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. SCTL is interpreted over networks of extended Timed Automata (TA) that either make synchronous moves or synchronize via joint actions. We consider several semantics regarding information: imperfect (i) and perfect (l), 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 SCTL is of the same complexity as for ATL, the model checking problem for STCTL is of the same complexity as for TCTL, while for STCTL it is undecidable as for ATL. The above results suggest to use SCTL and STCTL in practical applications. Therefore, we use the tool IMITATOR to support model checking of STCTL.

KEYWORDS

timed automata; model checking; timed logics; strategy logics

1 INTRODUCTION

Alternating-time temporal logics ATL and ATL [6, 7] extend the temporal logic 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 SCTL is of the same complexity as for ATL, the model checking problem for STCTL is of the same complexity as for TCTL, while for STCTL it is undecidable as for ATL. These results suggest to use SCTL and STCTL in practical applications. Therefore, we demonstrate the feasibility of STCTL 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 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 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 SCTL and STCTL considered here. They are shown to be EXPTIME-complete, demonstrating that model checking of SCTL over continuous time models retains the same complexity as over untimed ones (cf. Table 1). Analogously to ATL, the logic TATL over continuous time semantics, call it TATL-∗, would be a natural counterpart to the discrete-time 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 SCTL, 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.
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 \( \mathbb{R}_{\geq 0} \).

2.1 Syntax of STCTL and its Fragments

We begin by introducing the logical formulas of interest. Assume a synchronous continuous-time multi-agent system (CMAS) with continuous and discrete time, as well as untimed.

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 \( X = \{ x_1, \ldots, x_n \} \) (with a fixed ordering assumed for simplicity). A clock valuation on \( X \) is a \( \pi_X \)-tuple \( v \). We denote:
- by \( \pi(x_i) \) or \( \pi(i) \), the value of clock \( x_i \) in \( v \);
- by \( + \delta \), where \( \delta \in \mathbb{R}_{\geq 0} \) or \( v' \) s.t. \( v'(x) = (v + \delta) \) for all \( x \in X \);
- by \( \forall [X = 0] \), where \( X \subseteq X \), \( v' \) s.t. \( v'(x) = 0 \) for all \( x \in X \), and \( v'(x) = v(x) \) for all \( x \in X \setminus X \).

The clock constraints over \( X \) are defined by the following grammar:

- \( \mathfrak{c} \) is defined inductively as:
  - \( v \models \text{true} \) \iff \( v(x) \sim c \),
  - \( v \models (x_i - c) \iff v(x_i) \sim c \), and
  - \( v \models (\mathfrak{c} \land \mathfrak{c}' ) \iff v \models \mathfrak{c} \land v \models \mathfrak{c}' \).

The set of all valuations satisfying \( \mathfrak{c} \) is denoted by \( [\mathfrak{c}] \).

Definition 2.1 (CMAS). A continuous-time multi-agent system (CMS) consists of \( n \) agents \( \mathcal{A} = \{ 1, \ldots, n \} \), each associated with a 9-tuple \( \mathcal{A}_i = (L_i, u_i, Act_i, P_i, X_i, T_i, \mathcal{P}_i, V_i, I_i) \) including:
- a finite non-empty set of local states \( L_i \)
- an initial local state \( i \in L_i \)
- a finite non-empty set of local actions \( \text{Act}_i = \{ a_1, a_2, \ldots, a_m \} \)
- a local protocol \( P_i : L_i \rightarrow 2^{\text{Act}_i} \setminus \{ \emptyset \} \)
- a set of clocks \( X_i \)
- an invariant \( I_i : L_i \rightarrow C_{X_i} \) specifying a condition for the CMS to stay in a given local state
- a (partial) local transition function \( T_i : L_i \times \text{Act}_X \times C_X \times 2^{X_i} \rightarrow L_i \), where \( \text{Act}_X = \bigcap_{i \in \mathcal{A}} \text{Act}_i \) is the set of joint (global) actions of all agents, is s.t. \( T_i(l_i, a, \mathcal{C}_X, X_i) = l' \) for some \( l' \in L_i \) if \( a' \in P_i(l_i), \mathcal{C}_X \in C_{X_i} \), and \( X \subseteq X_i \)
- a finite non-empty set of local propositions \( P_{X_i} = \{ p_1, \ldots, p_r \} \)
- a local valuation function \( V_i : L_i \rightarrow 2^{P_{X_i}} \).

For a local transition \( t : = l \xrightarrow{a, \mathcal{C}_X, X_i} l' \) in a CMS, \( l \) and \( l' \) are the source and target states, \( a \) is the executed action, clock condition \( \mathcal{C}_X \) is called a guard, and \( X \) is the set of clocks to be reset.

Definition 2.2 (Model of CMS). The model of CMS is a 7-tuple \( M = (\mathcal{A}, S, t, X, I, T, V) \), where:
- \( \mathcal{A} = \{ 1, \ldots, n \} \) is the set of agents
- \( S = \prod_{i=1}^{n} L_i \) is the set of global states
- \( t : S \times \text{Act}_X \times C_X \times 2^{X_i} \rightarrow S \) s.t. \( T(s, a, \mathcal{C}_X, X_i) = s' \) iff \( T_i(s', a, \mathcal{C}_X, X_i) = s'' \) for each \( 1 \leq i \leq n \)
- a valuation function \( V : S \rightarrow 2^{P_{X_i}} \), where \( P_{X_i} = \bigcup_{i=1}^{n} P_{X_i} \).

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 CMS model \( M = (\mathcal{A}, S, t, X, I, T, V) \) is given by its Continuous Transition System (CTS) \( (\mathcal{A}, \mathcal{S}, q_0, t, V_0) \), where:
This definition enforces all synchronising actions to have the same duration in their respective local components. Another possibility could be to have individual durations $\delta_i$ 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) $(\mathcal{A}, T, S, q_0, E, V_d)$, where:

- $\mathcal{A} = \{1, \ldots, n\}$ is the set of agents;
- $T = S \times \mathbb{N}$ is a set of timed states;
- $q_0 = (1, 0) \in T S$ is the initial timed state;
- $E: T S \times \mathcal{JAct} \rightarrow T S$ is a (partial) transition function such that $E((s,d),a) = (s',d+\delta)$ iff $T(s,a) = (s',\delta)$, for $s, s' \in S, a \in \mathcal{JAct}, d \in \mathbb{N}$, 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 CMS with every $X_i = \emptyset$. 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 $\sigma_i: L_i \rightarrow \mathcal{Act}$ 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 $\sigma_i: S \rightarrow \mathcal{Act}$ such that $\sigma_i(s) \in P_i(s')$ for each $s \in S$.
- A perfect recall, imperfect information (ir) strategy for $i \in \mathcal{A}$ is a function $\sigma_i: L_i^r \rightarrow \mathcal{Act}$ s.t. $\sigma_i(h) \in P_i(l)$ for each $l \in L_i$.
- A perfect recall, perfect information (ir) strategy for $i \in \mathcal{A}$ is a function $\sigma_i: S^r \rightarrow \mathcal{Act}$ s.t. $\sigma_i(h) \in P_i(l)$ for each $s \in S$. By $H \in S^r$ (resp. $h \in L_i^r$), 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 generalised to an agent coalition $A \subseteq \mathcal{A}$, whose joint strategy $\sigma_A$ 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 \((\mathbb{A}, C, S, q_0, \rightarrow_c, V_c)\) be a CTS. Its execution from \(q_0 = (s_0, o_0)\) is \(\pi = q_0, \delta_0, q_0, o_0, q_1, \delta_1, q_1, o_1, \ldots\), where \(q_k = (s_{2k}, v_{2k})\), \(q_{k+1} = (s_{2k+1}, v_{2k+1})\), such that \(\delta_k \in \mathbb{R}_{\geq 0}\), \(\sigma_k \in JAct\), \(q_k \rightarrow_c \delta_k \rightarrow_c q_{k+1}\), for each \(k \geq 0\).

An execution of a DTS \((\mathbb{A}, T, S, q_0, V_D)\) from \(q_0 = \pi = q_0, \delta_0, q_0, \delta_0, q_1, \delta_1, q_{1}, \ldots\), where \(q_k = (s_{2k}, v_{2k})\), \(\delta_k \in \mathbb{R}_{\geq 0}\), \(\sigma_k \in JAct\), \(q_k \rightarrow_c \delta_k \rightarrow_c q_{k+1}\), for each \(k \geq 0\).

An execution of a (concrete) model of an untimed MAS from \(s_0\) is \(\pi = s_0, a_0, s_1, a_1, \ldots\), s.t. \(\sigma_k \in JAct\), \(s_k \rightarrow_c \delta_k \rightarrow_c s_{k+1}\), 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 \(\sigma_A\) represents executions where the agents in \(A\) adopt \(\sigma_A\), 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 \(A \subseteq \mathbb{A}, Y \subseteq \{ir, ir, iR, iR\}, M^C = \) (resp. \(M^D\)) be the model of a CMAS (resp. a MAS, an untimed MAS), and let \(\pi^C = q_0, \delta_0, q_0, \delta_0, a_0, \ldots\), (resp. \(\pi^D = q_0, \delta_0, q_0, \delta_0, \ldots\)) be an execution of the corresponding concrete model.

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

\[
\begin{align*}
(Y = \text{ir}) & : a_i^m = \sigma_i(s_m), & Z = C, \\
(Y = \text{ir}) & : a_i^m = \sigma_i(s_m), & Z = D, \\
(Y = \text{ir}) & : a_i^m = \sigma_i(s_m), & Z = U.
\end{align*}
\]

By \(L^M_S\) we denote the logical system being considered, where \(L\) is the syntactic variant (see Section 2.1), \(S\) is the class of strategies (cf. Section 3.1), and \(M \in \{C, D, U\}\) 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 \(\langle A \rangle p\), it starts with a candidate set of states (chosen appropriately depending on \(p\)) 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 ATL_{ir} and ATL_{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: \(\text{AL}^M_{ir}\) for ATL_{ir} and \#P-hard for simple instances of ATL_{ir} [19], and undecidable for ATL_{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 TATL_{ir} and TATL_{p} coincide [26]. Model checking TATL_{ir} and TATL_{p} 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 TATL_{ir} we can at least establish upper and lower bounds of PSPACE (since TATL \subset SCTL) and \Delta^P_2 (since ATL \subset TATL), respectively. Furthermore, TATL_{ir} model checking is undecidable since it is already the case for 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 SCTL_{ir}.

\begin{algorithm}
\textbf{Algorithm 1: mcheckSCTL}_{ir}(M, s, \varphi)
\begin{itemize}
  \item Guess a strategy \sigma_A.
  \item Prune \tilde{M} by removing transitions not consistent with \sigma_A.
  \item If \langle\langle A \rangle\rangle \varphi is an SCTL formula, run TCTL model checking on \gamma, else run CTL model checking on \gamma.
\end{itemize}
\end{algorithm}

\textbf{Theorem 4.1.} Model checking SCTL_{ir} is \Delta^P_2-complete.

Proof. Let \varphi = \langle\langle A \rangle\rangle \gamma be an SCTL_{ir} formula without nested strategic modalities. Consider the procedure in Alg. 1. It runs in NP for input \varphi, since \sigma_A can be guessed in non-deterministic polynomial time, while pruning transitions and model checking of CTL formula \gamma requires deterministic polynomial time. Note that for an arbitrary 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 PNP = \Delta^P_2.

The lower bound follows from the fact that SCTL_{ir} subsumes ATL_{ir} whose model checking complexity is \Delta^P_2 [19, Th. 34].

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

\textbf{Theorem 4.2.} Model checking SCTL_{ir} is \Delta^P_2-complete.

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

The lower bound is obtained by a reduction from the model checking problem for ATL_{ir}, which is \Delta^P_2-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 \varphi an 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, ..., a_n), we create a new state (q, a_1, ..., a_n), and direct the transition to that state. Moreover, we label the new state by fresh atomic propositions exec_{1,a_1}, ..., 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 \varphi is true in all the states s' such that (s')^i = s'). The uniformity of agent i's play can now be captured by the following CTLK formula: unif_i = \bigvee\{\forall a \in A_i K_i (y \land exec_{i,a})\}. We reconstruct formula \varphi by replacing every occurrence of (\langle\langle A \rangle\rangle \varphi) with (\langle\langle A \rangle\rangle (\forall y \land \bigwedge_{i \in A} \text{unif}_i)) . We denote the resulting SCTLK formula by \psi'. It is easy to see that M, s \models ATL_{ir} \varphi iff M', s \models SCTLK \psi'.

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 \psi' by appropriate strategic subformulas for e_i, see [18, Section 4.2] for the details. The resulting translations of M' and \psi' are denoted by M'' and \psi''. Analogously to [18, Theorem 1], we get that M'', s \models SCTLK \psi'' iff M', s \models SCTLK \psi' (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 SCTL_{ir}.

\textbf{Theorem 4.3.} Model checking 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 SCTL_{ir}, we obtain EXPTIME-completeness via a reduction of the module checking problem [27] for CTL as follows.

\begin{algorithm}
\textbf{Algorithm 2: mcheckSCTL}_{ir}(M, s, \langle\langle A \rangle\rangle \gamma)
\begin{itemize}
  \item Split agents into two groups: coalition A and opponents O.
  \item Merge A and O into single agents by creating auxiliary action labels for tuples of actions belonging to agents in A and O.
  \item Run CTL module checking on \gamma.
\end{itemize}
\end{algorithm}

\textbf{Theorem 4.4.} Model checking 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 \varphi 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 SCTL_{ir} formula.
We can see in Section 4 that using the broader syntax of STCTL whose model checking is PSPACE-complete, i.e. it remains unchanged from TCTL.

**Theorem 4.5.** Model checking STCTLIR is PSPACE-complete.

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

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

**Theorem 4.6.** Model checking STCTLIR is PSPACE-complete.

**Proof.** Both bounds follow exactly as in Th. 4.5 for STCTLIR. □

**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 STCTLIR is undecidable.

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

**Theorem 4.8.** Model checking STCTLIR is undecidable.

**Proof.** (Sketch) Undecidability follows from the fact that TCTL games [14] can be seen as a special case of STCTLIR 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 \( M \). \( L_1 \) is at least as expressive as \( L_2 \) (written \( L_2 \preceq_{\text{expr}} L_1 \)) if, for every formula \( \varphi_2 \) of \( L_2 \), there exists a formula \( \varphi_1 \) of \( L_1 \) such that \( \varphi_1 \) and \( \varphi_2 \) are satisfied in the same models from \( M \).

Moreover, \( L_1 \) is at least as distinguishing as \( L_2 \) (written \( L_2 \preceq_{\text{dist}} L_1 \)) if every pair of models \( M, M' \in M \) 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 \( L_2 \preceq_{\text{dist}} L_1 \) implies \( L_2 \preceq_{\text{expr}} L_1 \). By transposition, we also have that \( L_2 \not\preceq_{\text{dist}} L_1 \). The following is straightforward.

**Proposition 5.2.** For any strategy type \( S \) and model type \( M \), we have that \( \text{TATL}^M_S \preceq_{\text{dist}} \text{STCTL}^M_S \) (and thus \( \text{TATL}^M_S \preceq_{\text{expr}} \text{STCTL}^M_S \)).

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

**Proposition 5.3.** For any strategy type \( S \) and model type \( M \), we have that \( \text{STCTL}^M_S \preceq_{\text{dist}} \text{TATL}^M_S \) (and thus \( \text{STCTL}^M_S \preceq_{\text{expr}} \text{TATL}^M_S \)).

**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 $(M', q_0'q_0')$ are in alternating bisimulation [5], and thus they satisfy exactly the same formulas of ATL. By the above argument, they must satisfy the same formulas of TATL. Thus, they satisfy exactly the same formulas of TATL for all the strategy types $S$ and model types $M \in \{C, U\}$ and all the strategy types $S$ considered in this paper. On the other hand, we have that the STCTL (and hence also STCTL) formula $\varphi \equiv \langle\langle \exists F \left[ q_{\infty} \right] p \land \exists G \left[ q_{\infty} \right] \neg p \rangle \rangle$ holds in $(M, q_0q_0)$ but not in $(M', q_0'q_0')$ for all the strategy types $S$ and model types $M \in \{C, U\}$.

For $M = D$, 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 alternation-bisimilar with $M, M'$ [1]. In consequence, they satisfy the same formulas of TATL, for all strategy types $S$. On the other hand, the above STCTL and STCTL formula $\varphi$ holds in $(M, q_0q_0)$ but not in $(M', q_0'q_0')$ for $M = D$ and all $S$.

The following is a straightforward corollary.

**Theorem 5.4.** For any strategy type $S$ and model type $M$, STCTL$_S^M$ has strictly larger expressive and distinguishing power than TATL$_S^M$.

### 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)** 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 CAMAS except for the following component:

- a (partial) local transition function $T_i : L_i \times \text{Act}_i \times \mathcal{C}_X_i \times \mathcal{X}_i \rightarrow L_i$ such that $T_i(t_i, a, c, X_i) = t'_i$ for some $t'_i \in L_i$ iff $a \in P_i(t_i), c \in \mathcal{C}_X_i$, 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].

#### 6.3 Discrete Time AMAS

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

- a (partial) local transition function $T_i : L_i \times \text{Act}_i \rightarrow L_i \times \mathbb{N}$, such that $T_i(t_i, a)$ is defined iff $a \in P_i(t_i)$;
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:

- \( T : S \times Act \rightarrow S \times N_+ \) 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 \( i \in A \setminus Agent(a) \).

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:

- \( E : TS \times Act \rightarrow TS \) is a (partial) transition function such that \( E((s, d), a) = (s', d + \delta) \) iff \( T(s, a) = (s', \delta) \), for \( s, s' \in S \), \( a \in Act \), \( d \in N \), and \( \delta \in N_+ \).

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 \( X_i = \emptyset \). 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 \( a \) 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 STCTL\(_{ir}\) 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 \( \varphi_{[A]} = (\langle A \rangle)\exists F_{[0,8]} V_1 \) which specify that voter(s)\(^2\) in \( A = \{ \text{voter}_1, \ldots, \text{voter}_{|A|} \} \) 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.

While this already demonstrates the feasibility of STCTL\(_{ir}\) 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 \( \varphi_3 \) 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\(_{ir}\). 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.
Session 1F: Knowledge Representation and Reasoning I

ACKNOWLEDGMENTS

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

REFERENCES


