Electronic Notes in Theoretical Computer Science 85 No. 2 (2004)
URL: http://www.elsevier.nl/locate/entcs/volume85.html 12 pages

Alternating-time logic with imperfect recall

Pierre-Yves Schobbens

Institut d'Informatique
Facultés Universitaires de Namur
Rue Grandgagnage 21
5000 Namur, Belgium

Abstract

We study here a variant of the alternating-time temporal logic (ATL) where each agent has a given memory. We show that it is an interesting compromise, rather realistic but with a reasonable complexity. In contrast, most models with perfect recall and imperfect information have an undecidable model-checking problem.

Key words: alternating-time temporal logic (ATL), imperfect recall, imperfect information, games, verification, temporal logic

1 Introduction

ATL [2] comes with a modality expressing that a group has a strategy to ensure a temporal property, assuming that each member of the group knows the complete state of the system at each step (*complete information*). The strategies synthesized by ATL are thus often unrealistic for systems where global information is not easily accessible.

We propose instead to consider agents that have access to only part of the system state, called their *information*. This hypothesis is thus called *incomplete information*. It is natural for most problems; for instance, a network node can only know the messages it receives, a robot can only perceive its immediate surroundings, etc. Complete information is slightly different from *perfect information*, that assumes that all choices of the past are known, but as we will see that this difference is inessential in our setting.

With incomplete information, many authors assume *perfect recall* [7,10], i.e. that each agent can use the complete history of its information. For practical reasons, we propose here to consider also *imperfect recall*: the memory of each agent is described explicitly. This does not necessarily mean that what the agent will record is fixed: we can provide an agent with actions to update its memory at will; the strategy will then decide what to record and when.

Example 1.1 Before delving into the formal definitions, we provide a simple example.

A banker B at his desk d knows the code c of the safe in the vault v. A robber R wants to open the safe o. He can threaten B with his gun. When threatened t, B is paralysed by fear and tells the code. R can dial any number on the safe when in the vault. R can try to open the safe when in the vault, but it will only open if the last dialled number n is the code s; otherwise the vault door closes, jailing R.

We represent the fact that B knows the code by a private variable containing the code. If we use ATL with complete information, all variables are accessible to all agents: thus the robber has a simple strategy to open the safe, since he can go in the vault and type the code that he (magically) reads from the brain of the banker. This strategy is synthesized easily by an ATL model-checker, but it looks unrealistic.

A more realistic model is provided by incomplete information and perfect recall: the robber can threaten the banker to learn the code, remember it, go in the vault, type the code. Unfortunately, such strategies are not computable in general.

Thus, we propose here to include in the model an explicit description of the memory of the agents. We can select two types of solutions for this example: If the robber is modelled as absent-minded, i.e. has not enough memory to store the code, he will not be able to open the safe, since he would have forgotten the code while going to the vault. If he has enough memory, the the perfect recall strategy also applies here.

2 Definitions

Given a signature, containing:

- P, a set of atomic predicates;
- Σ, a finite set of agents (also called modules [1] or players);

An *imperfect information concurrent game structure* (*iCGS*), or *game*, consists of:

- Q, a finite set of system states;
- $q_0 \in Q$, the initial state;
- C, a finite set of choices (sometimes called actions [10] or alternatives);
- $e \subseteq Q \times \Sigma \times C$, a guard that indicates whether an agent is enabled to take a choice in a given state; at least one choice must be enabled in any state for each agent: $\forall q \in Q, a \in \Sigma, \exists c \in C, e(q, a, c)$.
- $\delta : Q \times C^{\Sigma} \to Q$, a transition function; we assume all agents decide their choice at the same time, and the transition function gives the next state obtained by the interaction of their choices.

- $\sim: \Sigma \to 2^{Q \times Q}$, a family of equivalence relations, one per agent, indicating the states that are undistinguishable from its viewpoint. For consistency, we assume that each agent knows which choices are open to him: $\forall q \sim_a q', e(q,a,c) \iff e(q',a,c)$.
- $\pi : Q \to 2^P$, a valuation giving the propositions that are true in a state.

$\sim_a$ determines equivalence classes on Q: Given a state q, we note $[q]_a$ its equivalence class (the view of agent a). Q_a is the set of such views $[q]_a$. In many systems, the state is structured as a valuation for a given set of typed variables. Often, the agent a can only observe a subset r_a of the variables called its *read variables*. $[q]_a$ can then more concretely understood as the value of the variables in r_a, which is a part of the global state q. When these variables are also writable, i.e. if the agent has choices that allow him to set the variable to any value he likes, the agent can use them as its memory.

In this paper, we will consider four types of strategies. In all cases, they will determine the choice of an agent, as a function of his view of the past: $f_a : V \to C$. This choice must be enabled.

(i) The model of perfect information and perfect recall, noted IR, is the usual model of ATL. The view of the past $V = Q^+$ is the sequence of all states that the system has been through until now.

(ii) The model of perfect information and imperfect recall, noted Ir. The view of the past $V = Q$ is limited to the present state of the system.

(iii) The model of imperfect information and perfect recall, noted iR. Only a designated part of the information, described by $\sim_a$, is available to each agent. Thus $V = Q_a^+$.

(iv) The model of imperfect information and imperfect recall, noted ir, that deserves more attention, we claim. An agent can only base its choice on its view of the current state, $V = Q_a$.

Note that all such strategies can also be viewed as functions from Q^+, that satisfy the supplementary constraint that sequences that are undistinguishable (e.g. if they have the same present in an imperfect recall model) yield the same choice.

An XY (where XY is IR, iR, Ir, or ir) strategy f_Γ for a group of agents $\Gamma \subseteq \Sigma$ is a family of XY strategies $(f_a)_{a \in \Gamma}$, one for each agent. A computation (sequence of states) $\lambda = q_0, q_1, \ldots$ is an outcome from q of a strategy f_Γ if $q = q_0$ and for each $i > 0$, there are $c_a \in C$, for $a \in \Sigma \setminus \Gamma$, such that $q_{i+1} = \delta(q_i, \boldsymbol{c})$ where $\boldsymbol{c}_a = f_a([q_i]_a)$ if $a \in \Gamma$, and $\boldsymbol{c}_a = c_a$ for the opponents. So the outcomes are the computations that follow the strategy. The set of outcomes is noted $out(q, f_\Gamma)$.

The syntax of ATL* is given by:

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\langle\Gamma\rangle\rangle\varphi_1 \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \mid \mathrm{X}\, \varphi_1$$

where $p \in P$ is an atomic predicate and $\Gamma \subseteq \Sigma$ is a set of agents. We will also use the usual logical and temporal abbreviations, we write $\mathrm{F}\,\varphi$ for $\top\,\mathcal{U}\,\varphi$ and $\mathrm{G}\,\varphi$ for $\neg\mathrm{F}\,\neg\varphi$, and $\mathrm{A}\,\phi$ for $\langle\langle\emptyset\rangle\rangle\phi$.

The semantics of ATL* is an extension of the temporal logic CTL*; we assume a fixed iCGS:

- $\lambda \vDash p$ iff $p \in \pi(\lambda[0])$
- $\lambda \vDash \top$, always
- $\lambda \vDash \neg\varphi$ iff $\lambda \nvDash \varphi$
- $\lambda \vDash \varphi_1 \vee \varphi_2$ iff $\lambda \vDash \varphi_1$ or $\lambda \vDash \varphi_2$
- $\lambda \vDash \langle\langle\Gamma\rangle\rangle_{XY}\psi$ iff there exists a set of XY strategies F_Γ, one for each agent in Γ, such that $\forall a \in \Gamma, \forall q' \sim_a \lambda[0], \forall\lambda' \in \mathit{out}(q', F_\Gamma)$ we have $\lambda' \vDash \psi$
- $\lambda \vDash \psi_1\,\mathcal{U}\,\psi_2$ iff $\exists i \geq 0.\lambda[i,\infty] \vDash \psi_2$ and $\forall 0 \leq j < i\lambda[j,\infty] \vDash \psi_1$.
- $\lambda \vDash \mathrm{X}\,\varphi$ iff $\lambda[1,\infty] \vDash \varphi$

[2] gives a different semantics to $\langle\langle\Gamma\rangle\rangle$:

$$q \vDash \langle\langle\Gamma\rangle\rangle\phi \text{ iff}$$
there is a strategy F_Γ such that all outcomes $\lambda \in \mathit{out}(q, F_\Gamma)$ satisfy ϕ

which has the counter-intuitive effect that, because F_Γ is the scope of q, we can make the strategy dependent on q, which amounts to give perfect information on the initial state, even for imperfect information models.

Example 2.1 For our robber example, for each code c consider the strategy of going in the vault and typing code c. This strategy is successful when the code is indeed c. For each state q, we would thus have a successful strategy for the robber with the definition of [2].

ATL is the subset of ATL* where each temporal operator (X, $\mathcal{U}$) is immediately preceded by a cooperation modality (similarly to CTL). $\langle\langle\Gamma\rangle\rangle$-ATL is further restricted to contain a single cooperation modality $\langle\langle\Gamma\rangle\rangle$. ATL$^+$ is the subset of ATL* where each temporal operator (X, $\mathcal{U}$) can occur in a boolean combination, preceded by a cooperation modality (similarly to CTL$^+$ [4]). In [6], we show that ATL$^+$ has the same expressivity as ATL, but can be exponentially more succinct.

Example 2.2 We will use the Mocha [1] conventions to describe our example: We describe the set of states Q as a set of valuations for finite-domain variables. For this example, the variables are listed in Table 1. The choices C of each agent will be listed as assignments. To ensure compatibility of the choices, each agent *controls* disjoint subsets of the variables, and he can only modify his controlled variables. Then the transition δ is defined as a usual assignment. To describe the undistinguishability relation $\sim$, each agent has a set of *read* variables. Two states that differ by non-readable variables only are deemed undistinguishable.

Name	Meaning	Type
LB	Location of Robber	{d,v}
LR	Location of Robber	{d,v}
t	is Banker threatened?	bool
m	code uttered by Banker	code
n	last code typed on the safe	code
o	is the safe open?	bool
j	is the robber jailed?	bool

Table 1
Variables of the example

```
module B /* Banker */
controls LB, m
reads c, LB, t, LR
update
[] ¬t → LB := v /* go down in the vault */
[] ¬t → LB := d /* go up at the desk */
[] t & LR=LB → m := c /* tell the code */
endmodule
```

Fig. 1. The banker

```
module R /* Robber */
controls LR, n, o, j, t
init
[] LR := d, j := false
update
[] LR := v /* go down in the vault */
[] LR := d /* go up at the desk */
[] LR=LB → t := true /* threaten the banker */
[] LR=v → n := ? /* type any number he likes */
[] LR=v → o := (n=c), j := ¬(n=c) /* try to open, might jail */
endmodule
```

Fig. 2. The robber

3 Complexity

Incomplete information is usually more complex than complete information. But imperfect recall, although restricting even more the capabilities of agents, usually reduces complexity. For the logics considered here, the model-checking problem is undecidable for perfect recall but incomplete information, and im-

perfect recall makes it decidable again.

logic	*ir*	*iR*	*Ir*	*IR*
$\langle\langle\Gamma\rangle\rangle - ATL$	NP	U [11]	$n.l$ [2]	$n.l$ [2]
ATL	$\Delta_2 P$	U [11]	$n.l$ [2]	$n.l$ [2]
ATL^+	$\Delta_3 P$	U [11]	$\Delta_3 P$	$\Delta_3 P$
ATL^*	$PSPACE$	U [11]	$PSPACE$	$DEXP$ [9]

NP	complete for nondeterministic polynomial time
$\Delta_2 P = P^{NP}$	complete for polynomial calls to an NP oracle
$\Delta_3 P = P^{NP^{NP}}$	complete for polynomial calls to a $\Sigma_2 P$ oracle
EXP	complete for deterministic exponential time
DEXP	complete for deterministic doubly exponential time
U	undecidable
l	size of the formula
n	size of the model

Table 2
Complexity of model-checking

This shows a second possible use of imperfect recall: If we are interested in iR, we can use IR and ir as cheap approximations from above and below:

$$\langle\langle\Gamma\rangle\rangle_{ir}\phi \;\Rightarrow\; \langle\langle\Gamma\rangle\rangle_{iR}\phi \;\Rightarrow\; \langle\langle\Gamma\rangle\rangle_{IR}\phi$$

When their value agree, we have an answer for iR, in spite of its undecidability.

3.1 NP-completeness

A $\langle\langle\Gamma\rangle\rangle_{ir}$-ATL formula is of the form $\langle\langle\Gamma\rangle\rangle_{ir}\phi$ with ϕ starting with a temporal operator. We nondeterministically choose a strategy by selecting a subset of the transition relation: we choose an action for each $a \in \Gamma$ and $[q]_a \in Q_a$. Then we check the CTL formula A ϕ for the structure so created, where A is the CTL modality quantifying on all paths of the system. This CTL check is done in time linear to the structure and to the formula [3]. So for each state formula, we make a number of choices limited by the size of the transition, and perform only polynomial operations. The total number of choices is thus still polynomial ($|\phi|.|Q|.|\Sigma|$) and the time also (viz. $|\phi|^2.|Q|^2.|\Sigma|$). This procedure returns a set of states on which the strategy is successful. To comply with our definition, we must then restrict to the states $\{q \in Q | \forall a \in \Gamma, \forall q' \sim_a q, q' \in S\}$, which can be done in linear time.

For NP-hardness, we encode the SAT problem in the structure. We are given a set of clauses, written using predicates $p_1, \ldots, p_n$. We introduce an agent per predicate, plus a dispatcher. A predicate agent will have two choices: to set his predicate to either "true" or "false". The dispatcher agent will choose for each clause, which literal to satisfy. For each clause (numbered c) we create a sub-structure with 3 layers: a single initial state q_c, which is connected to a state per literal, and the transition is under control of the dispatcher.The transition from a literal state is under control of the corresponding predicate agent. If it is a positive literal, the choice "true" will lead to q_{c+1}, and the choice "false" will lead to the dead state q_d. Conversely for a negative literal. All literal states are in an equivalence class of the corresponding predicate agent. Said otherwise, predicate agents have no information when they have to play, and their strategy is thus just the choice of a truth value. The structure will have a single initial state q_0, a dead state q_d, a success state q_f, where f is the number of the last clause plus one. There is a single logic predicate s (for success), true exactly in q_f. The model checking problem will be to check whether $q_0 \vDash \langle\langle\Sigma\rangle\rangle\Diamond s$, which is true iff the choices of the truth values gives a model of the clauses.

Note that we use a turn-based game with no opponent: all the complexity comes from the difficulty to make agents with imperfect recall cooperate.

3.2 Complexity of ATL

To show that the model-checking of ATL_{ir} is $\Delta_2 P$-easy, we notice that we can evaluate the formulae bottom-up, starting with the state subformulae. If it is a boolean subformula, we return the set of states that satisfy it. Else, the formula is in $\langle\langle\Gamma\rangle\rangle$-ATL, and we can use the procedure above. There are at most linearly many calls to this NP procedure, giving a $\Delta_2 P$ algorithm.

We only conjecture that this problem is also $\Delta_2 P$-hard; it is at least NP-hard by the proof above.

3.3 Complexity of ATL+

We now show that model-checking an ATL^+_{XY} formula is $\Delta_3 P$-complete, for $XY = ir, Ir$ or IR. Let us recall that $\Delta_3 P = P^{NP^{NP}}$ is in the polynomial hierarchy. For the easiness, we can use the fact that CTL^+ in in $\Delta_2 P = P^{NP}$ [3, Thm 6.2]. We proceed bottom-up in the formula, thus linearly, each time replacing a state formula of the form $\langle\langle\Gamma\rangle\rangle\phi$ (a modality, followed by a boolean combination of temporal formulae on boolean formulae) with a new predicate, true in the same states. For each such a formula, we can choose non-deterministically an ir strategy, then we check this strategy for the corresponding CTL^+ formula, with A replacing $\langle\langle\Gamma\rangle\rangle$. This can be done in P^{NP} [3, Thm 6.2]. The polynomial part of P^{NP} can on course be integrated in the NP choice of the imperfect recall strategy. We must call this for every cooperation subformula (at most n times, in P) yielding a $P^{NP^{NP}}$ algorithm.

We can also use this algorithm for ATL^+_{Ir}: it is just a special case where $\sim$ is the identity. It also works for ATL^+_{IR}, since it is translatable to ATL [6] and in ATL winning strategies without recall are sufficient [2], so that it is enough to enumerate Ir strategies.

For the hardness, we encode the typical $\Delta_3 P$-complete variant of SAT, into an ATL^+_{IR} model-checking problem.

A $\Delta_3 SAT$ problem is given by sequence of quantified boolean formula of the form: $z_i = \exists X_i \forall Y_i \phi_i(X_i \cup Y_i \cup \{z_j | j < i\})$, where each ϕ_i is a boolean formula using former variables z_j, and locally quantified variable X_i, Y_i. We assume that all variables are distinct. Let $X_i = \{x_{1i}, \ldots, x_{n_i i}\}, Y_i = \{y_{1i}, \ldots, y_{m_i i}\}$. We encode valuations very directly as paths in a game with two players $\{X, Y\}$. At the initial state z_n, player A chooses the value of x_{1n}. This can lead to two states, called x^+_{1n} and x^-_{1n}. There, X chooses x_{2n}, and so on. This is followed by Y choosing the values of the y_{in}. Then there is a single state labeled z_{n-1}, and so on until the last state z_0 which has a self-loop to comply with the definition of an iCGS. Predicate $p_i \in X_i, Y_i$ are true exactly in node p^+_i. We introduce a predicate z_i true in node z_i. We transform the formulae of the SAT problem into the ATL^+ formula $\langle\langle X \rangle\rangle \phi_n(\text{F } x_{in}, \text{F } y_{in}, \langle\langle X \rangle\rangle \phi_{n-1}(Fx_{in-1}, Fyin-1), \langle\langle X \rangle\rangle \phi_{n-2})$. Thus, the quantifications $\exists X_i \forall Y_i$ are replaced by $\langle\langle X \rangle\rangle$, and the basic predicates have an F prepended. The z_j are directly replaced by their definition. These replacements might make the textual form of the formula exponential in size, but fortunately the complexity actually depends on the size with subformula shared (the DAG form), which is still polynomial.

3.4 ATL complexity*

The PSPACE-hardness of ATL^*_{Ir} model-checking is deduced from the fact that LTL is PSPACE-complete and included in ATL^*.

The PSPACE-easiness is again obtained simply by enumerating the possible ir-strategies, and solving a CTL^* problem for each of them. This gives a $P^{NP^{PSPACE}} = PSPACE$ complexity.

4 Heuristics

A characteristic of NP-complete problems is that, although finding a solution (a strategy, here) is difficult in some cases, it is easy in other cases and it is always easy to check a solution. Model-checking ATL_{ir} is no exception to this. We thus provide here heuristic ways to construct ir strategies for ATL_{ir}.

A natural idea is to start from the perfect information strategies cheaply provided by ATL: they are constructed in time linear in the size of the structure. We first present this construction.

4.1 Strategies with perfect information

The efficiency of this search is based on the fact that recall plays no role for ATL: $IR = Ir$ [2]. Ir Strategies (also called positional strategies) exist whenever IR strategies exist.

4.1.1 Representing strategies

We can describe strategies in two ways: the usual *centralised* description uses one strategy for the whole group. It will be represented as a set of states ϕ_c for each vector of choices $\boldsymbol{c}$ for A (one per agent). This is noted as $\bigvee_c(\phi_c \wedge \boldsymbol{c})$, read "when in ϕ_c, you can do $\boldsymbol{c}$". The usual computation makes the ϕ_cs disjoint (for a deterministic strategy). However, it will be our interest to have the ϕ_c as large as possible, so that we will allow overlap. The *distributed* description uses one strategy f_a per agent a, each represented as a set of states ϕ_c^a for each choice c of a, saying when a will choose c. When no overlap is present, we can switch between the two representations by defining: $\phi_{\boldsymbol{c}} = \bigwedge_{c \in \boldsymbol{c}} \phi_c^a$, and $\phi_c^a = \bigvee_{c \in \boldsymbol{c}} \phi_{\boldsymbol{c}}$. However, in case of overlap, the two representations are not equivalent. It might be that only some combinations of individual choices lead to success. This dependency between the choice of several agents cannot be represented in distributed form.

4.1.2 Computing strategies

The algorithm proceeds in ascending order on the subformulae of the given formula. Only the cooperation subformulae are of interest:

- For $\phi' = \langle\langle\Gamma\rangle\rangle(\theta_1 \,\mathcal{U}\, \theta_2)$: The computation of these strategies can be performed by a backward search, starting from θ_2, and using only θ_1 states. We compute a new winning region r, by computing the states of θ_1 where a choice for each member of Γ will guarantee to be at next step in $r \cup \theta_2$. At step n, r is the set of states that can reach θ_2 in n steps while staying in θ_1. In the strategy f, we need to avoid loops, so we record the choices only for new r states, that were not in r at the previous iteration. (This easy condition sometimes removes choices that were not looping.)

  At the end, $r \cup \theta_2$ is the winning region, and thus the states where ϕ' is true. r is the domain of f, and on θ_2 any strategy can be followed, and is winning. Outside of the winning region, any strategy can be followed, since A will loose anyway.

  We call the strategy computed above the nondeterministic shortest path strategy, or S-strategy.

- For $\phi' = \langle\langle\Gamma\rangle\rangle(\mathrm{G}\,\theta_2)$: A greatest fixpoint be computed; Here, loops are included in this computation.

  r := θ_2
  repeat
  $r = Pre(A, r) \cap \theta_2$

until r stabilizes
$f = \{Pre(c, r) \wedge c\}$

$Pre(A, r)$ computes the controlled predecessors for all choices of A. The strategy simply keeps the successors inside the winning region r. Here, we obtain a most general nondeterministic strategy.

4.2 Strategies with imperfect information and recall

The strategies obtained above are Ir strategies, so we have to suppress in the conditions ϕ_c^a information that is not available to the agents that must execute the choices. We define an operator $K_a : 2^Q \to 2^Q$ that restricts ϕ_c^a to a stricter condition that only uses information available to a: $K_a(\phi) = \{s \in \phi | \forall s' \in I, s' \sim_a s \Rightarrow s' \in \phi\}$, where I is the set of states reachable from the initial state q_0. Given a distributed deterministic strategy $\phi_c^a \to c$, where c is a choice of a single agent a, $K_a(\phi_c^a) \to c$ is an imperfect information strategy. However, this technique is often too restrictive: ϕ_c^a is first restricted to have a deterministic strategy, and $K_a(\phi_c^a)$ is even smaller, often empty even though there are winning ir strategies. To reduce this risk, we start from centralized nondeterministic strategies, as given by the algorithm above. We complete it by including all choices when we are out of the winning region, giving a strategy sc. We can first include in the strategy *safe* choices: choices such that, whatever the other agents choose, their combination with the given choice c is inside the centralised strategy. The condition for c to be a safe choice of a is given by $K_a((\forall C_{\neq a} sc|_c)$.

Note that this technique is only heuristic and neglects several phenomena:

(i) The agents in a are willing to cooperate, but here we consider them as behaving randomly, because they might not have enough information to act adequately.

(ii) The strategies computed above make no effort to transmit information among agents.

(iii) Also, the agents could cooperate for remembering relevant parts of the past; again, the strategies above do not try this.

5 Conclusions

We have studied various possibilities for the model of agents: perfect or imperfect information, and perfect or imperfect recall. Although adding perfection yields more powerful agents, as expected, perfect information make the model-checking problems easier while perfect recall makes them more difficult. Imperfect information and recall seems a reasonable compromise, with realistic modelling powers and decidable problems.

A number of problems had to be left for future work:

- Axiomatization: The new modalities $\langle\langle\Gamma\rangle\rangle_{XY}$ have a different semantics, so that they obey different axioms from the ones of ATL [5]. They are still to be discovered.
- Experimentation: Many more heuristics than proposed here are possible. They should then be compared, and also with general heuristics, for instance SAT solvers.
- Models of the adversary: the $\langle\langle\Gamma\rangle\rangle$ modality must ensure its result whatever the adversary does, including unlikely lucky choices. It might be useful to quantify instead on strategies of a more realistic adversary, himself with imperfect information or imperfect recall.
- Knowledge: The knowledge of agents [8] plays an essential role in this paper, yet is never explicit in the logics we consider here. Adding an explicit knowledge operator seems useful, but what should be its meaning? Additional knowledge can be brought by the strategy used by the cooperating agents, but depends on which strategy is used. It seems thus necessary have a logic where strategies are explicit.

6 Acknowledgement

Thanks to Wiebe van der Hoek and Mark Ryan for first discussions on the future work mentioned in the conclusion.

References

[1] R. Alur, T. A. Henzinger, F. Y. C. Mang, S. Qadeer, S. K. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proc. 10th International Computer Aided Verification Conference*, pages 521–525, 1998.

[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*, pages 100–109. IEEE Computer Society Press, 1997.

[3] E. M. Clarke, E. Allen Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[4] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.

[5] Valentin Goranko and Govert van Drimmelen. Complete axiomatization of the alternating-time temporal logic, 2003.

[6] Aidan Harding, Mark Ryan, and Pierre-Yves Schobbens. Approximating ATL* in ATL. *Lecture Notes in Computer Science*, 2294:289–??, 2002.

[7] H. W. Kuhn. Extensive games and the problem of information. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 193–216. Princeton University Press, Princeton, NJ, 1953.

[8] John-Jules Ch. Meyer and Wiebe van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, Cambridge, 1995.

[9] R. Rosner. *Modular Synthesis of Reactive Systems*. PhD thesis, The Weizmann Institute of Science, 1992.

[10] Ron van der Meyden. Axioms for knowledge and time in distributed systems with perfect recall. In *Logic in Computer Science*, pages 448–457, 1994.

[11] M. Yannakakis. Synchronous multi-player games with incomplete information are undecidable, 1997. Personal communication (reference reproduced from [2]).