Model Checking Logics of Strategic Ability: Complexity*

Nils Bulling, Jürgen Dix, Wojciech Jamroga

Abstract This chapter is about model checking and its complexity in some of the main temporal and strategic logics, e.g. **LTL**, **CTL**, and **ATL**. We discuss several variants of **ATL** (perfect vs. imperfect *recall*, perfect vs. imperfect *information*) as well as two different measures for model checking with concurrent game structures (explicit vs. implicit *representation* of transitions). Finally, we summarize some results about higher order representations of the underlying models.

1 Introduction

Model checking is a powerful method used in verification. Given a model and a formula in a certain logic, model checking determines whether the formula is true in the model. Usually, it is used to check specifications of desirable properties for a system whose model is given. If the formula is true, then we know the property expressed by the formula is satisfied in the model. If not, it might lead us to change the system or give hints how to debug it.

Model checking was invented and pioneered by the work of Edward Melson Clarke, Ernest Allen Emerson, and by Joseph Sifakis and Jean Pierre Queille in the 80ies as a means for formal verification of finite-state concurrent systems. Specifica-

Nils Bulling and Jürgen Dix
Dept. of Informatics, Niedersächsische Technische Hochschule, Standort Clausthal, Germany
e-mail: {bulling,dix}@in.tu-clausthal.de

Wojciech Jamroga
Computer Science and Communications, University of Luxembourg, Luxembourg and Dept. of Informatics, Niedersächsische Technische Hochschule, Standort Clausthal, Germany
e-mail: wojtek.jamroga@uni.lu

* This work was partly funded by the NTH School for IT Ecosystems. NTH (Niedersächsische Technische Hochschule) is a joint university consisting of Technische Universität Braunschweig, Technische Universität Clausthal, and Leibniz Universität Hannover.

tions about the system were expressed as temporal logic formulae. It was especially suited for checking hardware designs, but also applied to checking software specifications. While it started as a new approach replacing the then common Floyd-Hoare style logic, it could only handle relatively small (though non-trivial) examples. Scalability was an important motivation right from the beginning. The last years have seen many industrial applications, and a number of powerful model checkers are available today. As founders of a new and flourishing area in computer science, Clarke, Emerson and Sifakis have been honored with the Turing award in 2007.

Logic-based verification of multi-agent systems has become an important subfield on its own. Some important model-checkers are:

- Mocha [1], available for download at http://www.cis.upenn.edu/ mocha/,
- VeriCS [11], available at http://pegaz.ipipan.waw.pl/verics/,
- MCMAS [45, 44], available at http://www-lai.doc.ic.ac.uk/mcmas/.

In this chapter, we do not deal with practical aspects of MAS verification. Instead, we offer a comprehensive survey of theoretical results concerning the computational complexity of model checking for relevant properties of agents and their teams. To this end, we focus on the class of properties that can be specified in Alternating-time Temporal Logic **ATL** (a logic that extends the classical branching time logic **CTL** with strategic modalities) and some of its extensions.

The aim of this chapter is twofold: (1) to give a comprehensive overview of the complexity of model checking in various strategic logics based on **ATL**, and (2) to discuss how the complexity can change when the models are not given explicitly but implicitly. Often, a model cannot be represented explicitly: It is given in a certain symbolic manner. Thus, the representation can be much smaller than the model itself, but it has to be (at least partially) *unfolded* when checking its properties.

While there are several chapters in this book that investigate model checking in multi-agent systems (cf. Chapter 3, *Model Checking Agent Communication*, Chapter 4, *Directions for Agent Model Checking*, Chapter 8, *Model Checking Goal-Oriented Agent Programming*), in this chapter we investigate mainly logics of *strategic ability* (variants of **ATL**). We determine the precise complexity of several variants of the logics and show when the problems become (probably) undecidable.

The plan of this chapter is as follows. In Section 2 we introduce the logics we are interested in: the temporal logics **LTL**, **CTL**, and **CTL*** and the strategic logics **ATL** and **ATL*** as well as their variants based on the assumption that agents have (im)perfect recall and (im)perfect information. We define syntax and semantics and introduce several running examples. Section 3 is devoted to standard complexity results for the logics. By standard, we mean that the input size is given by the number of transitions in the model and the length of the formula. In particular, we assume that the model and the formula are given explicitly. In Section 4 we consider the case when the transitions in the model are given in a more compact way, rather than by enumerating outcomes of all the possible combinations of agents' actions. Then, it makes more sense to measure complexity with respect to the number of states and the number of agents in the model. Finally, in Section 5, we investigate model checking for symbolic, very compact representations of multiagent systems:

concurrent programs and *modular interpreted systems*. This results in surprising complexity results, that can only be understood when looking closely at the size of the underlying structures (representations, models). We conclude in Section 6 with a discussion of our results, put them in perspective and point out future challenges.

2 The Logics: Syntax and Semantics

We begin by introducing temporal and strategic logics. We start with the linear-time logic **LTL** (Linear-time Temporal Logic) and the branching-time logics **CTL*** and **CTL** (Computation Tree Logic). Then, we present one of the most popular logics of strategic ability in multi-agent systems: **ATL** and **ATL*** (Alternating-time Temporal Logic). The relations between perfect vs. imperfect information on one hand, and perfect vs. imperfect recall on the other are discussed, and we show how they give rise to different semantics for **ATL** and **ATL***, yielding an interesting class of logics.

In the rest of this chapter we assume that Π is a non-empty set of *propositional symbols* and *St* a non-empty and finite set of *states*.

Remark 1 (Language, Semantics and Logic). In the following we proceed as follows. We introduce a logical language, say $\mathscr{L}$, which is defined as a set of formulae. Elements of $\mathscr{L}$ are called $\mathscr{L}$-formulae. Then, we consider (possibly several) semantics for the language. We look at each tuple consisting of a language and a suitable semantics (over a class of models) as a *logic*. The logic **CTL**, for instance, is given by the language $\mathscr{L}_{CTL}$ using the standard Kripke semantics.

2.1 Linear- and Branching-Time Logics

We begin by recalling two well-known classes of temporal logics: the *linear-time* logic **LTL** (Linear-Time Temporal Logic) and the *branching-time logics* **CTL** and **CTL*** (Computation Tree Logic).

2.1.1 The Languages $\mathscr{L}_{LTL}$, $\mathscr{L}_{CTL}$, and $\mathscr{L}_{CTL^*}$

$\mathscr{L}_{LTL}$ [42] extends the language of propositional logic with operators that allow to express temporal patterns over an infinite sequences of states, called *paths*. The basic temporal operators are $\mathscr{U}$ (*until*) and $\bigcirc$ (*in the next state*).

Definition 1 (Language $\mathscr{L}_{LTL}$ [42]). The *language* $\mathscr{L}_{LTL}$ is given by all formulae generated by the following grammar, where $p \in \Pi$ is a proposition: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathscr{U} \varphi \mid \bigcirc \varphi$.

The $\mathscr{L}_{LTL}$-formula $\bigcirc(\varphi \wedge \psi)$, for instance, expresses that φ and ψ hold in the next moment; $\varphi \mathscr{U} \psi$ states that the property φ is true at least until ψ becomes true which will eventually be the case. The additional operators $\Diamond$ (*sometime from now on*) and $\Box$ (*always from now on*) can be defined as macros by $\Diamond\varphi \equiv \top \mathscr{U} \varphi$ and $\Box\varphi \equiv \neg\Diamond\neg\varphi$, respectively. The standard Boolean connectives $\top, \bot, \vee, \rightarrow$, and $\leftrightarrow$ are defined in their usual way.

The logic is called *linear-time* since formulae are interpreted over infinite *linear* orders of states. The logic **CTL*** [13] explicitly refers to patterns of properties that can occur along a particular temporal path, *as well as* to the set of possible time series, and thus extends **LTL**. The latter dimension is handled by so called *path quantifiers*: E (*there is a path*) and A (*for all paths*) where the A quantifier is defined as macro: $\mathsf{A}\varphi \equiv \neg\mathsf{E}\neg\varphi$. Hence, the language of **CTL***, $\mathscr{L}_{CTL^*}$, extends $\mathscr{L}_{LTL}$ by adding the existential path quantifier E.

Definition 2 (Language $\mathscr{L}_{CTL^*}$ [13]). The *language* $\mathscr{L}_{CTL^*}$ is given by all formulae generated by the following grammar: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}\gamma$ where $\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \gamma \mathscr{U} \gamma \mid \bigcirc\gamma$ and $\mathsf{p} \in \Pi$. Formulae φ (resp. γ) are called *state* (resp. *path*) formulae.

Additionally, the same abbreviations as for $\mathscr{L}_{LTL}$ are defined. The $\mathscr{L}_{CTL^*}$-formula $\mathsf{E}\Diamond\varphi$, for instance, ensures that there is at least one path on which φ holds at some (future) time moment. Thus, $\mathscr{L}_{CTL^*}$-formulae do not only talk about temporal patterns on a given path but also quantify (existentially or universally) over such paths.

Finally, we define a fragment of **CTL*** called **CTL** [9] which is strictly *less expressive* but has *better computational properties*. The language $\mathscr{L}_{CTL}$ restricts $\mathscr{L}_{CTL^*}$ in such a way that each temporal operator must be directly preceded by a path quantifier. For example, $\mathsf{A}\Box\mathsf{E}\bigcirc\mathsf{p}$ is a $\mathscr{L}_{CTL}$-formula whereas $\mathsf{A}\Box\Diamond\mathsf{p}$ is not. Although this completely characterizes the language we also provide the original definition in which modalities are given by path quantifiers *coupled* with temporal operators. Note that, chronologically, **CTL** was proposed and studied before **CTL***.

Definition 3 (Language $\mathscr{L}_{CTL}$ [9]). The *language* $\mathscr{L}_{CTL}$ is given by all formulae generated by the following grammar, where $\mathsf{p} \in \Pi$ is a proposition: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}(\varphi \mathscr{U} \varphi) \mid \mathsf{E}\bigcirc\varphi \mid \mathsf{E}\Box\varphi$.

Again, the Boolean connectives are given by their usual abbreviations. In addition to that, we define the following: $\Diamond\varphi \equiv \top \mathscr{U} \varphi$, $\mathsf{A}\bigcirc\varphi \equiv \neg\mathsf{E}\bigcirc\neg\varphi$, $\mathsf{A}\Box\varphi \equiv \neg\mathsf{E}\Diamond\neg\varphi$, and $\mathsf{A}\varphi \mathscr{U} \psi \equiv \neg\mathsf{E}((\neg\psi) \mathscr{U} (\neg\varphi \wedge \neg\psi)) \wedge \neg\mathsf{E}\Box\neg\psi$. We note that in the definition of the language the existential quantifier cannot be replaced by the universal one without losing expressiveness (cf. [35]).

2.1.2 Semantics for $\mathscr{L}_{LTL}$, $\mathscr{L}_{CTL^*}$, and $\mathscr{L}_{CTL}$

As mentioned above, the semantics of **LTL** is given over *paths* that are infinite sequences of states from *St* and a labeling function $\pi : \Pi \rightarrow \mathscr{P}(St)$ that determines

which propositions are true at which states. Note that each path can be considered as a mapping $\mathbb{N} \to St$. We use $\lambda[i]$ to denote the ith position on path λ (starting from $i = 0$) and $\lambda[i,\infty]$ to denote the subpath of λ starting from i (i.e. $\lambda[i,\infty] = \lambda[i]\lambda[i+1]\ldots$).

Definition 4 (Semantics $\models^{\mathbf{LTL}}$). Let λ be a path and π be a valuation over St. The semantics of $\mathscr{L}_{LTL}$-formulae is defined by the satisfaction relation $\models^{\mathbf{LTL}}$ defined as follows:

$\lambda, \pi \models^{\mathbf{LTL}} \mathsf{p}$ iff $\lambda[0] \in \pi(\mathsf{p})$ and $\mathsf{p} \in \Pi$;
$\lambda, \pi \models^{\mathbf{LTL}} \neg\varphi$ iff not $\lambda, \pi \models^{\mathbf{LTL}} \varphi$ (we will also write $\lambda, \pi \not\models^{\mathbf{LTL}} \varphi$);
$\lambda, \pi \models^{\mathbf{LTL}} \varphi \wedge \psi$ iff $\lambda, \pi \models^{\mathbf{LTL}} \varphi$ and $\lambda, \pi \models^{\mathbf{LTL}} \psi$;
$\lambda, \pi \models^{\mathbf{LTL}} \bigcirc\varphi$ iff $\lambda[1,\infty], \pi \models^{\mathbf{LTL}} \varphi$; and
$\lambda, \pi \models^{\mathbf{LTL}} \varphi \mathscr{U} \psi$ iff there is an $i \in \mathbb{N}_0$ such that $\lambda[i,\infty], \pi \models \psi$ and $\lambda[j,\infty], \pi \models^{\mathbf{LTL}} \varphi$ for all $0 \leq j < i$;

Thus, according to Remark 1, the *logic* **LTL** is given by $(\mathscr{L}_{LTL}, \models^{\mathbf{LTL}})$. Paths are considered as (canonical) models for $\mathscr{L}_{LTL}$-formulae.

For model checking we require a finite representation of the input λ. To this end, we use a (pointed) Kripke model $\mathfrak{M}, q$ and consider the problem whether an $\mathscr{L}_{LTL}$-formula holds on *all* paths of $\mathfrak{M}$ starting in q.

A *Kripke model* (or *unlabeled transition system*) is given by $\mathfrak{M} = \langle St, \mathscr{R}, \Pi, \pi \rangle$ where St is a nonempty set of states (or possible worlds), $\mathscr{R} \subseteq St \times St$ is a *serial* transition relation on states, Π is a set of atomic propositions, and $\pi : \Pi \to \mathscr{P}(St)$ is a valuation of propositions. A *path* λ (or *computation*) in $\mathfrak{M}$ is an infinite sequence of states that can result from subsequent transitions, and refers to a possible course of action. We use the same notation for these paths as introduced above. For $q \in St$ we use $\Lambda_{\mathfrak{M}}(q)$ to denote the set of all paths of $\mathfrak{M}$ starting in q and we define $\Lambda_{\mathfrak{M}}$ as $\bigcup_{q \in St} \Lambda_{\mathfrak{M}}(q)$. The subscript "$\mathfrak{M}$" is often omitted when clear from context.

$\mathscr{L}_{CTL^*}$- and $\mathscr{L}_{CTL}$-formulae are interpreted over Kripke models but in addition to $\mathscr{L}_{LTL}$-(path) formulae (which can only occur as subformulae) it must be specified how state formulae are evaluated.

Definition 5 (Semantics $\models^{\mathbf{CTL}^*}$). Let $\mathfrak{M}$ be a Kripke model, $q \in St$ and $\lambda \in \Lambda$. The semantics of $\mathscr{L}_{CTL^*}$- and $\mathscr{L}_{CTL}$-formulae are given by the satisfaction relation $\models^{\mathbf{CTL}^*}$ for state formulae by

$\mathfrak{M}, q \models^{\mathbf{CTL}^*} \mathsf{p}$ iff $\lambda[0] \in \pi(\mathsf{p})$ and $\mathsf{p} \in \Pi$;
$\mathfrak{M}, q \models^{\mathbf{CTL}^*} \neg\varphi$ iff $\mathfrak{M}, q \not\models^{\mathbf{CTL}^*} \varphi$;
$\mathfrak{M}, q \models^{\mathbf{CTL}^*} \varphi \wedge \psi$ iff $\mathfrak{M}, q \models^{\mathbf{CTL}^*} \varphi$ and $\mathfrak{M}, q \models^{\mathbf{CTL}^*} \psi$;
$\mathfrak{M}, q \models^{\mathbf{CTL}^*} \mathsf{E}\varphi$ iff there is a path $\lambda \in \Lambda(q)$ such that $\mathfrak{M}, \lambda \models^{\mathbf{CTL}^*} \varphi$;

and for path formulae by:

$\mathfrak{M}, \lambda \models^{\mathbf{CTL}^*} \varphi$ iff $\mathfrak{M}, \lambda[0] \models^{\mathbf{CTL}^*} \varphi$;
$\mathfrak{M}, \lambda \models^{\mathbf{CTL}^*} \neg\gamma$ iff $\mathfrak{M}, \lambda \not\models^{\mathbf{CTL}^*} \gamma$;
$\mathfrak{M}, \lambda \models^{\mathbf{CTL}^*} \gamma \wedge \delta$ iff $\mathfrak{M}, \lambda \models^{\mathbf{CTL}^*} \gamma$ and $\mathfrak{M}, \lambda \models^{\mathbf{CTL}^*} \delta$;

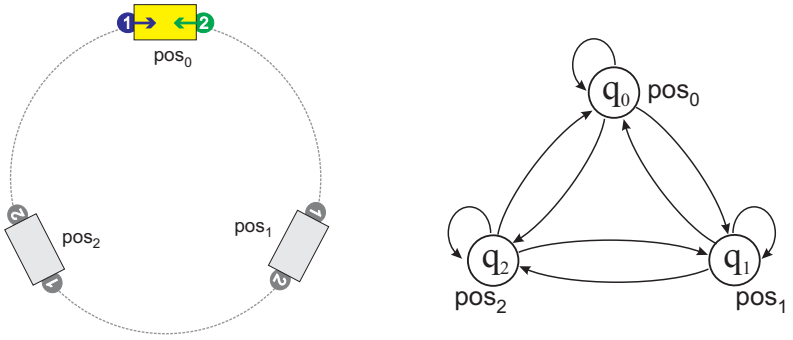

Fig. 1 Two robots and a carriage: a schematic view (left) and a transition system $\mathfrak{M}_0$ that models the scenario (right).

$\mathfrak{M},\lambda \models^{\mathbf{CTL}^*} \bigcirc\gamma$ iff $\lambda[1,\infty],\pi \models^{\mathbf{CTL}^*} \gamma$; and
$\mathfrak{M},\lambda \models^{\mathbf{CTL}^*} \gamma\mathscr{U}\delta$ iff there is an $i \in \mathbb{N}_0$ such that $\mathfrak{M},\lambda[i,\infty] \models^{\mathbf{CTL}^*} \delta$ and $\mathfrak{M},\lambda[j,\infty] \models^{\mathbf{CTL}^*} \gamma$ for all $0 \leq j < i$.

Alternatively, an equivalent *state-based* semantics for **CTL** can be given:

$\mathfrak{M},q \models^{\mathbf{CTL}} p$ iff $q \in \pi(p)$;
$\mathfrak{M},q \models^{\mathbf{CTL}} \neg\varphi$ iff $\mathfrak{M},q \not\models^{\mathbf{CTL}} \varphi$;
$\mathfrak{M},q \models^{\mathbf{CTL}} \varphi \wedge \psi$ iff $\mathfrak{M},q \models^{\mathbf{CTL}} \varphi$ and $\mathfrak{M},q \models^{\mathbf{CTL}} \psi$;
$\mathfrak{M},q \models^{\mathbf{CTL}} \mathsf{E}\bigcirc\varphi$ iff there is a path $\lambda \in \Lambda(q)$ such that $\mathfrak{M},\lambda[1] \models^{\mathbf{CTL}} \varphi$;
$\mathfrak{M},q \models^{\mathbf{CTL}} \mathsf{E}\Box\varphi$ iff there is a path $\lambda \in \Lambda(q)$ such that $\mathfrak{M},\lambda[i] \models^{\mathbf{CTL}} \varphi$ for every $i \geq 0$;
$\mathfrak{M},q \models^{\mathbf{CTL}} \mathsf{E}\varphi\mathscr{U}\psi$ iff there is a path $\lambda \in \Lambda(q)$ such that $\mathfrak{M},\lambda[i] \models^{\mathbf{CTL}} \psi$ for some $i \geq 0$, and $\mathfrak{M},\lambda[j,\infty] \models^{\mathbf{CTL}} \varphi$ for all $0 \leq j < i$.

This equivalent semantics underlies the model checking algorithm for **CTL** which can be implemented in *P* rather than *PSPACE* which is the case for $\mathbf{CTL}^*$ (cf. Section 3.1). Hence, the logics **CTL** and $\mathbf{CTL}^*$ are given by $(\mathscr{L}_{CTL},\models^{\mathbf{CTL}})$ and $(\mathscr{L}_{CTL^*},\models^{\mathbf{CTL}^*})$, respectively.

Remark 2. Note that model checking problem for an $\mathscr{L}_{LTL}$-formula φ with respect to a given Kripke model $\mathfrak{M}$ and a state q is equivalent to the $\mathbf{CTL}^*$ model checking problem $\mathfrak{M},q \models^{\mathbf{CTL}^*} \mathsf{A}\varphi$.

We end this section with an example.

Example 1 (Robots and Carriage). Consider the scenario depicted in Figure 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. The arrows in transition system M_0 indicate how the state of the system can change in a single step. We label the states with 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.

For example, we have $\mathfrak{M}_0, q_0 \models^{\mathbf{CTL}} \mathsf{E}\Diamond\mathsf{pos}_1$: In state q_0, there is a path such that the carriage will reach position 1 sometime in the future. Of course, the same is not true for *all* paths, so we also have that $\mathfrak{M}_0, q_0 \models^{\mathbf{CTL}} \neg\mathsf{A}\Diamond\mathsf{pos}_1$.

2.2 Strategic Abilities under Perfect Information

In this section we introduce logics that can be used to model and to reason about strategic abilities of agents with perfect information. Here "perfect information" is understood in such a way that agents know the current state of the system: The agents are able to distinguish all states of the system. This is fundamentally different from the imperfect information setting presented in Section 2.3 where *different* states possibly provide the same information to an agent and thus make them appear indistinguishable to it. This must be reflected in the agents' available strategies.

¿From now on, we assume that $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$ is a non-empty and finite set of *agents*. Sometimes, in order to make the examples easier to read, we may also use symbolic names $(a, b, c, \ldots)$ when referring to agents.

2.2.1 The Languages $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$

The logics **ATL*** and **ATL** [3, 4] (Alternating-time Temporal Logic) are generalizations of **CTL*** and **CTL**, respectively. In $\mathscr{L}_{ATL^*}/\mathscr{L}_{ATL}$ the path quantifiers E, A are replaced by *cooperation modalities* $\langle\langle A\rangle\rangle$ where $A \subseteq \mathbb{A}\mathrm{gt}$ is a team of agents. Formula $\langle\langle A\rangle\rangle\gamma$ expresses that team A has a *collective strategy* to enforce γ. The recursive definition of the language is given below.

Definition 6 (Language $\mathscr{L}_{ATL^*}$ [3]). The *language* $\mathscr{L}_{ATL^*}$ is given by all formulae generated by the following grammar: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle\gamma$ where $\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \gamma\,\mathscr{U}\,\gamma \mid \bigcirc\gamma$, $A \subseteq \mathbb{A}\mathrm{gt}$, and $\mathsf{p} \in \Pi$. Formulae φ (resp. γ) are called *state* (resp. *path*) formulae.

We use similar abbreviations to the ones introduced in Section 2.1.1. In the case of a single agent a we will also write $\langle\langle a\rangle\rangle$ instead $\langle\langle\{a\}\rangle\rangle$. An example $\mathscr{L}_{ATL^*}$-formula is $\langle\langle A\rangle\rangle\Box\Diamond\mathsf{p}$ which says that coalition A can guarantee that p is satisfied infinitely many times (ever and ever again in the future).

The language $\mathscr{L}_{ATL}$ restricts $\mathscr{L}_{ATL^*}$ in the same way as $\mathscr{L}_{CTL}$ restricts $\mathscr{L}_{CTL^*}$: Each temporal operator must be directly preceded by a cooperation modality.

Definition 7 (Language $\mathscr{L}_{ATL}$ [3]). The *language* $\mathscr{L}_{ATL}$ is given by all formulae generated by the following grammar: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box \varphi \mid \langle\langle A\rangle\rangle \varphi \mathscr{U} \varphi$ where $A \subseteq \mathbb{A}\mathrm{gt}$ and $\mathsf{p} \in \Pi$.

The $\mathscr{L}_{ATL^*}$-formula $\langle\langle A\rangle\rangle \Box \Diamond \mathsf{p}$ is obviously not a formula of $\mathscr{L}_{ATL}$ as it includes two consecutive temporal operators. In more general terms, $\mathscr{L}_{ATL}$ does not allow to express abilities related to, e.g., fairness properties. Still, many interesting properties are expressible. For instance, we can state that agent a has a strategy that permanently take away the ability to enforce $\bigcirc \mathsf{p}$ from coalition B: $\langle\langle a\rangle\rangle \Box \neg \langle\langle B\rangle\rangle \bigcirc \mathsf{p}$. As for the two computation tree logics, the choice between $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$ reflects the tradeoff between expressiveness and practicality.

2.2.2 Perfect Information Semantics for $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$

The semantics for $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$ are defined over a variant of transition systems where transitions are labeled with combinations of actions, one per agent. Formally, a *concurrent game structure* (CGS) is a tuple $\mathfrak{M} = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, Act, d, o\rangle$ which includes a nonempty finite set of all agents $\mathbb{A}\mathrm{gt} = \{1, \ldots, k\}$, a nonempty set of states St, a set of atomic propositions Π and their valuation $\pi : \Pi \to \mathscr{P}(St)$, and a nonempty finite set of (atomic) actions Act. Function $d : \mathbb{A}\mathrm{gt} \times St \to \mathscr{P}(Act)$ defines nonempty sets of actions available to agents at each state, and o is a (deterministic) transition function that assigns the outcome state $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to state q and a tuple of actions $\langle \alpha_1, \ldots, \alpha_k\rangle$ for $\alpha_i \in d(i,q)$ and $1 \leq i \leq k$, that can be executed by $\mathbb{A}\mathrm{gt}$ in q. We also write $d_a(q)$ instead of $d(a,q)$. So, it is assumed that all the agents execute their actions synchronously: The combination of the actions, together with the current state, determines the next transition of the system.

A *strategy* of agent a is a conditional plan that specifies what a is going to do in each situation. It makes sense, from a conceptual and computational point of view, to distinguish between two types of "situations" (and hence strategies): An agent might base his decision only on the current state or on the whole history of events that have happened. A *history* is considered as a finite sequence of states of the system.

A *perfect information perfect recall strategy* for agent a (*IR-strategy* for short)[2] is a function $s_a : St^+ \to Act$ such that $s_a(q_0 q_1 \ldots q_n) \in d_a(q_n)$. The set of such strategies is denoted by Σ_a^{IR}. On the other hand, a *perfect information memoryless strategy* for agent a (*Ir-strategy* for short) is given by a function $s_a : St \to Act$ where $s_a(q) \in d_a(q)$. The set of such strategies is denoted by Σ_a^{Ir}. We will use the term *strategy* to refer to any of these two types.

A *collective strategy* for a group of agents $A = \{a_1, \ldots, a_r\} \subseteq \mathbb{A}\mathrm{gt}$ is simply a tuple $s_A = \langle s_{a_1}, \ldots, s_{a_r}\rangle$ of strategies, one per agent from A. By $s_A|_a$, we denote agent a's part s_a of the collective strategy s_A where $a \in A$. The set of A's collective perfect information strategies is given by $\Sigma_A^{IR} = \prod_{a \in A} \Sigma_a^{IR}$ (in the perfect recall case)

[2] The notation was introduced in [49] where i (resp. I) stands for *imperfect* (resp. *perfect*) *information* and r (resp. R) for *imperfect* (resp. *perfect*) *recall*. Also compare with Section 2.3.

and $\Sigma_A^{Ir} = \prod_{a \in A} \Sigma_a^{Ir}$ (in the memoryless case). The set of all *strategy profiles* is given by $\Sigma^{IR} = \Sigma^{IR}_{\mathbb{A}\mathrm{gt}}$ (resp. $\Sigma^{Ir} = \Sigma^{Ir}_{\mathbb{A}\mathrm{gt}}$).

Function $out(q, s_A)$ returns the set of all paths that may occur when agents A execute strategy s_A from state q onward. For an *IR*-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 $i = 1, 2, \ldots$ there exists a tuple of agents' decisions $\langle \alpha_{a_1}^{i-1}, \ldots, \alpha_{a_k}^{i-1} \rangle$ such that $\alpha_a^{i-1} \in d_a(q_{i-1})$ for every $a \in \mathbb{A}\mathrm{gt}$, and $\alpha_a^{i-1} = s_A|_a(q_0 q_1 \ldots q_{i-1})$ for every $a \in A$, and $o(q_{i-1}, \alpha_{a_1}^{i-1}, \ldots, \alpha_{a_k}^{i-1}) = q_i\}$.

For an *Ir*-strategy s_A the outcome is defined analogously: "$s_A|_a(q_0 q_1 \ldots q_{i-1})$" is simply replaced by "$s_A|_a(q_{i-1})$"

The semantics for $\mathscr{L}_{ATL}$ and $\mathscr{L}_{ATL^*}$, one for each type of strategy, are shown below. Informally speaking, $\mathfrak{M}, q \models \langle\langle A \rangle\rangle \gamma$ if, and only if, there exists a collective strategy s_A such that γ holds for all computations from $out(q, s_A)$.

Definition 8 (Perfect Information Semantics $\models_{IR}$ and $\models_{Ir}$). Let $\mathfrak{M}$ be a CGS. The *perfect information perfect recall semantics* for $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$, *IR-semantics* for short, is defined as $\models^{\mathbf{CTL}^*}$ from Definition 5, denoted by $\models_{IR}$, but the rule for $\mathsf{E}\varphi$ is replaced by the following clause:

$\mathfrak{M}, q \models_{IR} \langle\langle A \rangle\rangle \gamma$ iff there is an *IR*-strategy $s_A \in \Sigma_A^{IR}$ for A such that for every path $\lambda \in out(s_A, q)$, we have $\mathfrak{M}, \lambda \models_{IR} \gamma$.

The *perfect information memoryless semantics* for $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$, *Ir-semantics* for short, is given as above but "*IR*" is replaced by "*Ir*" everywhere.

Remark 3. Note that cooperation modalities are neither "diamonds" nor "boxes" in terms of classical modal logic. Rather, they are combinations of both as their structure can be described by "$\exists\forall$": we ask for the *existence* of a strategy of the proponents which is successful against *all* responses of the opponents.

In [6] it is shown how the cooperation modalities can be decomposed into two parts in the context of **STIT** logic. A similar decomposition is considered in [27] for the analysis of stochastic multi-agent systems.

The $\mathscr{L}_{CTL^*}$ path quantifiers A and E can be embedded in $\mathscr{L}_{ATL^*}$ using the *IR*-semantics in the following way: $\mathsf{A}\gamma \equiv \langle\langle \varnothing \rangle\rangle \gamma$ and $\mathsf{E}\gamma \equiv \langle\langle \mathbb{A}\mathrm{gt} \rangle\rangle \gamma$.

Analogously to **CTL**, it is possible to provide a state-based semantics for $\mathscr{L}_{ATL}$. We only present the clause for $\langle\langle A \rangle\rangle \Box \varphi$ (the cases for the other temporal operators are given in a similar way):

$\mathfrak{M}, q, \models_{Ix}^{\mathbf{ATL}} \langle\langle A \rangle\rangle \Box \varphi$ iff there is an *Ix*-strategy $s_A \in \Sigma_A^{Ix}$ such that for all $\lambda \in out(q, s_A)$ and $i \in \mathbb{N}_0$ it holds that $\mathfrak{M}, q, \models_{Ix}^{\mathbf{ATL}} \varphi$

where x is either R or r.

This already suggests that dealing with $\mathscr{L}_{ATL}$ is computationally less expensive than with $\mathscr{L}_{ATL^*}$. On the other hand, $\mathscr{L}_{ATL}$ lacks expressiveness: There is no formula which is true for the memoryless semantics and false for the perfect recall semantics, and vice versa.

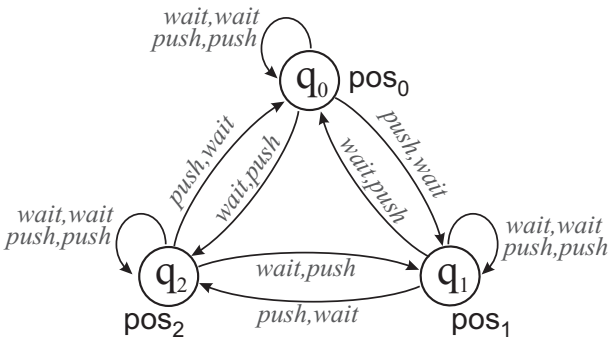

Fig. 2 The robots and the carriage: a concurrent game structure $\mathfrak{M}_1$.

Theorem 1. [3] *For $\mathscr{L}_{ATL}$, the perfect perfect recall semantics is equivalent to the memoryless semantics under perfect information, i.e., $\mathfrak{M},q \models_{IR} \varphi$ iff $\mathfrak{M},q \models_{Ir} \varphi$. Both semantics are different for $\mathscr{L}_{ATL^*}$.*

Thus, when referring to $\mathscr{L}_{ATL}$ using the perfect information semantics, we can omit the subscript in the satisfaction relation $\models$.

Definition 9 ($\mathbf{ATL_{Ix}}$, $\mathbf{ATL^*_{Ix}}$, ATL, $\mathbf{ATL^*}$). We define $\mathbf{ATL_{Ix}}$ and $\mathbf{ATL^*_{Ix}}$ as the logics $(\mathscr{L}_{ATL}, \models_{Ix})$ and $(\mathscr{L}_{ATL^*}, \models_{Ix})$ where $x \in \{r,R\}$, respectively. Moreover, we use **ATL** (resp. $\mathbf{ATL^*}$) as an abbreviation for $\mathbf{ATL_{IR}}$ (resp. $\mathbf{ATL^*_{IR}}$).

Note again, that $\mathbf{ATL_{IR}}$ and $\mathbf{ATL_{Ir}}$ are equivalent logics. We end our presentation of the language and semantics with an example.

Example 2 (Robots and Carriage, ctd.). Transition system $\mathfrak{M}_0$ from Figure 1 enabled us to study the evolution of the system as a whole. However, it did not allow us to represent *who* can achieve *what*, and how the possible actions of the agents interact. Concurrent game structure $\mathfrak{M}_1$, presented in Figure 2, fills the gap. We assume that each robot can either push (action *push*) or refrain from pushing (action *wait*). Moreover, they both use the same force when pushing. Thus, if the robots push simultaneously or wait simultaneously, the carriage does not move. When only one of the robots is pushing, the carriage moves accordingly.

As the outcome of each robot's action depends on the current action of the other robot, no agent can make sure that the carriage moves to any particular position. So, we have for example that $\mathfrak{M}_1,q_0 \models \neg\langle\langle 1\rangle\rangle \Diamond \mathsf{pos}_1$. On the other hand, the agent can at least make sure that the carriage will *avoid* particular positions. For instance, it holds that $\mathfrak{M}_1,q_0 \models \langle\langle 1\rangle\rangle \Box \neg \mathsf{pos}_1$, the right strategy being $s_1(q_0) = wait, s_1(q_2) = push$ (the action that we specify for q_1 is irrelevant).

[3] The property has been first observed in [49] but it follows from [4] in a straightforward way.

2.3 Strategic Abilities under Imperfect Information

ATL* and **ATL** include no way of addressing uncertainty that an agent or a process may have about the current situation. Several extensions capable of dealing with imperfect information have been proposed, e.g., in [4, 49, 29].

Here, we take Schobbens' version from [49] as the "core", minimal $\mathscr{L}_{ATL^*}$-based language for strategic ability under imperfect information. We take the already defined languages $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$ but here the cooperation modalities have an additional *epistemic flavor* by means of a modified semantics as we will show below.[4] The models, *imperfect information concurrent game structures* (ICGS), can be seen as concurrent game structures augmented with a family of indistinguishability relations $\sim_a \subseteq St \times St$, one per agent $a \in \mathbb{A}\text{gt}$. The relations describe agents' uncertainty: $q \sim_a q'$ means that agent a cannot distinguish between states q and q' of the system. Each $\sim_a$ is assumed to be an equivalence relation. It is also required that agents have the same choices in indistinguishable states: if $q \sim_a q'$ then $d(a,q) = d(a,q')$. Two *histories* $h = q_0 q_1 \dots q_n$ and $h' = q'_0 q'_1 \dots q'_{n'}$ are said to be *indistinguishable* for agent a, $h \sim_a h'$, if and only if, $n = n'$ and $q_i \sim_a q'_i$ for $i = 1, \dots, n$. This means that we deal with the synchronous notion of recall according to the classification in [18].

An *imperfect information strategy*[5] – memoryless or perfect recall – of agent a is a plan that takes into account a's epistemic limitations. An executable strategy must prescribe the *same choices for indistinguishable situations*. Therefore, we restrict the strategies that can be used by agents in the following way.

An *imperfect information perfect recall strategy* (*iR*-strategy for short) of agent a is an *IR*-strategy satisfying the following additional constraint: For all histories $h, h' \in St^+$, if $h \sim_a h'$ then $s_a(h) = s_a(h')$. That is, an *iR*-strategy is required to assign the same action to indistinguishable histories. Note that, as before, a perfect recall strategy (memoryless or not) assigns an action to each element from St^+.

An *imperfect information memoryless strategy* (*ir*-strategy for short) is an *Ir*-strategy satisfying the following constraint: if $q \sim_a q'$ then $s_a(q) = s_a(q')$. The set of a's *ir* (resp. *iR*) strategies is denoted by Σ_a^{ir} (resp. Σ_a^{iR}).

A *collective iR/ir-strategy* is a combination of individual *iR*/*ir*-strategies. The set of A's collective imperfect information strategies is given by $\Sigma_A^{iR} = \prod_{a \in A} \Sigma_a^{iR}$ (in the perfect recall case) and $\Sigma_A^{ir} = \prod_{a \in A} \Sigma_a^{ir}$ (in the memoryless case). The set of all strategy profiles is given by $\Sigma^{iR} = \Sigma_{\mathbb{A}\text{gt}}^{iR}$ (resp. $\Sigma^{ir} = \Sigma_{\mathbb{A}\text{gt}}^{ir}$). The outcome function $out(q, s_A)$ for the imperfect information cases is defined as before.

Definition 10 (Imperfect Information Semantics $\models_{iR}$ and $\models_{ir}$). Let $\mathfrak{M}$ be an ICGS, and let $\text{img}(q,\rho) = \{q' \mid \rho(q,q')\}$ be the image of state q wrt a binary relation ρ. The *imperfect information perfect recall* semantics (*iR-semantics*) for $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$, denoted by $\models_{iR}$, is given as in Definition 8 with the rule for $\langle\langle A \rangle\rangle \gamma$ replaced by the following clause:

---

[4] In [49] the cooperation modalities are presented with a subscript: $\langle\langle A \rangle\rangle_{ir}$ to indicate that they address agents with imperfect **i**nformation and imperfect **r**ecall. Here, we take on a rigorous semantic point of view and keep the syntax unchanged.

[5] Also called *uniform* strategy.

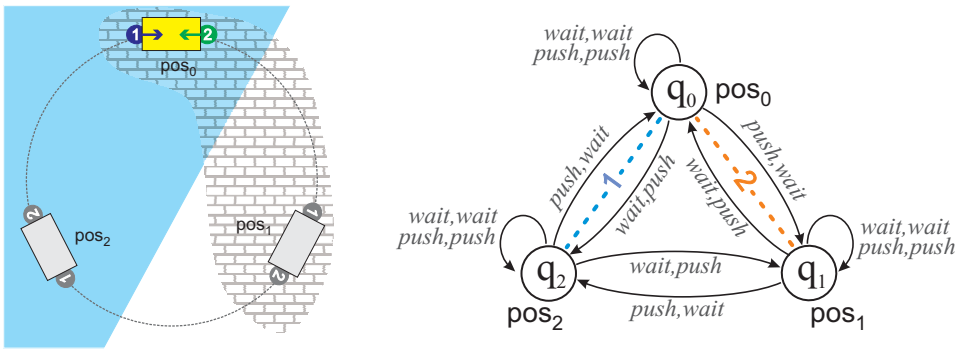

Fig. 3 Two robots and a carriage: a schematic view (left) and an imperfect information concurrent game structure $\mathfrak{M}_2$ that models the scenario (right).

$\mathfrak{M},q \models_{iR} \langle\langle A\rangle\rangle\gamma$ iff there is an *iR*-strategy $s_A \in \Sigma_A^{iR}$ such that, for each $q' \in \mathrm{img}(q,\sim_A)$ and every $\lambda \in out(s_A,q')$, we have $\mathfrak{M},\lambda \models_{iR} \gamma$ (where $\sim_A := \bigcup_{a\in A} \sim_a$).

The *imperfect information memoryless semantics* for $\mathscr{L}_{ATL^*}$ and $\mathscr{L}_{ATL}$, *ir-semantics for short*, is given as above but "*iR*" is replaced by "*ir*" everywhere.

Note that $\mathfrak{M},q \models_{ix} \langle\langle A\rangle\rangle\gamma$ requires A to have a single strategy that is successful in *all* states indistinguishable from q.

Remark 4 (Implicit knowledge operators). Note that some knowledge operators are *implicitly* given by the cooperation modalities if the imperfect information semantics is used. In this setting a formula $\langle\langle A\rangle\rangle\gamma$ is read as follows: every agent in A *knows* that they (the agents in A) have a collective strategy to enforce γ. In particular, one can express $K_a\varphi$ ("a knows that φ") by $\langle\langle a\rangle\rangle\varphi\,\mathscr{U}\varphi$, and $E_A\varphi$ ("everybody in A knows that φ") by $\langle\langle A\rangle\rangle\varphi\,\mathscr{U}\varphi$. More sophisticated epistemic versions of **ATL** which contain explicit knowledge operators (including ones for common and distributed knowledge) are, for instance, considered in [29, 25, 40, 21].

Definition 11 ($\mathbf{ATL_{ix}}$, $\mathbf{ATL^*_{ix}}$). We define $\mathbf{ATL_{ix}}$ and $\mathbf{ATL^*_{ix}}$ as the logics $(\mathscr{L}_{ATL},\models_{ix})$ and $(\mathscr{L}_{ATL^*},\models_{ix})$ where $x \in \{r,R\}$, respectively.

Example 3 (Robots and Carriage, ctd.). We refine the scenario from Examples 1 and 2 by restricting perception of the robots. Namely, 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 (cf. Figure 3). 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. We also assume that the agents are memoryless, i.e., they cannot memorize their previous observations.

With their observational capabilities restricted in such way, no agent can make the carriage reach or avoid any selected states singlehandedly. E.g., we have that

$\mathfrak{M}_2, q_0 \models_{ir} \neg\langle\langle 1\rangle\rangle \Box \neg \mathsf{pos}_1$. Note in particular that strategy s_1 from Example 2 cannot be used here because it is not uniform (indeed, the strategy tells robot 1 to wait in q_0 and push in q_2 but both states look the same to the robot). The robots cannot even be sure to achieve the task together: $\mathfrak{M}_2, q_0 \models_{ir} \neg\langle\langle 1,2\rangle\rangle \Box \mathsf{pos}_1$ (when in q_0, robot 2 considers it possible that the current state of the system is q_1, in which case all the hope is gone). 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: $\mathfrak{M}_2, q_0 \models_{ir} \langle\langle 1,2\rangle\rangle \Diamond \mathsf{pos}_1$ etc. – it suffices that one of the robots pushes all the time and the other waits all the time. Still, $\mathfrak{M}_2, q_0 \models_{ir} \neg\langle\langle 1,2\rangle\rangle \Diamond \Box \mathsf{pos}_x$ (for $x = 0,1,2$): there is no memoryless strategy for the robots to bring the carriage to a particular position and keep it there forever.

Most of the above properties hold for the *iR* semantics as well. Note, however, that for robots with perfect recall we do have that $\mathfrak{M}_2, q_0 \models_{iR} \langle\langle 1,2\rangle\rangle \Diamond \Box \mathsf{pos}_x$. The right strategy is that one robot pushes and the other waits for the first 3 steps. After that, they know their current position exactly, and can go straight the specified position.

2.4 Other Subsets of $\mathscr{L}_{ATL^*}$

2.4.1 Coalition Logic

***C**oalition **L**ogic* (**CL**), introduced in [41], is another logic for modeling and reasoning about strategic abilities of agents. The main construct of **CL**, $[A]\varphi$, expresses that coalition A can bring about φ in a single-step game.

Definition 12 (Language $\mathscr{L}_{CL}$ [41]). The *language* $\mathscr{L}_{CL}$ is given by all formulae generated by the following grammar: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid [A]\varphi$, where $\mathsf{p} \in \Pi$ and $A \subseteq \mathbb{A}\mathrm{gt}$.

In [41], *coalitional models* were chosen as semantics for $\mathscr{L}_{CL}$. These models are given by (St, E, π) consisting of a set of states St, a *playable effectivity function* E, and a valuation function π. The effectivity function determines the outcomes that a coalition is effective for, i.e., given a set $X \subseteq St$ of states a coalition C is said to be effective for X iff it can enforce the next state to be in X. However, in [21] it was shown that CGS provide an equivalent semantics, and that **CL** can be seen as the *next-time fragment* of **ATL**. Hence, for this presentation we will interpret $\mathscr{L}_{CL}$-formulae over CGS's, and consider $[A]\varphi$ as an abbreviation for $\langle\langle A\rangle\rangle \bigcirc \varphi$. The various logics $\mathbf{CL}_{xy}$ that we can obtain using the semantics $\models_{xy}$ for $x \in \{i, I\}$ and $y \in \{r, R\}$ are defined analogously to $\mathbf{ATL}_{xy}$.

2.4.2 ATL$^+$

The language $\mathscr{L}_{ATL^+}$ is the subset of $\mathscr{L}_{ATL^*}$ that requires each temporal operator to be followed by a state formula, but allows for Boolean combinations of path subformulae. The formula $\langle\langle A\rangle\rangle(\Box \mathsf{p} \wedge \Diamond \mathsf{q})$, for instance, is an $\mathscr{L}_{ATL^+}$-formula but not an $\mathscr{L}_{ATL}$-formula. Formally, the language is given as follows:

Definition 13 (Language $\mathscr{L}_{ATL^+}$). The *language* $\mathscr{L}_{ATL^+}$ is given by all formulae generated by the following grammar: $\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle\gamma$ where $\gamma ::= \neg\gamma \mid \gamma \wedge \gamma \mid \varphi \,\mathscr{U}\, \varphi \mid \bigcirc\varphi$, $A \subseteq \mathbb{A}\mathrm{gt}$ and $\mathsf{p} \in \Pi$.

We define the various logics emerging from $\mathscr{L}_{ATL^+}$ and the different semantics analogously to the case of $\mathscr{L}_{ATL}$. The logic **ATL**$^+$ is strictly more expressive than **ATL** (contrary to common belief, each **ATL**$^+$ formula can only be translated to an equivalent **ATL** formula if the "release" or "weak until" operator is added to the language of $\mathscr{L}_{ATL}$ [8, 37, 22]) but it enables a more succinct encoding of properties (this follows from the results in [53]). In Section 3 we will see that the more succinct language has its price: The model checking problem becomes computationally more expensive.

2.5 Summary, Notation, and Related Work

We have recalled the linear-time temporal logic and two versions of the computation tree logics for reasoning about purely temporal systems. Then, we presented several variants of the alternating-time temporal logics: the richest underlying language $\mathscr{L}_{ATL^*}$, somewhat restricted variants $\mathscr{L}_{ATL^+}$ and $\mathscr{L}_{ATL}$, and $\mathscr{L}_{CL}$ which can be seen as a very limited fragment of $\mathscr{L}_{ATL}$. All these languages were coupled with four alternative semantics that result from combining perfect/imperfect information with perfect recall/memoryless strategies (the *IR*, *Ir*, *iR*, and *ir*-semantics).

The resulting logics were defined with respect to the notation introduced by Schobbens [49] to refer to a strategic logic using a specific semantics. For $ID \in \{\mathbf{CL}, \mathbf{ATL}, \mathbf{ATL}^+, \mathbf{ATL}^*\}$, $x \in \{I, i\}$, and $y \in \{R, r\}$, we used ID_{xy} to refer to the logic over the language $\mathscr{L}_{ID}$ using the *xy*-semantics $\models_{xy}$.

In this chapter we are concerned with model checking strategic logics and thus take on a semantic view. Naturally, there is more than that to be studied. In [20] a complete *axiomatization* for $\mathbf{ATL}_{IR}$ is presented. Also the *satisfiability problem* of $\mathbf{ATL}_{IR}$ and $\mathbf{ATL}^*_{IR}$ has been considered by researchers: The problem was proven *EXPTIME*-complete for $\mathbf{ATL}_{IR}$ [12, 52] and even *2EXPTIME*-complete for $\mathbf{ATL}^*_{IR}$ [47]. Axiomatization and satisfiability of other variants of alternating-time temporal logic still remains open.

3 Standard Model Checking Complexity Results

In this section we consider model checking for the logics introduced in Section 2. The process of model checking seeks to answer the question whether a given formula φ is satisfied in a state q of model $\mathfrak{M}$. Formally, *local model checking* is the decision problem that determines membership in the set

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

where $\mathscr{L}$ is a logical language, Struc is a class of (pointed) models for $\mathscr{L}$ (i.e. a tuple consisting of a model and a state), and $\models$ is a semantic satisfaction relation compatible with $\mathscr{L}$ and Struc. We omit parameters if they are clear from context, e.g., we use $\mathsf{MC}(\mathbf{CTL})$ to refer to model checking of **CTL** over the class of (pointed) Kripke models and the introduced semantics.

It is often useful to compute the set of states in $\mathfrak{M}$ that satisfy formula φ instead of checking if φ holds in a particular state. This variant of the problem is known as *global model checking*. It is easy to see that, for the settings we 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.

In the following, we are interested in the decidability and the computational complexity of determining whether an input instance $(\mathfrak{M}, q, \varphi)$ belongs to $\mathsf{MC}(\dots)$. The complexity is always relative to the *size* of the instance; in the case of model checking, it is the size of the representation of the model and the representation of the formula that we use. Thus, in order to establish the complexity, it is necessary to fix how we *represent* the input and how we *measure* its size. In this section, we consider explicit representation of models and formulae, together with the "standard" input measure, where the size of the model ($|\mathfrak{M}|$) is given by the *number of transitions* in $\mathfrak{M}$, and the size of the formula ($|\varphi|$) is given by its *length* (i.e., the number of elements it is composed of, apart from parentheses). For example, the model in Figure 2 includes 12 (labeled) transitions, and the formula $\langle\langle 1\rangle\rangle \bigcirc (\mathsf{pos}_0 \vee \mathsf{pos}_1)$ has length 5.

3.1 Model Checking Temporal Logics

An excellent survey on the model checking complexity of temporal logics has been presented in [48]. Here, we only recall the results relevant for the subsequent analysis of strategic logics.

Let $\mathfrak{M}$ be a Kripke model and q be a state in the model. Model checking a $\mathscr{L}_{CTL}/\mathscr{L}_{CTL^*}$-formula φ in $\mathfrak{M}, q$ means to determine whether $\mathfrak{M}, q \models \varphi$, i.e., whether φ holds in $\mathfrak{M}, q$. For **LTL**, checking $\mathfrak{M}, q \models \varphi$ means that we check the *validity* of φ in the pointed model $\mathfrak{M}, q$, i.e., whether φ holds *on all the paths* in $\mathfrak{M}$ that start from q (equivalent to $\mathbf{CTL}^*$ model checking of formula $\mathsf{A}\varphi$ in $\mathfrak{M}, q$, cf. Remark 2).

function $mcheck(\mathfrak{M}, \varphi)$**.**
Model checking formulae of CTL. Returns the exact subset of St for which formula φ holds.
case $\varphi \equiv p$ **:** return $\{q \in St \mid p \in \pi(q)\}$
case $\varphi \equiv \neg\psi$ **:** return $St \setminus mcheck(\mathfrak{M}, \psi)$
case $\varphi \equiv \psi_1 \wedge \psi_2$ **:** return $mcheck(\mathfrak{M}, \psi_1) \cap mcheck(\mathfrak{M}, \psi_2)$
case $\varphi \equiv \mathsf{E}\bigcirc\psi$ **:** return $pre(mcheck(\mathfrak{M}, \psi))$
case $\varphi \equiv \mathsf{E}\Box\psi$ **:**
  $Q_1 := Q$; $Q_2 := Q_3 := mcheck(\mathfrak{M}, \psi)$;
  while $Q_1 \not\subseteq Q_2$ **do** $Q_1 := Q_1 \cap Q_2$; $Q_2 := pre(Q_1) \cap Q_3$ **od**;
  return Q_1
case $\varphi \equiv \mathsf{E}\psi_1 \mathscr{U} \psi_2$ **:**
  $Q_1 := \emptyset$; $Q_2 := mcheck(\mathfrak{M}, \psi_2)$; $Q_3 := mcheck(\mathfrak{M}, \psi_1)$;
  while $Q_2 \not\subseteq Q_1$ **do** $Q_1 := Q_1 \cup Q_2$; $Q_2 := pre(Q_1) \cap Q_3$ **od**;
  return Q_1
end case

Fig. 4 The **CTL** model checking algorithm from [9].

It has been known for a long time that formulae of **CTL** can be model-checked in time linear with respect to the size of the model and the length of the formula [10], whereas formulae of **LTL** and **CTL*** are significantly harder to verify.

Theorem 2 (CTL [10, 48]). *Model checking* **CTL** *is P-complete, and can be done in time* $\mathbf{O}(|\mathfrak{M}| \cdot |\varphi|)$*, where* $|\mathfrak{M}|$ *is given by the number of transitions.*

Proof (Sketch). The algorithm determining the states in a model at which a given formula holds is presented in Figure 4. The lower bound (*P*-hardness) can be for instance proven by a reduction of the tiling problem [48]. □

Theorem 3 (LTL [50, 39, 51]). *Model checking* **LTL** *is PSPACE-complete, and can be done in time* $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|\mathfrak{M}|)$*, where* $|\mathfrak{M}|$ *is given by the number of transitions.*

Proof (Sketch). We sketch the approach given in [51]. Firstly, given an $\mathscr{L}_{LTL}$-formula φ, a Büchi automaton $\mathscr{A}_{\neg\varphi}$ of size $2^{\mathbf{O}(|\varphi|)}$ accepting exactly the paths satisfying $\neg\varphi$ is constructed. The pointed Kripke model $\mathfrak{M}, q$ can directly be interpreted as a Büchi automaton $\mathscr{A}_{\mathfrak{M},q}$ of size $\mathbf{O}(|\mathfrak{M}|)$ accepting all possible paths in the Kripke model starting in q. Then, the model checking problem reduces to the non-emptiness check of $L(\mathscr{A}_{\mathfrak{M},q}) \cap L(\mathscr{A}_{\neg\varphi})$ which can be done in time $\mathbf{O}(|\mathfrak{M}|) \cdot 2^{\mathbf{O}(|\varphi|)}$ by constructing the product automaton. (Emptiness can be checked in linear time wrt to the size of the automaton.) A *PSPACE*-hardness proof can for instance be found in [50]. □

The hardness of **CTL*** model checking is immediate from Theorem 3 as $\mathscr{L}_{LTL}$ can be seen as a fragment of $\mathscr{L}_{CTL^*}$. For the proof of the upper bound one combines the **CTL** and **LTL** model checking techniques. Consider a $\mathscr{L}_{CTL^*}$-formula φ which contains a state subformula $\mathsf{E}\psi$ where ψ is a pure $\mathscr{L}_{LTL}$-formula. Firstly, we can use **LTL** model checking to determine all state which satisfy $\mathsf{E}\psi$ (these are all states q in which the $\mathscr{L}_{LTL}$-formula $\neg\psi$ is *not* true) and label them by a fresh propositional

symbol , say p, and replace $\mathsf{E}\psi$ in φ by p as well. Applying this procedure recursively yields a pure $\mathscr{L}_{CTL}$-formula which can be verified in polynomial time. Hence, the procedure can be implemented by an oracle machine of type $P^{PSPACE} = PSPACE$ (the **LTL** model checking algorithm might be employed polynomially many times). Thus, the complexity for $\mathbf{CTL}^*$ is the same as for **LTL**.

Theorem 4 ($\mathbf{CTL}^*$ [10, 16]). *Model checking $\mathbf{CTL}^*$ is PSPACE-complete, and can be done in time $2^{\mathbf{O}(|\varphi|)}\mathbf{O}(|\mathfrak{M}|)$, where $|\mathfrak{M}|$ is given by the number of transitions.*

In Section 2.4 we introduced $\mathbf{ATL}^+$, a variant of **ATL**. As the model checking algorithm for $\mathbf{ATL}^+$ will rely on the complexity of $\mathbf{CTL}^+$ model checking,[6] we mention the latter result here.

Theorem 5 ($\mathbf{CTL}^+$ [38]). *Model checking $\mathbf{CTL}^+$ is Δ_2^P-complete in the number of transitions in the model and the length of the formula.*

3.2 Model Checking ATL *and* CL*: Perfect Information*

One of the main results concerning **ATL** states that its formulae can also be model-checked in deterministic linear time, analogously to **CTL**. It is important to emphasize, however, that the result is relative to the number of transitions in the model and the length of the formula. In Section 4 we will discuss an alternative input measure in terms of agents, states, and the length of the formula, and show that this causes a substantial increase in complexity.

The **ATL** model checking algorithm from [4] is presented in Figure 5. The algorithm employs the well-known fixpoint characterizations of strategic-temporal modalities:

$$\langle\langle A\rangle\rangle \Box \varphi \leftrightarrow \varphi \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \Box \varphi$$
$$\langle\langle A\rangle\rangle \varphi_1 \mathscr{U} \varphi_2 \leftrightarrow \varphi_2 \vee \varphi_1 \wedge \langle\langle A\rangle\rangle \bigcirc \langle\langle A\rangle\rangle \varphi_1 \mathscr{U} \varphi_2,$$

and computes a winning strategy step by step (if it exists). That is, it starts with the appropriate candidate set of states ($\emptyset$ for $\mathscr{U}$ and the whole set *St* for $\Box$), and iterates backwards over *A*'s one-step abilities until the set gets stable. It is easy to see that the algorithm needs to traverse each transition at most once per subformula of φ. 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. Thus, for an $\mathscr{L}_{ATL}$-formula $\langle\langle A\rangle\rangle\gamma$, if *A* have an *IR*-strategy to enforce γ, they also have an *Ir*-strategy to obtain it.

Theorem 6 ($\mathbf{ATL}_{Ir}$ and $\mathbf{ATL}_{IR}$ [4]). *Model checking $\mathbf{ATL}_{Ir}$ and $\mathbf{ATL}_{IR}$ is P-complete, and can be done in time $\mathbf{O}(|\mathfrak{M}| \cdot |\varphi|)$, where $|\mathfrak{M}|$ is given by the number of transitions in $\mathfrak{M}$.*

[6] $\mathbf{CTL}^+$ is defined analogously to $\mathbf{ATL}^+$: Boolean combinations of path formulae are allowed in the scope of path quantifiers.

function $mcheck(M,\varphi)$**.**

ATL model checking. Returns the set of states in model $M = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, o\rangle$ for which formula φ holds.

case $\varphi \in \Pi$ **:** return $\pi(p)$
case $\varphi = \neg\psi$ **:** return $St \setminus mcheck(M,\psi)$
case $\varphi = \psi_1 \vee \psi_2$ **:** return $mcheck(M,\psi_1) \cup mcheck(M,\psi_2)$
case $\varphi = \langle\langle A\rangle\rangle \bigcirc \psi$ **:** return $pre(M,A,mcheck(M,\psi))$
case $\varphi = \langle\langle A\rangle\rangle \Box \psi$ **:**
  $Q_1 := St$; $Q_2 := mcheck(M,\psi)$; $Q_3 := Q_2$;
  while $Q_1 \not\subseteq Q_2$
  do $Q_1 := Q_2$; $Q_2 := pre(M,A,Q_1) \cap Q_3$ **od**;
  return Q_1
case $\varphi = \langle\langle A\rangle\rangle \psi_1 \mathscr{W} \psi_2$ **:**
  $Q_1 := \varnothing$; $Q_2 := mcheck(M,\psi_1)$;
  $Q_3 := mcheck(M,\psi_2)$;
  while $Q_3 \not\subseteq Q_1$
  do $Q_1 := Q_1 \cup Q_3$; $Q_3 := pre(M,A,Q_1) \cap Q_2$ **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 $\{q \mid \exists\alpha_A \forall\alpha_{\mathbb{A}\mathrm{gt}\setminus A}\ o(q,\alpha_A,\alpha_{\mathbb{A}\mathrm{gt}\setminus A}) \in Q\}$

Fig. 5 The **ATL** model checking algorithm from [4]

Proof (Sketch). Each case of the algorithm is called at most $\mathbf{O}(|\varphi|)$ times and terminates after $\mathbf{O}(|\mathfrak{M}|)$ steps [4]. The latter is shown by translating the model to a two-player game [4], and then solving the "invariance game" on it in polynomial time [5]. Hardness is shown by a reduction of reachability in And-Or-Graphs, which was shown to be *P*-complete in [26], to model checking the (constant) $\mathscr{L}_{ATL}$-formula $\langle\langle 1\rangle\rangle \Diamond \mathsf{p}$ in a two player game. In each Or-state it is the turn of player 1 and in each And-state it is player 2's turn [4]. □

In the next theorem, we show that the model checking of coalition logic is as hard as for **ATL**. To our knowledge, this is a new result; the proof is done by a slight variation of the hardness proof for **ATL** in [4] (cf. the proof of Theorem 6).

Theorem 7 ($\mathbf{CL}_{Ir}$ and $\mathbf{CL}_{IR}$). *Model checking $\mathbf{CL}_{Ir}$ and $\mathbf{CL}_{IR}$ is P-complete, and can be done in time $\mathbf{O}(|\mathfrak{M}| \cdot |\varphi|)$, where $|\mathfrak{M}|$ is given by the number of transitions in $\mathfrak{M}$.*

Proof. The upper bound follows from the fact that $\mathscr{L}_{CL}$ is a sublanguage of $\mathscr{L}_{ATL}$. We show *P*-hardness by the following adaption of the reduction of And-Or-Graph reachability from [4]. Firstly, we observe that if a state y is reachable from 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 6, we take G to be a turned-based CGS in which player 1 "owns" all the Or-states and player 2 "owns" 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 $G,x \models \underbrace{\langle\langle 1\rangle\rangle \bigcirc \ldots \langle\langle 1\rangle\rangle \bigcirc}_{n\text{-times}} \mathsf{y}$. The reduction uses only logarithmic space. □

It is worth pointing out, however, that checking strategic properties in one-step games is somewhat easier. We recall that AC^0 is the class corresponding to constant-depth, unbounded-fanin, polynomial-size Boolean circuits with AND, OR, and NOT gates [19]. We 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. For example, the language of "simple **CL**" consists only of formulae p and $\langle\langle A\rangle\rangle \bigcirc p$, for $p \in \Pi$ and $A \subseteq \mathbb{A}\mathrm{gt}$.

Theorem 8 (Simple $\mathbf{CL}_{Ir}$ and $\mathbf{CL}_{IR}$ [37]). *Model checking "Simple $\mathbf{CL}_{Ir}$" and "Simple $\mathbf{CL}_{IR}$" with respect to the number of transitions in the model and the length of the formula is in AC^0.*

Proof (Sketch). For $\mathfrak{M},q \models \langle\langle A\rangle\rangle \bigcirc \mathsf{p}$, we construct a 3-level circuit [37]. On the first level, we assign one AND gate for every possible coalition B and B's collective choice α_B; the output of the gate is "true" iff α_B leads to a state satisfying p for every response of $\mathbb{A}\mathrm{gt} \setminus B$. On the second level, there is one OR gate per possible coalition B that connects all the B's gates from the first level and outputs "true" iff there is any successful strategy for B. On the third level, there is a single AND gate that selects the right output (i.e., the one for coalition A). □

3.3 Model Checking ATL and CL: Imperfect Information

In contrast to the perfect information setting, analogous fixpoint characterizations do *not* hold for the incomplete information semantics over $\mathscr{L}_{ATL}$ because the choice of a particular action at a state q has non-local consequences: It automatically fixes choices at all states q' indistinguishable from q for the coalition A. 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. Note that, in order to check $\mathfrak{M},q \models \langle\langle A\rangle\rangle \gamma$ (where γ includes no nested cooperation modalities), the following procedure suffices. Firstly, we guess a uniform strategy s_A of team A (by calling an NP oracle), and then verify the strategy by pruning $\mathfrak{M}$ accordingly (removing all the transitions that are not going to be executed according to s_A) and model-checking the $\mathscr{L}_{CTL}$-formula $\mathsf{A}\gamma$ in the resulting model. For nested cooperation modalities, we proceed recursively (bottom up). Since model checking **CTL** can be done in polynomial deterministic time, the procedure runs in polynomial deterministic time with calls to an NP oracle, which demonstrates the inclusion in $\Delta_2^P = P^{NP}$ [49]. As it turns out, a more efficient procedure does not exist, which is confirmed by the following result.

Theorem 9 ($\mathbf{ATL}_{ir}$ [49, 33]). *Model checking $\mathbf{ATL}_{ir}$ is Δ_2^P-complete in the number of transitions in the model and the length of the formula.*

Proof (Sketch). The discussion above proves the membership in Δ_2^P. Δ_2^P-hardness was shown in [33] through a reduction of *sequential satisfiability* ($\mathbf{SNSAT}_2$), a standard Δ_2^P-complete problem [38]. The idea is that there are two agents where one agent tries to verify a (nested) propositional formula and a second agent tries to refute it. A winning strategy of the "verifier agent" corresponds to a satisfying valuation of the formula. Uniformity of the verifier's strategy is needed to ensure that identical proposition symbols, occurring at different places in the formula, are assigned the same truth values. □

Now we consider the incomplete information setting for Coalition Logic. It is easy to see that the *iR*- and *ir*-sematics are equivalent for $\mathscr{L}_{CL}$ since $\bigcirc$ 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 $\bigcirc\varphi$, then there is also an *ir*-strategy for A to obtain the same. Perfect recall of the history does not matter in one-step games.

Theorem 10 ($\mathbf{CL}_{ir}$ and $\mathbf{CL}_{iR}$). *Model checking $\mathbf{CL}_{ir}$ and $\mathbf{CL}_{iR}$ is P-complete wrt the number of transitions in the model and the length of the formula, and can be done in time $\mathbf{O}(|\mathfrak{M}| \cdot |\varphi|)$.*

Proof. The *P*-hardness follows from Theorem 7 (perfect information CGS's can be seen as a special kind of ICGSwhere the indistinguishability relations contain only the reflexive loops). For the upper bound, we use the following algorithm. For $\mathfrak{M}, q \models \langle\langle A\rangle\rangle \bigcirc \mathsf{p}$, we check if there is a collective action α_A such that for all responses $\alpha_{\mathbb{A}\mathrm{gt}\setminus A}$ we have that $\bigcup_{\{q' | q \sim_A q'\}} \{o(q', \alpha_A, \alpha_{\mathbb{A}\mathrm{gt}\setminus A})\} \subseteq \pi(\mathsf{p})$. For $\langle\langle A\rangle\rangle \bigcirc \varphi$ with nested cooperation modalities, we proceed recursively (bottom up). □

Theorem 11 (Simple $\mathbf{CL}_{ir}$ and $\mathbf{CL}_{iR}$). *Model checking "simple" formulae of $\mathbf{CL}_{ir}$ and $\mathbf{CL}_{iR}$ with respect to the number of transitions in the model and the length of the formula is in AC^0.*

Proof. For $\mathfrak{M}, q \models \langle\langle A\rangle\rangle \bigcirc \mathsf{p}$, we extend the procedure from [37] by creating one copy of the circuit per $q' \in \mathrm{img}(q, \sim_A)$. Then, we add a single AND gate on the fourth level of the circuit, that takes the output of those copies and returns "true" iff A have a strategy that is successful from all states indistinguishable from q. □

That leaves us with the issue of $\mathscr{L}_{ATL}$ with the semantics assuming imperfect information and perfect recall. To our knowledge, there is no formal proof in the literature regarding the complexity of model checking $\mathscr{L}_{ATL}$ with *iR*-strategies. However, the problem is commonly believed to be undecidable.

Conjecture 1 ($\mathbf{ATL}_{iR}$ [4]). Model checking $\mathbf{ATL}_{iR}$ is undecidable.

Theorems 10 and 11 are incorrect.
The correct result is that model checking of $\mathrm{CL}_{\mathrm{ir}}$ and $\mathrm{CL}_{\mathrm{iR}}$ is P-complete for coalitions of size up to 2, and NP-hard for larger coalitions. For Simple $\mathrm{CL}_{\mathrm{ir}}$ and $\mathrm{CL}_{\mathrm{iR}}$ the problem is respectively P-complete and NP-complete.
A precise formulation and proofs can be found e.g. in Jamroga (2015), pp. 112-114.

3.4 Model Checking ATL* *and* ATL$^+$

We now turn to model checking logics over broader subsets of $\mathscr{L}_{ATL^*}$. In the first case we consider perfect recall strategies in the perfect information setting. The complexity results established here are based on an automata-theoretic approach which is explained below.

Let $\mathfrak{M}$ be a CGS and $\langle\langle A\rangle\rangle\psi$ be an $\mathscr{L}_{ATL^*}$-formula (where we assume that ψ is an $\mathscr{L}_{LTL}$-formula). Given a strategy s_A of A and a state q in $\mathfrak{M}$ 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 strategy profile for A may result in a different execution tree. Now, a Büchi tree automaton $\mathscr{A}_{\mathfrak{M},q,A}$ can be constructed that accepts exactly the (q,A)-execution trees [4].

Secondly, it was shown that one can construct a Rabin tree automaton which accepts all trees that satisfy the $\mathscr{L}_{CTL^*}$-formula $\mathsf{A}\psi$ [17]. Hence, the $\mathscr{L}_{ATL^*}$-formula $\langle\langle A\rangle\rangle\psi$ is satisfied in $\mathfrak{M},q$ if there is a tree accepted by $\mathscr{A}_{\mathfrak{M},q,A}$ (i.e., it is a (q,A)-execution tree) and by $\mathscr{A}_\psi$ (i.e., it is a model of $\mathsf{A}\psi$).

Theorem 12 (ATL$^*_{IR}$ [4]). *Model checking* ATL$^*_{IR}$ *is 2EXPTIME-complete in the number of transitions in the model and the length of the formula.*

Proof (Sketch). We briefly analyze the complexity for the procedure described above. Firstly, the Büchi tree automaton $\mathscr{A}_{\mathfrak{M},q,A}$ is built by considering the states A is effective for [4]. That is, in a state of the automaton corresponding to a state $q \in St$ of $\mathfrak{M}$ the automaton nondeterministically chooses a sequence $(q'_1,q'_2,\ldots,q'_n)$ of successors of q such that A has a common action to guarantee that the system will end up in one of the states $\{q'_1,q'_2,\ldots,q'_n\}$ in the next step. It is assumed that the sequence is minimal. Incrementally, this models any s_A strategy of A and thus accepts all (q,A)-execution trees. The transition function of the automaton is constructed in the described way. As the number of transitions in each state of the automaton is bounded by the move combinations of agents A the size of the automaton, $|\mathscr{A}_{\mathfrak{M},q,A}|$, is bounded by $\mathbf{O}(|\mathfrak{M}|)$. All states are defined as acceptance states, such that $\mathscr{A}_{\mathfrak{M},q,A}$ accepts all possible execution trees of A.

Following the construction of [17], the automaton $\mathscr{A}_\psi$ is a Rabin tree automaton with $2^{2^{\mathbf{O}(|\psi|)}}$ states and $2^{\mathbf{O}(|\psi|)}$ Rabin pairs.

The product automaton $\mathscr{A}_\psi \times \mathscr{A}_{\mathfrak{M},q,A}$, accepting the trees accepted by both automata, is a Rabin tree automaton with $n := \mathbf{O}(|\mathscr{A}_\psi| \cdot |\mathscr{A}_{\mathfrak{M},q,A}|)$ many states and $r := 2^{\mathbf{O}(|\psi|)}$ many Rabin pairs (note that $\mathscr{A}_{\mathfrak{M},q,A}$ can be seen as a Rabin tree automaton with one Rabin pair composed of the states of the automaton and the empty set). Finally, to determine whether the language accepted by the product automaton is empty can be done in time $\mathbf{O}(n \cdot r)^{3r}$ [15, 43]; hence, the algorithm runs in time $|\mathfrak{M}|^{2^{\mathbf{O}(|\psi|)}}$ (it might be employed at each state of the model and for each subformula).

The lower bound is shown by a reduction of the *2EXPTIME*-complete problem of the realizability of **LTL**-formulae [43, 46, 4]. □

The next result shows that model checking $\mathscr{L}_{ATL^*}$ *with memoryless strategies* is no worse than for **LTL** and $\mathbf{CTL}^*$ for both perfect and imperfect information.

Theorem 13 ($\mathbf{ATL}^*_{ir}$ and $\mathbf{ATL}^*_{Ir}$ [49]). *Model checking $\mathbf{ATL}^*_{ir}$ and $\mathbf{ATL}^*_{Ir}$ is PSPACE-complete in the number of transitions in the model and the length of the formula.*

Proof (Sketch). $\mathscr{L}_{LTL}$ is contained in $\mathscr{L}_{ATL^*}$ which renders $\mathscr{L}_{ATL^*}$ with the perfect information memoryless semantics to be at least *PSPACE*-hard.

On the other hand, there is a *PSPACE* algorithm for model checking $\mathscr{L}_{ATL^*}$ with the imperfect information memoryless semantics. Consider the formula $\langle\langle A\rangle\rangle\psi$ where ψ is an $\mathscr{L}_{LTL}$-formula. Then, an *ir*-strategy s_A for A is guessed and the model is "trimmed" according to s_A, 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, and hence also using only polynomially many memory cells. In the new model the $\mathscr{L}_{CTL^*}$-formula $\mathsf{A}\psi$ is checked. This procedure can be performed in NP^{PSPACE}, which renders the complexity of the whole language to be in $P^{NP^{PSPACE}} = PSPACE$. □

We consider the more limited language $\mathscr{L}_{ATL^+}$. Boolean combinations of path formulae prevent us from using the fixed-point characterizations for model checking. Instead, given a formula $\langle\langle A\rangle\rangle\psi$ with no nested cooperation modalities, we can guess a (memoryless) strategy of A, "trim" the model accordingly, and model-check the $\mathscr{L}_{CTL^+}$-formula $\mathsf{A}\psi$ in the resulting model. Since the model checking problem for $\mathbf{CTL}^+$ is Δ_2^P-complete, we get that the overall procedure runs in time $\Delta_2^{P\Delta_2^P} = \Delta_3^P$ [49].

Theorem 14 ($\mathbf{ATL}^+_{ir}$ and$\mathbf{ATL}^+_{Ir}$ [49]). *Model checking $\mathbf{ATL}^+_{ir}$ and $\mathbf{ATL}^+_{Ir}$ is Δ_3^P-complete in the number of transitions in the model and the length of the formula.*

Proof (Sketch). The above procedure shows the membership. Note that in the incomplete information case one has to guess a *uniform* strategy. Again, it is essential that a strategy can be guessed in *polynomially many steps*, which is indeed the case for *Ir*- and *ir*-strategies. The hardness proof can be obtained by a reduction of the standard Δ_3^P-complete problem $\mathbf{SNSAT}_3$, cf. [49] for the construction. □

What about $\mathbf{ATL}^+_{IR}$? It has been believed that verification with $\mathscr{L}_{ATL^+}$ is Δ_3^P-complete for perfect recall strategies, too. However, it turns out that the complexity of $\mathbf{ATL}^+_{IR}$ model checking is much harder, namely *PSPACE* [8]. Since the Δ_3^P-completeness for memoryless semantics *is* correct, we get that memory makes verification harder already for $\mathscr{L}_{ATL^+}$, and not just for $\mathscr{L}_{ATL^*}$ as it was believed before.

Theorem 15 ($\mathbf{ATL}^+_{IR}$ [8]). *Model checking $\mathbf{ATL}^+_{IR}$ is PSPACE-complete with respect to the number of transitions in the model and the length of the formula. It is PSPACE-complete even for turn-based models with two agents and "flat" $\mathbf{ATL}^+$ formulae.*

	Ir	*IR*	*ir*	*iR*
Simple $\mathscr{L}_{CL}$	AC^0	AC^0	AC^0	AC^0
$\mathscr{L}_{CL}$	*P*	*P*	*P*	*P*
$\mathscr{L}_{ATL}$	*P*	*P*	Δ_2^P	Undecidable†
$\mathscr{L}_{ATL^+}$	Δ_3^P	*PSPACE*	Δ_3^P	Undecidable†
$\mathscr{L}_{ATL^*}$	*PSPACE*	*2EXPTIME*	*PSPACE*	Undecidable†

Fig. 6 Overview of the model checking complexity results for explicit models. All results except for "Simple **CL**" are completeness results. Each cell represents the logic over the language given in the row using the semantics given in the column. † These problems are believed to be undecidable, though no formal proof has been proposed yet (cf. Conjectures 1, 2, and 3).

Proof (Sketch). Consider the $\mathscr{L}_{ATL^+}$-formula $\langle\langle A\rangle\rangle\gamma$ where γ does not contain any further cooperation modalities. The upper bound can be proven by constructing an alternating Turing Machine that first produces (by alternatingly guessing the "best" choices of the proponents and the "most damaging" responses of the opponents) the relevant part of a path (whose length is asymptotically bounded by the product of the length of the formula and the number of states in the model) that suffices to determine the truth of an $\mathscr{L}_{ATL^+}$-formula. Then, we implement the game-theoretical semantics of propositional logic [23] as a game between the verifier (who controls disjunction) and the refuter (controlling conjunction). The machine runs in time $\mathbf{O}(nkl)$ where n (resp. k and l) denotes the number of states (resp. number of agents and length of the formula), cf. [8] for details.

Hardness is proved by a reduction of QSAT. A perfect recall strategy of the proponents is used to assign consistent valuations (step-by-step) to propositional variables that they control; analogously for the opponents. Thereby, the proponents control the existentially quantified variables and the opponents the universally quantified ones. An $\mathscr{L}_{ATL^+}$-formula is used to describe such valid assignments; i.e. truth values must be ascribed to variables in a uniform way. For the complete construction we refer to [8] again. □

Note that the input size only depends on the number of *states and agents* in the model and length of the formula which is important for the complexity result given in Theorem 25 about non-standard input measures.

The following conjectures are immediate consequences of Conjecture 1 as $\mathscr{L}_{ATL}$ is a fragment of $\mathscr{L}_{ATL^*}$ as well as $\mathscr{L}_{ATL^+}$.

Conjecture 2 ($\mathbf{ATL}^*_{\boldsymbol{iR}}$). Model checking $\mathbf{ATL}^*_{\boldsymbol{iR}}$ is undecidable.

Conjecture 3 ($\mathbf{ATL}^+_{\boldsymbol{iR}}$). Model checking $\mathbf{ATL}^+_{\boldsymbol{iR}}$ is undecidable.

Figure 3.4 presents an overview of the model checking complexity results for explicit models.

4 Complexity for Implicit Models: States and Agents

We have seen several complexity results for the model checking problem in logics like **LTL**, **CTL**, and **ATL**. Some of these results are quite attractive: one usually cannot hope to achieve verification with complexity better than *linear*.

However, it is important to remember that these results measure the complexity with respect to the *size of the underlying model*. Often, these models are so big, that an *explicit* representation is not possible and we have to represent the model in a "compressed" way. To give a simple illustration, consider the famed primality problem: checking whether a given natural number n is prime. The well-known algorithm uses $\sqrt{n}$-many divisions and thus runs in polynomial time *when the input is represented in unary*. But a symbolic representation of n needs only $\log(n)$ bits and thus the above algorithm runs in *exponential* time with respect to its size. This does not necessarily imply that the problem itself is of exponential complexity. In fact, the famous and deep result of Agrawal, Kayal and Saxena shows that the primality problem *can* be solved in polynomial time.

We will consider model checking of temporal and strategic logics for such highly compressed representations (in terms of *state space compression* and *modularization*) in Section 5. Such a rigorous compressed representation is not the only way in which the model checking complexity can be influenced. Another important factor is how we encode the transition function. So far, we assumed that the size of a model is measured with respect to the number of transitions in the model.

In this section we consider the complexity of the model checking problem *with respect to the number of states, agents, and an implicitly encoded transition function* rather than the (explicit) number of transitions. It is easy to see that, for CGS's, the number of transitions can be exponential in the number of states and agents. Therefore, all the algorithms presented in Section 3 give us only exponential time bounds provided that the transition function is encoded sufficiently small.

Observation 1 ([4, 31]) *Let n be the number of states in a concurrent game structure $\mathfrak{M}$, let k denote the number of agents, and d the maximal number of available decisions (moves) per agent per state. Then, $m = \mathbf{O}(nd^k)$. Therefore the* $\mathbf{ATL_{IR}}$ *model checking algorithm from [4] runs in time $\mathbf{O}(nd^k l)$, and hence its complexity is exponential if the number of agents is a parameter of the problem.*

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

Theorem 16. **CTL** *model checking over unlabeled transition systems is P-complete in the number of states and the length of the formula, and can be done in time $\mathbf{O}(n^2 l)$.*

For **ATL** and concurrent game structures, however, the situation is different. In the following we make precise what we mean by a compressed transition function.

Implicit concurrent game structures (called this way first in [36], but already present in the ISPL modeling language behind MCMAS [45, 44]) are defined similarly to a CGS but the transition function is encoded in a particular way often allowing for a more compact representation than the explicit transition table. Formally, an *implicit* CGS is given by $\mathfrak{M} = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, Act, d, \hat{o} \rangle$ where $\hat{o}$, the *encoded transition function*, is given by a sequence

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

where $t_r \in \mathbb{N}_0$, $q_i^r \in St$ and each φ_i^r is a Boolean combination of propositions exec_α^j where $j \in \mathbb{A}\mathrm{gt}$, $\alpha \in Act$, $i = 1, \ldots, t$ and $r = 1, \ldots, |Q|$. It is required that $\varphi_{t_r}^r = \top$. The term exec_α^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 exec_α^j with $\top$ (resp. $\bot$) if $\alpha_j = \alpha$ (resp. $\alpha_j \neq \alpha$). The encoded transition function induces a standard transition function $o_{\hat{o}}$ as follows:

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

That is, $o_{\hat{o}}(q_i, \alpha_1, \ldots, \alpha_k)$ 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 $\alpha_1, \ldots, \alpha_k$. We use $\hat{o}(q_i, \alpha_1, \ldots, \alpha_k)$ to refer to $o_{\hat{o}}(q_i, \alpha_1, \ldots, \alpha_k)$. 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 $|\hat{o}| = \sum_{r=1,\ldots,|Q|} \sum_{j=1,\ldots,t_r} |\varphi_j^r|$, that is, the sum of the sizes of all formulae. Hence, the size of an implicit CGS is given by $|St| + |\mathbb{A}\mathrm{gt}| + |\hat{o}|$. Recall, that the size of an explicit CGS is $|St| + |\mathbb{A}\mathrm{gt}| + m$ where m is the number of transitions. Finally, we require that the encoding of the transition function is reasonably compact, that is, $|\hat{o}| \leq \mathbf{O}(|o_{\hat{o}}|)$.

Now, why should the model checking complexity change for implicit CGS's? Firstly, one can observe that we can take the trivial encoding of an explicit transition function yielding an implicit CGS that has the same size as the explicit CGS. This implies that all the *lower bounds* proven before are still valid.

Proposition 1. *Model checking with respect to implicit* CGS*'s is at least as hard as model checking over explicit* CGS*'s for any logic discussed here.*

Therefore, we focus on the question whether model checking can become more difficult for implicit CGS's. Unfortunately, the answer is *yes*: Model checking can indeed become more difficult.

We illustrate this by considering the presented algorithm for solving the $\mathbf{ATL}_{IR}$ model checking problem. It traverses all transitions and since transitions are considered explicitly in the input, the algorithm runs in polynomial time. But if we choose an encoding $\hat{o}$ that is significantly smaller than the explicit number of transitions, the algorithm still has to check all transitions, yet now the number of transitions can be *exponential* with respect to the input of size $|St| + |\mathbb{A}\mathrm{gt}| + |\hat{o}|$.

Henceforth, we are interested in the cases in which the size of the encoded transition function is much smaller, in particular, when the size of the encoding is polyno-

mial with respect to the *number of states and agents*. This is the reason why we will often write that we measure the input in terms of states (n) and agents (k), neglecting the size of $\hat{o}$ when it is supposed to be polynomial in n,k.

Remark 5. An alternative view is to assume that the transition function is provided by an external procedure (a "black box") that runs in polynomial time, similar to an oracle [31]. This view comes along with some technical disadvantages, and we will not discuss it here.

4.1 Model Checking ATL *and* CL *in Terms of States and Agents*

As argued above the complexity of $\mathbf{O}(ml)$ may (but does not have to) include potential intractability if the transition function is represented more succinctly. The following result supports this observation.

Theorem 17 ([37, 31, 33]). *Model checking* $\mathbf{ATL}_{IR}$ *and* $\mathbf{ATL}_{Ir}$ *over implicit* CGS*'s is* Δ_3^P*-complete with respect to the size of the model and the length of the formula (l).*

Proof (Sketch). The idea of the proof for the lower bound is clear if we reformulate the model checking of $M,q \models \langle\langle a_1,\ldots,a_r\rangle\rangle \bigcirc \varphi$ as

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

which closely resembles QSAT_2, a typical Σ_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 $p_1,\ldots,p_r$ are "declared" by agents A and the values of $p_{r+1},\ldots,p_k$ 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 $p_1,\ldots,p_k$. Then, QSAT_2 reduces to model checking formula $\langle\langle a_1,\ldots,a_r\rangle\rangle \bigcirc \mathsf{yes}$ [31]. In order to obtain Δ_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 [37].

For the upper bound, we consider the following algorithm for checking $\mathfrak{M},q \models \langle\langle A\rangle\rangle \gamma$ with no nested cooperation modalities. Firstly, guess a strategy s_A of the proponents and fix A's actions to the ones described by s_A. Then 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). □

Surprisingly, the imperfect information variant of **ATL** is no harder than the perfect information one under this measure:

Theorem 18 ([33]). *Model checking* $\mathbf{ATL}_{ir}$ *over implicit* CGS*'s is* Δ_3^P*-complete with respect to the size of the model and the length of the formula. This is the same complexity as for model checking* $\mathbf{ATL}_{Ir}$ *and* $\mathbf{ATL}_{IR}$*.*

Proof (Sketch). For the upper bound, we use the same algorithm as in checking $\mathbf{ATL}_{Ir}$. For the lower bound, we observe that $\mathbf{ATL}_{Ir}$ can be embedded in $\mathbf{ATL}_{ir}$ by explicitly assuming perfect information of agents (through the minimal reflexive indistinguishability relations). □

The Δ_3^P-hardness proof in Theorem 17 uses the "nexttime" and "until" temporal operators in the construction of an **ATL** formula that simulates SNSAT_3 [37]. However, the proof can be modified so that only the "nexttime" sublanguage of $\mathscr{L}_{ATL}$ is used. We obtain thus an analogous result for coalition logic. Details of the new construction can be found in the technical report [7].

Theorem 19. *Model checking* $\mathbf{CL}_{IR}$*,* $\mathbf{CL}_{Ir}$*,* $\mathbf{CL}_{ir}$*, and* $\mathbf{CL}_{iR}$ *over implicit* CGS*'s is* Δ_3^P*-complete with respect to the size of the model and the length of the formula. Moreover, it is* Σ_2^P*-complete for the "simple" variants of* **CL***.*

It is worth mentioning that model checking "Positive **ATL**" (i.e., the fragment of $\mathscr{L}_{ATL}$ where negation is allowed only on the level of literals) is Σ_2^P-complete with respect to the size of implicit CGS's, and the length of formulae for the *IR*, *Ir*, and *ir*-semantics [33]. The same applies to "Positive **CL**", the analogous variant of coalition logic.

4.2 CTL *and* CTL$^+$ *Revisited*

At the beginning of Section 4, we mentioned that the complexity of model checking computation tree logic is still polynomial even if we measure the size of models with the number of states rather than transitions. That is certainly true for unlabeled transition systems (i.e., the original models of **CTL**). For concurrent game structures, however, this is no longer the case.

Theorem 20. *Model checking* **CTL** *over implicit* CGS*'s is* Δ_2^P*-complete with respect to the size of the model and the length of the formula.*

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

For the lower bound, we sketch the reduction of SAT to model checking $\mathscr{L}_{CTL}$-formulae with only one path quantifier. For propositional variables $p_1,\ldots,p_k$ and boolean formula φ, we construct an implicit CGS 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, SAT reduces to model checking formula $\langle\langle \mathbb{A}\mathrm{gt}\rangle\rangle \bigcirc \mathsf{yes}$. The reduction of SNSAT_2 (to model checking $\mathscr{L}_{CTL}$-formulae with nested path quantifiers) is an extension of the SAT reduction, analogous to the one in [32, 33]. □

As it turns out, the complexity of $\mathbf{CTL}^+$ does not increase when we change the models to implicit concurrent game structures: It is still Δ_2^P.

Theorem 21. *Model checking* $\mathbf{CTL}^+$ *over implicit* CGS*'s is* Δ_2^P*-complete with respect to the size of the model and the length of the formula.*

Proof (Sketch). The lower bound follows from Theorem 5 and Proposition 1.

For the upper bound, we observe that the $\mathbf{CTL}^+$ model checking algorithm in [38] verifies $\mathfrak{M},q \models \mathsf{E}\gamma$ by guessing a finite history h with length $|St_M|\cdot|\gamma|$, and then checking γ on h. We recall that $\mathsf{E}\gamma \equiv \langle\langle \mathbb{A}\mathrm{gt}\rangle\rangle\gamma$. Thus, for a concurrent game structure, each transition in h can be determined by guessing an action profile in $O(|\mathbb{A}\mathrm{gt}|)$ steps, calculating $\hat{o}$ wrt the guessed profile, and the final verification whether γ holds on the *finite* sequence h which can be done in deterministic polynomial time (cf. [8]). Consequently, we can implement this procedure by a nondeterministic Turing machine that runs in polynomial time. For nested path quantifiers, we proceed recursively which shows that the model checking problem can be solved by a polynomial time Turing machine with calls to an *NP*-oracle. □

We will use the last result in the analysis of $\mathbf{ATL}^+$ in Section 4.3.

4.3 $\mathbf{ATL}^*$ *and* $\mathbf{ATL}^+$

Theorem 22. *Model checking* $\mathbf{ATL}^*_{Ir}$ *and* $\mathbf{ATL}^*_{ir}$ *over implicit* CGS*'s is PSPACE-complete with respect to the size of the model and the length of the formula.*

Proof. The lower bound follows from Theorem 13 and Proposition 1.

For the upper bound, we model-check $\mathfrak{M},q \models \langle\langle A\rangle\rangle\gamma$ by guessing a memoryless strategy s_A for coalition A, then we guess a counterstrategy $s_{\bar{A}}$ of the opponents. Having a complete strategy profile, we proceed as in the proof of Theorem 20 and check the **LTL** path formula γ on the resulting (polynomial model) $\mathfrak{M}'$ which can be done in polynomial space (Theorem 13). For nested cooperation modalities, we proceed recursively. □

Theorem 23 ([37]). *Model checking* $\mathbf{ATL}^*_{IR}$ *over implicit* CGS*'s is 2EXPTIME-complete with respect to the size of the model and the length of the formula.*

	Ir	*IR*	*ir*	*iR*
Simple $\mathscr{L}_{CL}$	Σ_2^P	Σ_2^P	Σ_2^P	Σ_2^P
$\mathscr{L}_{CL}$	Δ_3^P	Δ_3^P	Δ_3^P	Δ_3^P
$\mathscr{L}_{ATL}$	Δ_3^P	Δ_3^P	Δ_3^P	Undecidable†
$\mathscr{L}_{ATL^+}$	Δ_3^P	*PSPACE*	Δ_3^P	Undecidable†
$\mathscr{L}_{ATL^*}$	*PSPACE*	*2EXPTIME*	*PSPACE*	Undecidable†

Fig. 7 Overview of the model checking complexity results for implicit CGS. All results are completeness results. Each cell represents the logic over the language given in the row using the semantics given in the column. † These problems are believed to be undecidable, though no formal proof has been proposed yet.

Proof. The lower bound follows from Theorem 12 and Proposition 1. For the upper bound, we have to modify the algorithm given in the proof of Theorem 12 such that it is capable of dealing with implicit models. More precisely, we need to modify the construction of the Büchi automaton $\mathscr{A}_{\mathfrak{M},q,A}$ 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 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 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. □

Theorem 24 ([37]). *Model checking* $\mathbf{ATL}^+_{\boldsymbol{Ir}}$ *and* $\mathbf{ATL}^+_{\boldsymbol{ir}}$ *over implicit* CGS*'s is* Δ_3^P*-complete with respect to the size of the model and the length of the formula.*

Proof. The lower bounds follow from Theorem 14 and Proposition 1. For the upper bound we model-check $\mathfrak{M},q \models \langle\langle A\rangle\rangle\gamma$ by guessing a memoryless strategy s_A for coalition A, and constructing an *unlabeled transition system* $\mathfrak{M}'$ as follows. For each state q_i we evaluate formulae contained in $((\varphi_0^i,q_0^i),\ldots,(\varphi_{t_i}^i,q_{t_i}^i))$ according to the guessed strategy. Then, we introduce a transition from q_i to q_j^i if $(\bigwedge_{k=0,\ldots,j-1}\neg\varphi_k^i)\wedge\varphi_j^i$ is satisfiable (i.e., there is a countermove of the opponents such that φ_i^j is true and j is the minimal index) . This is the case iff the opponents have a strategy to enforce the next state to be q_j^i. These polynomially many tests can be done by *independent* calls of an *NP*-oracle. The resulting model $\mathfrak{M}'$ is an explicit CGS of polynomial size regarding the number of states and agents. Finally, we apply $\mathbf{CTL}^+$ model checking to $\mathsf{A}\gamma$ which can be done in time Δ_2^P. □

Finally, we consider the case for perfect recall strategies. The lower and upper bound directly follow from the proof of Theorem 15.

Theorem 25 ([8]). *Model checking* $\mathbf{ATL}^+_{\boldsymbol{IR}}$ *over implicit* CGS*'s is PSPACE-complete with respect to the size of the model and the length of the formula.*

A summary of complexity results for the alternative representation/measure of the input is presented in Figure 7. It turns out that, when considering the finer-grained representation that comes along with a measure based on the number of

states, agents, and an encoded transition function rather than just the number of transitions, the complexity of model checking $\mathscr{L}_{ATL}$ seems distinctly harder than before for games with perfect information, and only somewhat harder for imperfect information. In particular, the problem falls into the same complexity classes for imperfect and perfect information analysis, which is rather surprising, considering the results from Section 3. Finally, the change of perspective does not influence the complexity of model checking of $\mathscr{L}_{ATL^*}$ as well as $\mathscr{L}_{ATL^+}$ at all.

5 Higher-Order Representations of Models

In this section, we summarize very briefly the results for higher-level representations of multi-agent systems (e.g., concurrent programs, reactive modules, modular interpreted systems etc.). Sections 3 and 4 presented complexity results for model checking with respect to models where global states of the system were represented *explicitly*. Most multi-agent systems, however, are characterized by an immensely huge state space. In such cases, one would like to define the model in terms of a *compact high-level representation*, plus an unfolding procedure that defines the relationship between representations and actual models of the logic (and hence also the semantics of the logic with respect to the compact representation). Of course, unfolding a higher-level description 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 $n = 2^r$. A more general approach is presented in [34], where the "high-level description" is defined in terms of *concurrent programs*, that can be used for simulating Boolean variables, but also for processes or agents acting in parallel.

A concurrent program P is composed of k concurrent processes, each described by a labeled transition system $P_i = \langle St_i, Act_i, \mathscr{R}_i, \Pi_i, \pi_i \rangle$, where St_i is the set of local states of process i, Act_i is the set of local actions, $\mathscr{R}_i \subseteq St_i \times Act_i \times St_i$ is a transition relation, and Π_i, π_i are the set of local propositions and their valuation. The behavior of program P is given by the product automaton of $P_1, \ldots, P_k$ under the assumption that processes work asynchronously, actions are interleaved, and synchronization is obtained through common action names.

Theorem 26 ([34]). *Model checking* **CTL** *in concurrent programs is PSPACE-complete with respect to the number of local states and agents (processes), and the length of the formula.*

Concurrent programs seem to be sufficient to reason about purely temporal properties of systems, but not quite so for reasoning about agents' strategies and abilities. For the latter kind of analysis, we need to allow for more sophisticated interference between agents' actions (and enable modeling agents that play synchronously). **ATL** model checking for higher-order representations was first analyzed in [24]

over a class of *simple reactive modules*, based on synchronous product of local models. However, even simple reactive modules do not allow to model interference between agents' actions. Because of that, we use *modular interpreted systems* [30, 28], that draw inspiration from interpreted systems [18], reactive modules [2], and are in many respects similar to ISPL specifications [44].

A modular interpreted system (MIS) is defined as a tuple $\mathfrak{M} = \langle \mathbb{A}\mathrm{gt}, env, Act, \mathscr{I}n \rangle$, where $\mathbb{A}\mathrm{gt} = \{a_1, \ldots, a_k\}$ is a set of agents, *env* is the environment, *Act* is a set of actions, and $\mathscr{I}n$ is a set of symbols called *interaction alphabet*. Each agent has the following internal structure: $a_i = \langle St_i, d_i, out_i, in_i, o_i, \Pi_i, \pi_i \rangle$, where:

- St_i is a set of local states,
- $d_i : St_i \to \mathscr{P}(Act)$ defines local availability of actions; for convenience we additionally define the set of *situated actions* as $D_i = \{\langle q_i, \alpha \rangle \mid q_i \in St_i, \alpha \in d_i(q_i)\}$,
- out_i, in_i are *interaction functions*; $out_i : D_i \to \mathscr{I}n$ refers to the influence that a given situated action (of agent a_i) may possibly have on the external world, and $in_i : St_i \times \mathscr{I}n^k \to \mathscr{I}n$ translates external manifestations of the other agents (and the environment) into the "impression" that they make on a_i's transition function depending on the local state of a_i,
- $o_i : D_i \times \mathscr{I}n \to St_i$ is a (deterministic) local transition function,
- Π_i is a set of local propositions of agent a_i where we require that Π_i and Π_j are disjunct when $i \neq j$, and
- $\pi_i : \Pi_i \to \mathscr{P}(St_i)$ is a valuation of these propositions.

The environment *env* has the same structure as an agent except that it does not perform actions.

The unfolding of a MIS $\mathfrak{M}$ to a concurrent game structure follows by the synchronous product of the agents (and the environment) in $\mathfrak{M}$, with interaction symbols being passed between local transition functions at every step. The unfolding can also determine indistinguishability relations as follows $\langle q_1, \ldots, q_k, q_{env} \rangle \sim_i \langle q'_1, \ldots, q'_k, q'_{env} \rangle$ iff $q_i = q'_i$, thus yielding a full *i*CGS. This way the semantics of both $\mathbf{ATL_{IR}}$/$\mathbf{ATL_{Ir}}$ and $\mathbf{ATL_{ir}}$ is extended to MIS.

Theorem 27 ([24]). *Model checking* $\mathbf{ATL_{Ir}}$ *and* $\mathbf{ATL_{IR}}$ *in simple reactive modules is EXPTIME-complete with respect to the number of local states and agents, and the length of the formula.*

Since simple reactive modules can be embedded in modular interpreted systems, and the model checking algorithm from [24] can be extended to MIS, we get the following.

Theorem 28 ([28]). *Model checking* $\mathbf{ATL_{Ir}}$ *and* $\mathbf{ATL_{IR}}$ *in modular interpreted systems is EXPTIME-complete with respect to the number of local states and agents, and the length of the formula.*

Note that this means that systems with no interference between agents are *not easier to handle* than the general case.

The real surprise, however, comes to light when we study the model checking complexity for imperfect information agents.

Theorem 29 ([30, 28]). *Model checking* $\mathbf{ATL}_{ir}$ *in modular interpreted systems is PSPACE-complete with respect to the number of local states and agents, and the length of the formula.*

Thus, model checking in modular interpreted systems seems to be easier for imperfect rather than perfect information strategies (while it appears to be distinctly harder for explicit models, cf. Section 3). There are two reasons for that. 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 agents. More precisely, the number of perfect information strategies is *double exponential* with respect to the number of 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 MIS, modular interpreted systems may provide more compact representation to systems where all the agents have perfect information by definition. In particular, the most compact MIS representation of a given ICGS $\mathfrak{M}$ can be exponentially larger than the most compact MIS representation of $\mathfrak{M}$ with the epistemic relations removed. In the former case, the MIS must encode the epistemic relations explicitly. In the latter case, the epistemic aspect is ignored, which gives some extra room for encoding the transition relation more efficiently.

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 $\mathbf{ATL}_{IR}$ is *EXPTIME*-complete and model checking $\mathbf{ATL}_{ir}$ is *PSPACE*-complete, which suggests that imperfect information can be beneficial in practical verification.

Finally, we report two results that are straightforward extensions of Theorem 19 and Theorem 29, respectively.

Theorem 30. *Model checking* $\mathbf{CL}_{IR}$, $\mathbf{CL}_{Ir}$, $\mathbf{CL}_{ir}$, *and* $\mathbf{CL}_{iR}$ *is* Δ_3^P*-complete with respect to the number of local states and agents in the* MIS *and the length of the formula. Moreover, it is* Σ_2^P*-complete for the "simple" variants of* $\mathbf{CL}$.

Theorem 31. *Model checking* $\mathbf{ATL}^{+}_{ir}$ *and* $\mathbf{ATL}^{*}_{ir}$ *in modular interpreted systems is PSPACE-complete with respect to the number of local states and agents, and the length of the formula.*

6 Summary

Figure 8 gives a summary of the results. The results for $\mathbf{ATL}_{Ir}$ and $\mathbf{ATL}_{ir}$ form an intriguing pattern. When we compare model checking agents with *perfect vs. imperfect* information, the first problem appears to be much easier against explicit

Logic \ Input	m,l	n,k,l	n_{local},k,l
Simple $\mathbf{CL}_{\mathbf{iR,ir,IR,Ir}}$	AC^0 [37]	Σ_2^P-complete	Σ_2^P-complete
$\mathbf{CL}_{\mathbf{iR,ir,IR,Ir}}$	P-complete	Δ_3^P-complete	Δ_3^P-complete
$\mathbf{ATL}_{\mathbf{Ir,IR}}$	P-complete [4]	Δ_3^P-compl. [37, 33]	*EXPTIME*-compl. [24]
$\mathbf{ATL}_{\mathbf{ir}}$	Δ_2^P-compl. [49, 33]	Δ_3^P-compl. [33]	*PSPACE*-compl. [28]
$\mathbf{ATL}_{\mathbf{iR}}$	Undecidable†	Undecidable†	Undecidable†
$\mathbf{ATL}^+_{\mathbf{Ir}}$	Δ_3^P-complete [49]	Δ_3^P-complete	*EXPTIME*-hard
$\mathbf{ATL}^+_{\mathbf{ir}}$	Δ_3^P-complete [49]	Δ_3^P-complete	*PSPACE*-complete
$\mathbf{ATL}^+_{\mathbf{IR}}$	*PSPACE*-complete [8]	*PSPACE*-complete [8]	*EXPTIME*-hard
$\mathbf{ATL}^+_{\mathbf{iR}}$	Undecidable†	Undecidable†	Undecidable†
$\mathbf{ATL}^*_{\mathbf{Ir}}$	*PSPACE*-complete [49]	*PSPACE*-complete	*EXPTIME*-hard
$\mathbf{ATL}^*_{\mathbf{ir}}$	*PSPACE*-compl. [49]	*PSPACE*-complete	*PSPACE*-complete
$\mathbf{ATL}^*_{\mathbf{IR}}$	*2EXPTIME*-compl. [4]	*2EXPTIME*-compl.	*EXPTIME*-hard
$\mathbf{ATL}^*_{\mathbf{iR}}$	Undecidable†	Undecidable†	Undecidable†

Fig. 8 Overview of the complexity results. All results except for "Simple **CL**" are completeness results. The results with no given reference have been established in this chapter for the first time (usually by a simple extension of existing proofs). The fields that report only hardness results correspond to problems which are still open. Symbols n and m stand for the number of states and transitions, respectively, and k is the number of agents in the model, l is the length of the formula, and n_{local} is the number of local states in a concurrent program, simple reactive module, or modular interpreted system. † These problems are believed to be undecidable, though no formal proof has been proposed yet (cf. Conjectures 1, 2, and 3).

models measured with the number of transitions. Then, we get the same complexity class against explicit models measured with the number of states and agents. Finally, model checking imperfect information turns out to be *easier* than model checking perfect information for modular interpreted systems. Why is that so?

The *number of available strategies* (relative to the size of input parameters) is the crucial factor here. It is *exponential in the number of global states*. For uniform strategies, there are usually much less of them but still exponentially many in general. Thus, the fact that perfect information strategies can be synthesized incrementally has a substantial impact on the complexity of the problem. However, measured in terms of local states and agents, the number of all strategies is *double exponential*, while there are "only" exponentially many uniform strategies– which settles the results in favor of imperfect information. It should be also noted that the *representation* of a concurrent game structure by a MIS can be in general more compact than that of an *i*CGS. In the latter case, the MIS is assumed to encode the epistemic relations explicitly. In the case of CGS, the epistemic aspect is ignored, which gives some extra room for encoding the transition relation more efficiently.

What have we learned and what are the challenges ahead? An important outcome of theoretical research on verification is to determine the precise boundary between model checking problems that are decidable and those that are not. As we have shown, decidability depends very much on the underlying language, the chosen logic and whether we consider perfect or imperfect recall. But even if the

problem is decidable, the precise complexity of the problem depends on the chosen representation and ranges from *P*- to *2EXPTIME*-completeness.

It must be noted that Figure 8 is filled mostly with complexity classes that are generally considered intractable. Of these, the undecidability hypotheses for $\mathbf{ATL}_{iR}$ are obviously the most pessimistic. But what does an undecidability result tell us? It shows that there is no general algorithm solving the problem at hand. Yet one is often not interested in model checking *all* possible specifications that can be expressed in the underlying logic. For most practical purposes, the set of interesting formulas to be model checked is quite small. This raises the question: *Which subsets of the logics are decidable?* A similar question can be stated for the complexity results reported here: *2EXPTIME* completeness concerns all formulas of $\mathscr{L}_{ATL^*}$, but suitable fragments can have much lower complexity. These are interesting questions to be investigated in the future.

References

1. Alur, R., Henzinger, T., Mang, F., Qadeer, S., Rajamani, S., Tasiran, S.: MOCHA user manual. In: Proceedings of CAV'98, *Lecture Notes in Computer Science*, vol. 1427, pp. 521–525 (1998)
2. Alur, R., Henzinger, T.A.: Reactive modules. Formal Methods in System Design **15**(1), 7–48 (1999)
3. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS), pp. 100–109. IEEE Computer Society Press (1997)
4. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM **49**, 672–713 (2002)
5. Beeri, C.: On the menbership problem for functional and multivalued dependencies in relational databases. ACM Trans. Database Syst. **5**(3), 241–259 (1980)
6. Broersen, J., Herzig, A., Troquard, N.: A STIT-extension of ATL. In: JELIA, pp. 69–81 (2006)
7. Bulling, N.: Model checking coalition logic on implicit models is Δ_3-complete. In: IfI Technical Reports (2010)
8. Bulling, N., Jamroga, W.: Verifying agents with memory is harder than it seemed. In: Proceedings of AAMAS 2010. ACM Press, Toronto, Canada (2010)
9. Clarke, E., Emerson, E.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Proceedings of Logics of Programs Workshop, *Lecture Notes in Computer Science*, vol. 131, pp. 52–71 (1981)
10. Clarke, E., Emerson, E., Sistla, A.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Transactions on Programming Languages and Systems **8**(2), 244–263 (1986)
11. Dembiński, P., Janowska, A., Janowski, P., Penczek, W., Półrola, A., Szreter, M., Woźna, B., Zbrzezny, A.: Verics: A tool for verifying timed automata and estelle specifications. In: Proceedings of the of the 9th Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'03), *LNCS*, vol. 2619, pp. 278–283. Springer (2003)
12. van Drimmelen, G.: Satisfiability in alternating-time temporal logic. In: Proceedings of LICS'2003, pp. 208–217. IEEE Computer Society Press (2003)
13. Emerson, E., Halpern, J.: "sometimes" and "not never" revisited: On branching versus linear time temporal logic. Journal of the ACM **33**(1), 151–178 (1986)
14. Emerson, E.A.: Temporal and modal logic. In: J. van Leeuwen (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 995–1072. Elsevier Science Publishers (1990)

15. Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs. In: SFCS '88: Proceedings of the 29th Annual Symposium on Foundations of Computer Science, pp. 328–337. IEEE Computer Society, Washington, DC, USA (1988)
16. Emerson, E.A., Lei, C.L.: Modalities for model checking: Branching time logic strikes back. Science of Computer Programming **8**(3), 275–306 (1987)
17. Emerson, E.A., Sistla, A.P.: Deciding branching time logic. In: STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing, pp. 14–24. ACM, New York, NY, USA (1984)
18. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press: Cambridge, MA (1995)
19. Furst, M., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Math. Systems Theory **17**, 13–27 (1984)
20. Goranko, V., van Drimmelen, G.: Complete axiomatization and decidability of the Alternating-time Temporal Logic (2003)
21. Goranko, V., Jamroga, W.: Comparing semantics of logics for multi-agent systems. Synthese **139**(2), 241–280 (2004)
22. Harding, A., Ryan, M., Schobbens, P.Y.: Approximating ATL* in ATL. In: VMCAI '02: Revised Papers from the Third International Workshop on Verification, Model Checking, and Abstract Interpretation, pp. 289–301. Springer-Verlag, London, UK (2002)
23. Hintikka, J.: Logic, Language Games and Information. Clarendon Press : Oxford (1973)
24. van der Hoek, W., Lomuscio, A., Wooldridge, M.: On the complexity of practical ATL model checking. In: P. Stone, G. Weiss (eds.) Proceedings of AAMAS'06, pp. 201–208 (2006)
25. van der Hoek, W., Wooldridge, M.: Cooperation, knowledge and time: Alternating-time Temporal Epistemic Logic and its applications. Studia Logica **75**(1), 125–157 (2003)
26. Immerman, N.: Number of quantifiers is better than number of tape cells. Journal of Computer and System Sciences **22**(3), 384 – 406 (1981)
27. Jamroga, W.: A temporal logic for stochastic multi-agent systems. In: Proceedings of PRIMA'08, *LNCS*, vol. 5357, pp. 239–250 (2008)
28. Jamroga, W., Ågotnes, T.: Modular interpreted systems: A preliminary report. Tech. Rep. IfI-06-15, Clausthal University of Technology (2006)
29. Jamroga, W., Ågotnes, T.: Constructive knowledge: What agents can achieve under incomplete information. Journal of Applied Non-Classical Logics **17**(4), 423–475 (2007)
30. Jamroga, W., Ågotnes, T.: Modular interpreted systems. In: Proceedings of AAMAS'07, pp. 892–899 (2007)
31. Jamroga, W., Dix, J.: Do agents make model checking explode (computationally)? In: M. Pěchouček, P. Petta, L. Varga (eds.) Proceedings of CEEMAS 2005, *Lecture Notes in Computer Science*, vol. 3690, pp. 398–407. Springer Verlag (2005)
32. Jamroga, W., Dix, J.: Model checking ATL$_{ir}$ is indeed Δ_2^P-complete. In: Proceedings of EUMAS'06 (2006)
33. Jamroga, W., Dix, J.: Model checking abilities of agents: A closer look. Theory of Computing Systems **42**(3), 366–410 (2008)
34. Kupferman, O., Vardi, M., Wolper, P.: An automata-theoretic approach to branching-time model checking. Journal of the ACM **47**(2), 312–360 (2000)
35. Laroussinie, F.: About the expressive power of CTL combinators. Information Processing Letters **54**(6), 343–345 (1995)
36. Laroussinie, F., Markey, N., Oreiby, G.: Expressiveness and complexity of ATL. Tech. Rep. LSV-06-03, CNRS & ENS Cachan, France (2006)
37. Laroussinie, F., Markey, N., Oreiby, G.: On the expressiveness and complexity of atl. LMCS **4**, 7 (2008)
38. Laroussinie, F., Markey, N., Schnoebelen, P.: Model checking CTL+ and FCTL is hard. In: Proceedings of FoSSaCS'01, *Lecture Notes in Computer Science*, vol. 2030, pp. 318–331. Springer (2001)
39. Lichtenstein, O., Pnueli, A.: 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, pp. 97–107. ACM, New York, NY, USA (1985)

40. van Otterloo, S., van der Hoek, W., Wooldridge, M.: Knowledge as strategic ability. Electronic Lecture Notes in Theoretical Computer Science **85**(2) (2003)
41. Pauly, M.: A modal logic for coalitional power in games. Journal of Logic and Computation **12**(1), 149–166 (2002)
42. Pnueli, A.: The temporal logic of programs. In: Proceedings of FOCS, pp. 46–57 (1977)
43. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL '89: Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 179–190. ACM, New York, NY, USA (1989)
44. Raimondi, F.: Model checking multi-agent systems. Ph.D. thesis, University College London (2006)
45. Raimondi, F., Lomuscio, A.: Automatic verification of deontic interpreted systems by model checking via OBDD's. In: R. de Mántaras, L. Saitta (eds.) Proceedings of ECAI, pp. 53–57 (2004)
46. Rosner, R.: Modular synthesis of reactive systems. Ph.D. thesis, Weizmann Institute of Science (1992)
47. Schewe, S.: ATL* satisfiability is 2ExpTime-complete. In: Proceedings of ICALP 2008, *Lecture Notes in Computer Science*, vol. 5126, pp. 373–385. Springer-Verlag (2008)
48. Schnoebelen, P.: The complexity of temporal model checking. In: Advances in Modal Logics, Proceedings of AiML 2002. World Scientific (2003)
49. Schobbens, P.Y.: Alternating-time logic with imperfect recall. Electronic Notes in Theoretical Computer Science **85**(2) (2004)
50. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. Journal of ACM **32**(3), 733–749 (1985)
51. Vardi, M.Y., Wolper, P.: 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), pp. 332–344. IEEE Computer Society Press (1986)
52. Walther, D., Lutz, C., Wolter, F., Wooldridge, M.: ATL satisfiability is indeed EXPTIME-complete. Journal of Logic and Computation **16**(6), 765–787 (2006)
53. Wilke, T.: CTL+ is exponentially more succint than CTL. In: Proceedings of FST&TCS '99, *LNCS*, vol. 1738, pp. 110–121 (1999)