TEMPORAL LOGICS FOR GAMES

François Laroussinie
LIAFA, Univ. Paris Diderot – Paris 7 & CNRS UMR 7089, IUF
Francois.Laroussinie@liafa.jussieu.fr

1 Introduction

Model-checking is now a well-established method in the area of formal verification. It has been initiated roughly 25 years ago with the seminal papers by Pnueli, Clarke and Sifakis [25, 13, 26], and since then many theoretical and practical results have been obtained. In this framework, the behavior of the system to be analyzed is described with some formal model, the property to be verified is stated with a specification language and the analysis is done automatically with a tool (a model checker).

One can use many different types of formalism to represent the behavior of a system. This choice depends on the nature and the features of the system we deal with: Which kind of data is manipulated by the system? Is it necessary to handle real-time constraints? Are there probabilities to consider? The same holds for the specification language: for example, there is a wide family of temporal logics allowing to express different kinds of properties. Of course there is a trade-off between expressiveness and efficiency: at the very end, one aims at using a model-checker to decide whether the specification is satisfied by the model and then the efficiency of the decision procedures is crucial.

An important aspect of complex systems is that they are usually composed by several components that interact together. One can consider them as a whole and verify that the resulting system satisfies some property: given a complete description of some protocol (with a sender, several receivers, a network,...) one can ask whether there is no deadlock (with CTL, this could be expressed by $AG \neg \text{deadlock}$ where $\text{deadlock}$ would be an atomic proposition characterizing the deadlocked states). We could also verify that every new message is followed by a reception with $AG (\text{new-message} \Rightarrow AF \text{reception})$.

Sometimes it is interesting to consider one component $C$ (or a subset of components) and to express properties over it within its environment (i.e. the other components): one can then analyze its ability to act upon the whole system in order to ensure some property. For example, it can be useful to know whether
C can always ensure that some request will be granted. By always, we mean *whatever the other components do*. Thus we need a formalism to model the interaction of components and a specification language that allows us to express this kind of property. And note that it cannot be easily stated with classical temporal logics because it requires an *existential* quantification over the actions of C (“C can ensure…”) and an *universal* quantification over the actions of the remaining part of the system (“whatever the other components do”). Such problems occur when considering the verification of *open systems* that have to behave correctly whatever their environments do.

Games are a natural and interesting model to represent this kind of problem. Consider for example the classical train crossing problem. Trains and cars can arrive at the crossing, and the gate has to be controlled in order to (1) avoid crash and (2) ensure liveness in the whole system (neither trains nor cars can be blocked forever). Such a problem can be seen as a game: one player deals with trains, another one drives cars and the last one, let’s say Alice, has to control (open or close) the gate. The question is then: Has Alice a winning strategy in this game? If so, the corresponding control problem has a solution.

Then we need special specification languages to deal with games: it is important to be able to state the existence of *strategies* for a given player. This has motivated the introduction of Alternating-time Temporal Logics (ATL, ATL*, . . . ) where explicit quantification over the strategies is possible [5]. The aim of this document is to give an overview of results about these temporal logics. We will mainly focus on semantic questions (Which kind of properties can be stated? How can we increase the expressive power?), and we will also consider complexity results for model checking problems.

**Plan of the paper.** In Section 2 we give the formal definition of Concurrent Game Structures. In Section 3 the temporal logic ATL and its variant ATL* are presented. Several questions about the expressive power and the complexity of these logics will be discussed. In Sections 5 and 6 we will present two extensions of ATL: the first one uses strategy contexts to express complex properties over strategies and the second one allows us to add real-time constraints in the specifications.

## 2 Concurrent Game Structures

Let AP be a set of atomic propositions. Concurrent Game Structures are a multi-player extension of Kripke structures:

**Definition:** [Concurrent Game Structure [5]]
A Concurrent Game Structure (CGS) is a 7-tuple \( \mathcal{S} = (Q, q_0, \ell, \text{Agt}, \mathcal{M}, \text{Mv}, \text{Edg}) \) where:

- \( Q \) is a finite set of control states (or locations) and \( q_0 \in Q \) is the initial location;
- \( \ell : Q \to \mathcal{P}(\text{AP}) \) is a labeling of locations with atomic propositions;
- \( \text{Agt} = \{A_1, \ldots, A_k\} \) is a finite set of agents (or players);
- \( \mathcal{M} \) is a finite alphabet of moves;
- \( \text{Mv} : Q \times \text{Agt} \to \mathcal{P}(\mathcal{M}) \) gives the set of possible moves for every player in every location;
- \( \text{Edg} : Q \times \mathcal{M}^k \to Q \) is the transition table of \( \mathcal{S} \): \( \text{Edg}(q, m_1, \ldots, m_k) \) is the new location of \( \mathcal{S} \) when \( A_i \) plays the move \( m_i \) for \( i = 1, \ldots, k \) from location \( q \).

The size \( |\mathcal{S}| \) of a CGS \( \mathcal{S} \) is defined as \( |Q| + |\text{Edg}| \), where \( |\text{Edg}| \) is the size of the transition table.

From the current location \( q \), any player \( A_i \) independently chooses a move \( m_i \in \text{Mv}(q, A_i) \) and the transition table \( \text{Edg} \) provides the new location \( q' = \text{Edg}(q, m_1, \ldots, m_k) \). We use \( \text{Next}(q, A_i, m_i) \) to denote the set of locations that are reachable from \( q \) when Player \( A_i \) chooses \( m_i \) (i.e. \( q' \in \text{Next}(q, A_i, m_i) \) iff \( \exists \tilde{m} \in \mathcal{M}^k \) with \( \tilde{m}(j) \in \text{Mv}(q, A_j) \) for any \( j, \tilde{m}(i) = m_i \) and \( q' = \text{Edg}(q, \tilde{m}) \)). And \( \text{Next}(q) \) is the set of locations reachable from \( q \) for some set of moves of the agents (i.e. \( q' \in \text{Next}(q) \) iff \( \exists \tilde{m} \in \mathcal{M}^k \) with \( \tilde{m}(i) \in \text{Mv}(q, A_i) \) for any \( i \), and \( q' = \text{Edg}(q, \tilde{m}) \)).

**Example:** Figure 1 presents a CGS corresponding to the game "Rock-paper-scissors". There are two players and each one has to choose R (rock), P (paper) or S (scissors). The alphabet of moves is \{R, P, S\}. The transitions of the CGS are labeled with the corresponding set of moves (\langle R,P \rangle means that the move of Player 1 (resp. Player 2) is R (resp. P).

Note that the transition table may be exponential in the number of agents. This has motivated the introduction of a succinct encoding of \( \text{Edg} \) by using Boolean functions \[18\]. In this model, namely the implicit CGSs, one can specify the possible transitions from \( q \) as a sequence of "If-Then-Else" tests whose conditions are Boolean combinations of propositions of the form "Player \( i \) chooses \( m_i \)". Formally the transition function from \( q \) is defined by a finite sequence \(((\varphi_0, q_0), \ldots, (\varphi_n, q_n))\) s.t.: \( q_i \in Q \) and \( \varphi_i \) is a Boolean combination of propositions "\( A_j \rightarrow m_i \)" (i.e. "Agent \( A_j \) plays \( m_i \)") and \( \varphi_n = \top \). Then we define \( \text{Edg}(q, m_1, \ldots, m_k) \) as \( q_j \).
with \( j = \min \{m_1 \ldots m_k \models \varphi_i \} \). An example of implicit CGS with three players is shown in Figure 2 with \( \operatorname{Mv}(q_0) \equiv (A_1 \rightarrow 1, q_0), (A_2 \rightarrow 2, q_1), (\top, q_2) \): the right part of the figure corresponds to the equivalent "explicit" CGS. We will see in Section 4 that this encoding changes the complexity of model checking.

**Alternating transition systems.** In [4], another game model is studied: the Alternating Transition Systems (ATS). In this setting, a move of a player consists in a subset of successor locations. \( \operatorname{Mv}(q, A) \) is then a set of subsets of \( Q \). When every player has chosen a move, the new location is the intersection of every move. By construction, this intersection has to be a singleton (this point makes it quite difficult to design an ATS in practice). Translations from ATSs to CGSs and back are possible (w.r.t. alternating bisimulation [6]) but may be expensive [21].

**Turn-based games.** Finally note also that the turn-based CGSs are an important subclass with interesting semantic and algorithmic properties. In a turn-based CGS, for every location \( q \), there is at most one player \( A_i \) who has several choices: \( |\operatorname{Mv}(q, A_i)| = 1 \) for any \( j \neq i \). Thus the set of locations is partitioned into \( k \) sets \( Q_1, \ldots, Q_k \): \( Q_i \) contains locations that "belong" to Player \( A_i \) (who decides the successor location).

\footnote{with the convention: \( m_1 \ldots m_k \models \text{"} A_j \rightarrow m \text{"} \) iff \( m_j = m \).}
Coalitions, executions and strategies. A coalition $A$ is a subset of agents. A move for $A$ is a move for every player in $A$. The previous definitions for $\text{Next}$ can easily be extended to deal with coalitions instead of single players. Given $m$ a move for $A$ and $m'$ a move for $\text{Agt}\setminus A$, we use $m \cdot m'$ to denote the corresponding move for $\text{Agt}$. Of course we have $\text{Next}(q, \text{Agt}, m \cdot m') = \{\text{Edg}(q, m \cdot m')\}$.

An execution is an infinite sequence $\rho = q_0 \rightarrow q_1 \rightarrow q_2 \ldots$ such that $q_{i+1} \in \text{Next}(q_i)$ for any $i$. We use $\rho[i]$ to denote the state $q_i$ and $\rho[0 \ldots i]$ (or $\rho_i$) for the prefix $q_0 \rightarrow q_1 \rightarrow \ldots \rightarrow q_i$.

A strategy for Player $A_i$ is a function $f_{A_i}$ that maps any finite prefix $q_0 \ldots q_j$ of some execution to a possible move for $A_i$, i.e. satisfying $f_{A_i}(q_0 \ldots q_j) \in Mv(q_j, A_i)$. We use $\text{Strat}(A_i)$ to denote the set of strategies for $A_i$. This notion can also be extended to coalitions.

A memoryless (or state-based) strategy $f_{A_i}$ only depends on the current state of the system (i.e. the last state of the prefix): $f_{A_i}$ is then a mapping from $Q$ to $M$. More generally we could also consider $k$-bounded-memory strategies for any integer $k$ [23, 29], this is a way to characterize the resources needed by a strategy.

Given a strategy $F_A$ for a coalition $A$, we use $\text{Out}(q, F_A)$ to denote the set of executions enforced by the strategy $F_A$: $q_0 \rightarrow q_1 \rightarrow q_2 \ldots \in \text{Out}(q, F_A)$ if $q_0 = q$ and for any $i$ we have $q_{i+1} \in \text{Next}(q_i, A, F_A(q_0 \ldots q_i))$. If $F_\emptyset$ is a strategy for the empty coalition, $\text{Out}(q, F_\emptyset)$ is the set of all executions starting from $q$ (denoted by $\text{Exec}(q)$).

3 Temporal logics for games

Temporal logic (TL) is an interesting framework to express properties over reactive systems [25]: the temporal modalities offer a natural way to deal with the ordering of events along executions. There are many different TLs differing from the nature of the underlying models, the allowed temporal modalities, etc. First we can mention two main families: the linear-time temporal logics and the branching-time temporal logics. In the first case, a system is viewed as a set of executions and formulas are interpreted over these runs. This is the case of the well-known LTL. One can specify that "any problem is followed by an alarm" by using the following formula: "$\varphi \defeq G(\text{problem} \Rightarrow F\text{alarm})"$. Modality $G$ (resp. $F$) means "Always along the run" (resp. "Eventually along the run"). When we say that an LTL formula $\varphi$ holds for a Kripke structure $S$, we usually mean that every run of $S$ satisfies $\varphi$. There is an implicit universal quantification over the executions of $S$.

Branching-time temporal logics (such as CTL or CTL*) are interpreted over states of Kripke structures. Every state may have several successors. In addition
to classical temporal modalities, we can use the existential (E) or universal (A) quantification over the runs starting from the current state. The previous property can be stated in CTL with the following formula: "AG (problem ⇒ AF alarm)". The formula AG (problem ⇒ EF alarm) is very different: it specifies that from any problem state, it is possible to find a run leading to alarm. This property cannot be expressed in linear-time temporal logic (see [15] for a detailed overview of temporal logics). Adding path quantifications allows us to express complex properties on the behavior of reactive systems. In this framework CTL* is very powerful and contains both CTL and LTL.

When considering games, it is natural to deal with the strategies for agents to enforce temporal properties. For example, in the game Rock-paper-scissors, one could ask whether there exists a strategy for Player 1 to reach the state 1-Win. Such a query is in fact an existential quantification over a subset of runs (generated by a strategy for Player 1) followed by an universal quantification over the runs of this subset. It cannot be expressed by using the existential or universal quantification over paths and this has motivated the introduction of modalities ⟨⟨A⟩⟩ in [5]: ⟨⟨A⟩⟩ Φ means that "there exists a strategy for A such that Φ holds for any run". This provides a new family of temporal logics: ATL, ATL*,... that correspond to branching-time TLs where E and A have been replaced by modalities ⟨⟨A⟩⟩ for A ⊆ Agt.

Formally ATL is defined as follows:

Definition: [Syntax of ATL]

\[ ATL \ni \phi_s, \psi_s \ ::= \ P \mid \neg \phi_s \mid \phi_s \lor \psi_s \mid \langle\langle A \rangle\rangle \phi_p \]
\[ \phi_p \ ::= \ X \phi_s \mid \phi_s U \psi_s \mid \phi_s R \psi_s \]

with \( P \in AP \) and \( A \subseteq Agt \).

As usual we can easily define \( \Rightarrow, \land, \top, \bot \ldots \) We will use \( F \varphi \) (resp. \( G \varphi \)) as a shorthand for \( \top U \varphi \) (resp. \( \neg F \neg \varphi \)). ATL formulas are interpreted over states of CGS (the case of Boolean operators is omitted):

\[ q \models_S \langle\langle A \rangle\rangle \phi_p \iff \exists F_A \in Strat(A). \forall \rho \in Out(q, F_A) \text{ we have } \rho \models_S \phi_p, \]
\[ \rho \models_S X \phi_s \iff \rho[1] \models_S \phi_s, \]
\[ \rho \models_S \phi_s U \psi_s \iff \exists i. \rho[i] \models_S \psi_s \text{ and } \forall 0 \leq j < i \text{ we have } \rho[j] \models_S \phi_s \]
\[ \rho \models_S \phi_s R \psi_s \iff \forall i : \left( \exists j < i. \rho(j) \models_S \phi_s \right) \lor \rho(i) \models_S \psi_s \]

ATL* is defined in the same manner as CTL* with \( \langle\langle A \rangle\rangle \) modalities: there is no restriction on the embedding of temporal modalities and any strategy quantifier
⟨⟨A⟩⟩ can be followed by a general path formula defined as:

\[ \phi_p, \psi_p ::= \phi_s | \phi_p \lor \psi_p | \neg \phi_p | X \phi_p | \phi_p U \psi_p \]

For example, ⟨⟨A⟩⟩ G F P is an ATL* formula.

Note that classical quantifications E and A can be easily expressed with modalities ⟨⟨A⟩⟩:

\[ E \Phi \equiv \langle \langle \text{Agt} \rangle \rangle \Phi \quad A \Phi \equiv \langle \langle \emptyset \rangle \rangle \Phi \]

Indeed the existential quantification corresponds to a case where all agents cooperate together in order to satisfy a property. Conversely if \( \Phi \) is true when nobody tries to make it to be true, it means that \( \Phi \) is always true. This implies that ATL contains CTL, and ATL* contains CTL* (note also that a Kripke structure can be seen as a one-player CGS).

**Example of properties.** Now we give examples of properties that can be expressed with ATL.

- ⟨⟨Controller⟩⟩ G (¬Pb): there exists a strategy for the controller to ensure that Pb is never true.
- ⟨⟨A⟩⟩ F (¬⟨⟨B⟩⟩ F P ∧ ¬⟨⟨C⟩⟩ F P): there exists a strategy for A to reach a state where neither B nor C can manage alone reach P.
- ⟨⟨A⟩⟩ F (¬⟨⟨B⟩⟩ F P ∧ ¬⟨⟨C⟩⟩ F P ∧ ⟨⟨B, C⟩⟩ F P): there exists a strategy for A to reach a state where neither B nor C can manage alone reach P, but where B and C can cooperate together to reach P.
- ⟨⟨Sender, Receiver⟩⟩ F (msg-ok): there exists a strategy for the sender and the receiver to reach a state where msg-ok is true.

Dealing with strategy quantifiers is not always easy. For example, consider two path formulas \( \Phi_1 \) and \( \Phi_2 \), it is clear that \( E(\Phi_1 \lor \Phi_2) \) is equivalent to \( E\Phi_1 \lor E\Phi_2 \) (and \( A(\Phi_1 \land \Phi_2) \equiv A\Phi_1 \land A\Phi_2 \)). But such rules are not true when considering strategy quantifiers:

\[ \langle \langle A \rangle \rangle (\Phi_1 \lor \Phi_1) \not\equiv \langle \langle A \rangle \rangle \Phi_1 \lor \langle \langle A \rangle \rangle \Phi_1 \]

CGSs are not determined for temporal properties \([5, 16]\): if a coalition A does not have a strategy to enforce \( \Phi \), it does not imply that Agt \( \setminus A \) has a strategy to ensure \( \neg \Phi \). Thus \( \neg \langle \langle A \rangle \rangle \varphi \) is not equivalent to \( \langle \langle \text{Agt} \setminus A \rangle \rangle \neg \varphi \). For example, in the
"Rock-paper-scissors" game, nobody has a strategy to win or to prevent a defeat and thus we have: $q_0 \not\Vdash \langle A_1 \rangle X 1\text{-Win}$ and $q_0 \not\Vdash \langle A_2 \rangle X \neg 1\text{-Win}$.

Finally note that memoryless strategies are sufficient to deal with ATL specifications \cite{5, 28} but this is not the case for ATL\*: for example, the formula $\langle A \rangle (F P_1 \land F P_2)$ – "there exists a strategy for $A$ to reach $P_1$ and to reach $P_2$" – may require to choose two different moves in the same location in order to reach first $P_1$ and then $P_2$.

In \cite{9} there is an extension of ATL\* with strategy quantifiers of the form $\langle A, k \rangle$ with $k \in \mathbb{N}$ in order to quantify over strategies using a memory of size $k$. These modalities increase the expressive power (and even the distinguishing power) of ATL\*.

**Other specification languages.** In addition to ATL and ATL\*, there are different specification languages in the literature. First \cite{2} we can mention the Alternating-time $\mu$-Calculus (AMC) \cite{5} based on the modality $\langle A \rangle X$ and fixed-point operators, its semantics is similar to that of propositional $\mu$-calculus.

Game Logic \cite{5} allows us to deal explicitly with the trees obtained by the choice of a given strategy for a coalition. In GL, there is an existential quantifier over the strategies of coalition $A$ ($\exists A$), and path quantifiers ($\exists$ and $\forall = \neg \exists \neg$) to deal with the paths in the underlying tree corresponding to the chosen strategy. The syntax is as follows (GL$\_t$ contains the tree formulas, and GL$\_p$ contains the path formulas):

**Definition:** [Syntax of GL \cite{5}]

- $\text{GL} \triangleright \phi_s, \psi_s ::= P | \neg \phi_s | \phi_s \lor \psi_s | \exists A. \phi_t$
- $\text{GL}_t \triangleright \phi_t, \psi_t ::= \phi_t | \neg \phi_t | \phi_t \lor \psi_t | \exists \phi_p$
- $\text{GL}_p \triangleright \phi_p, \psi_p ::= \phi_t | \neg \phi_p | \phi_p \lor \psi_p | X \phi_p | \phi_p U \psi_p$

with $A \subseteq \text{Agt}$ and $P \in \text{AP}$.

Semantics is natural: $\exists A. \phi_t$ holds for a location $q$ iff there is a strategy $F_A \in \text{Strat}(A)$ such that the tree $T_{F_A}$ produced by $F_A$ from $q$ (i.e. whose branches belong to $\text{Out}(q, F_A)$) satisfies $\phi_t$.

For example, $\exists A. (\exists G P_1 \land \exists G P_2)$ holds for $q$ iff there is a strategy $F_A$ for $A$ such that there exist a run where $P_1$ is always true and another one verifying $P_2$: both runs belong to $\text{Out}(q, F_A)$. This formula cannot be stated with ATL\* \cite{5}. We will see in Section \cite{5} an extension of ATL allowing to express such properties.

\footnote{We will consider other logics in Section \cite{5}}
4 Model checking

Given a CGS \( S \) and an ATL formula \( \Phi \), the model checking problem consists in deciding whether the initial location of \( S \) satisfies \( \Phi \) or not.

The decision procedure computes the subset \([ [ \psi ] ]\) of locations of \( S \) satisfying \( \psi \) for any subformula \( \psi \) in \( \Phi \). It proceeds in a bottom-up manner starting by atomic propositions and then dealing with outermost formulas. Boolean operators can be easily handled. The set \([ [ \langle A \rangle X \varphi ] ]\) corresponds to the controllable predecessors of \([ [ \varphi ] ]\) by \( A \), denoted \( \text{CPre}(A, [ [ \varphi ] ] ) \). Given a set of locations \( S \) and a coalition \( A \), we have:

\[
\text{CPre}(A, S) \overset{\text{def}}{=} \{ q \in Q \mid \exists m_A \in \text{Mv}(q, A) \text{ such that } \text{Next}(q, A, m_A) \subseteq S \} 
\]

From any location in \( \text{CPre}(A, S) \), \( A \) can enforce to reach a location in \( S \).

And the treatment of \( \langle A \rangle \_ U \_ \) and \( \langle A \rangle \_ R \_ \) is based on a standard fixed-point computation in which we use the controllable predecessors. Given \( \Psi \overset{\text{def}}{=} \langle A \rangle P_1 U P_2 \), the set \([ [ \Psi ] ]\) is defined as the least fixed-point of the function:

\[
f(Z) \overset{\text{def}}{=} [[P_2]] \cup ([[P_1]] \cap \text{CPre}(A, Z))
\]

(1)

The formula \( \Psi \) can also be expressed by using the following Alternating-time \( \mu \)-calculus formula:

\[
\mu Z. (P_2 \lor (P_1 \land \langle A \rangle X Z)).
\]

The difference from CTL model checking is then the use of \( \langle A \rangle X \) instead of \( \text{EX} \): we need to consider controllable predecessors instead of classical predecessors. Note also that if we consider turn-based games, it is possible to express \( \langle A \rangle P_1 U P_2 \) with the standard propositional \( \mu \)-calculus because for every state we can use either the existential modality \( \text{EX} \) or the universal modality \( \text{AX} \) depending on the "owner" (\( A \) or \( \bar{A} \)) of the state.

The complexity of ATL model checking depends on the complexity of the computation \( \text{CPre} \) (see [21] for these results for ATS, CGS and implicit CGS).

Finally we have:

**Theorem:** Model checking ATL formulas...

- is P-complete on CGSs [5];
- is \( \Delta^p_3 \)-complete on implicit CGSs [21];
- is \( \Delta^p_2 \)-complete on ATSs [21].

For ATL*, we have:

**Theorem:** ([5] [21]) Model checking ATL* is 2EXPTIME-complete on ATSs as well as on CGSs and implicit CGSs.
Satisfiability. For ATL (and ATL*) the satisfiability problem ("given $\Phi$, does there exist a CGS satisfying $\Phi$?") can be defined in several ways [30]:

(Pb 1.) Given $\Phi$ and a finite set of agents $\text{Agt}$, does there exist a multi-agent model over $\text{Agt}$ satisfying $\Phi$?

(Pb 2.) Given $\Phi$, does there exist a finite set of agents $\text{Agt}$ such that there is a multi-agent model over $\text{Agt}$ satisfying $\Phi$?

(Pb 3.) Given $\Phi$, does there exist a multi-agent model over $\text{Agt}_\Phi$ (i.e. the set of agents occurring in $\Phi$) satisfying $\Phi$?

These three variations are discussed in [30]. The importance of the agents is highlighted by the following example: consider $\neg\langle A \rangle X P \land \neg\langle A \rangle X P' \land \langle A \rangle X (P \lor P')$. This formula is not satisfiable in a model where $A$ is the unique player. But it is clearly satisfiable when considering two players. In [30] it is shown that for ATL specifications, we have: (Pb 1.) and (Pb 3.) are polynomially reducible to each other and (Pb 2.) can be reduced to (Pb 1.) by considering one extra player in addition to $\text{Agt}_\Phi$. These remarks could also be extended to ATL*. Finally we have:

Theorem:

- Satisfiability problems for ATL are EXPTIME-complete [30].
- Satisfiability problems for ATL* are 2EXPTIME-complete [27].

Finally note that in [30] the problem is considered for Alternating Transition Systems (and not for CGSs), but as there exist translations between ATS and CGS (where the set of agents is preserved), the decision problems for the existence of ATS or CGS have the same complexity.

5 Extension with strategy contexts

In this section, we present an extension of ATL and ATL* with strategy contexts that allow us to express complex properties over strategies [9]. The basic idea is to deal with properties of the following form: given a strategy $F_A$ for $A$, does there exist (1) a strategy $F_B$ for $B$ such that the combination of $F_B$ with $F_A$ (denoted "$F_B \circ F_A"$) ensures some property $\Phi$, and (2) a strategy $F_C$ for $C$ such that "$F_C \circ F_A"$ ensures some property $\Psi$? The choices of $F_B$ and $F_C$ are independent (we do not assume any kind of cooperation between $B$ and $C$) but they rely on the choice of $F_A$. 
Consider the following example. A server $S$ has to treat the requests of $n$ agents $A_1, \ldots, A_n$. We want to check whether there is a strategy for $S$ in order to grant any request of the agents. Expressing such a property with ATL or ATL* is not possible. Something like $\langle S \rangle G (\bigwedge_i \text{req}_i \Rightarrow \langle A_i \rangle F \text{grant}_i)$ is not correct because the strategy for $S$ is not taken into account in the choices of $A_1, A_2, \ldots$ This would mean that every agent can enforce the server to grant its request. The property $\langle S \rangle G (\bigwedge_i \text{req}_i \Rightarrow \langle S, A_i \rangle F \text{grant}_i)$ is not right: because we want to specify that one given strategy for $S$ ensures the property for every agent. Finally $\langle S, A_1, \ldots, A_n \rangle (G \bigwedge_i \text{req}_i \Rightarrow F \text{grant})$ does not work because we do not want to assume that agents cooperate together. This has motivated the use of new modalities $\langle A \rangle$ that are interpreted in strategy contexts containing the strategies fixed by outermost quantifiers: the previous property could then be expressed by the following formula:

$$\langle S \rangle \left( \bigwedge_{i=1}^{n} (\text{req}_i \Rightarrow \langle A_i \rangle F \text{grant}_i) \right)$$

The innermost modalities $\langle A_i \rangle$s are strategy quantifiers within the context of the strategy for $S$ selected by the outermost modality $\langle S \rangle$.

Another example to illustrate the semantics of the new modality $\langle A \rangle$ is given in Figure 3. The left part of the figure gives a graphical interpretation of the formula $\langle A \rangle G \langle B \rangle F P$ in a location $q$: the first strategy quantifier $\langle A \rangle$ selects a set of runs satisfying $G \langle B \rangle F P$, thus from every state $q'$ along these executions there is a strategy for $B$ to ensure $F P$ (and this strategy does not rely on the previous strategy for $A$). In the right part of figure, we consider the formula $\langle A \rangle G \langle B \rangle F P : B$ has to select a set of executions among those provided by the strategy fixed by $A$.

To formally define the new modalities $\langle \cdot \rangle$, we first introduce some notions over the strategies. A strategy context is a strategy for a subset of agents in Agt. Let $F$ be a strategy for $A \subseteq \text{Agt}$. We use $\text{dom}(F)$ to denote $A$ (the domain of $F$). Given $B \subseteq \text{Agt}$, $(F_A)_B$ (resp. $(F_A)_B$) denotes the restriction of $F_A$ to the coalition $A \cap B$ (resp. $A \setminus B$). It is also possible to combine two strategies $F, F' \in \text{Strat}(A)$ and $F' \in \text{Strat}(B)$, resulting in a strategy $F \circ F' \in \text{Strat}(A \cup B)$ defined as follows: $(F \circ F')|_{A_j, \{q_0, \ldots, q_m\}}$ equals $F|_{A_j, \{q_0, \ldots, q_m\}}$ if $A_j \in A$, and it equals $F'|_{A_j, \{q_0, \ldots, q_m\}}$ if $A_j \in B \setminus A$. Finally, given a strategy $F$ and a finite execution $\rho$, we define the strategy $F^\pi$ corresponding to the behaviour of $F$ after $\rho$ as follows: $F^\pi(\pi) = F(\rho \cdot \pi)$.

Now we define $\text{ATL}_{sc}$:

**Definition:** [Syntax of $\text{ATL}_{sc}$][9]

$$\text{ATL}_{sc} \ni \phi_s, \psi_s := P \mid \neg \phi_s \mid \phi_s \lor \psi_s \mid \langle A \rangle \phi_p \mid \langle A \rangle \phi_s$$

$$\phi_p := X \phi_s \mid \phi_s U \psi_s \mid \phi_s R \psi_s$$
with \(P \in \text{AP}\) and \(A \subseteq \text{Agt}\).

\(\text{ATL}_{sc}\) formulas are interpreted over states of a CGS \(S\) within a context \(F\):

\[
\begin{align*}
q \models_{S,F} \langle A \rangle \phi_p & \text{ iff } \exists F_A \in \text{Strat}(A). \forall \rho \in \text{Out}(q, F_A \circ F) \text{ we have } \rho \models_{S,F \circ F} \phi_p, \\
\rho \models_{S,F} X \phi_s & \text{ iff } \rho[1] \models_{S,F[0]} \phi_s, \\
\rho \models_{S,F} \phi_s U \psi_s & \text{ iff } \exists i. \rho[i] \models_{S,F[0...i]} \psi_s \text{ and } \forall 0 \leq j < i \text{ we have } \rho[j] \models_{S,F[0...j]} \phi_s \\
\rho \models_{S,F} \phi_s R \psi_s & \text{ iff } \forall i : (\exists j < i. \rho(j) \models_{S,F[0...j]} \phi_s) \lor \rho(i) \models_{S,F[0...i]} \psi_s
\end{align*}
\]

The modality \(\langle A \rangle\) is used to remove the strategy for \(A\) from the current strategy context. This modality allow us to easily express \(\text{ATL}\) strategy quantifiers \(\langle A \rangle\):

\[
\langle A \rangle \Phi \equiv \langle A \rangle \text{gt} \langle A \rangle \Phi.
\]

We define \(\text{ATL}_{sc}^{\ast}\) as the variant of \(\text{ATL}^{\ast}\) with \(\langle \neg \rangle\) modalities.

Note that \(\text{ATL}_{sc}^{\ast}\) formula like \(\langle A_1 \rangle (\langle A_2 \rangle \Phi \land \neg \langle A_3 \rangle \Phi')\) can be written in \(\text{ATL}_{sc}\) as follows: \(\langle A_1 \rangle \perp U (\langle A_2 \rangle \Phi \land \neg \langle A_3 \rangle \Phi')\). Thus we can nest strategy quantifiers in \(\text{ATL}_{sc}\).

In [9] several results about the expressiveness of \(\text{ATL}_{sc}\) and \(\text{ATL}_{sc}^{\ast}\) are given. In particular, we can see that these logics have a stronger distinguishing power than \(\text{ATL}\) (and \(\text{ATL}^{\ast}\)):: alternating-bisimilar CGSs can be distinguished with \(\langle A \rangle\) modalities.

Moreover, examples of complex properties – like Nash equilibrium and winning secure equilibrium – are given in [9]. Note that these properties have been used to motivate the introduction of Strategy Logic (SL) in [12]. SL extends
linear-time temporal logics with first-order quantification over strategies, this logic has been studied for the two-player turn-based games.

The model checking problem for $\text{ATL}_{sc}$ is decidable. Indeed $\text{ATL}_{sc}$ can be translated into $\mathcal{OD}_\mu$ [24] that is a powerful extension of $\mu$-calculus with special modalities to deal with strategies.

Finally note also that $\text{ATL}_{sc}$ and $\text{ATL}^*_{sc}$ are uncomparable with the Alternating-time $\mu$-calculus and strictly more expressive than $\text{GL}$ [9].

There exist other logics that have been introduced to deal with such complex properties over strategies. In [1], a variant of ATL called IATL is proposed: the main idea is to consider irrevocable strategies (the difference with strategy contexts is that with IATL as soon as a player has chosen a strategy, he cannot modify it in the sequel). Decision procedures are given when one consider memoryless strategies. Stochastic Game Logic [8] is a probabilistic extension of ATL that uses a variant of strategy contexts, for stochastic games (model checking is undecidable in the general case, but proved decidable when restricting to memoryless strategies).

6 Timed extensions

Classical model checking framework has been adapted to handle timed verification: both models and specification languages have been extended to deal with real-time constraints. Several models have been proposed: the most interesting is probably the well-known timed automata introduced by Alur and Dill [3]. Temporal logics have also been extended in several manners to measure time elapsing between system events.

In this section we will consider a simple real-time extension of CGS with integer durations and two variants of games over dense time. In these models, we have a notion of duration (denoted $\text{Duration}(\pi)$) associated with any finite prefix $\pi$ of an execution. We will only present them informally and mention the main related results.

From the specification language point of view, we will consider an extension of $\text{ATL}$ where real-constraints are associated with $\mathbf{U}$ modalities (exactly as it is done for many timed temporal logics as $\text{TCTL}$ [2]).

6.1 Timed ATL

$\text{TATL}$ has been introduced in [17].
Definition: [Syntax of TATL]

TATL \ni \phi, \psi ::=
P \mid \neg \phi \mid \phi \lor \psi \mid \langle \langle A \rangle \rangle \phi_p

\phi_p ::=
\phi \mathbf{U}_c \psi \mid \phi \mathbf{R}_c \psi

with \( P \in \text{AP} \) and \( A \subseteq \text{Agt} \).

The integer constants are assumed to be encoded in binary (this is important for the complexity results). TATL formulas are interpreted over states in a timed game model. The semantics is defined as follows:

\[ q \models_S \langle \langle A \rangle \rangle \phi \] if \( \exists F_A \in \text{Strat}(A) \). \( \forall \rho \in \text{Out}(q, F_A) \) we have \( \rho \models_S \phi_p \),

\[ \rho \models_S \phi \mathbf{U}_c \psi \] if \( \exists \rho, \rho[p] \models_S \phi \) and \( \text{Duration}(\rho[p]) \sim d \)

and \( \forall \rho' <_\rho p \) we have \( \rho[p'] \models_S \phi \)

\[ \rho \models_S \phi \mathbf{R}_c \psi \] if \( \rho \models_S \neg(\neg \phi \mathbf{U}_c \neg \psi) \).

Note that we do not consider X modality because as soon as we consider dense time models, the notion of successor state is not relevant.

In the definition of the semantics of U, we use p and p' to denote positions along the run \( \rho \) and <p is the precedence relation over these positions. In the discrete-time case, a position is an integer but for the dense-time case a position is defined as a pair \((k, \delta) \in \mathbb{N} \times \mathbb{R} \geq 0\) to represent the configuration reached after the k-th action-transition followed by a δ time units delay.

fo discrete time

First we consider a simple timed variant of CGSs: A Durational CGS (DCGS) is a CGS where integer durations are associated with every transition \([20]\). More precisely the transition table is of the form \( \text{Edg} : Q \times M^k \rightarrow I \times Q \) where \( I \) denotes the set of intervals over \( \mathbb{N} \cup \{\infty\} \). For every transition \( \text{Edg}(q, m) = ([d; D], q') \), the final choice of the duration among the interval \([d; D]\) is done by an additional agent. This allows us to use these agents in the strategy quantifiers within the formulas. Note also that we call TDCGS (Tight DCGS) the class of DCGS where every interval is restricted to a single value.

Given a formula \( \phi \) def \( \langle \langle A \rangle \rangle P_1 \mathbf{U}_d P_2 \) with \( P_1, P_2 \in \text{AP} \), we can label the locations satisfying \( \phi \) by computing the minimal duration \( d' \) required by \( A \) to enforce reaching \( P_2 \) along a path satisfying \( P_1 \). This can be done recursively over the number of turns in the game. See \([20]\) for a complete description of the algorithms.

Theorem: [Model checking DCGS and TDCGS \([20]\).]
- Model checking TATL formula over DCGS or Tight DCGS is EXPTIME-complete.
- Model checking TATL_{≤,≥} over DCGS or Tight DCGS is P-complete.

**EXPTIME-hardness.** The complexity lower bound for TATL is based on the complexity of model checking \(\langle\langle A\rangle\rangle F = cP\) in a tight DCGS. It corresponds to a two-players game – the *countdown game* – in a weighted graph \((V, E, w)\) with \(w : E \rightarrow \mathbb{N}_{>0}\). A configuration of this game is a pair \((q, \alpha)\) with \(q \in V\) and \(\alpha \in \mathbb{N}\). At every turn, Player \(A_1\) chooses an integer \(d\) such that (1) \(0 < d \leq \alpha\) and (2) \(\exists (q, r) \in E\) s.t. \(w(q, r) = d\). Then Player \(A_2\) chooses a transition from \(q\) whose duration is \(d\) and leading to some \(q'\). The new configuration is then \((q', \alpha - d)\). A configuration \((q, 0)\) is winning for \(A_1\) (for any \(q\)). A configuration \((q, \alpha)\) with \(\alpha > 0\) and no transition from \(q\) whose duration is less than \(\alpha\) is winning for \(A_2\). Deciding whether there is a winning strategy for \(A_1\) in such a game is EXPTIME-complete [19]. It can easily be reduced to a model checking problem of \(\langle\langle A_1\rangle\rangle F = cP\) over a Tight DCGS.

### 6.3 Dense time

Here we present two variants of game models based on timed automata (other models exist, see for example [22, 7]).

**Timed Games.** The most well-known timed models for games are the Timed Games Automata (TGA) introduced in [14]. As in timed automata, there are clocks that progress synchronously with time and every transition is guarded by a constraint over the clocks. The main difference with TA is that the set of transitions is partitioned in two subsets: one for each player (NB: this is a two-player game).

At every turn, from a configuration \((q, v)\) where \(q\) is a location and \(v\) is a valuation for the clocks, Player \(A_i\) has to choose a delay \(d_i\) and a transition \(t_i\) from his set of transitions (starting from \(q\)). Then the new configuration of the game is computed as follows: assume \(d_1 < d_2\) (resp. \(d_2 < d_1\)) then the transition \(t_1\) (resp. \(t_2\)) is fired after \(d_1\) (resp. \(d_2\)) time units. If \(d_1 = d_2\), the system chooses non-deterministically to perform \(t_1\) or \(t_2\) after \(d_1\) time units.

Consider the example below:

\[
\begin{aligned}
c_2; x := 0 \\
q_1 \quad c_1; x > 0 \quad q_2
\end{aligned}
\]
If Player $A_1$ plays $(1.5, c_1)$ and $A_2$ chooses $(0.8, c_2)$ from $(q_1, 0)$, then the new configuration is $(q_1, 0)$ after 0.8 time units. If $A_1$ chooses $(0.5, c_1)$ and $A_2$ plays $(0.8, c_2)$ then the new configuration is $(q_2, 0.5)$.

On this example, one can see that $A_2$ can avoid to reach $q_2$ by always choosing $(0, c_2)$ since Player $A_1$ has to play $(d, c_1)$ with $d > 0$. But such a strategy for $A_2$ is not fair: it is a Zeno strategy, it makes time converge. In TGA, we usually require strategies to be non-Zeno: a player cannot block time elapsing. For example, in [14] it is assumed that a player win a game only if its winning objective is satisfied and either time diverges or this player is not responsible for the convergence of time. With such a requirement the previous strategy for $A_2$ is not winning any more even if his control objective is to avoid $q_2$. And in this game, Player $A_1$ has a strategy to reach $q_2$.

Timed CGS. Timed CGSs are CGS with real-time constraints [10]. The main difference compared with the previous TGA is that the concurrency of CGS is preserved: the next location depends on the choices of all agents. An example is given in the Figure below.

In these games, every player chooses a delay $d$ (after which he wants to play) and a move function from $\mathbb{R}^+$ in $M$ that gives a move for any possible delay. This function allows the player to participate to the move even if another player asks to play before $d$. From the choices $(d_i, f_i)$ for $i = 1, \ldots, k$, one can deduce the new location as follows: the system will delay for $d = \min(d_1, \ldots, d_i)$ time units and then perform the transition $Edg(q, f_1(d_1), \ldots, f_k(d_k))$.

Consider the previous example. First note that some transitions are missing: for example, when $x < 2$ there is no move $(1.2)$ from $q_1$. In this case we assume that these moves correspond to a self-loop in the TCGS. From $(q_1, x = 1.2)$, assume that Player $A_1$ chooses $(d_1, f_1)$ with $d_1 = 0.9$ and $f_1(d) = 2$ if $d \leq 0.5$ and $f_1(d) = 1$ when $d > 0.5$. Moreover assume that $A_2$ chooses $(d_2, f_2)$ with $d_2 = 1.4$ and $f_2(d) = 2$ when $d \leq 1$ and $f_2(d) = 1$ otherwise. Then with these moves, the system will perform the transition $q_2 \rightarrow q_3$ labeled with $(1.2)$ after 0.9 time units.

Model checking TATL specifications over TCGS is decidable (one can build a finite "region" CGS that is time-abstract game-bisimilar to the infinite CGS corresponding to the semantics of the TCGS).
And we have:

**Theorem:**

- Model checking $\text{TATL}$ over $\text{TGA}$ is EXPTIME-complete \[14\].
- Model checking $\text{TATL}$ over $\text{TCGS}$ is EXPTIME-hard and can be done in EXPSPACE \[10\].

Finally note that there is a tool – UppAal Tiga \[1\] – to automatically analyze a variant of $\text{TGA}$. One can check reachability properties with an on-the-fly algorithm \[11\].

**Acknowledgments**

Special thanks to Nicolas Markey, Arnaud Da Costa, Thomas Brihaye and Ghasan Oreiby for all the work we have done together on $\text{ATL}$ and its variants. Many thanks to Luca Aceto and Béatrice Bérard for the interesting discussions about temporal logics.

**References**


\[http://www.cs.aau.dk/~adavid/tiga/\]


