Strategic (Timed) Computation Tree Logic

Jaime Arias
LIPN, CNRS UMR 7030,
Université Sorbonne Paris Nord
Villetaneuse, France
arias@lipn.univ-paris13.fr

Wojciech Jamroga
Institute of Computer Science,
Polish Academy of Sciences
and SnT, University of Luxembourg
jamroga@ipipan.waw.pl

Wojciech Penczek
Institute of Computer Science,
Polish Academy of Sciences
Warsaw, Poland
penczek@ipipan.waw.pl

Laure Petrucci
LIPN, CNRS UMR 7030,
Université Sorbonne Paris Nord
Villetaneuse, France
petrucci@lipn.univ-paris13.fr

Teofil Sidoruk
Institute of Computer Science, PAS
and Faculty of Math. and Inf. Science,
Warsaw University of Technology
t.sidoruk@ipipan.waw.pl

ABSTRACT

We define extensions of **CTL** and **TCTL** with strategic operators, called Strategic **CTL** (**SCTL**) and Strategic **TCTL** (**STCTL**), respectively. For each of the above logics we give a synchronous and asynchronous semantics, i.e. **STCTL** is interpreted over networks of extended Timed Automata (TA) that either make synchronous moves or synchronise via joint actions. We consider several semantics regarding information: imperfect (i) and perfect (I), and recall: imperfect (r) and perfect (R). We prove that **SCTL** is more expressive than **ATL** for all semantics, and this holds for the timed versions as well. Moreover, the model checking problem for $\mathbf{SCTL}_{\mathrm{ir}}$ is of the same complexity as for $\mathbf{ATL}_{\mathrm{ir}}$, the model checking problem for $\mathbf{STCTL}_{\mathrm{ir}}$ is of the same complexity as for **TCTL**, while for $\mathbf{STCTL}_{\mathrm{iR}}$ it is undecidable as for $\mathbf{ATL}_{\mathrm{iR}}$. The above results suggest to use $\mathbf{SCTL}_{\mathrm{ir}}$ and $\mathbf{STCTL}_{\mathrm{ir}}$ in practical applications. Therefore, we use the tool IMITATOR to support model checking of $\mathbf{STCTL}_{\mathrm{ir}}$.

KEYWORDS

timed automata; model checking; timed logics; strategy logics

ACM Reference Format:
Jaime Arias, Wojciech Jamroga, Wojciech Penczek, Laure Petrucci, and Teofil Sidoruk. 2023. Strategic (Timed) Computation Tree Logic. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Alternating-time temporal logics $\mathbf{ATL}^*$ and **ATL** [6, 7] extend the temporal logic $\mathbf{CTL}^*$ and **CTL**, resp., with the notion of *strategic ability*. These logics allow for expressing properties of agents (or groups of agents) referring to what they can achieve. Such properties can be useful for specification, verification, and reasoning about interaction in multi-agent systems [17, 21, 24, 25, 31].

In this paper we investigate timed extensions of strategy logics, these already known as well as newly introduced ones. One of our main aims is to identify the most expressive logics for which the model checking problem is not only decidable, but also of complexity acceptable in practice. We start with recalling the syntax of **ATL** [6, 7] and **TATL** [28]. Then, we put forward definitions of two new logics: Strategic CTL (**SCTL**), and its timed extension, Strategic Timed CTL (**STCTL**). For each (timed) strategy logic we consider two types of interpretations, over models of synchronous (Time) Multi-Agent Systems MAS and asynchronous (Time) Multi-Agent Systems AMAS. In addition, Time MAS and Time AMAS can be either discrete (D), or continuous (C). We investigate the model checking problem for **SCTL** and **STCTL** for all the semantics, and compare their complexity with other strategy logics. Notably, we prove that **SCTL** is more expressive than **ATL** for all semantics, and this holds for the timed versions as well. Moreover, the model checking problem for $\mathbf{SCTL}_{\mathrm{ir}}$ is of the same complexity as for $\mathbf{ATL}_{\mathrm{ir}}$, the model checking problem for $\mathbf{STCTL}_{\mathrm{ir}}$ is of the same complexity as for **TCTL**, while for $\mathbf{STCTL}_{\mathrm{iR}}$ it is undecidable as for $\mathbf{ATL}_{\mathrm{iR}}$. These results suggest to use $\mathbf{SCTL}_{\mathrm{ir}}$ and $\mathbf{STCTL}_{\mathrm{ir}}$ in practical applications. Therefore, we demonstrate the feasibility of $\mathbf{STCTL}_{\mathrm{ir}}$ model checking on a small scalable example using IMITATOR.

Related Work. **TATL** [28] is a discrete-time extension of **ATL** [6, 7], the subset of $\mathbf{ATL}^*$ where each strategic modality is immediately followed by a single temporal operator. A hierarchy of semantic variants of **TATL** was established and studied in [26], including counting strategies. Game Logic (**GL**) [7], similarly to **SCTL**, combines path quantifiers with the notion of strategic ability. **GL** is a generalisation of $\mathbf{ATL}^*$ over perfect information, where quantification is possible separately over paths within a strategy outcome. **CTL** timed games [13] are defined over timed automata with continuous time, but with specifications given using **CTL** and **LTL**, placing them somewhere between untimed $\mathbf{SCTL}^{\mathbf{U}}_{\mathrm{IR}}$ and $\mathbf{STCTL}^{\mathbf{C}}_{\mathrm{IR}}$ considered here. They are shown to be **EXPTIME**-complete, demonstrating that model checking of $\mathbf{SCTL}_{\mathrm{IR}}$ over continuous time models retains the same complexity as over untimed ones (cf. Table 1). Analogously to $\mathbf{ATL}^*$, the logic $\mathbf{TATL}^*$ over continuous time semantics, call it $\mathbf{TATL}^{*\mathbf{C}}$, would be a natural counterpart to the discrete-time $\mathbf{TATL}^*$. However, even without the strategic modality, model checking is undecidable for continuous time extensions of **LTL** (**MTL** and **TPTL**) [10]. This has motivated our choice of **STCTL**, which is applicable where discrete time is insufficient, more expressive than **TATL**, and yet its model checking is decidable for r-strategies.

Outline. First, Sec. 2 recalls the basic notions of strategic logics. Synchronous systems are tackled in untimed, discrete time and continuous time settings, all described in an homogeneous manner.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

Sec. 3 discusses different types of strategies and gives the semantics of considered logics. Theoretical results regarding the model checking complexity and expressiveness of **S(T)CTL** are introduced in Sec. 4 and Sec. 5, respectively. Sec. 6 considers asynchronous systems, pointing out differences from the synchronous case wherever applicable. Sec. 7 reports experimental results using the IMITATOR model checker. Finally, Sec. 8 concludes the paper.

2 REASONING ABOUT STRATEGIES AND TIME

In this section, we define the logical framework to reason about strategic abilities in timed synchronous multi-agent systems. Our definitions are based on [4, 7, 26, 28, 32, 33]. We will denote the set of natural numbers including (resp. without) zero by $\mathbb{N}$ (resp. $\mathbb{N}_+$), and the set of non-negative real numbers by $\mathbb{R}_{0+}$.

2.1 Syntax of STCTL and its Fragments

We begin by introducing the logical formulas of interest. Assume a countable set PV of atomic propositions, and a finite set $\mathcal{A}$ of agents. The syntax of *Strategic Timed Computation Tree Logic* (**STCTL**) resembles that of Game Logic [7], and can be defined by the grammar:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A \rangle\!\rangle \gamma,$$
$$\gamma ::= \varphi \mid \neg\gamma \mid \gamma \wedge \gamma \mid \forall \mathsf{X}\, \gamma \mid \forall \gamma \,\mathsf{U}_I \gamma \mid \exists \gamma \,\mathsf{U}_I \gamma \mid \forall \gamma \,\mathsf{R}_I \gamma \mid \exists \gamma \,\mathsf{R}_I \gamma,$$

where $\mathsf{p} \in PV$ is an atomic proposition, $A \subseteq \mathcal{A}$ is a subset of agents, and $\langle\!\langle A \rangle\!\rangle$ is the strategic operator expressing that the agents in A have a strategy to enforce the temporal property that follows after it. $\forall$ ("for all paths") and $\exists$ ("there exists a path") are the usual path quantifiers of **CTL**. The temporal operators X, U, R stand for "next", "strong until", and "release," respectively. Boolean connectives and the remaining operators F ("eventually"), G ("always") can be derived as usual. Intervals $I \subseteq \mathbb{R}_{0+}$ with bounds of the form $[n, n']$, $[n, n')$, $(n, n']$, (n, n'), (n, ∞), or $[n, \infty)$, where $n, n' \in \mathbb{N}$, are subscribed to temporal operators U, R to denote time constraints on their evaluation, e.g. $\gamma_1 \,\mathsf{U}_I \gamma_2$ specifies that γ_1 holds until a time within I when γ_2 becomes true. Note that we added the next step operator X to the syntax of **STCTL** (and **TCTL**) in order to be able to define other logics as syntactic fragments of **STCTL**:

SCTL: untimed Strategic **CTL**, obtained from **STCTL** by restricting the time intervals I in temporal operators U_I, R_I to $I = [0, \infty)$. So, they are removed from the syntax of **STCTL**;

TCTL: Timed **CTL**, obtained from **STCTL** by removing the strategic modality $\langle\!\langle A \rangle\!\rangle$ from the syntax;

CTL: "vanilla" **CTL**, obtained from **SCTL** by removing $\langle\!\langle A \rangle\!\rangle$;

TATL: Timed Alternating-time Temporal Logic, the fragment of **STCTL** where each instance of $\langle\!\langle A \rangle\!\rangle$ is immediately followed by $\forall\mathsf{X}$, $\forall\mathsf{U}_I$, or $\forall\mathsf{R}_I$, then $\forall$ is removed from the syntax;

ATL: "vanilla" **ATL**, obtained from **TATL** by restricting the time intervals to $I = [0, \infty)$, thus removing them from the syntax;

We can now introduce the syntax and semantics of synchronous MAS with continuous and discrete time, as well as untimed.

2.2 Continuous Time Synchronous MAS

In *synchronous continuous-time multi-agent systems*, all agents have an associated set of clocks. All clocks evolve at the same rate (across agents), thus allowing for delays and instantaneous actions. We first recall the formal definitions for key notions of *timed systems* [4]. Then, we combine them with the concept of *interpreted systems* [32] which has been successful in modelling synchronous MAS.

Clocks are non-negative, real-valued variables; we denote a finite set of clocks by $\mathcal{X} = \{x_1, \ldots, x_{n_\mathcal{X}}\}$ (with a fixed ordering assumed for simplicity). A *clock valuation* on $\mathcal{X}$ is a $n_\mathcal{X}$-tuple v. We denote:

- by $v(x_i)$ or $v(i)$, the value of clock x_i in v;
- by $v + \delta$, where $\delta \in \mathbb{R}_{0+}$, v' s.t. $v'(x) = v(x) + \delta$ for all $x \in \mathcal{X}$;
- by v[X := 0], where $X \subseteq \mathcal{X}$, v' s.t. $v'(x) = 0$ for all $x \in X$, and $v'(x) = v(x)$ for all $x \in \mathcal{X} \setminus X$.

The *clock constraints* over $\mathcal{X}$ are defined by the following grammar: $\mathfrak{cc} := true \mid x_i \sim c \mid x_i - x_j \sim c \mid \mathfrak{cc} \wedge \mathfrak{cc}$, where $x_i, x_j \in \mathcal{X}$, $c \in \mathbb{N}$, and $\sim \in \{\leq, <, =, >, \geq\}$. The set $C_\mathcal{X}$ collects all constraints over $\mathcal{X}$. For $\mathfrak{cc} \in C_\mathcal{X}$, the satisfaction relation $\models$ is inductively defined as:

$v \models true$,
$v \models (x_i \sim c)$ iff $v(x_i) \sim c$,
$v \models (x_i - x_j \sim c)$ iff $v(x_i) - v(x_j) \sim c$, and
$v \models (\mathfrak{cc} \wedge \mathfrak{cc}')$ iff $v \models \mathfrak{cc}$ and $v \models \mathfrak{cc}'$.

The set of all valuations satisfying $\mathfrak{cc}$ is denoted by $[\![\mathfrak{cc}]\!]$.

Definition 2.1 (CMAS). A *continuous-time multi-agent system (CMAS)* consists of n agents $\mathcal{A} = \{1, \ldots, n\}$, each associated with a 9-tuple $AG_i = (L_i, \iota_i, Act_i, P_i, \mathcal{X}_i, \mathcal{I}_i, T_i, PV_i, V_i)$ including:

- a finite non-empty set of *local states* $L_i = \{l_i^1, l_i^2, \ldots, l_i^{n_i}\}$;
- an *initial local state* $\iota_i \in L_i$;
- a finite non-empty set of *local actions* $Act_i = \{a_i^1, a_i^2, \ldots, a_i^{m_i}\}$;
- a *local protocol* $P_i : L_i \to 2^{Act_i} \setminus \{\emptyset\}$;
- a set of *clocks* $\mathcal{X}_i$;
- an *invariant* $\mathcal{I}_i \colon L_i \to C_{\mathcal{X}_i}$ specifying a condition for the CMAS to stay in a given local state;
- a (partial) *local transition function* $T_i : L_i \times \mathcal{J}Act \times C_{\mathcal{X}_i} \times 2^{\mathcal{X}_i} \rightharpoonup L_i$, where $\mathcal{J}Act \triangleq \prod_{i \in \mathcal{A}} Act_i$ is the set of *joint (global) actions* of all agents, is s.t. $T_i(l_i, \alpha, \mathfrak{cc}, X) = l_i'$ for some $l_i' \in L_i$ iff $a^i \in P_i(l_i)$, $\mathfrak{cc} \in C_{\mathcal{X}_i}$, and $X \subseteq \mathcal{X}_i$;
- a finite non-empty set of *local propositions* $PV_i = \{\mathsf{p}_i^1, \ldots, \mathsf{p}_i^{\mathsf{r}_i}\}$;
- a *local valuation function* $V_i : L_i \to 2^{PV_i}$.

For a local transition $t := l \xrightarrow{\alpha, \mathfrak{cc}, X} l'$ in a CMAS, l and l' are the *source* and *target* states, α is the executed *action*, clock condition $\mathfrak{cc}$ is called a *guard*, and X is the set of clocks to be *reset*.

Definition 2.2 (Model of CMAS). The *model* of CMAS is a 7-tuple $M = (\mathcal{A}, S, \iota, \mathcal{X}, \mathcal{I}, T, V)$, where:

- $\mathcal{A} = \{1, \ldots, n\}$ is the set of *agents*;
- $S = \prod_{i=1}^{n} L_i$ is the set of *global states*;
- $\iota = (\iota_1, \ldots, \iota_n) \in S$ is the *initial global state*;
- $\mathcal{X} = \bigcup_{i \in \mathcal{A}} \mathcal{X}_i$ is the set of *clocks*;
- $\mathcal{I}(s) = \bigwedge_{i \in \mathcal{A}} \mathcal{I}_i(s^i)$ is the *global invariant*, where s^i is the i-th local state of s;
- $T : S \times \mathcal{J}Act \times C_\mathcal{X} \times 2^\mathcal{X} \to S$ s.t. $T(s, \alpha, \bigwedge_i \mathfrak{cc}_i, \bigcup_{i=1}^n X_i) = s'$ iff $T_i(s^i, \alpha, \mathfrak{cc}_i, X_i) = s'^i$ for each $1 \leq i \leq n$;
- a *valuation function* $V \colon S \to 2^{PV}$, where $PV = \bigcup_{i=1}^n PV_i$.

The continuous (dense) semantics of time defines *concrete states* as tuples of global states and non-negative real clock valuations.

Definition 2.3 (CTS). The *concrete model* of a CMAS model $M = (\mathcal{A}, S, \iota, \mathcal{X}, \mathcal{I}, T, V)$ is given by its *Continuous Transition System* (CTS) $(\mathcal{A}, \mathcal{CS}, q_\iota, \to_c, V_c)$, where:

- $\mathcal{A} = \{1, \ldots, n\}$ is the set of *agents*;
- $\mathcal{CS} = S \times \mathbb{R}_{0+}^{n_X}$ is the set of *concrete states*;
- $q_\iota = (\iota, v) \in \mathcal{CS}$, such that $v(x_i) = 0$ for each $x_i \in \mathcal{X}$, is the *concrete initial state*;
- $\rightarrow_c \subseteq \mathcal{CS} \times (\mathcal{J}Act \cup \mathbb{R}_{0+}) \times \mathcal{CS}$ is the *transition relation*, defined by time- and action successors as follows:
  $(s, v) \xrightarrow{\delta}_c (s, v + \delta)$ for $\delta \in \mathbb{R}_{0+}$ and $v, v + \delta \in [\![\mathcal{I}(s)]\!]$,
  $(s, v) \xrightarrow{\alpha}_c (s', v')$ iff there are $\alpha \in \mathcal{J}Act$, $\mathfrak{cc} \in \mathcal{C}_\mathcal{X}$, $X \subseteq \mathcal{X}$ s.t.: $s \xrightarrow{\alpha, \mathfrak{cc}, X} s' \in T, v \in [\![\mathfrak{cc}]\!], v \in [\![\mathcal{I}(s)]\!], v' = v[X := 0], v' \in [\![\mathcal{I}(s')]\!]$,
- $V_c(s, v) = V(s)$ is the *valuation function*.

Intuitively, there are two types of transitions: delay transitions $\xrightarrow{\delta}_c$, which increase the clock valuation(s) by a given δ but do not change the global state, and action transitions $\xrightarrow{\alpha}_c$ which correspond to executing an enabled action in the CMAS and move the latter to a successor state, possibly resetting some clocks. Note that if the set of clocks is empty, the concrete model contains only action transitions, and thus it is identical with the model itself.

2.3 Discrete Time Synchronous MAS

Synchronous models with discrete time were considered for reasoning in **TATL** in [26, 28] using Tight Durational Concurrent Game Structures (TDCGS), which is a flat model, as opposed to a network of synchronising models, used in this paper. This gives an equivalent model whose definition is consistent with MAS. Indeed, *synchronous discrete time* MAS extend MAS with a constant duration associated with each individual transition.

Definition 2.4 (DMAS). A *discrete-time multi-agent system (DMAS)* consists of n agents $\mathcal{A} = \{1, \ldots, n\}$, where each agent $i \in \mathcal{A}$ is associated with a 7-tuple $AG_i = (L_i, \iota_i, Act_i, P_i, T_i, PV_i, V_i)$ including:

- a finite non-empty set of *local states* $L_i = \{l_i^1, l_i^2, \ldots, l_i^{n_i}\}$;
- an *initial local state* $\iota_i \in L_i$;
- a finite non-empty set of *local actions* $Act_i = \{a_i^1, a_i^2, \ldots, a_i^{m_i}\}$;
- a *local protocol* $P_i : L_i \to 2^{Act_i} \setminus \{\emptyset\}$ selecting the actions available at each local state;
- a (partial) *local transition function* $T_i : L_i \times \mathcal{J}Act \to L_i \times \mathbb{N}_+$ such that $T_i(l_i, \alpha)$ is defined iff $\alpha^i \in P_i(l_i)$, where α^i is the action of agent $i \in \mathcal{A}$ in the *joint action* $\alpha \in \mathcal{J}Act$; $T_i(l_i, \alpha) = (l_i', \delta_i)$ where δ_i is the *duration* of the transition;
- a finite non-empty set of *local propositions* $PV_i = \{\mathrm{p}_\mathrm{i}^1, \ldots, \mathrm{p}_\mathrm{i}^{\mathrm{r}_\mathrm{i}}\}$;
- a *local valuation function* $V_i : L_i \to 2^{PV_i}$.

This definition is similar to definition 2.7 with a duration associated with each transition. As for CMAS, the model of a DMAS is defined, that describes its behaviour.

Definition 2.5 (Model of DMAS). The *model* of DMAS is a 5-tuple $M = (\mathcal{A}, S, \iota, T, V)$, where:

- $\mathcal{A} = \{1, \ldots, n\}$ is the set of *agents*;
- $S = \prod_{i=1}^n L_i$ is the set of *global states*;
- $\iota = (\iota_1, \ldots, \iota_n) \in S$ is the *initial global state*;
- $T : S \times \mathcal{J}Act \to S \times \mathbb{N}_+$ is the partial *global transition function*, such that $T(s, \alpha) = (s', \delta)$ iff $T_i(s^i, \alpha) = (s'^i, \delta)$ for all $i \in \mathcal{A}$, where s^i is the i-th local state of s;
- $V : S \to 2^{PV}$ is the *valuation function* such that $V((l_1, \ldots, l_n)) = \bigcup_{i=1}^n V_i(l_i)$, where $PV = \bigcup_{i=1}^n PV_i$.

This definition enforces all synchronising actions to have the same duration in their respective local components. Another possibility could be to have individual durations δ_i for each component, and the longest duration $\max_{i=1}^n \delta_i$ for the synchronised transition. This would mimic actions of the different components taking place together, with the longest slowing down the whole execution.

Definition 2.6 (DTS). The concrete model of a DMAS is given by its *Duration Transition System* (DTS) $(\mathcal{A}, \mathcal{TS}, q_\iota, \mathcal{E}, V_d)$, where:

- $\mathcal{A} = \{1, \ldots, n\}$ is the set of *agents*;
- $\mathcal{TS} = S \times \mathbb{N}$ is a set of *timed states*;
- $q_\iota = (\iota, 0) \in \mathcal{TS}$ is the *initial timed state*;
- $\mathcal{E} : \mathcal{TS} \times \mathcal{J}Act \to \mathcal{TS}$ is a (partial) *transition function* such that $\mathcal{E}((s, d), \alpha) = (s', d+\delta)$ iff $T(s, \alpha) = (s', \delta)$, for $s, s' \in S$, $\alpha \in \mathcal{J}Act$, $d \in \mathbb{N}$, and $\delta \in \mathbb{N}_+$;
- $V_d(s, d) = V(s)$ is the *valuation function*.

It is straightforward to see that these compositional definitions correspond to those of the flat structures (TDCGS and DTS) in [26].

2.4 Untimed MAS

Untimed MAS can be defined as Timed MAS with no clocks, see below. Note that the definition is essentially equivalent to the concept of an *interpreted system* in [32].

Definition 2.7 (MAS). An *untimed multi-agent system*, simply MAS, is a CMAS with every $\mathcal{X}_i = \emptyset$. The model and concrete model of MAS are equal and defined as in Definition 2.2, without clocks.

3 SEMANTICS OF LOGICS

We start with defining strategies and their outcomes.

3.1 Strategies

The taxonomy proposed by Schobbens [33] defines four strategy types based on agents' *state information*: perfect (I) vs. imperfect (i), and their *recall of state history*: perfect (R) vs. imperfect (r).

Intuitively, a strategy can be seen as a conditional plan that dictates the choice of an agent in each possible situation. In *perfect information* strategies, agents have complete knowledge about global states of the model and thus can make different choices in each one. Under *imperfect information*, decisions can only be made based on local states. *Perfect recall* assumes that agents have access to a full history of previously visited states, whereas under *imperfect recall* only the current state is explicitly known. Formally:

- A *memoryless imperfect information* (ir) *strategy* for $i \in \mathcal{A}$ is a function $\sigma_i : L_i \to Act_i$ such that $\sigma_i(l) \in P_i(l)$ for each $l \in L_i$.
- A *memoryless perfect information* (Ir) *strategy* for $i \in \mathcal{A}$ is a function $\sigma_i : S \to Act_i$ such that $\sigma_i(s) \in P_i(s^i)$ for each $s \in S$.
- A *perfect recall, imperfect information* (iR) *strategy* for $i \in \mathcal{A}$ is a function $\sigma_i : L_i^+ \to Act_i$ s.t. $\sigma_i(h) \in P_i(last(h))$ for each $l \in L_i$.
- A *perfect recall, perfect information* (IR) *strategy* for $i \in \mathcal{A}$ is a function $\sigma_i : S^+ \to Act_i$ s.t. $\sigma_i(H) \in P_i(last(H)^i)$ for each $s \in S$.

By $H \in S^+$ (resp. $h \in L_i^+$), we denote a history of global (resp. i's local) states, and $last(H)$ (resp. $last(h)$) refers to its last state. The notion of a strategy can be generalized to an agent coalition $A \subseteq \mathcal{A}$, whose *joint strategy* σ_A is a tuple of strategies, one for each $i \in A$.

We now formally define executions in concrete models of MAS.

Definition 3.1 (Execution). Let $(\mathcal{A}, \mathcal{CS}, q_\iota, \rightarrow_c, V_c)$ be a CTS. Its execution from $q_0 = (s_0, v_0)$ is $\pi = q_0, \delta_0, q'_0, \alpha_0, q_1, \delta_1, q'_1, \alpha_1, \ldots$, where $q_k = (s_{2k}, v_{2k})$, $q'_k = (s_{2k+1}, v_{2k+1})$, such that $\delta_k \in \mathbb{R}_{0+}$, $\alpha_k \in \mathcal{J}Act$, $q_k \xrightarrow{\delta_k}_c q'_k \xrightarrow{\alpha_k}_c q_{k+1}$, for each $k \geq 0$.

An execution of a DTS $(\mathcal{A}, \mathcal{TS}, q_\iota, \mathcal{E}, V_d)$ from q_0 is $\pi = q_0, \delta_0, \alpha_0, q_1, \delta_1, \alpha_1, \ldots$, where $q_k = (s_k, d_k)$, $d_0 = 0$, s.t. $\alpha_k \in \mathcal{J}Act$, $\delta_k \in \mathbb{N}_+$, $\mathcal{E}((s_k, d_k), \alpha_k) = (s_{k+1}, d_{k+1})$, $d_{k+1} = d_k + \delta_k$, for each $k \geq 0$.

An execution of a (concrete) model of an untimed MAS from s_0 is $\pi = s_0, \alpha_0, s_1, \alpha_1, \ldots$, s.t. $\alpha_k \in \mathcal{J}Act$, $s_k \xrightarrow{\alpha_k}_c s_{k+1}$, for each $k \geq 0$.

Note that if the set of clocks is empty in CMAS, then the executions of CTS contain only action transitions, as in an untimed MAS.

The outcome of a strategy σ_A represents executions where the agents in A adopt σ_A, i.e. it is the set of all paths in the model that may occur when the coalition strictly follows the strategy, while opponents freely choose from actions permitted by their protocols.

Definition 3.2 (Outcome). Let $A \subseteq \mathcal{A}$, $Y \in \{\mathrm{ir}, \mathrm{Ir}, \mathrm{iR}, \mathrm{IR}\}$, M^{C} (resp. $M^{\mathrm{D}}, M^{\mathrm{U}}$) be the model of a CMAS (resp. a DMAS, an untimed MAS), and let $\pi^{\mathrm{C}} = q_0, \delta_0, q'_0, \alpha_0, \ldots$ (resp. $\pi^{\mathrm{D}} = q_0, \delta_0, \alpha_0, \ldots$; $\pi^{\mathrm{U}} = s_0, \alpha_0, \ldots$) be an execution of the corresponding concrete model.

The *outcome* of Y-strategy σ_A in state g^Z of the concrete model of M^Z, where $Z \in \{\mathrm{C}, \mathrm{D}, \mathrm{U}\}$, $g^Z = q_0$ for $Z \in \{\mathrm{C}, \mathrm{D}\}$, and $g^Z = s_0$ for $Z = \mathrm{U}$, is the set $out^Y_{M^Z}(g^Z, \sigma_A)$, such that $\pi^Z \in out^Y_{M^Z}(g^Z, \sigma_A)$ iff for each $m \geq 0$ and each agent $i \in A$:

$(Y = \mathrm{ir})$: $\alpha^i_m = \sigma_i(s^i_m)$,

$(Y = \mathrm{iR})$: $\alpha^i_m = \sigma_i(h^Z)$, $\quad h^Z = \begin{cases} s^i_0, s^i_0, s^i_1, s^i_1, \ldots, s^i_m & \text{if } Z = \mathrm{C} \\ s^i_0, s^i_1, \ldots, s^i_m & \text{otherwise} \end{cases}$

$(Y = \mathrm{Ir})$: $\alpha^i_m = \sigma_i(s_m)$,

$(Y = \mathrm{IR})$: $\alpha^i_m = \sigma_i(H^Z)$, $\quad H^Z = \begin{cases} s_0, s_0, s_1, s_1, \ldots, s_m & \text{if } Z = \mathrm{C} \\ s_0, s_1, \ldots, s_m & \text{otherwise} \end{cases}$

where $s_j \in S$ is the global state component of q_j in π^{C} and π^{D}.

3.2 Semantics of TATL and STCTL

We give the discrete-time semantics of **TATL** [28] and the continuous-time semantics of **STCTL** for strategies of type $Y \in \{\mathrm{ir}, \mathrm{Ir}, \mathrm{iR}, \mathrm{IR}\}$.

*Definition 3.3 (**TATL** Semantics).* Let $M = (S, \iota, T, V)$ be a DMAS model, $(\mathcal{A}, \mathcal{TS}, q_\iota, \mathcal{E}, V_d)$ be its DTS, $s = s_0 \in S$ a state, $\varphi, \psi \in$ **TATL**, $\pi = (s_0, d_0), \delta_0, \alpha_0, \ldots$ an execution of the DTS, and $A \subseteq \mathcal{A}$ a set of agents. The Y-semantics of **TATL** is given by the clauses:

- $M, s \models \mathrm{p}$ iff $\mathrm{p} \in V(s)$,
- $M, s \models \neg\varphi$ iff $M, s \not\models \varphi$,
- $M, s \models \varphi \wedge \psi$ iff $M, s \models \varphi$ and $M, s \models \psi$,
- $M, s \models \langle\langle A \rangle\rangle X\varphi$ iff there exists a joint Y-strategy σ_A s.t. for each $\pi \in out^Y_M(s, \sigma_A)$ we have $M, s_1 \models \varphi$,
- $M, s \models \langle\langle A \rangle\rangle \varphi U_I \psi$ iff there exists a joint Y-strategy σ_A s.t. for each $\pi \in out^Y_M(s, \sigma_A)$ there exists $i \in \mathbb{N}$ s.t. $d_i \in I$ and $M, s_i \models \psi$ and for all $0 \leq j < i : M, s_j \models \varphi$,
- $M, s \models \langle\langle A \rangle\rangle \varphi R_I \psi$ iff there exists a joint Y-strategy σ_A s.t. for each $\pi \in out^Y_M(s, \sigma_A)$ for each $i \in \mathbb{N}$ such that $d_i \in I$ we have $M, s_i \models \psi$ or there exists $0 \leq j < i : M, s_j \models \varphi$.

*Definition 3.4 (**STCTL** Semantics).* Let $M = (\mathcal{A}, \mathcal{CS}, q_\iota, \rightarrow_c, V_c)$ be a CTS, $q_0 = (s, v) = (s_0, v_0) \in \mathcal{CS}$ a concrete state, $A \subseteq \mathcal{A}$, $\varphi, \psi \in$ **STCTL**, $\pi = q_0, \delta_0, q'_0, \alpha_0, \ldots$ an execution of M where $q_k = (s_{2k}, v_{2k})$, $q'_k = (s_{2k+1}, v_{2k+1})$, and let $time_\pi(q_k) = \sum_{j=0}^{k-1} \delta_j$, $time_\pi(q'_k) = \sum_{j=0}^{k} \delta_j$. The Y-semantics of **STCTL** is given as:

- $M, (s, v) \models \mathrm{p}$ iff $\mathrm{p} \in V_c(s, v)$,
- $M, (s, v) \models \neg\varphi$ iff $M, (s, v) \not\models \varphi$,
- $M, (s, v) \models \varphi \wedge \psi$ iff $M, (s, v) \models \varphi$ and $M, (s, v) \models \psi$,
- $M, (s, v) \models \langle\langle A \rangle\rangle \gamma$ iff there exists a joint Y-strategy σ_A such that we have $M, out^Y_M((s, v), \sigma_A) \models \gamma$, where:
  - $M, out^Y_M((s, v), \sigma_A) \models \mathrm{p}$ iff $\mathrm{p} \in V_c(s, v)$,
  - $M, out^Y_M((s, v), \sigma_A) \models \neg\varphi$ iff $M, out^Y_M((s, v), \sigma_A) \not\models \varphi$,
  - $M, out^Y_M((s, v), \sigma_A) \models \varphi \wedge \psi$ iff $M, out^Y_M((s, v), \sigma_A) \models \varphi$ and $M, out^Y_M((s, v), \sigma_A) \models \psi$,
  - $M, out^Y_M((s, v), \sigma_A) \models \forall \mathrm{X}\, \varphi$ iff for each $\pi \in out^Y_M((s, v), \sigma_A)$ we have $M, (\pi, \sigma_A, Y) \models \mathrm{X}\, \varphi$,
  - $M, out^Y_M((s, v), \sigma_A) \models \forall \varphi U_I \psi$ iff for each $\pi \in out^Y_M((s, v), \sigma_A)$ we have $M, (\pi, \sigma_A, Y) \models \varphi U_I \psi$,
  - $M, out^Y_M((s, v), \sigma_A) \models \exists \varphi U_I \psi$ iff for some $\pi \in out^Y_M((s, v), \sigma_A)$ we have $M, (\pi, \sigma_A, Y) \models \varphi U_I \psi$,
  - $M, out^Y_M((s, v), \sigma_A) \models \forall \varphi R_I \psi$ iff for each $\pi \in out^Y_M((s, v), \sigma_A)$ we have $M, (\pi, \sigma_A, Y) \models \varphi R_I \psi$,
  - $M, out^Y_M((s, v), \sigma_A) \models \exists \varphi R_I \psi$ iff for some $\pi \in out^Y_M((s, v), \sigma_A)$ we have $M, (\pi, \sigma_A, Y) \models \varphi R_I \psi$, where
    - $M, (\pi, \sigma_A, Y) \models \mathrm{X}\, \varphi$ iff $M, out^Y_M(s_1, \sigma_A) \models \varphi$ (untimed only)
    - $M, (\pi, \sigma_A, Y) \models \varphi U_I \psi$ iff there is $i \in \mathbb{N}$ s.t. $time_\pi(s_i, v_i) \in I$ and we have $M, out^Y_M((s_i, v_i), \sigma_A) \models \psi$ and for all $0 \leq j < i$: $M, out^Y_M((s_j, v_j), \sigma_A) \models \varphi$,
    - $M, (\pi, \sigma_A, Y) \models \varphi R_I \psi$ iff for each $i \in \mathbb{N}$ s.t. $time_\pi(s_i, v_i) \in I$ we have $M, out^Y_M((s_i, v_i), \sigma_A) \models \psi$ or there exists $0 \leq j < i$: $M, out^Y_M((s_j, v_j), \sigma_A) \models \varphi$.
  - $M, out^Y_M((s, v), \sigma_A) \models \langle\langle A' \rangle\rangle \gamma$ iff $M, (s, v) \models \langle\langle A' \rangle\rangle \gamma$.

By $\mathcal{L}^{\mathcal{M}}_{\mathcal{S}}$, we denote the logical system being considered, where $\mathcal{L}$ is the syntactic variant (see Section 2.1), $\mathcal{S}$ is the class of strategies (cf. Section 3.1), and $\mathcal{M} \in \{\mathrm{C}, \mathrm{D}, \mathrm{U}\}$ is the class of continuous-time, discrete-time, and untimed models, respectively. Superscripts C and U, if omitted, are assumed to follow from the syntactic variant.

4 MODEL CHECKING RESULTS

We now recall complexity results under different semantics for **ATL** (Sec. 4.1) and **TATL** (Sec. 4.2). Then, in Secs. 4.3 to 4.6, we provide new results for **SCTL** and **STCTL**. It is important to note that complexity results are given wrt. the *model* size, as defined in Sec. 2. In particular, note that the model and the concrete model are not equal in each case. The results are summarised in Table 1.

4.1 Model Checking ATL

The standard fixpoint algorithm for model checking **ATL** under perfect information was presented in the original paper by Alur, Henzinger and Kupferman [7]. In a nutshell, to verify a formula $\langle\langle A \rangle\rangle \varphi$, it starts with a candidate set of states (chosen appropriately depending on φ) and then iterates backwards over the abilities of coalition A at each step [19]. Model checking of **ATL** is **PTIME**-complete [7], for both memoryless and perfect recall strategies, since the satisfaction semantics for $\mathbf{ATL}_{\mathrm{Ir}}$ and $\mathbf{ATL}_{\mathrm{IR}}$ coincide [19].

On the other hand, the fixpoint-based approach cannot be adapted to imperfect information [2, 11], making model checking significantly more complex in this setting: $\mathbf{\Delta^P_2}$ for $\mathbf{ATL}_{\mathrm{ir}}$ and NP-hard for simple instances of $\mathbf{ATL}_{\mathrm{ir}}$ [19], and undecidable for $\mathbf{ATL}_{\mathrm{iR}}$ [12].

4.2 Model Checking TATL

Algorithms and complexity results are given in [28] for model checking **TATL** over TDCGS, which are analogous to DMAS. The strategies considered are defined on histories of concrete states, i.e. of type IRT, where T refers to the timed strategy. Furthermore, analogously to **ATL***, the semantics of $\mathbf{TATL_{IrT}}$ and $\mathbf{TATL_{IRT}}$ coincide [26]. Model checking $\mathbf{TATL_{IrT}}$ and $\mathbf{TATL_{IRT}}$ is **EXPTIME**-complete in the general case [28, Theorem 13], and **PTIME**-complete for the subset that excludes equality in time constraints [28, Theorem 14].

It is important to note, though, that these results are given wrt. the formula size; in particular, the exponential blowup in the general case is attributed solely to the binary encoding of constraints in **TATL** formulas, while the algorithm is actually in **PTIME** wrt. the model size [28, Theorem 12]. Hence, we put the latter in Table 1, as it cannot be otherwise compared with our results for **S(T)CTL**.

We are not aware of any works investigating **TATL** under imperfect information thus far. However, for $\mathbf{TATL_{ir}}$ we can at least establish upper and lower bounds of **PSPACE** (since $\mathbf{TATL} \subset \mathbf{STCTL}$) and $\mathbf{\Delta_2^P}$ (since $\mathbf{ATL} \supset \mathbf{TATL}$), respectively. Furthermore, $\mathbf{TATL_{iR}}$ model checking is undecidable since it is already the case for $\mathbf{ATL_{iR}}$.

4.3 Model Checking SCTL with r-Strategies

Under the memoryless semantics of strategic ability, the complexity of model checking **SCTL** can be established analogously to the way it was done for **ATL** [19]. We begin with $\mathbf{SCTL_{ir}}$.

Algorithm 1: MCHECK$\mathbf{STCTL_{ir,Ir}}(M, s, \langle\langle A\rangle\rangle\gamma)$

- Guess a strategy σ_A.
- Prune M by removing transitions not consistent with σ_A.
- If $\langle\langle A\rangle\rangle\gamma$ is an **STCTL** formula, run **TCTL** model checking on γ, else run **CTL** model checking on γ.

Theorem 4.1. *Model checking* $\mathbf{SCTL_{ir}}$ *is* $\mathbf{\Delta_2^P}$*-complete.*

Proof. Let $\varphi = \langle\langle A\rangle\rangle\gamma$ be an $\mathbf{SCTL_{ir}}$ formula without nested strategic modalities. Consider the procedure in Alg. 1. It runs in **NP** for input φ, since σ_A can be guessed in non-deterministic polynomial time, while pruning transitions and model checking of **CTL** formula γ requires deterministic polynomial time. Note that for an arbitrary $\mathbf{SCTL_{ir}}$ formula, nested strategic modalities can be replaced with fresh propositions by calling Alg. 1 recursively, bottom up. This requires polynomially many calls (wrt. the formula size) to an **NP** oracle executing Alg. 1, hence the upper bound of $\mathbf{P^{NP}} = \mathbf{\Delta_2^P}$.

The lower bound follows from the fact that $\mathbf{SCTL_{ir}}$ subsumes $\mathbf{ATL_{ir}}$ whose model checking complexity is $\mathbf{\Delta_2^P}$ [19, Th. 34]. □

For $\mathbf{SCTL_{Ir}}$, i.e., strategies with perfect information, a more involved construction is required to establish the lower bound.

Theorem 4.2. *Model checking* $\mathbf{SCTL_{Ir}}$ *is* $\mathbf{\Delta_2^P}$*-complete.*

Proof. The upper bound follows exactly as for $\mathbf{SCTL_{ir}}$ (Th. 4.1).

The lower bound is obtained by a reduction from the model checking problem for $\mathbf{ATL_{Ir}}$, which is $\mathbf{\Delta_2^P}$-complete [20]. We present an overview of the reduction below, and refer the reader to [9, Theorem 4.2] for more technical details on the constructions used.

Let M be an untimed model, and φ an $\mathbf{ATL_{ir}}$ formula. First, we reconstruct the model by cloning states in M so that they record the latest action profile that has been executed, as in [16, Section 3.4]. That is, for each state q in M and incoming transition labeled by $(a_1, \dots, a_n)$, we create a new state $(q, a_1, \dots, a_n)$, and direct the transition to that state. Moreover, we label the new state by fresh atomic propositions $\mathsf{exec}_{1,a_1}, \dots, \mathsf{exec}_{n,a_n}$ that can be used to capture the latest decision of each agent within the formulas of the logic. We denote the resulting model by M'.

Then, we temporarily add epistemic operators K_i to the language, with the standard observational semantics (i.e., $K_i\varphi$ holds in state s iff φ is true in all the states s' such that $(s')^i = s^i$). The uniformity of agent i's play can be now captured by the following **CTLK** formula: $unif_i \equiv \forall\mathsf{G}\,(\bigvee_{a\in Act_i} K_i\forall\mathsf{X}\,\mathsf{exec}_{i,a})$. We reconstruct formula φ by replacing every occurrence of $\langle\langle A\rangle\rangle\psi$ with $\langle\langle A\rangle\rangle(\forall\psi \wedge \bigwedge_{i\in A} unif_i)$. We denote the resulting **SCTLK** formula by φ'. It is easy to see that $M, s \models_{\mathbf{ATL_{ir}}} \varphi$ iff $M', s \models_{\mathbf{SCTL_{Ir}K}} \varphi'$.

Finally, we do a translation from **SCTLK** to **SCTL** by a straightforward adaptation of the construction in [18, Section 4.2]. For each agent $i \in A$, we add an "epistemic ghost" e_i to the set of agents. Then, we simulate the indistinguishability of states in M' with transitions effected by the epistemic ghosts. That is, we add transitions controlled by e_i between each pair of states s, s' with $s^i = (s')^i$. We also replace the knowledge operators in φ' by appropriate strategic subformulas for e_i, see [18, Section 4.2] for the details. The resulting translations of M' and φ' are denoted by M'' and φ''. Analogously to [18, Theorem 1], we get that $M', s \models_{\mathbf{SCTL_{Ir}K}} \varphi'$ iff $M'', s \models_{\mathbf{SCTL_{Ir}}} \varphi''$ (note that we need to extend the proof of [18, Theorem 1] to Boolean combinations of reachability/safety objectives, but that is also straightforward). This completes the reduction. □

4.4 Model Checking SCTL with R-Strategies

As with the corresponding variants of **ATL** and logics that extend it, we immediately obtain undecidability for $\mathbf{SCTL_{iR}}$.

Theorem 4.3. *Model checking* $\mathbf{SCTL_{iR}}$ *is undecidable.*

Proof. Follows from the fact **SCTL** subsumes **ATL**, whose model checking is undecidable in the iR semantics [12, Theorem 1]. □

For $\mathbf{SCTL_{IR}}$, we obtain **EXPTIME**-completeness via a reduction of the *module checking* problem [27] for **CTL** as follows.

Algorithm 2: MCHECK$\mathbf{SCTL_{IR}}(M, s, \langle\langle A\rangle\rangle\gamma)$

- Split agents into two groups: coalition A and opponents O.
- Merge A and O into single agents by creating auxiliary action labels for tuples of actions belonging to agents in A and O.
- Run **CTL** module checking on γ.

Theorem 4.4. *Model checking* $\mathbf{SCTL_{IR}}$ *is* **EXPTIME**-*complete.*

Proof. (Sketch) The upper bound follows from the procedure in Alg. 2, which runs in **EXPTIME**, since **CTL** module checking is **EXPTIME**-complete. If φ has nested coalition operators, they can be eliminated by proceeding recursively bottom up, requiring polynomially many calls to Alg. 2 (wrt. formula size). Thus the procedure still runs in **EXPTIME** for an arbitrary $\mathbf{SCTL_{IR}}$ formula.

The lower bound follows from the fact **CTL** module checking, which is **EXPTIME**-complete [27, Th. 3.1], can be seen as a special case of $\mathbf{SCTL_{IR}}$ model checking where we have a single strategic operator at the beginning of a formula and only two agents. □

4.5 Model Checking STCTL with r-Strategies

For **STCTL** with memoryless strategies, we have that model checking is **PSPACE**-complete, i.e. it remains unchanged from **TCTL**.

Theorem 4.5. *Model checking* $\mathbf{STCTL_{ir}}$ *is* **PSPACE***-complete.*

Proof. Let $\varphi = \langle\langle A \rangle\rangle \gamma$ be an $\mathbf{STCTL_{ir}}$ formula without nested strategic modalities. The upper bound follows analogously to the case of $\mathbf{SCTL_{ir}}$ (cf. Theorem 4.1). Alg. 1 runs in **NPSPACE** = **PSPACE** for input φ (since rather than **CTL**, **TCTL** model checking, which is in **PSPACE**, is now called on γ). For an arbitrary $\mathbf{STCTL_{ir}}$ formula, eliminating nested modalities requires polynomially many calls to Alg. 1, thus we obtain the upper bound of $\mathbf{P}^{\mathbf{PSPACE}}$ = **PSPACE**.

The lower bound follows from the fact **STCTL** subsumes **TCTL**, whose model checking is **PSPACE**-complete [3]. □

Theorem 4.6. *Model checking* $\mathbf{STCTL_{Ir}}$ *is* **PSPACE***-complete.*

Proof. Both bounds follow exactly as in Th. 4.5 for $\mathbf{STCTL_{ir}}$. □

4.6 Model Checking STCTL with R-strategies

Finally, under perfect recall model checking of **STCTL** is undecidable, which for iR semantics directly follows from prior results, and for IR semantics is obtained via a reduction to **TCTL** games [14].

Theorem 4.7. *Model checking* $\mathbf{STCTL_{iR}}$ *is undecidable.*

Proof. Follows from the fact **STCTL** subsumes **ATL**, whose model checking is undecidable for iR strategies [12, Theorem 1]. □

Theorem 4.8. *Model checking* $\mathbf{STCTL_{IR}}$ *is undecidable.*

Proof. (Sketch) Undecidability follows from the fact that **TCTL** games [14] can be seen as a special case of $\mathbf{STCTL_{IR}}$ model checking, with a single strategic operator at the beginning of the formula, and two agents (obtained from grouping together all coalition agents and all opponents as in Alg. 2). Since TCTL games are undecidable for unrestricted **TCTL** [14, Theorem 3], clearly this is also the case for the more general case with nested strategic modalities. □

5 EXPRESSIVITY RESULTS

We can see in Section 4 that using the broader syntax of **STCTL** (**SCTL**), rather than **TATL** (**ATL**, resp.), does not significantly worsen the complexity of model checking, especially for the imperfect information semantics. In this section, we show that, in addition, it strictly increases the expressivity of the logic. We start by recalling the formal definitions of expressive and distinguishing power.

Definition 5.1 (Expressive power and distinguishing power [35]). Consider two logical systems L_1 and L_2, with their semantics defined over the same class of models $\mathcal{M}$. L_1 is *at least as expressive as* L_2 (written $L_2 \preceq_e L_1$) if, for every formula φ_2 of L_2, there exists a formula φ_1 of L_1, such that φ_1 and φ_2 are satisfied in the same models from $\mathcal{M}$.

Moreover, L_1 is *at least as distinguishing as* L_2 ($L_2 \preceq_d L_1$) if every pair of models $M, M' \in \mathcal{M}$ that can be distinguished by a formula of L_2 can also be distinguished by some formula of L_1.

It is easy to see that $L_2 \preceq_e L_1$ implies $L_2 \preceq_d L_1$. By transposition, we also have that $L_2 \not\preceq_d L_1$ implies $L_2 \not\preceq_e L_1$. The following is straightforward.

[1] **PTIME** wrt. model size, **EXPTIME** wrt. formula length (cf. Sec. 4.2).

	Ir	IR	ir	iR
$\mathbf{ATL^U}$	**PTIME** [7, Th. 5.2] (Ir, IR semantics coincide)		$\Delta_2^{\mathbf{P}}$ [19, Th. 34]	**Undecidable** [12, Th. 1] ($\supseteq \mathbf{ATL_{iR}}$)
$\mathbf{SCTL^U}$	$\Delta_2^{\mathbf{P}}$ [Th. 4.2]	**EXPTIME** [Th. 4.4]	$\Delta_2^{\mathbf{P}}$ [Th. 4.1]	
$\mathbf{ATL^{*U}}$	**PSPACE** [19, Th. 23]	**2EXPTIME** [7, Th. 5.6]	**PSPACE** [19, Th. 38]	
$\mathbf{TATL^D}$	**PTIME**[1] [Sec. 4.2] (Ir, IR semantics coincide)		$\Delta_2^{\mathbf{P}}$−**PSPACE**	
$\mathbf{STCTL^C}$	**PSPACE** [Th. 4.6]	**Undecidable** [Th. 4.8]	**PSPACE** [Th. 4.5]	
$\mathbf{TATL^{*C}}$	**Undecidable** [10, Th. 4.3] ($\supset$ **MTL**)			

Table 1: Model checking complexity wrt. the model size, for STCTL, its subsets, and $\mathbf{ATL^{*U}}$, $\mathbf{TATL^{*C}}$ for comparison. Undecidability ($\mathbf{ATL^U_{iR}}$, MTL) propagates to more expressive logics.

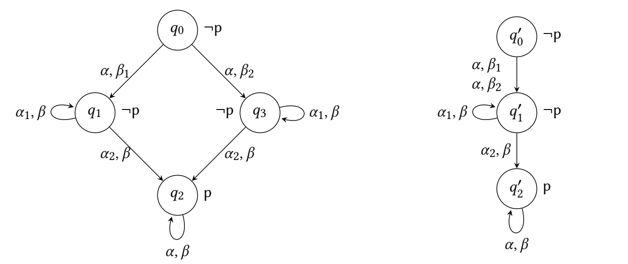

Figure 1: Agent template a (left) and agent a' (right)

Proposition 5.2. *For any strategy type* $\mathcal{S}$ *and model type* $\mathcal{M}$*, we have that* $\mathbf{TATL}^{\mathcal{M}}_{\mathcal{S}} \preceq_e \mathbf{STCTL}^{\mathcal{M}}_{\mathcal{S}}$ *(and thus* $\mathbf{TATL}^{\mathcal{M}}_{\mathcal{S}} \preceq_d \mathbf{STCTL}^{\mathcal{M}}_{\mathcal{S}}$*).*

Proof. Follows as **TATL** is a syntactic restriction of **STCTL**. □

Proposition 5.3. *For any strategy type* $\mathcal{S}$ *and model type* $\mathcal{M}$*, we have that* $\mathbf{STCTL}^{\mathcal{M}}_{\mathcal{S}} \not\preceq_d \mathbf{TATL}^{\mathcal{M}}_{\mathcal{S}}$ *(and thus* $\mathbf{STCTL}^{\mathcal{M}}_{\mathcal{S}} \not\preceq_e \mathbf{TATL}^{\mathcal{M}}_{\mathcal{S}}$*).*

Proof. The proof is inspired by the proof of [1, Proposition 4]. Let us construct two multi-agent systems S, S', each with $\mathcal{A} = \{1, 2\}$. Both agents in S are based on the agent template a, depicted in Figure 1 (left), with the empty sets of clocks. Note that the model M of the system is isomorphic with the agent template, and the concrete model is identical with the model.

Similarly, both agents in S' are based on the agent template a', depicted in Figure 1 (right), again with no clocks. The model M' of the system is isomorphic with the agent template, and its concrete model identical with M'. Moreover, M and M' are models with perfect information, in the sense that the local state of agent 1 (resp. 2) always uniquely identifies the global state in the model. Thus, the sets of available strategies with perfect and imperfect information coincide, and likewise of untimed vs. timed strategies. Furthermore, the strategic abilities for strategies with perfect vs. imperfect recall are the same for properties expressible in **ATL** [7].

It is easy to see that the pointed models (M, q_0q_0) and $(M', q'_0q'_0)$ are in alternating bisimulation [5], and thus they satisfy exactly the same formulas of $\mathbf{ATL}_{\mathrm{ir}}$. By the above argument, they must satisfy the same formulas of $\mathbf{TATL}^{\mathcal{M}}_{\mathcal{S}}$, for any $\mathcal{M} \in \{\mathrm{C}, \mathrm{U}\}$ and all the strategy types $\mathcal{S}$ considered in this paper. On the other hand, we have that the **SCTL** (and hence also **STCTL**) formula $\varphi \equiv \langle\langle 1 \rangle\rangle(\exists \mathsf{F}_{[0,\infty)} \mathsf{p} \wedge \exists \mathsf{G}_{[0,\infty)} \neg \mathsf{p})$ holds in (M, q_0q_0) but not in $(M', q'_0q'_0)$ for all the strategy types $\mathcal{S}$ and model types $\mathcal{M} \in \{\mathrm{C}, \mathrm{U}\}$.

For $\mathcal{M} = \mathrm{D}$, we adapt the above construction by assuming that each transition consumes 1 unit of time. The models M, M' of S, S' are still isomorphic with S, S', and their concrete models CM, CM' are the tree-unfoldings of M, M', thus they are alternating-bisimilar with M, M' [1]. In consequence, they satisfy the same formulas of $\mathbf{TATL}^{\mathrm{D}}_{\mathcal{S}}$, for all strategy types $\mathcal{S}$. On the other hand, the above **SCTL** and **STCTL** formula φ holds in (M, q_0q_0) but not in $(M', q'_0q'_0)$ for $\mathcal{M} = \mathrm{D}$ and all $\mathcal{S}$. □

The following is a straightforward corollary.

Theorem 5.4. *For any strategy type $\mathcal{S}$ and model type $\mathcal{M}$, $\mathbf{STCTL}^{\mathcal{M}}_{\mathcal{S}}$ has strictly larger expressive and distinguishing power than $\mathbf{TATL}^{\mathcal{M}}_{\mathcal{S}}$.*

6 THE ASYNCHRONOUS CASE

This section considers the case of asynchronous multi-agent systems (AMAS), providing the syntax and semantics of continuous time, discrete time, and untimed AMAS.

6.1 Asynchronous MAS

Asynchronous Multi-Agent Systems (AMAS [22]) are a modern semantic model for the study of agents' strategies in asynchronous systems. Technically, AMAS are similar to networks of automata that synchronise on shared actions, and interleave local transitions to execute asynchronously [15, 22, 29]. However, to deal with agents coalitions, automata semantics (e.g. for Timed Automata) must resort to algorithms and additional attributes. In contrast, by linking protocols to agents, AMAS are a natural compositional formalism to analyse multi-agent systems.

6.2 Continuous Time AMAS

Definition 6.1 (CAMAS). A continuous time AMAS (CAMAS) is defined as CMAS except for the following component:

- a (partial) *local transition function* $T_i : L_i \times Act_i \times C_{\mathcal{X}_i} \times 2^{\mathcal{X}_i} \rightharpoonup L_i$ such that $T_i(l_i, a, \mathfrak{cc}, X) = l'_i$ for some $l'_i \in L_i$ iff $a \in P_i(l_i)$, $\mathfrak{cc} \in C_{\mathcal{X}_i}$, and $X \subseteq \mathcal{X}_i$;

Note that as opposed to synchronous MAS in Def. 2.7, the local transition function of AMAS is defined on local actions only. This is also reflected in the formal definition of AMAS models, also called Interleaved Interpreted Systems [22, 30].

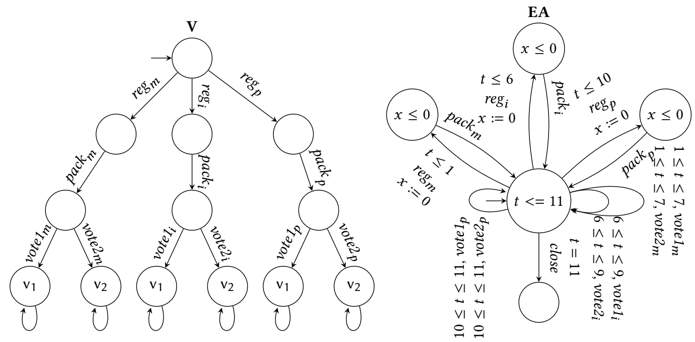

Figure 2: The CAMAS of the voting scenario from Ex. 6.4.

Definition 6.2 (Model of a CAMAS). Let $PV = \bigcup_{i=1}^{n} PV_i$ be the union of the local propositions. The *model* of a CAMAS is defined as the model for a CMAS except for the following component:

- $T : S \times Act \times C_{\mathcal{X}} \times \mathcal{X} \to S$ s.t. $T(s, a, \bigwedge_{i \in Agent(a)} \mathfrak{cc}_i, \mathcal{X}) = s'$ iff $\forall i \in Agent(a)$, $T_i(s^i, a, \mathfrak{cc}_i, X_i) = s'^i$, whereas $\forall i \in A \setminus Agent(a)$, $s^i = s'^i$;

Definition 6.3 (ACTS). The *concrete model* of a CAMAS model is defined as the CTS of a CMAS model except for the component:

- $\to_c \subseteq \mathcal{CS} \times (Act \cup \mathbb{R}_{0+}) \times \mathcal{CS}$ is the *transition relation*, defined by time- and action successors as follows:
  $(s, v) \xrightarrow{\delta}_c (s, v + \delta)$ for $\delta \in \mathbb{R}_{0+}$ and $v, v + \delta \in [\![\mathcal{I}(s)]\!]$,
  $(s, v) \xrightarrow{a}_c (s', v')$ iff there are $a \in Act$, $\mathfrak{cc} \in C_{\mathcal{X}}$, $X \subseteq \mathcal{X}$ s.t.: $s \xrightarrow{a, \mathfrak{cc}, X} s' \in T, v \in [\![\mathfrak{cc}]\!], v \in [\![\mathcal{I}(s)]\!], v' = v[X := 0]\; v' \in [\![\mathcal{I}(s')]\!]$.

Example 6.4. Consider the simple voting model in Fig. 2, inspired by the election procedures in Estonia [34]. The voter (V) needs to register first, selecting one of the three voting modalities: postal vote by mail (reg_m), e-vote over the internet (reg_i), or a traditional paper ballot at a polling station (reg_p). The election authority (EA) accepts V's registration by synchronising with the registration transition. It then proceeds to send a voting package appropriate for V's chosen modality ($pack_m$, $pack_i$ or $pack_p$), e.g. a postal ballot for voting by mail, e-voting access credentials, or the address of the local election office. After receiving the package, V casts a vote for the selected candidate ($vote1_m$, etc.), which is registered by EA. The local proposition $\mathsf{v_i}$ denotes that V voted for candidate i.

Time frames are associated with the voting process in the EA automaton, which accepts votes by mail between times 1 and 7, by internet between 6 and 9, and at the polling station between 10 and 11. The ballot is closed at time 11. Moreover, a voter must be registered for a modality before its respective voting period starts.

6.3 Discrete Time AMAS

Discrete time asynchronous multi-agent systems extend AMAS with discrete time, in a way similar to the synchronous case.

Definition 6.5 (DAMAS). A discrete time AMAS (DAMAS) is defined as DMAS except for the following component:

- a (partial) *local transition function* $T_i : L_i \times Act_i \rightharpoonup L_i \times \mathbb{N}_+$ such that $T_i(l_i, a)$ is defined iff $a \in P_i(l_i)$;

However, when agents share an action, the time the action takes is not enforced to be the same for all participants. Instead, the duration of the global action is the maximum of the participating agents' durations. Thus, the slowest agent slows down its partners.

Definition 6.6 (Model of DAMAS). The *model* of a DAMAS is defined as the model of a DMAS except for the component:

- $T : S \times Act \to S \times \mathbb{N}_+$ is the partial *transition function*, such that $T(s,a) = (s', \max_{i \in Agent(a)} \delta_i)$ iff $T_i(s_i, a) = (s'_i, \delta_i)$ for all $i \in Agent(a)$, and $s'_i = s_i$ for all $i \in \mathcal{A} \setminus Agent(a)$.

These changes to local and global transitions are incorporated in the concrete DAMAS model, otherwise identical to that of a DMAS.

Definition 6.7 (ADTS). The *concrete model* of a DAMAS model is defined as the DTS of a DMAS model except for the component:

- $\mathcal{E}\colon \mathcal{TS} \times Act \to \mathcal{TS}$ is a (partial) transition function such that $\mathcal{E}((s,d),a) = (s', d+\delta)$ iff $T(s,a) = (s', \delta)$, for $s, s' \in S$, $a \in Act$, $d \in \mathbb{N}$, and $\delta \in \mathbb{N}_+$.

6.4 Untimed AMAS

Untimed AMAS can be defined as Timed AMAS with no clocks, see below. Note that the definition is essentially equivalent to the concept of an *interleaved interpreted system* in [30].

Definition 6.8 (Untimed AMAS). An *untimed asynchronous multi-agent system*, simply AMAS, is a CAMAS with every $\mathcal{X}_i = \emptyset$. The model and concrete model of an AMAS are equal and defined as in Definition 6.2 without clocks.

6.5 Model Checking in AMAS

The semantics of **STCTL** (**TATL**) is the same as in the synchronous case except for each α to be replaced by a in the paths. In principle, the model checking procedures and complexity results for **STCTL** and **TATL** and their untimed variants given in Sec. 4 also apply to asynchronous models. Note, however, that complexity is specified wrt. the model size, which in AMAS is significantly larger due to asynchronous interleaving of agents' actions. On the other hand, the associated blow-up of state- and transition-space can be alleviated via techniques such as partial order reductions [23].

7 EXPERIMENTS

In this section, we aim to show that model checking $\mathbf{STCTL}_{\mathrm{ir}}$ is practically feasible. To that end, we implemented the CAMAS from Ex. 6.4 in the IMITATOR model checker [8], and conducted a set of initial experiments using formulas $\varphi_{|A|} = \langle\!\langle A \rangle\!\rangle \exists \mathsf{F}_{[0,8]} \mathsf{v}_1$, which specify that voter(s)[2] in $A = \{voter_1, \ldots, voter_{|A|}\}$ have a strategy to vote for the first candidate within 8 time units, i.e., reach a state labelled with the local proposition v_1 before 8.

IMITATOR allows for **TCTL** model checking and uses an asynchronous semantics on networks of timed automata, which fits our purposes. Furthermore, as a state-of-the-art tool for Parametric Timed Automata, it enables us to encode agents' strategies as parameters: for each coalition agent, we add a parameter for each transition and a guard such that the parameter corresponding to the transition is 1 while those corresponding to the other transitions exiting the same location are 0. Note that this is not necessary when a single transition exits a location as there is no choice and thus no influence on the strategy.

[2]Coalition specified explicitly here for clarity; all voters in the CAMAS are symmetrical.

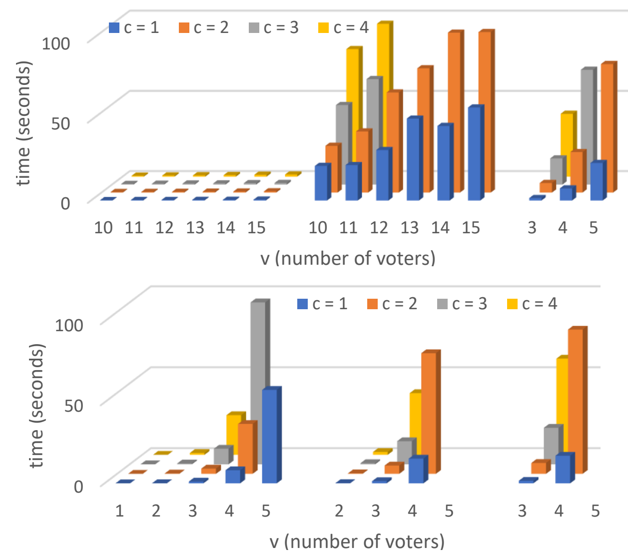

Figure 3: Top: model checking φ_1 (left), φ_2 (middle), and φ_3 (right) with v voters and c candidates. Bottom: also synthesising all strategies. Missing bars indicate timeout (> 120s).

Our model is scaled with the number of voters v and the number of candidates c, and we verify formulas φ_1, φ_2 and φ_3, i.e., with one to three agents (voters) in the coalition A (Fig. 3, top). The expected result is obtained: the voter(s) have a strategy to enforce $\exists \mathsf{F}_{[0,8]} \mathsf{v}_1$, which consists in choosing either the mail or the internet modality, and then voting for candidate 1.

While this already demonstrates the feasibility of $\mathbf{STCTL}_{\mathrm{ir}}$ model checking, the use of IMITATOR additionally provides (for free) the synthesis of *all* strategies (Fig. 3, bottom). However, this quickly faces a blowup in computation time. On the other hand, a single strategy of one agent in the formula φ_1 can be obtained within the same timeout (120s) for significantly larger models, with as many as 180 voters and 2 candidates, or 200 voters and 1 candidate. The code and binaries required to replicate the experiments are accessible at https://depot.lipn.univ-paris13.fr/mosart/publications/stctl.

8 CONCLUSIONS AND FUTURE WORK

This paper shows that **STCTL**, being a syntactic extension of **TATL**, but interpreted over timed models with continuous semantics, in both synchronous and asynchronous settings, is of theoretical and practical interest in model checking with ir- and Ir-strategies. Our plans for future research include: investigating also counting and timed strategies, a finer tuning of a model checking practical approach to easily capture all **STCTL** properties, and extending **STCTL** to **STCTL***. Moreover, since we have observed that synthesis of all strategies is too time consuming, but feasible even with the existing tool, we plan to implement a smarter, dedicated algorithm.

ACKNOWLEDGMENTS

This work was funded by the CNRS IEA project MoSART, and by NCBR Poland & FNR Luxembourg under the PolLux/FNR-CORE projects STV (POLLUX-VII/1/2019 & C18/IS/12685695/IS/STV/Ryan) and SpaceVote (POLLUX-XI/14/SpaceVote/2023).

REFERENCES

[1] T. Ågotnes, V. Goranko, and W. Jamroga. 2007. Alternating-time Temporal Logics with Irrevocable Strategies. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge (TARK '07)*. ACM, New York, NY, USA, 15–24.

[2] T. Ågotnes, V. Goranko, W. Jamroga, and M. Wooldridge. 2015. Knowledge and Ability. In *Handbook of Epistemic Logic*. College Publications, Rickmansworth, UK, 543–589.

[3] R. Alur, C. Courcoubetis, and D. L. Dill. 1993. Model-Checking in Dense Real-time. *Inf. Comput.* 104, 1 (1993), 2–34.

[4] R. Alur and D. L. Dill. 1990. Automata For Modeling Real-Time Systems. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming, ICALP90 (Lecture Notes in Computer Science, Vol. 443)*. Springer, Berlin, Germany, 322–335.

[5] R. Alur, T.A. Henzinger, O. Kupferman, and M.Y. Vardi. 1998. Alternating Refinement Relations. In *Proceedings of the 9th International Conference on Concurrency Theory, CONCUR '98 (Lecture Notes in Computer Science, Vol. 1466)*. Springer, Berlin, Germany, 163–178.

[6] R. Alur, T. A. Henzinger, and O. Kupferman. 1997. Alternating-Time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*. IEEE Computer Society, Palo Alto, CA, USA, 100–109.

[7] R. Alur, T. A. Henzinger, and O. Kupferman. 2002. Alternating-Time Temporal Logic. *J. ACM* 49 (2002), 672–713.

[8] É. André, L. Fribourg, U. Kühne, and R. Soulat. 2012. IMITATOR 2.5: A Tool for Analyzing Robustness in Scheduling Problems. In *FM 2012 (Lecture Notes in Computer Science, Vol. 7436)*. Springer, Berlin, Germany, 33–36.

[9] Jaime Arias, Wojciech Jamroga, Wojciech Penczek, Laure Petrucci, and Teofil Sidoruk. 2023. Strategic (Timed) Computation Tree Logic. *CoRR* abs/2302.13405 (2023), 1–10.

[10] P. Bouyer. 2009. Model-checking Timed Temporal Logics. *Electronic Notes in Theoretical Computer Science* 231 (2009), 323–341.

[11] N. Bulling, V. Goranko, and W. Jamroga. 2015. Logics for Reasoning About Strategic Abilities in Multi-Player Games. In *Models of Strategic Reasoning. Logics, Games, and Communities*. Lecture Notes in Computer Science, Vol. 8972. Springer, Berlin, Germany, 93–136.

[12] C. Dima and F.L. Tiplea. 2011. Model-checking ATL under Imperfect Information and Perfect Recall Semantics is Undecidable. *CoRR* abs/1102.4225 (2011), 1–17.

[13] M. Faella, S. La Torre, and A. Murano. 2014. Automata-theoretic Decision of Timed Games. *Theoretical Computer Science* 515 (2014), 46–63.

[14] M. Faella, S. La Torre, and A. Murano. 2002. Dense Real-Time Games. In *Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS 2002)*. IEEE Computer Society, Palo Alto, CA, USA, 167–176.

[15] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. 1995. *Reasoning about Knowledge*. MIT Press, Cambridge, MA, USA.

[16] V. Goranko and W. Jamroga. 2004. Comparing Semantics of Logics for Multi-agent Systems. *Synthese* 139, 2 (2004), 241–280.

[17] X. Huang and R. van der Meyden. 2014. Symbolic Model Checking Epistemic Strategy Logic. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI14)*. AAAI Press, Palo Alto, CA, USA, 1426–1432.

[18] W. Jamroga. 2008. Knowledge and Strategic Ability for Model Checking: A Refined Approach. In *Proceedings of the 6th German Conference on Multiagent Systems Technologies, MATES 2008 (Lecture Notes in Computer Science, Vol. 5244)*. Springer, Berlin, Germany, 99–110.

[19] W. Jamroga. 2015. *Logical Methods for Specification and Verification of Multi-Agent Systems*. ICS PAS Publishing House, Warsaw, Poland.

[20] W. Jamroga and J. Dix. 2006. Model Checking ATL$_{ir}$ is Indeed Δ_2^P-complete. In *Proceedings of the 4th European Workshop on Multi-Agent Systems, EUMAS '06 (CEUR Workshop Proceedings, Vol. 223)*. CEUR-WS.org, Aachen, Germany, 13–24.

[21] W. Jamroga, B. Konikowska, and W. Penczek. 2016. Multi-Valued Verification of Strategic Ability. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*. ACM, New York, NY, USA, 1180–1189.

[22] W. Jamroga, W. Penczek, P. Dembinski, and A. W. Mazurkiewicz. 2018. Towards Partial Order Reductions for Strategic Ability. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*. IFAAMAS, Richland, SC, USA, 156–165.

[23] W. Jamroga, W. Penczek, T. Sidoruk, P. Dembinski, and A. W. Mazurkiewicz. 2020. Towards Partial Order Reductions for Strategic Ability. *JAIR* 68 (2020), 817–850.

[24] M. Kacprzak and W. Penczek. 2004. Unbounded Model Checking for Alternating-Time Temporal Logic. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*. IEEE Computer Society, Palo Alto, CA, USA, 646–653.

[25] M. Kacprzak and W. Penczek. 2005. Fully Symbolic Unbounded Model Checking for Alternating-time Temporal Logic. *Autonomous Agents and Multi-Agent Systems* 11, 1 (2005), 69–89.

[26] M. Knapik, É. André, L. Petrucci, W. Jamroga, and W. Penczek. 2019. Timed ATL: Forget Memory, Just Count. *Journal of Artificial Intelligence Research* 66 (2019), 197–223.

[27] O. Kupferman, M. Y. Vardi, and P. Wolper. 2001. Module Checking. *Information and Computation* 164, 2 (2001), 322–344.

[28] F. Laroussinie, N. Markey, and G. Oreiby. 2006. Model-Checking Timed ATL for Durational Concurrent Game Structures. In *Proceedings of the 4th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2006 (Lecture Notes in Computer Science, Vol. 4202)*. Springer, Berlin, Germany, 245–259.

[29] A. Lomuscio, W. Penczek, and H. Qu. 2010. Partial Order Reductions for Model Checking Temporal Epistemic Logics Over Interleaved Multi-agent Systems. In *AAMAS 2010*, Vol. 1-3. IFAAMAS, Richland, SC, USA, 659–666.

[30] A. Lomuscio, W. Penczek, and H. Qu. 2010. Partial Order Reductions for Model Checking Temporal-epistemic Logics over Interleaved Multi-agent Systems. *Fundamenta Informaticae* 101, 1-2 (2010), 71–90.

[31] A. Lomuscio, H. Qu, and F. Raimondi. 2015. MCMAS: An Open-Source Model Checker for the Verification of Multi-Agent Systems. *International Journal on Software Tools for Technology Transfer* 24 (2015), 84–90.

[32] A. Lomuscio and M. Ryan. 1997. On the Relation Between Interpreted Systems and Kripke Models. In *Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications*. Lecture Notes in Artificial Intelligence, Vol. 1441. Springer, Berlin, Germany, 46–59.

[33] P. Y. Schobbens. 2004. Alternating-time Logic with Imperfect Recall. In *1st International Workshop on Logic and Communication in Multi-Agent Systems (LCMAS 2003) (Electronic Notes in Theoretical Computer Science, Vol. 85)*. Elsevier, Amsterdam, 1–12. Issue 2.

[34] D. Springall, T. Finkenauer, Z. Durumeric, J. Kitcat, H. Hursti, M. MacAlpine, and J. A. Halderman. 2014. Security Analysis of the Estonian Internet Voting System. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14)*. ACM, New York, NY, USA, 703–715.

[35] Y. Wang and F. Dechesne. 2009. On expressive power and class invariance. *CoRR* abs/0905.4332 (2009), 1–18.