Information Processing Letters 116 (2016) 75–79

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

On the semantics of Strategy Logic ☆

Patricia Bouyer, Patrick Gardy, Nicolas Markey *

LSV, CNRS, ENS Cachan, Univ. Paris-Saclay, France

ARTICLE INFO

Article history:
Received 21 November 2014
Received in revised form 14 October 2015
Accepted 14 October 2015
Available online 27 October 2015
Communicated by A. Muscholl

Keywords:
Formal methods
Multi-agent systems
Strategic reasoning
Temporal logics

ABSTRACT

We define and study a slight variation on the semantics of Strategy Logic: while in the classical semantics, all strategies are shifted during the evaluation of temporal modalities, we propose to only shift the strategies that have been assigned to a player, thus matching the intuition that we can assign the *very same* strategy to the players at different points in time. We prove that surprisingly, this renders the model-checking problem undecidable.

1. Introduction

Model checking [5,3] is a model-based technique for automatically verifying properties of computerized systems. Model-checking algorithms exhaustively explore the set of behaviours of the (model of the) system under study, and compare this set against properties being checked. *Temporal logics*, and in particular the *Linear-time Temporal Logic* (LTL) [14] and the *Computation-Tree Logic* (CTL) [15,4], provide a convenient formalism for expressing such properties: they extend boolean logics in order to state properties of sequences of boolean valuations. Using temporal modalities, they can constrain the order in which various events occur along such sequences. It is then possible to express, for instance, that any problem is eventually followed by an alarm.

During the last 15 years, temporal logics—and model checking—have been extended to deal with *multi-agent systems*: there, the behaviour of the global system depends on the actions of individual agents, and the new logic, *Alternating-time Temporal Logic* (ATL) [1,2], can now express what some agent can (or cannot) achieve, or what happens in the whole system when some agent tries to achieve their goal. This extension is particularly relevant in the setting of *controller synthesis*, as it provides a way of expressing the existence of a controller (often seen as a *strategy* in a game against the other agents) enforcing a given property.

However, it has been noticed recently that ATL is not expressive enough to express many interesting properties of multi-agent systems. In particular, ATL is mainly usable for expressing properties of antagonistic agents, and cannot express real interactions or collaborations between agents. It has thus been enriched in order to allow for such collaborations: *Strategy Logic* (SL) [6,7,13,12], in particular, deals with strategies as first-class citizens, with (first-order) quantification, and assignment to one or several agents.

Consider for instance a network of several clients, that may ask a central server for accessing a shared resource.

☆ This work was partly supported by ERC project EQualIS (FP7-308087) and FET project Cassting (FP7-601148).

* Corresponding author.
E-mail addresses: patricia.bouyer@lsv.fr (P. Bouyer), patrick.gardy@lsv.fr (P. Gardy), nicolas.markey@lsv.fr (N. Markey).

http://dx.doi.org/10.1016/j.ipl.2015.10.004
0020-0190/

One (or several) user can turn the clients on and off, and when turned on, each client then requests access to the resource. The server then has two objectives: one is to enforce that no two clients access the resource at the same time, whatever the clients do; the second property is that the clients must have a strategy that each of them can apply when turned on, and that ensures access to the resource (by collaborating with the server). This, in SL (with adapted syntax to make the formula readable), would be written

$$\exists \sigma_{\text{server}}.\ \texttt{if}\ \text{server}\ \texttt{applies}\ \sigma_{\text{server}}\ \texttt{then}$$
$$\Big[(\texttt{always}\ \text{mutual exclusion}) \wedge$$
$$(\exists \sigma_{\text{client}}.\ \text{always}\ (\texttt{if}\ \text{client}\ \texttt{applies}\ \sigma_{\text{client}}\ \texttt{then}$$
$$\texttt{eventually}\ \text{access}))\Big].$$

SL model checking is decidable [6,12]. In this paper, we prove that this result heavily relies on a semantical choice that is silently made in the previous papers about SL. We argue in this paper that this semantical choice does *not* achieve the expected meaning for the sample formula above (intuitively, because it gives to the subformula "client $\texttt{applies}\ \sigma_{\text{client}}$" a meaning that depends on the history of the system, whereas when a client is turned on, it should start applying its strategy with no prior knowledge about what has happened previously). We propose an alternative semantics, which assumes that strategies starts being applied with empty history, and prove that this minor change makes the model-checking problem undecidable.

2. Definitions

2.1. Turn-based games

Logics for multi-agent systems are usually interpreted over structures involving multiple agents (hence the name...). In the context of this note, we only focus on *two-player turn-based* games, since this is enough for proving our result.

Definition 1. A two-player turn-based game is a tuple $\mathcal{G} = \langle S_{\bigcirc}, S_{\Box}, T\rangle$ where $S_{\bigcirc}$ and $S_{\Box}$ are pairwise-disjoint finite sets of states, $T \subseteq S^2$ (where $S = S_{\bigcirc} \cup S_{\Box}$) is the set of transitions. It is assumed that for all $s \in S$, there exists $s' \in S$ s.t. $(s, s') \in T$.

A path in such a game is a (finite or infinite) sequence $(s_i)_{1\le i<L+1}$ (with $L \in \mathbb{N}\cup\{+\infty\}$) of states such that, for every $1 \le i < L$, it holds $(s_i, s_{i+1}) \in T$. The length of a path $(s_i)_{1\le i<L+1}$ is the number L of elements of the sequence. A strategy for Player $\bigcirc$ is a mapping $\sigma^{\bigcirc}\colon S^* \times S_{\bigcirc} \to S$ such that for all finite path $(s_i)_{1\le i\le n}$ with $s_n \in S_{\bigcirc}$, it holds $(s_n, \sigma^{\bigcirc}((s_i)_{1\le i\le n})) \in T$. In other terms, a strategy for Player $\bigcirc$ tells which transition to follow after any finite play ending in a state controlled by that player. Strategies for Player $\Box$ are defined symmetrically. We write $\mathsf{Strat}^{\bigcirc}$ and $\mathsf{Strat}^{\Box}$ for the sets of strategies of Players $\bigcirc$ and $\Box$, and Strat for the set of all strategies.

Given a strategy $\sigma^{\bigcirc}$ for Player $\bigcirc$, a strategy $\sigma^{\Box}$ for Player $\Box$, and a state s, the outcome of $\sigma^{\bigcirc}$ and $\sigma^{\Box}$ from s is the infinite path $(s_i)_{i\ge1}$ s.t. $s_1 = s$, and $s_{n+1} = \sigma^{\bigcirc}((s_i)_{1\le i\le n})$ if $s_n \in S_{\bigcirc}$, and $s_{n+1} = \sigma^{\Box}((s_i)_{1\le i\le n})$ if $s_n \in S_{\Box}$.

2.2. Strategy Logic (SL)

2.2.1. Syntax and semantics of SL

We now present logics for expressing properties of the games defined above. For this, we first fix a finite set AP of atomic propositions, and consider *labelled* games, with a mapping $\ell\colon S \to 2^{\mathsf{AP}}$.

Strategy Logic (SL for short) was introduced in [6], and further extended and studied in [13,12], as a rich logical formalism for expressing properties of games. Formulas in SL are built along the following grammar[1]:

$$\mathsf{SL} \ni \varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi \mid \exists x.\ \varphi$$
$$\mid \mathsf{assign}(a \mapsto x).\ \varphi$$

where p ranges over AP, x ranges over a set Var of variables, and a ranges over a finite set Agt of agents (in our setting, $\mathsf{Agt} = \{\bigcirc, \Box\}$). Thus, SL can be seen as an extension of LTL [14] with strategy quantification ($\exists x.\ \varphi$, which selects a strategy and stores it in variable x, before evaluating φ) and strategy assignments ($\mathsf{assign}(a \mapsto x).\ \varphi$, which assigns the strategy stored in variable x to Player a, and then evaluates φ).

Formally, formulas of SL are evaluated at a state s of a game $\mathcal{G}$, under a valuation χ mapping (part of the) agents and variables to strategies. We write $\mathsf{dom}(\chi)$ for the subset of $\mathsf{Agt}\cup\mathsf{Var}$ on which χ is defined. The semantics of atomic propositions and boolean combinators is the natural one.

In order to define the semantics of strategy quantifiers and assignments, we need several intermediary notions. The set of *free agents and variables* of a formula φ, which we write $\mathsf{free}(\varphi)$, contains the agents and variables that have to be associated with a strategy before φ can be evaluated. It is defined inductively as follows:

$$\mathsf{free}(p) = \varnothing \quad \text{for all } p \in \mathsf{AP}$$
$$\mathsf{free}(\neg\varphi) = \mathsf{free}(\varphi)$$
$$\mathsf{free}(\varphi \vee \psi) = \mathsf{free}(\varphi) \cup \mathsf{free}(\psi)$$
$$\mathsf{free}(\mathbf{X}\varphi) = \mathsf{Agt} \cup \mathsf{free}(\varphi)$$
$$\mathsf{free}(\varphi\mathbf{U}\psi) = \mathsf{Agt} \cup \mathsf{free}(\varphi) \cup \mathsf{free}(\psi)$$
$$\mathsf{free}(\exists x.\ \varphi) = \mathsf{free}(\varphi) \setminus \{x\} \begin{cases} \mathsf{free}(\varphi) & \text{if } a \notin \mathsf{free}(\varphi) \\ (\mathsf{free}(\varphi) \cup \{x\}) \setminus \{a\} & \text{otherwise} \end{cases}$$

Let s be a state of $\mathcal{G}$, χ be a valuation, $x \in \mathsf{Var}$, and $\varphi \in \mathsf{SL}$ s.t. $\mathsf{free}(\varphi) \setminus \{x\} \subseteq \mathsf{dom}(\chi)$. Then

[1] We mainly follow the syntax of SL from [12], but write $\exists x.\ \varphi$ instead of $\langle\langle x\rangle\rangle\varphi$ (to avoid confusions with the ATL strategy quantified $\langle\langle -\rangle\rangle$), and $\mathsf{assign}(a \mapsto x).\ \varphi$ instead of $(a, x)\varphi$ (thus avoiding overloading parentheses).