land \langle\langle a\rangle\rangle \mathrm{X}Z)$.

Combinations of achievement and maintenance, captured by the "until" operator U, can be treated analogously.

7.3.3 Expressive Power of AEMC

We have already indicated that $\mathbf{AEMC}$ captures a different kind of ability than $\mathbf{ATL}_{\mathrm{ir}}$ and $\mathbf{ATL}_{\mathrm{iR}}$. This can be shown formally by comparing their expressivity.

Definition 15 (Expressive and distinguishing power) *Let $L_1 = (\mathcal{L}_1, \models_1)$ and $L_2 = (\mathcal{L}_2, \models_2)$ be two logical systems over the same class of models $\mathcal{M}$. By $[\![\phi]\!]_{\models} =$*

$\{(M,q) \mid M,q \models \phi\}$, *we denote the class of pointed models that satisfy* ϕ *according to the semantics given by* $\models$. *Likewise,* $[\![\phi, M]\!]_{\models} = \{q \mid M,q \models \phi\}$ *is the set of states (or, equivalently, pointed models) that satisfy* ϕ *in a given structure* M.

L_2 *is* at least as expressive *as* L_1 *iff for every formula* $\phi_1 \in \mathcal{L}_1$ *there exists* $\phi_2 \in \mathcal{L}_2$ *such that* $[\![\phi_1]\!]_{\models_1} = [\![\phi_2]\!]_{\models_2}$.

It is well known that alternating μ-calculus is strictly more expressive than $\mathbf{ATL}$ and $\mathbf{ATL}^*$ in the perfect information setting. We start our analysis of fixpoints in imperfect information scenarios by showing that, analogously, $\mathbf{AEMC}$ allows to capture properties that cannot be expressed in $\mathbf{ATL}_{\mathrm{ir}}$.

Theorem 46 $\mathbf{ATL}_{\mathrm{ir}}$ *is not as expressive as* $\mathbf{AEMC}$ *over the class of CEGM's.*

Proof sketch. Alternation-free modal μ-calculus is strictly more expressive than $\mathbf{CTL}$ over the class of Kripke models. Now suppose that $\mathbf{ATL}_{\mathrm{ir}}$ is at least as expressive as $\mathbf{AEMC}$ over the class of CEGM's. Then, $\mathbf{ATL}_{\mathrm{ir}}$ would also be at least as expressive as $\mathbf{AEMC}$ over Kripke models, as Kripke models can be seen as single-agent CEGM's with perfect information. Thus, $\mathbf{CTL}$ would be at least as expressive as alternation-free μ-calculus over the class of Kripke models, which is a contradiction.

Note that epistemic operators are not used in the proof, so they are *not* the reason for the expressivity gap. ■

It also turns out, rather unexpectedly, that $\mathbf{AEMC}$ does not cover the whole expressivity of $\mathbf{ATL}_{\mathrm{ir}}$. The main idea is to prove that the pointed models (M_{pub}, q_0) and (M_{pub2}, q_0) in Figure 7.12 satisfy exactly the same formulae of $\mathbf{AEMC}$, but they are distinguished by the $\mathbf{ATL}_{\mathrm{ir}}$ formula $\langle\!\langle 1 \rangle\!\rangle \mathrm{F}\mathsf{p}$.

Theorem 47 $\mathbf{AEMC}$ *is not as expressive as* $\mathbf{ATL}_{\mathrm{ir}}$ *over the class of CEGM's.*

Corollary 2 $\mathbf{AEMC}$ *is incomparable to* $\mathbf{ATL}_{\mathrm{ir}}$ *in expressive power over the class of CEGM's.*

How does the picture change if agents have memory? Let us compare $\mathbf{ATL}_{\mathrm{iR}}$ and $\mathbf{AEMC}$ (recall that it does not make sense to consider memory-based strategies in $\mathbf{AEMC}$ since the strategic modalities of $\mathbf{AEMC}$ refer only to one-step abilities). Clearly, the analogue of Theorem 47 holds for $\mathbf{ATL}_{\mathrm{iR}}$ by the same argument. But also analogue of Theorem 46 holds, since $\mathbf{ATL}_{\mathrm{iR}}$ and $\mathbf{ATL}_{\mathrm{ir}}$ are equally expressive over perfect information CEGM's. Thus, if the claim were not true, then $\mathbf{ATL}_{\mathrm{iR}}$ would also be at least as expressive as $\mathbf{AEMC}$ over Kripke models. As before this yields a contradiction, since alternation-free μ-calculus is strictly more expressive than $\mathbf{CTL}$.

Corollary 3 $\mathbf{AEMC}$ *is incomparable to* $\mathbf{ATL}_{\mathrm{iR}}$ *in expressive power over the class of CEGM's.*

7.3.4 Complexity of Model Checking

$\mathbf{AEMC}$ allows to specify interesting properties that cannot be expressed in the "standard" variants of $\mathbf{ATL}$ with imperfect information. A natural question arises: How costly is the verification of those properties? We recall that:

- Model checking of strategic logics with perfect information ($\mathbf{ATL}_{\mathrm{Ir}}$, $\mathbf{ATL}_{\mathrm{IR}}$, alternation-free $\mathbf{AMC}$) is in $\mathbf{P}$;

- Verification of $\mathbf{ATL}_{\mathrm{ir}}$ (imperfect information and memoryless strategies) is $\mathbf{\Delta_2^P}$-complete even for turn-based systems and singleton coalitions;
- Model checking $\mathbf{ATL}_{\mathrm{iR}}$ (imperfect information, memory-based strategies) is undecidable.

It turns out that model checking of alternating epistemic μ-calculus is in $\mathbf{P}$ for coalitions consisting of at most two agents (in a system including arbitrarily many agents). Moreover, the problem is between $\mathbf{NP}$ and $\mathbf{\Delta_2^P}$ if larger coalitions are involved. Thus, verification with $\mathbf{AEMC}$ is distinctly cheaper than with $\mathbf{ATL}$ for abilities of small coalitions, and no harder in the general case. To see this, we point out that the μ-operator in $\mathbf{AEMC}$ is a standard least fixpoint operator. Thus, the crucial part of model checking is the computation of the preimage for $\langle\langle A\rangle\rangle\mathrm{X}$ (i.e., the set of states satisfying $\langle\langle A\rangle\rangle\mathrm{X}\mathsf{p}$ for a given proposition p). Then, an arbitrary formula can be verified through a polynomial number of calls to preimage computations within the standard iterative algorithm (cf. Figure 5.1). The complexity of the preimage computation was already established in Theorems 35 and 36. Thus, we obtain the following.

Theorem 48 *Model checking* $\mathbf{AEMC}$ *for the formulae that include only operators* $\langle\langle A\rangle\rangle\mathrm{X}$ *with* $|A| \leq 2$ *can be performed in polynomial time with respect to the number of the transitions in the model and the length of the formula.*

For arbitrary formulae, the problem is between $\mathbf{NP}$ *and* $\mathbf{\Delta_2^P}$*. We conjecture that it is actually* $\mathbf{\Delta_2^P}$*-complete, but this has yet to be formally proved.*

Summary

The most important result for $\mathbf{AEMC}$ is that it admits polynomial-time verification for formulae involving only small coalitions. The logic offers specifications for an intuitive class of strategic properties ("persistent knowing how to play") which neither $\mathbf{ATL}_{\mathrm{ir}}$ nor $\mathbf{ATL}_{\mathrm{iR}}$ can capture. In fact, we argue that $\mathbf{AEMC}$ formalizes Moore's fixpoint concept of knowledge-based ability that was discussed in Section 6.1.

At the same time, the logic has tractable model checking – though in a somewhat limited scope. This is significant, both theoretically and in practical terms. Theoretically, $\mathbf{AEMC}$ is the first strategic logic of imperfect information with tractable verification for a relevant subset of formulae. Even solving 2-player extensive form games with binary payoffs is $\mathbf{NP}$-complete in the sense of surely winning. Model checking of all existing variants of $\mathbf{ATL}$ with imperfect information is at least $\mathbf{NP}$-hard, and this applies even to verification of 1-player(!) turn-based CEGM's.

Practically, verification of abilities for small coalitions is useful in a number of important scenarios, including fault-tolerance of communication protocols, correctness of security protocols, etc. A typical property in such scenarios is "Alice and Bob can communicate/interact in such a way that the correctness property γ is guaranteed," where γ can for instance refer to inability of the eavesdropper to intercept the message. Note that the model can include an arbitrary number of eavesdroppers, intruders, and neutral agents.

For larger groups of agents, model checking $\mathbf{AEMC}$ is in the same complexity class as for $\mathbf{ATL}_{\mathrm{ir}}$. However, the scope of backtracking for $\mathbf{AEMC}$ is limited to information sets of the coalition, whereas the straightforward verification algorithm for $\mathbf{ATL}_{\mathrm{ir}}$ searches through the set of complete strategies. In consequence, model checking $\mathbf{AEMC}$ can be exponentially faster than $\mathbf{ATL}_{\mathrm{ir}}$ for models that include reasonably small indistinguishability classes.

References and Further Reading. The idea of **AEMC**, as well as the technical results, were proposed in [38]. Other works on recomputable abilities under imperfect information include attempts at more refined fixpoint charcterizations of strategic modalities [54, 19] and invariance under bisimulation of abilities in 2-player reachability games with perfect recall [22].

Chapter 8

Conclusions

In this book, we have surveyed several important logics for specification of multi-agent systems, and the basic approaches to automatic verification for MAS. We paired specification languages with complexity results for the corresponding model checking problems. We also demonstrated how the complexity changes when we change the representation of systems. Many examples were shown of how to represent important properties of agents, and how the model checking algorithms proceed on concrete instances of the problem.

We would like to emphasize that specification and verification of multi agent systems is a very rapidly developing area of computer science. Therefore, even at the time of writing this text, new methods are emerging. In order to become acquainted with the latest developments, we refer the interested reader to the proceedings of conferences such as AAMAS, IJCAI, AAAI, ECAI, SR, LAMAS, ICSOC, ICEFM, MoChart, ATVA, etc. Below we hesitantly list some open problems within the area, expecting that – at the time this book is read – some of them may have been already solved, and certainly many more research questions will have emerged:

1. Logics for specification of MAS with perfect information are relatively well studied. However, many issues regarding modeling, analysis, and reasoning about systems with incomplete information are barely touched. This regards e.g. axiomatization of **ATL** with imperfect information (some preliminary results were obtained in [74] but the research is far from complete). Moreover, to our best knowledge, no satisfiability-checking algorithms for any variant of **ATL** with imperfect information have been proposed. It is not even known whether the problem is decidable at all.

2. Model checking of strategies with incomplete information is largely left untouched, besides the theoretical complexity results that we presented in Chapter 7. This is somewhat easy to understand, since the theoretical results are quite pessimistic. Moreover, the imperfect information semantics of **ATL** does not admit fixpoint equivalences, which makes incremental synthesis of strategies impossible, or at least cumbersome. Some practical attempts at tackling the problem started to emerge only very recently [137, 43, 86]. Up until now, experimental results confirm that the initial intuition was right: model checking strategic modalities for imperfect information is hard, and dealing with it requires innovative algorithms and verification techniques.

3. The coordination issue: the ATL formula $\langle\langle A\rangle\rangle\gamma$ only requires that there *exists* a winning strategy for A to achieve γ. However, the agents in A may not be able to successfully coordinate if there are multiple joint strategies with that property. Some work on this issue was reported in [81] and [175], but the research is still preliminary.

4. Correspondence between abstract and concrete models of agents' strategic play is becoming to attract interest again. The issue is relatively well studied for perfect information games, both one-step (cf. [129, 69]) and multi-step [68]. On the other hand, no correspondence result exists for abilities under imperfect information.

5. The connection between strategic logics and mainstream game theory is still weak in the sense that game theory uses much more sophisticated models of agents' incentives and concepts of agents' rationality. There have been several attempts at characterizing game-theoretic solution concepts in modal logics (e.g., [78, 77, 167, 181, 40, 162, 185]) but none of them match the elegance and simplicity of preference models and solution concepts in game theory.

The Road goes ever on and on... We may even meet somewhere around the bend. 🙂

Bibliography

[1] J. Abdou and H. Keiding. *Effectivity Functions in Social Choice*. Springer, 1991.

[2] T. Ågotnes. A note on syntactic characterization of incomplete information in ATEL. In *Procedings of Workshop on Knowledge and Games*, pages 34–42, 2004.

[3] T. Ågotnes. Action and knowledge in alternating-time temporal logic. *Synthese*, 149(2):377–409, 2006. Section on Knowledge, Rationality and Action.

[4] T. Ågotnes, V. Goranko, and W. Jamroga. Alternating-time temporal logics with irrevocable strategies. In *Proceedings of TARK XI*, pages 15–24, 2007.

[5] T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge. Knowledge and ability. In H.P. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B.P. Kooi, editors, *Handbook of Epistemic Logic*, pages 543–589. College Publications, 2015.

[6] T. Ågotnes and D. Walther. A logic of strategic ability under bounded memory. *Journal of Logic, Language and Information*, 18(1):55–77, 2009.

[7] E.A. Akkoyunlu, K. Ekanadham, and R.V. Huber. Some constraints and tradeoffs in the design of network communications. 1975.

[8] N. Alechina, B. Logan, N.H. Nga, and A. Rakib. A logic for coalitions with bounded resources. In *Proceedings of IJCAI*, pages 659–664, 2009.

[9] N. Alechina, B. Logan, H.N. Nguyen, and A. Rakib. Resource-bounded alternating-time temporal logic. In *Proceedings of AAMAS*, pages 481–488, 2010.

[10] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997.

[11] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Lecture Notes in Computer Science*, 1536:23–60, 1998.

[12] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[13] R. Alur, T.A. Henzinger, O. Kupferman, and M.Y. Vardi. Alternating refinement relations. In *Proceedings of CONCUR*, volume 1466 of *LNCS*, pages 163–178, 1998.

[14] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[15] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[16] T. Ball and O. Kupferman. An abstraction-refinement framework for multi-agent systems. In *Proceedings of LICS*, pages 379–388. IEEE Computer Society, 2006.

[17] A. Baltag. A logic for suspicious players. *Bulletin of Economic Research*, 54(1):1–46, 2002.

[18] C. Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5(3):241–259, 1980.

[19] F. Belardinelli. A logic of knowledge and strategies with imperfect information. In *Proceedings of LAMAS*. IFAAMAS, 2015.

[20] N. Belnap and M. Perloff. Seeing to it that: a canonical form for agentives. *Theoria*, 54:175–199, 1988.

[21] N. Belnap, M. Perloff, and M. Xu. *Facing the future: agents and choices in our indeterminist world*. Oxford, 2001.

[22] D. Berwanger and L. Kaiser. Information tracking in games on graphs. *Journal of Logic, Language and Information*, 19(4):395–412, 2010.

[23] P. Blackburn, M. de Rijke, and V. Venema. *Modal Logic*. Cambridge Univ. Press, 2001.

[24] E. Boros, K. Elbassioni, V. Gurvich, and K. Makino. On effectivity functions of game forms. *Games and Economic Behavior*, 68(2):512–531, 2010.

[25] T. Brihaye, A. Da Costa Lopes, F. Laroussinie, and N. Markey. ATL with strategy contexts and bounded memory. In *Proceedings of LFCS*, volume 5407 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2009.

[26] J. Broersen. CTL.STIT: enhancing ATL to express important multi-agent system verification properties. In *Proceedings of (AAMAS*.

[27] J. Broersen. A stit-logic for extensive form group strategies. In *Proceedings of Web Intelligence and Intelligent Agent Technology – Workshops*, pages 484–487. IEEE Computer Society, 2009.

[28] J. Broersen, A. Herzig, and N. Troquard. Embedding Alternating-time Temporal Logic in Strategic STIT logic of agency. *Journal of Logic and Computation*, 16(5):559–578, 2006.

[29] J. Broersen, A. Herzig, and N. Troquard. A STIT-extension of ATL. In *Proceedings of JELIA*, pages 69–81, 2006.

[30] M.A. Brown. On the logic of ability. *Journal of Philosophical Logic*, 17:1–26, 1988.

[31] N. Bulling. Model checking coalition logic on implicit models is Δ_3-complete. Technical Report IfI-10-02, Clausthal University of Technology, 2010.

[32] N. Bulling and J. Dix. Modelling and verifying coalitions using argumentation and ATL. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 14(46):45–73, 2010.

[33] N. Bulling, J. Dix, and W. Jamroga. Model checking logics of strategic ability: Complexity. In M. Dastani, K. Hindriks, and J.-J. Meyer, editors, *Specification and Verification of Multi-Agent Systems*, pages 125–159. Springer, 2010.

[34] N. Bulling and B. Farwer. Expressing properties of resource-bounded systems: The logics RTL* and RTL. In *Proceedings of CLIMA*, volume 6214 of *Lecture Notes in Computer Science*, pages 22–45, 2010.

[35] N. Bulling and B. Farwer. On the (un-)decidability of model checking resource-bounded agents. In *Proceedings of ECAI*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 567–572. IOS Press, 2010.

[36] N. Bulling, V. Goranko, and W. Jamroga. Logics for reasoning about strategic abilities in multi-player games. In J. van Benthem, S. Ghosh, and R. Verbrugge, editors, *Models of Strategic Reasoning. Logics, Games, and Communities*, volume 8972 of *LNCS*, pages 93–136. Springer, 2015.

[37] N. Bulling and W. Jamroga. What agents can probably enforce. *Fundamenta Informaticae*, 93(1-3):81–96, 2009.

[38] N. Bulling and W. Jamroga. Alternating epistemic mu-calculus. In *Proceedings of IJCAI-11*, pages 109–114, 2011.

[39] N. Bulling and W. Jamroga. Comparing variants of strategic ability: How uncertainty and memory influence general properties of games. *Journal of Autonomous Agents and Multi-Agent Systems*, 28(3):474–518, 2014.

[40] N. Bulling, W. Jamroga, and J. Dix. Reasoning about temporal properties of rational play. *Annals of Mathematics and Artificial Intelligence*, 53(1-4):51–114, 2008.

[41] N. Bulling, W. Jamroga, and M. Popovici. Agents with truly perfect recall in alternating-time temporal logic (extended abstract). In *Proceedings of AAMAS*, pages 1561–1562, 2013.

[42] N. Bulling, W. Jamroga, and M. Popovici. Agents with truly perfect recall: Expressivity and validities. In *Proceedings of ECAI*, pages 177–182, 2014.

[43] S. Busard, C. Pecheur, H. Qu, and F. Raimondi. Improving the model checking of strategies under partial observability and fairness constraints. In *Formal Methods and Software Engineering*, volume 8829 of *Lecture Notes in Computer Science*, pages 27–42. 2014.

[44] K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. In *Proceedings of CONCUR*, pages 59–73, 2007.

[45] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.

[46] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[47] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[48] P.R. Cohen and H.J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[49] Mads Dam. CTL* and ECTL* as fragments of the modal mu-calculus. *Theoretical Computer Science*, 126(1):77–96, 1994.

[50] M. Dastani, K. Hindriks, and J.-J. Meyer, editors. *Specification and Verification of Multi-Agent Systems*. Springer, 2010.

[51] L. de Alfaro, M. Faella, T.A. Henzinger, R. Majumdar, and M. Stoelinga. Model checking discounted temporal properties. *Theoretical Computer Science*, 345:139–170, 2005.

[52] R. Diaconu and C. Dima. Model-checking alternating-time temporal logic with strategies based on common knowledge is undecidable. *Applied Artificial Intelligence*, 26(4):331–348, 2012.

[53] C. Dima, C. Enea, and D.P. Guelev. Model-checking an alternating-time temporal logic with knowledge, imperfect information, perfect recall and communicating coalitions. In *Proceedings of GANDALF*, pages 103–117, 2010.

[54] C. Dima, B. Maubert, and S. Pinchinat. Relating paths in transition systems: The fall of the modal mu-calculus. In *Proceedings of MFCS*, volume 9234 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2015.

[55] C. Dima and F.L. Tiplea. Model-checking atl under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[56] L. Doyen and J.-F. Raskin. Games with imperfect information: Theory and algorithms. In *Lecture Notes in Game Theory for Computer Scientists*, pages 185–212. Cambridge University Press, 2011.

[57] E. A. Emerson and C-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *Proc. of the 1st Symp. on Logic in Computer Science (LICS'86)*, pages 267–278. IEEE Computer Society, 1986.

[58] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[59] E.A. Emerson and E.M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Automata, Languages and Programming*, volume 85 of *Lecture Notes in Computer Science*, pages 169–181, 1980.

[60] E.A. Emerson and J.Y. Halpern. ”sometimes” and ”not never” revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[61] E.A. Emerson and Ch.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.

[62] E.A. Emerson and A.P. Sistla. Deciding branching time logic. In *STOC ’84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 14–24, New York, NY, USA, 1984. ACM.

[63] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[64] M. Fisher. *Temporal Logics*. Kluwer, 2006.

[65] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Systems Theory*, 17:13–27, 1984.

[66] V. Goranko. Coalition games and alternating temporal logics. In J. van Benthem, editor, *Proceedings of TARK VIII*, pages 259–272. Morgan Kaufmann, 2001.

[67] V. Goranko and W. Jamroga. Comparing semantics of logics for multi-agent systems. *Synthese*, 139(2):241–280, 2004.

[68] V. Goranko and W. Jamroga. State and path coalition effectivity models of concurrent multi-player games. *Autonomous Agents and Multi-Agent Systems*, pages 1–40, 2015.

[69] V. Goranko, W. Jamroga, and P. Turrini. Strategic games and truly playable effectivity functions. In *Proceedings of AAMAS2011*, pages 727–734, 2011.

[70] V. Goranko, W. Jamroga, and P. Turrini. Strategic games and truly playable effectivity functions. *Journal of Autonomous Agents and Multi-Agent Systems*, 26(2):288–314, 2013.

[71] V. Goranko and D. Shkatov. Tableau-based decision procedures for logics of strategic ability in multiagent systems. *ACM Trans. Comput. Log.*, 11(1), 2009.

[72] V. Goranko and G. van Drimmelen. Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science*, 353(1):93–117, 2006.

[73] D.P. Guelev and C. Dima. Model-checking strategic ability and knowledge of the past of communicating coalitions. In *Proceedings of DALT*, pages 75–90, 2008.

[74] D.P. Guelev, C. Dima, and C. Enea. An alternating-time temporal logic with knowledge, perfect recall and past: axiomatisation and model-checking. *Journal of Applied Non-Classical Logics*, 21(1):93–131, 2011.

[75] J.Y. Halpern. Reasoning about knowledge: a survey. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming. Vol. 4: Epistemic and Temporal Reasoning*, pages 1–34. Oxford University Press, Oxford, 1995.

[76] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[77] B.P. Harrenstein, W. van der Hoek, J.-J. Meyer, and C. Witteveen. A modal characterization of Nash equilibrium. *Fundamenta Informaticae*, 57(2–4):281–321, 2003.

[78] P. Harrenstein, W. van der Hoek, J-J. Meijer, and C. Witteveen. Subgame-perfect Nash equilibria in dynamic logic. In M. Pauly and A. Baltag, editors, *Proceedings of the ILLC Workshop on Logic and Games*, pages 29–30. University of Amsterdam, 2002. Tech. Report PP-1999-25.

[79] P. Harrenstein, W. van der Hoek, J.-J.Ch. Meyer, and C. Witteveen. On modal logic interpretations of games. In *Proceedings of ECAI*, pages 28–32. IOS Press, 2002.

[80] S. Hart. Games in extensive and strategic forms. In R.J. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications, Volume 1*, pages 19–40. Elsevier/North-Holland, 1992.

[81] P. Hawke. Coordination, almost perfect information and strategic ability. In *Proceedings of LAMAS*, 2010.

[82] A. Herzig and N. Troquard. Knowing how to play: Uniform choices in logics of agency. In *Proceedings of AAMAS'06*, pages 209–216, 2006.

[83] J. F. Horty. *Agency and Deontic Logic*. Oxford University Press, 2001.

[84] J.F. Horty and N. Belnap. The deliberative stit: A study of action, omission, ability, and obligation. *Journal of Philosophical Logic*, 24:583–644, 1995.

[85] X. Huang, K. Su, and C. Zhang. Probabilistic alternating-time temporal logic of incomplete information and synchronous perfect recall. In *Proceedings of AAAI-12*, 2012.

[86] X. Huang and R. van der Meyden. Symbolic model checking epistemic strategy logic. In *Proceedings of AAAI*, pages 1426–1432, 2014.

[87] M. Huth and M. Ryan. *Logic in Computer Science: Modeling and Reasoning about Systems*. Cambridge University Press, 2000.

[88] N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.

[89] W. Jamroga. Some remarks on alternating temporal epistemic logic. In B. Dunin-Keplicz and R. Verbrugge, editors, *Proceedings of Formal Approaches to Multi-Agent Systems (FAMAS 2003)*, pages 133–140, 2003.

[90] W. Jamroga. On the relationship between playing rationally and knowing how to play: A logical account. In C. Freksa, M. Kohlhase, and K. Schill, editors, *Proceedings of KI 2006*, volume 4314 of *Lecture Notes in Artificial Intelligence*, pages 419–433. Springer Verlag, 2006.

[91] W. Jamroga. A temporal logic for stochastic multi-agent systems. In *Proceedings of PRIMA'08*, volume 5357 of *LNCS*, pages 239–250, 2008.

[92] W. Jamroga and T. Ågotnes. Modular interpreted systems: A preliminary report. Technical Report IfI-06-15, Clausthal University of Technology, 2006.

[93] W. Jamroga and T. Ågotnes. Constructive knowledge: What agents can achieve under incomplete information. *Journal of Applied Non-Classical Logics*, 17(4):423–475, 2007.

[94] W. Jamroga and T. Ågotnes. Modular interpreted systems. In *Proceedings of AAMAS'07*, pages 892–899, 2007.

[95] W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In M. Pěchouček, P. Petta, and L.Z. Varga, editors, *Proceedings of CEEMAS 2005*, volume 3690 of *Lecture Notes in Computer Science*, pages 398–407. Springer Verlag, 2005.

[96] W. Jamroga and J. Dix. Model checking ATL$_{ir}$ is indeed Δ_2^P-complete. In *Proceedings of EUMAS'06*, 2006.

[97] W. Jamroga and J. Dix. Model checking abilities of agents: A closer look. *Theory of Computing Systems*, 42(3):366–410, 2008.

[98] W. Jamroga and W. Penczek. Specification and verification of multi-agent systems. In N. Bezhanishvili and V. Goranko, editors, *Lectures on Logic and Computation*, volume 7388 of *LNCS*, pages 210–263. Springer, 2012.

[99] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3):185–219, 2004.

[100] W. Jamroga, W. van der Hoek, and M. Wooldridge. Intentions and strategies in game-like scenarios. In Carlos Bento, Amílcar Cardoso, and Gaël Dias, editors, *Progress in Artificial Intelligence: Proceedings of EPIA 2005*, volume 3808 of *Lecture Notes in Artificial Intelligence*, pages 512–523. Springer Verlag, 2005.

[101] G. Jonker. Feasible strategies in Alternating-time Temporal Epistemic Logic. Master thesis, University of Utrecht, 2003.

[102] M. Kacprzak and W. Penczek. Unbounded model checking for Alternating-time Temporal Logic. In *Proceedings of AAMAS-04*, 2004.

[103] P. Kaźmierczak, T. Ågotnes, and W. Jamroga. Multi-agency is coordination and (limited) communication. In *Proceedings of PRIMA*, volume 8861 of *LNCS*, pages 91–106. Springer, 2014.

[104] M. Köster and P. Lohmann. Abstraction for model checking modular interpreted systems over ATL. In *Proceedings of AAMAS*, pages 1129–1130, 2011.

[105] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[106] S. Kripke. Semantical considerations on modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.

[107] O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[108] F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. *Logical Methods in Computer Science*, 4:7, 2008.

[109] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Morgan & Claypool, 2008.

[110] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *POPL '85: Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 97–107, New York, NY, USA, 1985. ACM.

[111] A. Lomuscio and J. Michaliszyn. An abstraction technique for the verification of multi-agent systems against ATL specifications. In *Proceedings of KR*. AAAI Press, 2014.

[112] A. Lomuscio and W. Penczek. Symbolic model checking for temporal-epistemic logics. *SIGACT News*, 38(3):77–99, 2007.

[113] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 2015. To appear.

[114] A. Lomuscio and F. Raimondi. MCMAS : A model checker for multi-agent systems. In *Proceedings of TACAS*, volume 4314 of *LNCS*, pages 450–454, 2006.

[115] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[116] F. Mogavero, A. Murano, G. Perelli, , and M.Y. Vardi. What makes ATL* decidable? a decidable fragment of strategy logic. In *Proceedings of CONCUR*, pages 193–208, 2012.

[117] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic*, 15(4):1–42, 2014.

[118] F. Mogavero, A. Murano, and M.Y. Vardi. Reasoning about strategies. In *Proceedings of FSTTCS*, pages 133–144, 2010.

[119] R. C. Moore. Reasoning about knowledge and action. In *Proceedings of IJCAI*, 1977.

[120] R.C. Moore. A formal theory of knowledge and action. In J. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*. Ablex Publishing Corp., 1985.

[121] L. Morgenstern. A first-order theory of planning, knowledge, and action. In *Proceedings of TARK*, pages 99–114. Morgan Kaufmann, 1986.

[122] L. Morgenstern. Knowledge preconditions for actions and plans. In *Proceedings of IJCAI*, pages 867–874, 1987.

[123] L. Morgenstern and L. Stein. Why things go wrong: A formal theory of causal reasoning. In *Proceedings of AAAI*, 1988.

[124] H. Moulin and B. Peleg. Cores of effectivity functions and implementation theory. *Journal of Mathematical Economics*, 10(1):115–145, 1982.

[125] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[126] C.H. Papadimitriou. *Computational Complexity*. Addison Wesley : Reading, 1994.

[127] R. Parikh. The logic of games and its applications. *Ann. of Discrete Mathematics*, 24:111–140, 1985.

[128] M. Pauly. Game logic for game theorists. Technical Report INS-R0017, CWI, 2000.

[129] M. Pauly. *Logic for Social Software*. PhD thesis, University of Amsterdam, 2001.

[130] M. Pauly. A logical framework for coalitional effectivity in dynamic procedures. *Bulletin of Economic Research*, 53(4):305–324, 2001.

[131] M. Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149–166, 2002.

[132] W. Penczek and A. Polrola. *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*, volume 20 of *Studies in Computational Intelligence*. Springer-Verlag, 2006.

[133] G. Peterson and J. Reif. Multiple-person alternation. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 348–363. IEEE Computer Society Press, 1979.

[134] G. Peterson, J. Reif, and S. Azhar. Lower bounds for multiplayer noncooperative games of incomplete information. *Computers and Mathematics with Applications*, 41(7):957–992, 2001.

[135] M. Piccione and A. Rubinstein. The absent-minded drivers paradox: Synthesis and responses. *Games and Economic Behavior*, 20:121–130, 1997.

[136] M. Piccione and A. Rubinstein. On the interpretation of decision problems with imperfect recall. *Games and Economic Behavior*, 20:3–24, 1997.

[137] J. Pilecki, M.A. Bednarczyk, and W. Jamroga. Synthesis and verification of uniform strategies for multi-agent systems. In *Proceedings of the 15th Workshop on Computational Logic in Multi-Agent Systems CLIMA XV*, volume 8624 of *LNCS*, pages 166–182. Springer, 2014.

[138] S. Pinchinat. A generic constructive solution for concurrent games with expressive constraints on strategies. In *5th International Symposium on Automated Technology for Verification and Analysis*, volume 4762 of *LNCS*, pages 253–267. Springer-Verlag, 2007.

[139] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *POPL '89: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages*, pages 179–190, New York, NY, USA, 1989. ACM.

[140] A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings of the 31th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 746–757. IEEE Computer Society Press, 1990.

[141] J.P. Queille and J. Sifakis. Specification and verification of concurrent programs in Cesar. In *Symposium on Programming*, volume 137 of *LNCS*, pages 337–351. Springer, 1981.

[142] F. Raimondi. *Model Checking Multi-Agent Systems*. PhD thesis, University College London, 2006.

[143] F. Raimondi and A. Lomuscio. Automatic verification of deontic interpreted systems by model checking via OBDD's. In R.L. de Mántaras and L. Saitta, editors, *Proceedings of ECAI*, pages 53–57, 2004.

[144] A.S. Rao and M.P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, 1991.

[145] R. Rosner. *Modular Synthesis of Reactive Systems*. PhD thesis, Weizmann Institute of Science, 1992.

[146] S. Russel and P. Norvig. *Artificial Intelligence: a Modern Approach*. Prentice Hall, 1995.

[147] G. Ryle. *The Concept of Mind*. Chicago University Press, 1949.

[148] Tuomas W. Sandholm. Distributed rational decision making. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 201–258. The MIT Press, Cambridge, MA, USA, 1999.

[149] Sven Schewe. ATL* satisfiability is 2ExpTime-complete. In *Proceedings of ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 373–385. Springer-Verlag, 2008.

[150] Ph. Schnoebelen. The complexity of temporal model checking. In *Advances in Modal Logics, Proceedings of AiML 2002*. World Scientific, 2003.

[151] H. Schnoor. Deciding epistemic and strategic properties of cryptographic protocols. Technical Report tr_1012, Kiel University, 2010.

[152] Henning Schnoor. Strategic planning for probabilistic games with incomplete information. In *Proceedings of AAMAS'10*, pages 1057–1064, 2010.

[153] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93, 2004.

[154] N. V. Shilov, N. O. Garanina, and K.-M. Choe. Update and abstraction in model checking of knowledge and branching time. *Fundamenta Informaticae*, 72(1-3):347–361, 2006.

[155] N. V. Shilov, N. O. Garanina, and N. A. Kalinina. Model checking knowledge, actions and fixpoints. In *Proceedings of CS&P2004*, pages 351–357, 2004.

[156] Y. Shoham and K. Leyton-Brown. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[157] M.P. Singh. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications*, volume 799 of *Lecture Notes in Computer Science*. Springer, 1994.

[158] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of ACM*, 32(3):733–749, 1985.

[159] J. Stanley and T. Williamson. Knowing how. *The Journal of Philosophy*, 98(8):411–444, 2001.

[160] C. Stirling. *Modal and Temporal Properties of Processes*. Springer-Verlag, 2001.

[161] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[162] A. Toumi, J. Gutierrez, and M. Wooldridge. A tool for the automated verification of nash equilibria in concurrent games. In *Proceedings of ICTAC*, volume 9399 of *Lecture Notes in Computer Science*, pages 583–594. Springer, 2015.

[163] N. Troquard. *Independent Agents in Branching Time*. PhD thesis, Université Paul Sabatier, Toulouse, France and Università degli studi di Trento, 2007.

[164] N. Troquard and D. Walther. On satisfiability of ATL with strategy contexts. In *Proceedings of JELIA'12*, volume 7519 of *LNAI*, pages 398–410. Springer, 2012.

[165] J. van Benthem. Extensive games as process models. *Journal of Logic, Language and Information*, 2002.

[166] J. van Benthem, S. Ghosh, and R. Verbrugge, editors. *Models of Strategic Reasoning. Logics, Games, and Communities*, volume 8972 of *LNCS*. Springer, 2015.

[167] W. van der Hoek, W. Jamroga, and M. Wooldridge. A logic for strategic reasoning. In *Proceedings of AAMAS'05*, pages 157–164, 2005.

[168] W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In P. Stone and G. Weiss, editors, *Proceedings of AAMAS'06*, pages 201–208, 2006.

[169] W. van der Hoek and R. Verbrugge. Epistemic logic: A survey. *Game Theory and Applications*, 8:53–94, 2002.

[170] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proceedings of SPIN)*, volume 2318 of *Lecture Notes in Computer Science*, pages 95–111. Springer, 2002.

[171] W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In C. Castelfranchi and W.L. Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, pages 1167–1174. ACM Press, New York, 2002.

[172] W. van der Hoek and M. Wooldridge. Cooperation, knowledge and time: Alternating-time Temporal Epistemic Logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[173] R. van der Meyden and N.V. Shilov. Model checking knowledge and time in systems with perfect recall (extended abstract). In *Proceedings of Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *LNCS*, pages 432–445. Springer, 1999.

[174] H. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B. Kooi, editors. *Handbook of Epistemic Logic*. College Publications, 2015.

[175] H. van Ditmarsch and S. Knight. Partial information and uniform strategies. In *Proceedings of CLIMA*, LNCS, pages 183–198. Springer, 2014.

[176] S. van Otterloo and G. Jonker. On Epistemic Temporal Strategic Logic. *Electronic Notes in Theoretical Computer Science*, 126:77–92, 2004. Proceedings of LCMAS'04.

[177] S. van Otterloo, W. van der Hoek, and M. Wooldridge. Preferences in game logics. In *Proceedings of AAMAS-04*, pages 152–159, 2004.

[178] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the First Annual IEEE Symposium on Logic in Computer Science (LICS 1986)*, pages 332–344. IEEE Computer Society Press, 1986.

[179] S. Vester. Alternating-time temporal logic with finite-memory strategies. In *Proceedings of GandALF*, EPTCS, pages 194–207, 2013.

[180] D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. ATL satisfiability is indeed EXPTIME-complete. *Journal of Logic and Computation*, 16(6):765–787, 2006.

[181] D. Walther, W. van der Hoek, and M. Wooldridge. Alternating-time temporal logic with explicit strategies. In D. Samet, editor, *Proceedings TARK XI*, pages 269–278. Presses Universitaires de Louvain, 2007.

[182] G. Weiss, editor. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. MIT Press: Cambridge, Mass, 1999.

[183] M. Wooldridge. *Reasoning about Rational Agents*. MIT Press : Cambridge, Mass, 2000.

[184] M. Wooldridge. *An Introduction to Multi Agent Systems*. John Wiley & Sons, 2002.

[185] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational verification: From model checking to equilibrium checking. In *Proceedings of AAAI*, 2016. To appear.

10

MONOGRAPH SERIES: W. JAMROGA - LOGICAL METHODS FOR SPECIFICATION AND VERIFICATION OF MULTI-AGENT SYSTEMS

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

PhD STUDIES

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

The Project is co-financed by the European Union from resources of the European Social Fund

ISBN 978-83-63159-25-2