Easy Yet Hard: Model Checking Strategies of Agents

Wojciech Jamroga

Department of Informatics, Clausthal University of Technology, Germany
wjamroga@in.tu-clausthal.de

Abstract. I present an overview of complexity results for model checking of temporal and strategic logics. Unfortunately, it is possible to manipulate the context so that different complexity results are obtained for the same problem. Among other things, this means that the results are often distant from the "practical" complexity which is encountered when one tries to use the formalisms in reality.

1 Introduction

A study of computational complexity is nowadays almost obligatory in a paper on logic in AI. Authors usually study the complexity of model checking and/or satisfiability checking of their logic in order to back the usefulness of the proposal with a formal argument. Unfortunately, the results are often far from the "practical" complexity which is encountered when one tries to use the formalisms in reality. Moreover, it is possible to manipulate the context so that different complexity results are obtained for the same problem. In this paper, I present a brief overview of complexity results for model checking temporal and strategic logics. Three logics are discussed here, namely computation tree logic CTL, alternating-time temporal logic ATL, and alternating-time logic with imperfect information and imperfect recall ATL_{ir}. For these logics, I show how the complexity class of the model checking problem changes when we change the way we represent input and/or measure its size.

Does it mean that theoretical complexity results are not worth anything in practice? Not necessarily – but certainly one needs to take these results with a grain of salt. In most cases, only a more extensive study (carried out from several different perspectives) can give us a meaningful picture of the *real* computational difficulty behind the problem.

2 The Logics

2.1 CTL: Branching Time and Temporal Evolution

Computation tree logic CTL [4, 6] explicitly refers to patterns of properties that can occur along a particular temporal path, as well as to the set of possible time series. The first dimension is captured by *temporal operators*: "$\bigcirc$" ("in the

next state"), $\Box$ ("always from now on") and $\mathcal{U}$ ("until"). Additional operator $\Diamond$ ("sometime from now on") can be defined as $\Diamond\varphi \equiv \top\,\mathcal{U}\,\varphi$. The second dimension is handled by so called *path quantifiers*: E ("there is a path") and A ("for all paths"). In CTL, every occurrence of a temporal operator is preceded by exactly one path quantifier.[1] Formally, the recursive definition of CTL formulae is:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{E}\bigcirc\varphi \mid \mathsf{E}\Box\varphi \mid \mathsf{E}\varphi\,\mathcal{U}\,\varphi.$$

A is derived from E in the usual way (cf., e.g., [13]).

The semantics of CTL is defined over *unlabeled transition systems*, i.e., tuples $M = \langle St, \mathcal{R}, \Pi, \pi\rangle$ where St is a nonempty set of states (or possible worlds), $\mathcal{R} \subseteq St \times St$ is a serial transition relation on states, Π is a set of atomic propositions, and $\pi : \Pi \to 2^{St}$ is a valuation of propositions. A *path* (or *computation*) in M is an infinite sequence of states that can result from subsequent transitions, and refers to a possible course of action. Let $\lambda[i]$ denote the ith position in computation λ (starting from $i = 0$). The meaning of CTL formulae is given by the following clauses:

$M, q \models p$ iff $q \in \pi(p)$ (where $p \in \Pi$);
$M, q \models \neg\varphi$ iff $M, q \not\models \varphi$;
$M, q \models \varphi \wedge \psi$ iff $M, q \models \varphi$ and $M, q \models \psi$;
$M, q \models \mathsf{E}\bigcirc\varphi$ iff there is a path λ such that $\lambda[0] = q$ and $M, \lambda[1] \models \varphi$;
$M, q \models \mathsf{E}\Box\varphi$ iff there is a path λ such that $\lambda[0] = q$ and $M, \lambda[i] \models \varphi$ for every $i \geq 0$;
$M, q \models \mathsf{E}\varphi\,\mathcal{U}\,\psi$ iff there is a path λ with $\lambda[0] = q$, and position $i \geq 0$ such that $M, \lambda[i] \models \psi$ and $M, \lambda[j] \models \varphi$ for each $0 \leq j < i$.

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.

As an example CTL property, we have $M_0, q_0 \models \mathsf{E}\Diamond\mathsf{pos}_1$: 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 $M_0, q_0 \models \neg\mathsf{A}\Diamond\mathsf{pos}_1$.

2.2 ATL: A Logic of Strategic Ability

ATL [2, 3] is a generalization of CTL, in which path quantifiers are replaced with so called *cooperation modalities*. Formula $\langle\langle A\rangle\rangle\varphi$ expresses that coalition A has

[1] This variant of the language is sometimes called "vanilla" CTL. The broader language of CTL*, where no such restriction is imposed, is not discussed here.

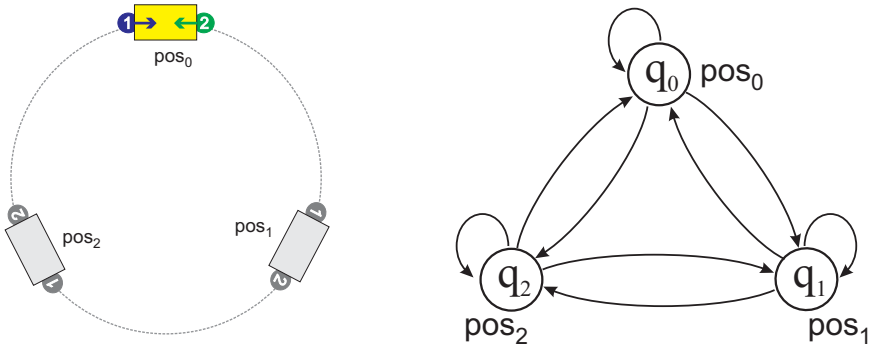

Fig. 1. Two robots and a carriage: a schematic view (left) and a transition system M_0 that models the scenario (right).

a collective strategy to enforce φ. The recursive definition of ATL formulae is:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle A\rangle\rangle \bigcirc \varphi \mid \langle\langle A\rangle\rangle \Box \varphi \mid \langle\langle A\rangle\rangle \varphi \mathcal{U} \varphi.$$

The semantics of ATL is defined in 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 $M = \langle \mathbb{A}\mathrm{gt}, St, \Pi, \pi, Act, d, o\rangle$ 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 π, and a nonempty finite set of (atomic) actions Act. Function $d : \mathbb{A}\mathrm{gt} \times St \to 2^{Act}$ 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 $\langle \alpha_1, \ldots, \alpha_k\rangle$, $\alpha_i \in d(i, q)$, that can be executed by $\mathbb{A}\mathrm{gt}$ in 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 possible state. Thus, a strategy can be represented with a function $s_a : St \to Act$, such that $s_a(q) \in d_a(q)$. A *collective strategy* for a group of agents $A = \{a_1, ..., a_r\}$ is simply a tuple of strategies $s_A = \langle s_{a_1}, ..., s_{a_r}\rangle$, one per agent from A.[2] By $s_A[a]$, we will denote agent a's part of the collective strategy s_A. 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:

[2] This is an important deviation from the original semantics of ATL [2, 3], where strategies assign agents' choices to *sequences* of states. While the choice of one or another notion of strategy affects the semantics of most extensions of ATL (e.g. for abilities under imperfect information), it should be pointed out that both types of strategies yield equivalent semantics for "pure" ATL [17].

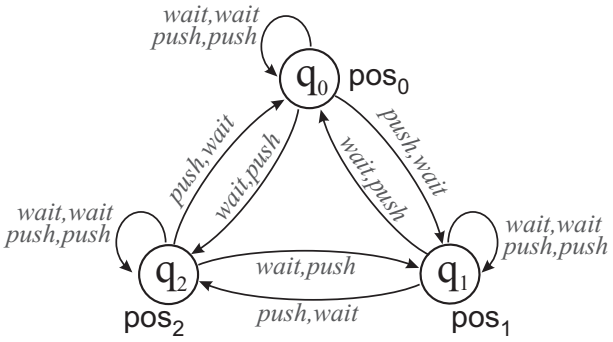

Fig. 2. The robots and the carriage: a concurrent game structure M_1.

$out(q, s_A) = \{\lambda = q_0 q_1 q_2 ... \mid q_0 = q$ and for each $i = 1, 2, ...$ there exists a tuple of agents' decisions $\langle \alpha_{a_1}^{i-1}, ..., \alpha_{a_k}^{i-1} \rangle$ such that $\alpha_a^{i-1} \in d_a(q_{i-1})$ for every $a \in \mathbb{A}$gt, and $\alpha_a^{i-1} \in s_A[a](q_{i-1})$ for every $a \in A$, and $o(q_{i-1}, \alpha_{a_1}^{i-1}, ..., \alpha_{a_k}^{i-1}) = q_i\}$.

The semantics of cooperation modalities is defined through the clauses below. Informally speaking, $M, q \models \langle\langle A \rangle\rangle \Phi$ iff there exists a collective strategy s_A such that Φ holds for all computations from $out(q, s_A)$.

- $M, q \models \langle\langle A \rangle\rangle \bigcirc \varphi$ iff there is a collective strategy s_A such that, for each path $\lambda \in out(s_A, q)$, we have $M, \lambda[1] \models \varphi$;
- $M, q \models \langle\langle A \rangle\rangle \Box \varphi$ iff there exists s_A such that, for each $\lambda \in out(s_A, q)$, we have $M, \lambda[i] \models \varphi$ for every $i \geq 0$;
- $M, q \models \langle\langle A \rangle\rangle \varphi \mathcal{U} \psi$ iff there exists s_A such that, for each $\lambda \in out(s_A, q)$, there is $i \geq 0$ for which $M, \lambda[i] \models \psi$, and $M, \lambda[j] \models \varphi$ for each $0 \leq j < i$.

Example 2 (Robots and Carriage, ctd.). Transition system M_0 enabled us to study the evolution of the system as a whole. However, it did not allow us to represent *who* can do *what*, and how the possible actions of the agents interact. Concurrent game structure 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 $M_1, q_0 \models \neg\langle\langle 1 \rangle\rangle \Diamond \mathsf{pos}_1$. On the other hand, the agent can at least make sure that the carriage will *avoid* particular positions. For instance, it holds that $M_1, q_0 \models \langle\langle 1 \rangle\rangle \Box \neg \mathsf{pos}_1$, the right strategy being $s_1(q_0) = wait, s_1(q_2) = push$ (the action that we specify for q_1 is irrelevant).

Note that the CTL path quantifiers A and E can be embedded in ATL in the following way: $\mathsf{A}\varphi \equiv \langle\langle \varnothing \rangle\rangle \varphi$ and $\mathsf{E}\varphi \equiv \langle\langle \mathbb{A}\text{gt} \rangle\rangle \varphi$.

2.3 Strategic Abilities under Imperfect Information

ATL and its models include no way of addressing uncertainty that an agent or a process may have about the current situation. Here, we take Schobbens' ATL_{ir} [17] as the "core", minimal ATL-based language for strategic ability under imperfect information. ATL_{ir} includes the same formulae as ATL, only 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. Models of ATL_{ir}, *imperfect information concurrent game structures* (i-CGS), 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, while the system is in state q, agent a considers it possible that it is in q'. Each $\sim_a$ is assumed to be an equivalence. It is also required that agents have the same choices in indistinguishable states: if $q \sim_a q'$ then $d(a, q) = d(a, q')$.

Again, a *strategy* of an agent a is a conditional plan that specifies what a is going to do in each possible state. An executable (deterministic) plan must prescribe the same choices for indistinguishable states. Therefore ATL_{ir} restricts the strategies that can be used by agents to the set of so called uniform strategies. A *uniform strategy* of agent a is defined as a function $s_a : St \to Act$, such that: (1) $s_a(q) \in d(a, q)$, and (2) if $q \sim_a q'$ then $s_a(q) = s_a(q')$. A collective strategy is uniform if it contains only uniform individual strategies. Again, function $out(q, s_A)$ returns the set of all paths that may result from agents A executing strategy s_A from state q onward. The semantics of cooperation modalities in ATL_{ir} is defined as follows:

- $M, q \models \langle\langle A\rangle\rangle_{ir} \bigcirc \varphi$ iff there exists a uniform collective strategy s_A such that, for each $a \in A$, q' such that $q \sim_a q'$, and path $\lambda \in out(s_A, q')$, we have $M, \lambda[1] \models \varphi$;
- $M, q \models \langle\langle A\rangle\rangle_{ir} \Box \varphi$ iff there is a uniform s_A such that, for each $a \in A$, q' such that $q \sim_a q'$, and $\lambda \in out(s_A, q')$, we have $M, \lambda[i] \models \varphi$ for each $i \geq 0$;
- $M, q \models \langle\langle A\rangle\rangle_{ir} \varphi \,\mathcal{U}\, \psi$ iff there exists a uniform strategy s_A such that, for each $a \in A$, q' such that $q \sim_a q'$, and $\lambda \in out(s_A, q')$, there is $i \geq 0$ for which $M, \lambda[i] \models \psi$, and $M, \lambda[j] \models \varphi$ for every $0 \leq j < i$.

That is, $\langle\langle A\rangle\rangle_{ir} \Phi$ if agents A have a uniform strategy such that, for each path that can possibly result from execution of the strategy *according to at least one agent from* A, Φ is the case.

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 2.3). In 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: every time

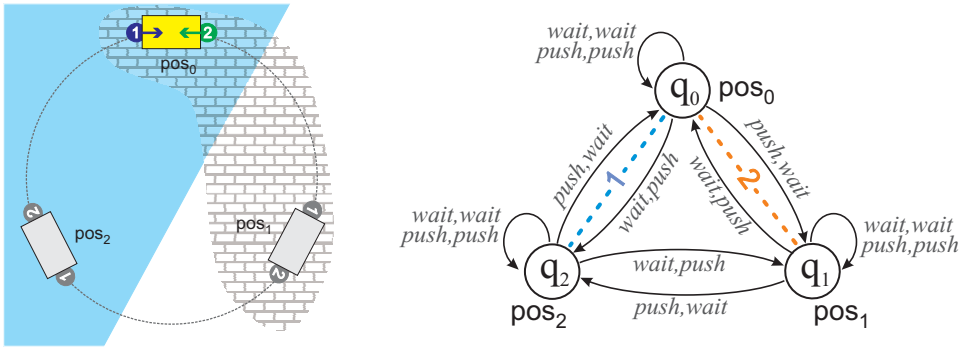

Fig. 3. Two robots and a carriage: a schematic view (left) and an imperfect information concurrent game structure M_2 that models the scenario (right).

they come back to the same position, their knowledge of the current situation is limited in the same way as before.

With its observational capabilities restricted in such way, no agent can make the carriage reach or avoid any selected states singlehandedly. E.g., we have that $M_2, q_0 \models \neg\langle\langle 1\rangle\rangle_{ir} \Box \neg \mathsf{pos}_1$; 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 it). The robots cannot even be sure to achieve the task together: $M_2, q_0 \models \neg\langle\langle 1,2\rangle\rangle_{ir} \Box \mathsf{pos}_1$ (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: $M_2, q_0 \models \langle\langle 1,2\rangle\rangle_{ir} \Diamond \mathsf{pos}_1$ etc. – it suffices that one of the robots pushes all the time and the other waits all the time.

3 A Survey of Model Checking Complexity Results

3.1 Model Checking Is Easy

It has been known for a long time that formulae of CTL can be checked in time linear with respect to the size of the model and the length of the formula. One of the main results concerning ATL states that its formulae can also be model-checked in deterministic linear time.

Theorem 1 ([5]). *Model checking* CTL *is* **P***-complete, and can be done in time* $\mathbf{O}(ml)$*, where* m *is the number of transitions in the model and* l *is the length of the formula.*

Theorem 2 ([3]). *Model checking* ATL *is* **P***-complete, and can be done in time* $\mathbf{O}(ml)$*, where* m *is the number of transitions in the model and* l *is the length of the formula.*

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

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 $\varphi = \langle\langle A\rangle\rangle \psi_1 \,\mathcal{U}\, \psi_2$ **:**
  $Q_1 := \varnothing$; $Q_2 := mcheck(M, \psi_1)$;
  $Q_3 := mcheck(M, \psi_2)$;
  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. 4. The ATL model checking algorithm from [3]

The ATL model checking algorithm from [3] is presented in Figure 4. The algorithm uses the well-known fixpoint characterizations of strategic-temporal modalities:

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

and computes a winning strategy step by step (if it exists). That is, it starts with the appropriate candidate set of states ($\emptyset$ for $\mathcal{U}$ and St for $\Box$), and iterates 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.

In contrast, analogous fixpoint characterizations do not hold for ATL$_{ir}$ modalities because the choice of a particular action at state q has non-local consequences: it automatically fixes choices at all states q' indistinguishable from q for the coalition A. Moreover, 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, which is indeed confirmed by the following (rather pessimistic) result.

Theorem 3 ([17, 11]). *Model checking* ATL$_{ir}$ *is* $\mathbf{\Delta_2^P}$*-complete in the number of transitions in the model and the length of the formula.*

Still, model checking CTL and ATL appear to be tractable. So... let's model check! Unfortunately, it turns out to be not as easy as it seems.

3.2 Model Checking Is Harder

The results from [5, 3] are certainly attractive, but it should be kept in mind that they are only relative to the size of models and formulae, and these can be very large for most application domains. Indeed, it is known that the number of states in a model is usually exponential in the size of a higher-level description of the problem domain for both CTL and ATL models. We will discuss this case in more detail in Section 3.3. In this section, we still consider model checking with respect to transition systems, concurrent game structures, and i-CGS's,[3] but we measure the size of the input in a slightly different way.

The size of a model has been defined as the number of transitions (m). Why not states then? For CTL this would not change the picture much. Let us denote the number of states by n; then, for any unlabeled transition system we have that $m = \mathbf{O}(n^2)$. In consequence, CTL model checking is in $\mathbf{P}$ also with respect to the number of states in the model (and the length of the formula). For ATL, however, the situation is different.

Observation 1 ([3, 10]) *Let n be the number of states in a concurrent game structure M. The number of transitions in M is* not *bounded by n^2, because transitions are labeled with tuples of agents' choices.*

Let k denote the number of agents, and d the maximal number of available decisions per agent per state. Then, $m = \mathbf{O}(nd^k)$. In consequence, the ATL *model checking algorithm from [3] 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.*

As we see, the complexity of $\mathbf{O}(ml)$ may (but does not have to) include potential intractability *even on the level of explicit models* if the size of models is defined in terms of states rather than transitions, and the number of agents is a parameter of the problem.

Corollary 1 (of Theorem 1). CTL *model checking is $\mathbf{P}$-complete, and can be done in time $\mathbf{O}(n^2 l)$.*

Theorem 4 ([10, 14]). *Model checking* ATL *is $\mathbf{\Delta_3^P}$-complete with respect to the number of states and agents, and the length of the formula.*

It also turns out that model checking of abilities under imperfect information looks no harder than perfect information from this perspective.

Theorem 5 ([11]). *Model checking* ATL_{ir} *is $\mathbf{\Delta_3^P}$-complete with respect to the number of states and agents, and the length of the formula.*

[3] Such structures are sometimes called *explicit models* [15] because global states and global transitions are represented explicitly in them.

3.3 Model Checking Is Hard

Sections 3.1 and 3.2 presented complexity results for model checking CTL, ATL, and ATL_{ir} with respect to explicit models. Most multi-agent systems, however, are characterized by an immensely huge state space and transition relation. In such cases, one would like to define the model in terms of a compact higher-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 representations). 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 [12], where the "higher level description" is defined in terms of so called *concurrent programs*, that can be used for simulating Boolean variables, but also processes or agents acting in parallel. A concurrent program P is composed of k concurrent processes, each described by a labeled transition system $P_i = \langle St_i, Act_i, \mathcal{R}_i, \Pi_i, \pi_i \rangle$, where St_i is the set of local states of process i, Act_i is the set of local actions, $\mathcal{R}_i \subseteq St_i \times Act_i \times St_i$ 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 $P_1, ..., P_k$ under the assumption that processes work asynchronously, actions are interleaved, and synchronization is obtained through common action names.

Theorem 6 ([12]). *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 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). Here, we use *modular interpreted systems* [9, 8], that draw inspiration from interpreted systems [7], reactive modules [1], and are in many respects similar to ISPL specifications [16]. A modular interpreted system (MIS) is defined as a tuple $s = \langle \mathbb{A}\text{gt}, env, Act, \mathcal{I}n \rangle$, where $\mathbb{A}\text{gt} = \{a_1, ..., a_k\}$ is a set of agents, env is the environment, Act is a set of actions, and $\mathcal{I}n$ 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,
- $d_i : St_i \to 2^{Act}$ defines local availability of actions; for convenience of the notation, 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*; $out_i : D_i \to \mathcal{I}n$ refers to the influence that a given situated action (of agent a_i) may possibly have on the external

world, and $in_i : St_i \times \mathcal{I}n^k \to \mathcal{I}n$ 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,
- $o_i : D_i \times \mathcal{I}n \to St_i$ 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
- $\pi_i : \Pi_i \to 2^{St_i}$ 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 s to a concurrent game structure follows by the synchronous product of the agents (and the environment) in s, with interaction symbols being passed between local transition functions at every step. The unfolding can also determine indistinguishability relations as $\langle q_1, ..., q_k, q_{env}\rangle \sim_i \langle q'_1, ..., q'_k, q'_{env}\rangle$ iff $q_i = q'_i$, thus yielding a full iCGS. This way semantics of both ATL and ATL$_{ir}$ is extended to MIS. Regarding model checking complexity, the following holds.

Theorem 7 ([18, 8]). *Model checking* ATL *in modular interpreted systems is* **EXPTIME***-complete with respect to the number of local states and agents, and the length of the formula.*

Theorem 8 ([9, 8]). *Model checking* 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.*

3.4 Summary of the Results

A summary of complexity results for model checking temporal and strategic logics is given in the table below. Symbols n, m stand for the number of states and transitions in an explicit model; 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 or a modular interpreted system.

	m, l	n, k, l	n_{local}, k, l
CTL	**P**-complete [5]	**P**-complete [5]	**PSPACE**-complete [12]
ATL	**P**-complete [3]	$\mathbf{\Delta_3^P}$-compl. [10, 14]	**EXPTIME**-compl. [18, 8]
ATL$_{ir}$	$\mathbf{\Delta_2^P}$-compl. [17, 11]	$\mathbf{\Delta_3^P}$-complete [11]	**PSPACE**-complete [9, 8]

Note that the results for ATL and 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 models measured with the number of transitions; next, 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 can it be so?

The amount of available strategies (relative to the size of input parameters) is the crucial factor here. The number of all strategies 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 *doubly exponential*, while there are "only" exponentially many uniform strategies – which settles the results in favor of imperfect information. It must be also noted that *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.

4 Conclusions

This paper recalls some important complexity results for model checking temporal and strategic properties of multi-agent systems. But, most of all, it tells a story with a moral. A single complexity result is often not enough to understand the *real* difficulty of the decision problem in question. When the perspective changes, so does the complexity class in which the problem belongs, and sometimes even its relative complexity with respect to other problems. Does it mean that theoretical complexity studies are worthless? Of course not, but the computational difficulty of a problem is usually more intricate than most computer scientists suspect.

There is yet another moral, too. Experimental studies where performance of algorithms and tools is measured in practice are no less needed than theoretical analysis.

References

1. R. Alur and T. A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
2. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997.
3. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.
4. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.
5. E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
6. E.A. Emerson and J.Y. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

7. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press: Cambridge, MA, 1995.
8. W. Jamroga and T. Ågotnes. Modular interpreted systems: A preliminary report. Technical Report IfI-06-15, Clausthal University of Technology, 2006.
9. W. Jamroga and T. Ågotnes. Modular interpreted systems. In *Proceedings of AAMAS'07*, pages 892–899, 2007.
10. W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In M. Pěchouček, P. Petta, and L.Z. Varga, editors, *Proceedings of CEEMAS 2005*, volume 3690 of *Lecture Notes in Computer Science*, pages 398–407. Springer Verlag, 2005.
11. W. Jamroga and J. Dix. Model checking abilities of agents: A closer look. *Theory of Computing Systems*, 42(3):366–410, 2008.
12. O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.
13. F. Laroussinie. About the expressive power of CTL combinators. *Information Processing Letters*, 54(6):343–345, 1995.
14. F. Laroussinie, N. Markey, and G. Oreiby. Expressiveness and complexity of ATL. Technical Report LSV-06-03, CNRS & ENS Cachan, France, 2006.
15. K.L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic Publishers, 1993.
16. F. Raimondi. *Model Checking Multi-Agent Systems*. PhD thesis, University College London, 2006.
17. P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004.
18. W. van der Hoek, A. Lomuscio, and M. Wooldridge. On the complexity of practical ATL model checking. In P. Stone and G. Weiss, editors, *Proceedings of AAMAS'06*, pages 201–208, 2006.