Verifying Pushdown Multi-Agent Systems against Strategy Logics*

Taolue Chen
Department of Computer Science
Middlesex University London, UK

Fu Song
Shanghai Key Laboratory of
Trustworthy Computing
East China Normal University, China

Zhilin Wu
State Key Laboratory of
Computer Science
Institute of Software
Chinese Academy of Sciences, China

Abstract

In this paper, we investigate model checking algorithms for variants of strategy logic over pushdown multi-agent systems, modeled by pushdown game structures (PGSs). We consider various fragments of strategy logic, i.e., SL[CG], SL[DG], SL[1G] and BSIL. We show that the model checking problems on PGSs for SL[CG], SL[DG] and SL[1G] are 3EXTIME-complete, which are not harder than the problem for the subsumed logic ATL*. When BSIL is concerned, the model checking problem becomes 2EXPTIME-complete. Our algorithms are automata-theoretic and based on the saturation technique, which are amenable to implementations.

1 Introduction

A *multi-agent system* (MAS), in a nutshell, is a complex decentralized computing system composed of multiple interacting intelligent agents within an environment, in which the behavior of each agent is determined by its observed information of the system. One of the most important models for multi-agent systems is (finite-state) *concurrent game structures*. Very recently (at IJCAI'15), a class of *infinite-state* multi-agent systems, i.e., pushdown multi-agent systems, was also studied [Murano and Perelli, 2015]. These infinite MASs are modeled naturally by *pushdown game structures* (PGSs), which are the main focus of the current paper.

To specify the behavior of MASs, a well-known logical formalism is *Alternating-Time Logic* (ATL), or its extension ATL* where more complex temporal properties can be expressed [Alur *et al.*, 2002]. In contrast to traditional reactive systems, for MASs, properties expressing *cooperation and enforcement* of agents must be taken into account. When these properties are concerned, ATL-like logics suffer, unfortunately, from significant limitations, which has been observed in a number of recent papers (e.g., [Chatterjee *et al.*, 2010; Mogavero *et al.*, 2012; 2014]). In particular, in these logics one is unable to refer explicitly to specific strategies a group of agents might take, which handicap the specification of many important MAS-specific properties, typically involving game-theoretic notions of agents in a cooperative and/or adversarial setting.

To remedy these shortcomings, *strategy logic* (SL, [Mogavero *et al.*, 2014]) has recently been put forward. In SL, strategies are explicitly referred to by using first-order quantifiers and bindings to agents. As a result, sophisticated concepts such as Nash equilibria, which cannot be expressed in ATL*, can naturally be encoded in SL. On the other hand, it is probably not surprising that the expressiveness of SL comes with a price of high computational complexity. For instance, its satisfiability problem is at least NON-ELEMENTARY hard. In light of this, several fragments of SL have been studied, for instance, *Nested-Goal*, *Boolean-Goal*, *Conjunctive-goal*, *Disjunctive-goal*, and *One-Goal* Strategy Logic, respectively denoted by SL[NG], SL[BG], SL[CG], SL[DG], SL[1G] [Mogavero *et al.*, 2013; 2014; Cermák *et al.*, 2015]. Independently, Wang et al [Wang *et al.*, 2015] put forward *basic strategy-interaction logic* (BSIL), which is a proper extension of ATL (but incomparable to ATL*). The main technical ingredient of BSIL is a new modal operator, viz, strategy interaction quantifier. As a specification language, BSIL bears an appropriate and natural balance between the expressiveness and the verification complexity.

For verification, we are mostly interested in *model checking*, a well-established formal method that allows to automatically verify correctness of systems. Model checking finite-state concurrent game structures is well-understood now. In particular, it is known that model checking SL[NG] or SL[BG] is already NON-ELEMENTARY hard [Mogavero *et al.*, 2014; Bouyer *et al.*, 2015], SL[CG], SL[DG] or SL[1G] is 2EXPTIME-complete, and BSIL is PSPACE-complete. In contrast, much less is known for PGSs. Only very recently, model checking ATL* and alternating-time μ-calculus is shown to be 3EXPTIME-complete and EXPTIME-complete respectively [Murano and Perelli, 2015; Chen *et al.*, 2016]. An obvious question is, how to model check PGSs against strategy logic? The current paper aims to fill in this gap.

It is known that SL[NG]/SL[BG] semantics might admit non-behavioral strategies, meaning that a choice of an agent at a given point of a play may depend on choices other agents can make in the future or in counter-factual plays [Mogavero *et al.*, 2013; Wang *et al.*, 2015]. This is not interesting from an MAS perspective. For this reason, we only consider the following subclasses: SL[CG], SL[DG], SL[1G]

*Equal contribution.

and BSIL. We show that the model checking problems on PGSs for SL[CG], SL[DG] and SL[1G] are 3EXTIME-complete, which is not harder than the problem for the subsumed logic ATL*, while the problem becomes 2EXPTIME-complete when considering BSIL. These results confirm the observation that SL[CG], SL[DG], SL[1G] do not increase the verification complexity in the asymptotic sense, comparing to ATL*, and that BSIL indeed is "simpler" in terms of verification complexity.

Our model checking algorithms are automata-theoretic and evidently are also applicable to concurrent (finite) game structures. One of the distinguished features is that they are able to perform *global* model checking, i.e., to compute (a finite representation of) the set of states that satisfy a given property. The importance of global model checking has been discussed in, e.g., [Piterman and Vardi, 2004]. It is crucial when repeated checks are required, or where the model checking is only a component of the verification process. It is also very useful when studying coverage metrics [Chockler *et al.*, 2006] for model checking, which could be used when an MAS practitioner cannot guarantee the correctness of specifications or system models. Specifically for the pushdown structures, our model checking algorithms are also saturation-based, which are amenable to implementations. Moreover, they can deal with regular valuations rather than simple valuations of atomic propositions. By regular valuations one atomic proposition can denote an infinite (but regular) set of configurations. For two reasons we consider regular valuations of atomic propositions: (1) the algorithm iteratively computes, for a (sub-)formula, a possibly infinite, but regular set of configurations. This (sub-)formula will then be replaced by a fresh atomic proposition with the regular set as the valuation, (2) Regular valuations do not bring extra cost to our algorithm, but may make the specification more convenient, see e.g. [Esparza *et al.*, 2003].

As another contribution, we also clarify the expressiveness of BSIL and SL. While it seems that BSIL is incomparable with SL[1G], we show that it is strictly less expressive than SL[CG] and SL[DG]. This was not known before to the best of our knowledge.

Related Work. LTL/CTL model checking on pushdown systems were well studied in the literature which can be used to verify infinite-state closed systems (see [Carayol and Hague, 2014] for a survey). Two-player games or module checking on pushdown systems were also extensively studied; see, e.g., [Walukiewicz, 2001; Hague and Ong, 2009; Löding *et al.*, 2004; Serre, 2003; Aminof *et al.*, 2013; Bozzelli *et al.*, 2010] which can be used to verify infinite-state open systems. However, as discussed in [Jamroga and Murano, 2014], module checking (model checking open systems) is incomparable to model checking MAS.

Model checking techniques were extended to verify *finite-state* MASs against variants of temporal logics, typically based on ATL. For instance, [Bourahla and Benmohamed, 2005; Bulling and Jamroga, 2011; Jamroga and Murano, 2015]; see [Chen *et al.*, 2016] for further references. The most closely related work is [Cermák *et al.*, 2015]. The authors provided symbolic model checking algorithms for SL[1G], but restricted to finite MASs. In contrast, we consider infinite MASs and much more expressive fragments of SL. We note that [Wang *et al.*, 2015] also presented automata-based model checking algorithms for BSIL. However, their method was based on tree automata, which is considerably different from ours.

2 Pushdown Game Structures

We write $[n] = \{1, 2, \ldots, n\}$. Let AP be a finite set of *atomic propositions*, Ag be a finite set of *agents*, Ac be a finite set of *actions* that can be made by agents, $Dc = Ac^{Ag}$ be the set of *decisions* of the agents in Ag. For each agent $a \in Ag$ and decision $d \in Dc$, let $d(a)$ denote the action chosen by a in d.

Definition 1 (Pushdown Game Structures, [Murano and Perelli, 2015])**.** *A Pushdown Game Structure (PGS) is a tuple $\mathcal{P} = (P, \Gamma, \Delta, \lambda)$, where P is a finite set of control states, Γ is a finite stack alphabet, $\Delta : P \times \Gamma \times Dc \to P \times \Gamma^*$ is a transition function, $\lambda : P \times \Gamma^* \to 2^{AP}$ is a labeling function that assigns to each $\langle p, \omega\rangle \in P \times \Gamma^*$ a set of atomic propositions. W.l.o.g., we assume that $\perp \in \Gamma$ is a special bottom stack symbol never popped up from the stack.*

A *configuration* $\langle p, \omega\rangle$ of the PGS $\mathcal{P}$ consists of a state $p \in P$, a stack content $\omega \in \Gamma^*$. We denote by $\mathcal{C}_\mathcal{P}$ the set $P \times \Gamma^*$. For every $(p, \gamma, d) \in P \times \Gamma \times Dc$ with $\Delta(\langle p, \gamma\rangle, d) = (p', \omega)$, we usually write $\langle p, \gamma\rangle \overset{d}{\hookrightarrow}_\mathcal{P} \langle p', \omega\rangle$ instead.

The transition relation $\Longrightarrow_\mathcal{P}$: $\mathcal{C}_\mathcal{P} \times Dc \times \mathcal{C}_\mathcal{P}$ of the PGS $\mathcal{P}$ is defined by the following rule: for every $\omega' \in \Gamma^*$, $\langle p, \gamma\omega'\rangle \overset{d}{\Longrightarrow}_\mathcal{P} \langle p', \omega\omega'\rangle$ if $\langle p, \gamma\rangle \overset{d}{\hookrightarrow}_\mathcal{P} \langle p', \omega\rangle$. The transition relation $\Longrightarrow_\mathcal{P}$ represents possible concurrent moves of the players involved in the game. A *track* of $\mathcal{P}$ is a *finite* sequence $\pi = c_0...c_n$ over $\mathcal{C}_\mathcal{P}^*$ such that $\forall i : 0 \le i < n$, $c_i \overset{d}{\Longrightarrow}_\mathcal{P} c_{i+1}$. A *path* of $\mathcal{P}$ is an *infinite* sequence $\pi = c_0c_1...$ over $\mathcal{C}_\mathcal{P}$ such that $\forall i \ge 0$, $c_i \overset{d}{\Longrightarrow}_\mathcal{P} c_{i+1}$. Given a track $\pi = c_0...c_n$ (resp. path $\pi = c_0c_1...$), for every $i : 0 \le i \le n$ (resp. $i \ge 0$), let π_i denote c_i, $\pi_{\ge i}$ denote $c_i...c_n$ (resp. $c_ic_{i+1}...$) of π, $\pi_{<i}$ denote the prefix sequence $c_0...c_{i-1}$ of π. Let $T_\mathcal{P} \subseteq \mathcal{C}_\mathcal{P}^+$ denote the set of all tracks in $\mathcal{P}$, $\prod_\mathcal{P} \subseteq \mathcal{C}_\mathcal{P}^\omega$ denote the set of all paths in $\mathcal{P}$. Furthermore, given a configuration c, we denote by $T_\mathcal{P}(c)$ (resp. $\prod_\mathcal{P}(c)$) the set $\{\pi \in T_\mathcal{P} \mid \pi_0 = c\}$ (resp. $\{\pi \in \prod_\mathcal{P} \mid \pi_0 = c\}$).

A *strategy* for an agent in a PGS $\mathcal{P}$ is a function $\theta : T_\mathcal{P} \to Ac$ that contains all the possible choices of actions depending upon the tracks (i.e., the history the agent saw so far). Let Θ denote the set of all the possible strategies. A path π is *compatible* with an assignment $\upsilon_A : A \to \Theta$ over the set A of agents, if for every $i \ge 0$, there is a decision $d \in Dc$ such that $\pi_i \overset{d}{\Longrightarrow}_\mathcal{P} \pi_{i+1}$ and $d(a) = \upsilon_A(a)(\pi_{<i})$ for all $a \in A$. Given a configuration $c \in \mathcal{C}_\mathcal{P}$ and an assignment υ_A over the set A of agents, let $\prod_\mathcal{P}(c, \upsilon_A) = \{\pi \mid \pi \in \prod_\mathcal{P}(c) \wedge \pi$ is compatible with $\upsilon_A\}$.

A *valuation* $\upsilon : V \cup Ag \to \Theta$ is a function assigning to each agent and element from V a strategy. Given a valuation υ, a configuration c of $\mathcal{P}$, a play corresponding to (c, υ) is the unique outcome of $\mathcal{P}$ determined by the strategies $\upsilon(a)$ of all agents $a \in Ag$ participating to it. Formally, a *play* corresponding to (c, υ) is a path $\pi \in \prod_\mathcal{P}(c)$ such that for every $\forall i \geq 1$, $\pi_{i-1} \xRightarrow{d_i}_{\mathcal{P}} \pi_i$ and $d_i(a) = \upsilon(a)(\pi_{<i})$ for every $a \in Ag$. Given an integer $i \geq 0$, let $(c, \upsilon)^i$ denote the pair (π_i, υ_i) such that π_i is the i-th configuration of the play π corresponding to (c, υ), and υ_i is the updated valuation after the i-steps of the play π, more precisely, υ_i is the valuation such that $\forall x \in V \cup Ag$, $\upsilon_i(x)$ is a strategy satisfying that $\forall \pi' \in T_{\mathcal{P}}(c_i)$, $\upsilon_i(x)(\pi') = \upsilon(x)(\pi_{<i}\pi')$.

3 Strategy Logic

3.1 SL[CG] and SL[DG]

SL[CG] and SL[DG] are extensions of the logic LTL by introducing quantification prefixes and binding prefixes. A *quantification prefix* $\wp \in \{\langle x \rangle, [x] \mid x \in V\}^{|V|}$ over a set of *strategy variables* V is a $|V|$-length word in which each variable $x \in V$ is either existentially quantified $\langle x \rangle$, or universally quantified $[x]$. Let $|\wp|$ denote the length of $\wp$, that is, $|V|$, $\wp(i)$ denote the i^{th} symbol $\langle x \rangle$ or $[x]$ in $\wp$, and $\wp(x)$ denote the position of x in $\wp$. Let $QPre_V$ denote the set of all quantification prefixes over V. A *partial* binding prefix is a word $(a_1, x_1) \ldots (a_n, x_n) \in (Ag \times V)^*$ such that $a_i \neq a_j$ for $i \neq j$. A *binding prefix* is a partial binding prefix of length $|Ag|$, i.e., $a_1, \ldots, a_n$ is an enumeration of all agents in Ag. An LTL formula preceded by a binding prefix is called a *goal*. In SL[CG], goals are restricted into conjunctive form. Dually, goals are restricted into disjunctive form in SL[DG]. Given a formula ϕ which is a Boolean combination of goals $\flat_i \varphi_i$, let $free(\phi)$ denote the set of strategy variables appearing in the binding prefixes $\flat_i' s$.

Definition 2 (SL[CG] and SL[DG])**.** *[Mogavero* et al.*, 2014; 2013] The syntax of SL[CG] and SL[DG] is defined as follows, with $\star = \wedge$ for SL[CG] and $\star = \vee$ for SL[DG]:*

$$\varphi ::= q \mid \neg\varphi \mid \varphi \wedge \varphi \mid \boldsymbol{X}\varphi \mid \varphi \boldsymbol{U} \varphi \mid \wp\psi$$
$$\psi ::= \flat\varphi \mid \psi \star \psi$$

where $q \in AP$, $\flat$ is a binding prefix, $\wp \in QPre_{free(\psi)}$ is a quantification prefix over $free(\psi)$.

Given a valuation $\upsilon : V \cup Ag \to \Theta$, a configuration c of a PGS $\mathcal{P}$ and an SL[CG] or SL[DG] formula φ, the satisfaction relation $\mathcal{P}, c, \upsilon \models \varphi$ is inductively defined as follows:

- $\mathcal{P}, c, \upsilon \models q$ iff $q \in \lambda(c)$;
- $\mathcal{P}, c, \upsilon \models \neg\varphi$ iff $\mathcal{P}, c, \upsilon \not\models \varphi$;
- $\mathcal{P}, c, \upsilon \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{P}, c, \upsilon \models \varphi_1$ and $\mathcal{P}, c, \upsilon \models \varphi_2$;
- $\mathcal{P}, c, \upsilon \models \langle x \rangle\varphi$ iff $\exists \theta \in \Theta$, $\mathcal{P}, c, \upsilon[\theta/x] \models \varphi$, where $\upsilon[\theta/x]$ is equal to υ except for $\upsilon[\theta/x](x) = \theta$;
- $\mathcal{P}, c, \upsilon \models [x]\varphi$ iff $\forall \theta \in \Theta$, $\mathcal{P}, c, \upsilon[\theta/x] \models \varphi$;
- $\mathcal{P}, c, \upsilon \models (a, x)\varphi$ iff $\mathcal{P}, c, \upsilon[\upsilon(x)/a] \models \varphi$;
- $\mathcal{P}, c, \upsilon \models \mathbf{X}\varphi$ iff $\mathcal{P}, (c, \upsilon)^1 \models \varphi$;
- $\mathcal{P}, c, \upsilon \models \varphi_1 \mathbf{U} \varphi_2$ iff $\exists i \geq 0$, $\mathcal{P}, (c, \upsilon)^i \models \varphi_2$ and for every $j : 0 \leq j < i$, $\mathcal{P}, (c, \upsilon)^j \models \varphi_1$.

Note that, all agents are bound to some strategies in υ when interpreting $\mathbf{X}\varphi$ or $\varphi_1 \mathbf{U} \varphi_2$. A configuration c of a PGS $\mathcal{P}$ satisfies a formula φ, denoted by $\mathcal{P}, c \models \varphi$, iff there is a valuation υ such that $\mathcal{P}, c, \upsilon \models \varphi$. Let $\|\varphi\|_{\mathcal{P}}$ denote the set of configurations c of $\mathcal{P}$ such that $\mathcal{P}, c \models \varphi$.

Proposition 1. *SL[CG] is as expressive as SL[DG].*

One-Goal SL (SL[1G]) is a special class of SL[CG] and SL[DG] in which quantification and binding prefixes merge into one rule, i.e., $\wp\flat\varphi$.

3.2 Basic Strategy-Interaction Logic

BSIL [Wang *et al.*, 2015] is an extension of alternating-time temporal logic (ATL) [Alur *et al.*, 2002] for specifying collaboration among agents. BSIL has three types of formulae: *state formulae*, *tree formulae* and *path formulae*, where state formulae and path formulae are used to express properties on states and paths of MAS respectively, while tree formulae are used to describe the interaction of strategies.

Definition 3 (BSIL)**.** *BSIL formulae are defined by the following three syntax rules:*

(*State formula*)	$\phi ::= q \mid \neg\phi \mid \phi \wedge \phi \mid \langle A \rangle\tau \mid \langle A \rangle\varphi$;
(*Tree formula*)	$\tau ::= \tau \wedge \tau \mid \tau \vee \tau \mid \langle +A \rangle\tau \mid \langle +A \rangle\varphi$;
(*Path formula*)	$\varphi ::= \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \boldsymbol{X}\phi \mid \phi\boldsymbol{U}\phi \mid \phi\boldsymbol{R}\phi$;

where, $q \in AP$, $A \subseteq Ag$.

$\langle A \rangle$ is called a *strategy quantifier* (SQ for short) and $\langle +A \rangle$ is called a *strategy-interaction quantifier* (SIQ for short). One can observe that each SIQ $\langle +A \rangle$ is bounded by some SQ $\langle A' \rangle$, that is, $\langle +A \rangle\tau$ or $\langle +A \rangle\varphi$ only appears as a subformula of $\langle A' \rangle\tau'$. Moreover, SIQs $\langle +A \rangle$ do not cross path modal operators **X**, **U** or **R**, which is important and allows us to analyze the interaction of strategies locally in a configuration and then enforce the interaction along all paths from this configuration, as pointed out by Wang *et al.* [Wang *et al.*, 2015]. We will use ψ to denote $\langle +\emptyset \rangle\psi$. Let $|\phi|$ denote the length of ϕ. State formulae are called BSIL formulae. In the rest of this paper, we use $\phi, \phi_1, ...$ to denote state formulae, $\tau, \tau_1, ...$ to denote tree formulae and $\varphi, \varphi_1, ...$ to denote path formulae.

As in SL, the semantics of BSIL is defined over PGSs. Let $\mathcal{P} = (P, \Gamma, \Delta, \lambda)$ be a PGS. Given a state formula ϕ and a configuration $c \in \mathcal{C}_{\mathcal{P}}$, the satisfiability relation $\mathcal{P}, c \models \phi$ is defined inductively as follows:

- $\mathcal{P}, c \models q$ iff $q \in \lambda(c)$;
- $\mathcal{P}, c \models \neg\phi$ iff $\mathcal{P}, c \not\models \phi$;
- $\mathcal{P}, c \models \phi_1 \wedge \phi_2$ iff $\mathcal{P}, c \models \phi_1$ and $\mathcal{P}, c \models \phi_2$;
- $\mathcal{P}, c \models \langle A \rangle\tau$ iff $\exists \upsilon_A : A \to \Theta, \mathcal{P}, c, \upsilon_A \models \tau$;
- $\mathcal{P}, c \models \langle A \rangle\varphi$ iff $\exists \upsilon_A : A \to \Theta$, $\forall \pi \in \prod_{\mathcal{P}}(c, \upsilon_A)$, $\mathcal{P}, \pi \models \varphi$.

Given a tree formula τ, a valuation υ and a configuration $c \in \mathcal{C}_{\mathcal{P}}$, the satisfiability relation $\mathcal{P}, c, \upsilon \models \tau$ is defined inductively as follows:

- $\mathcal{P}, c, \upsilon \models \tau_1 \vee \tau_2$ iff $\mathcal{P}, c, \upsilon \models \tau_1$ or $\mathcal{P}, c, \upsilon \models \tau_2$;
- $\mathcal{P}, c, \upsilon \models \tau_1 \wedge \tau_2$ iff $\mathcal{P}, c, \upsilon \models \tau_1$ and $\mathcal{P}, c, \upsilon \models \tau_2$;
- $\mathcal{P}, c, \upsilon \models \langle +A \rangle\tau$ iff $\exists \upsilon_A : A \to \Theta, \mathcal{P}, c, \upsilon \otimes \upsilon_A \models \tau$;
- $\mathcal{P}, c, \upsilon \models \langle +A \rangle\varphi$ iff $\exists \upsilon_A : A \to \Theta$, $\forall \pi \in \prod_{\mathcal{P}}(c, \upsilon \otimes \upsilon_A)$, $\mathcal{P}, \pi \models \varphi$,

where, $\upsilon \otimes \upsilon_A$ is the valuation such that for every $a \in Ag$, $(\upsilon \otimes \upsilon_A)(a)$ is $\upsilon_A(a)$ if $a \in A$, otherwise $\upsilon(a)$. The semantics of path formulae is entirely standard, hence is omitted.

It was shown that BSIL is incomparable with ATL* [Wang *et al.*, 2015], while SL[1G] subsumes ATL* [Mogavero *et al.*, 2012]. Therefore, there are some SL[1G] formulae (as well as SL[CG] and SL[DG]) that cannot be expressed in BSIL. On the other hand, it is difficult to construct an equivalent SL[1G] formula for the BSIL formula $\langle\{1\}\rangle((\langle\{+2\}\rangle\mathbf{G}p \wedge \langle\{+2\}\rangle\mathbf{G}q) \vee \langle\{+2\}\rangle\mathbf{G}q')$. We note that, however, this formula can be translated into an SL[CG] formula $\langle x_1\rangle\langle y_1\rangle\langle y_2\rangle((1,x_1)(2,y_1)\mathbf{G}p \wedge (1,x_1)(2,y_2)\mathbf{G}q) \vee \langle x_1\rangle\langle y_3\rangle((1,x_1)(2,y_3)\mathbf{G}q')$, where $Ag = \{1,2\}$. In general, we have the following result.

Theorem 1. *For each BSIL formula ϕ, an equivalent SL[CG] or SL[DG] formula ϕ' can be constructed.*

3.3 Automata for Logic Formulae

In this section, we recall some connection of logic and automata which will be used in our model checking algorithms. First, recall the syntax of LTL:

$$\phi ::= q \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{X}\phi \mid \phi\mathbf{U}\phi.$$

Definition 4. *A* parity automaton $\mathcal{PA}$ *is a tuple* $(G, \Sigma, \delta, g^0, F)$ *where G is a finite set of states, Σ is the input alphabet, $\delta : G \times \Sigma \to 2^G$ is a transition function, $g^0 \in G$ is the initial state and $F : G \to \{0, ..., k\}$ is a rank function assigning each state $g \in G$ a priority $F(g)$, where k is some natural number called index.*

A *run* of $\mathcal{PA}$ over an ω-word $\alpha_0\alpha_1...$ from Σ^ω is a sequence of states $\pi = g_0g_1...$ such that $g_0 = g^0$, and for every $i \geq 0$, $g_{i+1} \in \delta(g_i, \alpha_i)$. Let $inf(\pi)$ be the set of states visited infinitely often in π. A run π is *accepting* iff the smallest number of $\{F(g) \mid g \in inf(\pi)\}$ is *even*. $\mathcal{PA}$ is called *deterministic* if for every $(g, \alpha) \in G \times \Sigma$, $|\delta(g, \alpha)| \leq 1$. The transition function δ in a *deterministic parity automaton* (DPA) is written as $\delta : G \times \Sigma \to G$.

Theorem 2. *[Kupferman and Vardi, 2001; Piterman, 2007] For every LTL formula ϕ, we can construct a DPA with $2^{2^{\mathbf{O}(|\phi|)}}$ states and $2^{\mathbf{O}(|\phi|)}$ indices such that the DPA recognizes all of the ω-words satisfying ϕ.*

Let $BL(\mathbf{X}, \mathbf{U}, \mathbf{R})$ denote the set of all Boolean combinations ($\wedge, \vee$) of the formulae in the forms $\mathbf{X}\phi$, $\phi\mathbf{U}\phi$ and $\phi\mathbf{R}\phi$ such that $\phi ::= q \mid \neg q \mid \phi \wedge \phi \mid \phi \vee \phi$, $q \in AP$.

Definition 5. *A* deterministic Büchi automaton *(DBA) $\mathcal{BA}$ is a DPA $(G, \Sigma, \delta, g^0, F)$ such that $F : G \to \{0, 1\}$. A DBA $\mathcal{BA}$ is called* 1-weak *(1W-DBA for short) if each SCC (strongly connected component) in the transition graph of $\mathcal{BA}$ contains at most one state [Vardi, 1995].*

For each $BL(\mathbf{X}, \mathbf{U}, \mathbf{R})$ formula of the form $\mathbf{X}\phi_1$, $\phi_1\mathbf{U}\phi_2$ or $\phi_1\mathbf{R}\phi_2$, one can construct an equivalent 1W-DBA with at most 2 states. Furthermore, 1W-DBA is closed under intersection and union. Then, we get that:

Proposition 2. *For every $BL(\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{R})$ formula ϕ, we can construct a 1W-DBA with $2^{\mathbf{O}(|\phi|)}$ states recognizing all of the ω-words that satisfy ϕ.*

3.4 The Model Checking Problem

In this work, we consider the *global* model checking problem. Namely, given a PGS $\mathcal{P}$ and a BSIL, SL[CG] or SL[DG] formula ϕ, we compute $\|\phi\|_{\mathcal{P}}$, the set of configurations of $\mathcal{P}$ satisfying ϕ. Note that for the PGS model, we consider *regular valuations* [Esparza *et al.*, 2003], i.e., the labeling function is given as $\mathfrak{l} : AP \to 2^{\mathcal{C}_{\mathcal{P}}}$ such that for every $q \in AP$, $\mathfrak{l}(q)$ is a *regular* set (technically, it is represented by an alternating multi-automaton; see below for definition). The labeling function $\mathfrak{l}$ can be lifted to the function $\lambda_{\mathfrak{l}} : P \times \Gamma^* \to 2^{AP}$: for every $c \in \mathcal{C}_{\mathcal{P}}$, $\lambda_{\mathfrak{l}}(c) = \{q \in AP \mid c \in \mathfrak{l}(q)\}$.

4 Model Checking Algorithms

Our approaches rely crucially on alternating pushdown systems which we first review, as follows.

4.1 Alternating Pushdown Systems

Given a set X, let $\mathcal{B}^+(X)$ denote the set of *positive Boolean formulae* over X. For a set $Y \subseteq X$ and a formula $\psi \in \mathcal{B}^+(X)$, Y satisfies ψ if assigning *true* to elements of Y and assigning *false* to elements of $X \setminus Y$ makes ψ *true*.

Definition 6 (Alternating Pushdown Systems)**.** *An* Alternating Pushdown System (APDS) *is a tuple $\mathcal{P} = (P, \Gamma, \Delta)$, where P is a finite set of control states, Γ is a finite stack alphabet, and $\Delta : P \times \Gamma \to \mathcal{B}^+(P \times \Gamma^*)$ is a transition function that assigns to each element of $P \times \Gamma$ a positive Boolean formula over $P \times \Gamma^*$.*

For every set $\{\langle p_1, \omega_1\rangle, ..., \langle p_n, \omega_n\rangle\} \subseteq P \times \Gamma^*$ and every pair $\langle p, \gamma\rangle \in P \times \Gamma$, if $\{\langle p_1, \omega_1\rangle, ..., \langle p_n, \omega_n\rangle\}$ satisfies the positive Boolean formula $\Delta(p, \gamma)$, we sometimes write $\langle p, \gamma\rangle \longrightarrow_{\mathcal{P}} \{\langle p_1, \omega_1\rangle, ..., \langle p_n, \omega_n\rangle\}$. If $\langle p, \gamma\rangle \longrightarrow_{\mathcal{P}} \{\langle p_1, \omega_1\rangle, ..., \langle p_n, \omega_n\rangle\}$, then $\langle p, \gamma\omega\rangle \Longrightarrow_{\mathcal{P}} \{\langle p_1, \omega_1\omega\rangle, ..., \langle p_n, \omega_n\omega\rangle\}$ for every $\omega \in \Gamma^*$. For every pair $\langle p, \gamma\rangle \in P \times \Gamma$, we suppose in this work that the Boolean formula $\Delta(\langle p, \gamma\rangle)$ is in the *disjunctive normal form*. The size $|\Delta|$ of Δ is defined as $\sum_{(p,\gamma)\in P\times\Gamma} |\Delta(\langle p, \gamma\rangle)|$, where $|\Delta(\langle p, \gamma\rangle)|$ denotes the number of satisfying sets of the Boolean formula $\Delta(\langle p, \gamma\rangle)$.

A *run* ρ of the APDS $\mathcal{P}$ from a configuration $\langle p, \omega\rangle$ is a $\mathcal{C}_{\mathcal{P}}$-labeled tree (T_r, r) such that $r(\epsilon) = \langle p, \omega\rangle$, and for every node $t \in T_r$ with $r(t) = \langle p', \omega'\rangle$ and its children $t0, ..., tn$, it must be the case that $\langle p', \omega'\rangle \Longrightarrow_{\mathcal{P}} \{\langle p'_0, \omega'_0\rangle, ..., \langle p'_n, \omega'_n\rangle\}$ where $r(ti) = \langle p'_i, \omega'_i\rangle$ for every $i : 0 \leq i \leq n$. W.l.o.g., we assume that all of the runs of APDSs are infinite. Given a path π of the run ρ, let $r(\pi)$ be the sequence of configurations along π, and $inf(\pi)$ denote the set of control states appearing infinitely often in $r(\pi)$.

For an APDS, we consider the following acceptance conditions:

- parity: an APDS (P, Γ, Δ) is equipped with a function $F : P \to \{0, ..., k\}$, where k is referred to as the *index*.
- conjunctive parity: in this case, an APDS (P, Γ, Δ) is equipped with $F = \{F_i\}_{i\in[m]}$ such that for every $i \in [m]$, $F_i : P \to \{0, ..., k_i\}$. We call $k = \max\{k_i \mid i \in [m]\}$ as the index of the APDS with the conjunctive parity acceptance condition.

Given a run ρ, a path π in the run ρ is *accepting* if

- parity: the *smallest* number in $\{F(p) \mid p \in inf(\pi)\}$ is *even*.

- conjunctive parity: $\forall i \in [m]$ such that the *smallest* number in $\{F_i(p) \mid p \in inf(\pi)\}$ is *even*.

A run ρ in the APDS $\mathcal{P}$ is *accepting* iff all the paths in ρ are accepting. Let $\mathcal{L}(\mathcal{P})$ denote the set of all configurations from which $\mathcal{P}$ has an accepting run. In the sequel, we usually write APDS-P, and APDS-CP for APDS with parity and conjunctive parity acceptance conditions respectively.

We observe that APDS-CP with $F = \{F_i\}_{i\in[m]}$ can be seen as APDS with $\mathbf{O}(mk)$ Streett pairs [Chatterjee *et al.*, 2007], which can be transformed into APDS-P by using *index appearance records* (e.g. [Gurevich and Harrington, 1982; Schwoon, 2001]).

Theorem 3. *[Gurevich and Harrington, 1982; Schwoon, 2001] Given an APDS-CP $\mathcal{P} = (P, \Gamma, \Delta, F)$ with $F = \{F_i\}_{i\in[m]}$, we can construct an APDS-P $\mathcal{P}' = (P', \Gamma, \Delta', F')$ in $\mathbf{O}(|\Delta|(mk)!)$ time such that $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{P}')$. Moreover, $|P'| = \mathbf{O}(|P|(mk)!)$, $|\Delta'| = \mathbf{O}(|\Delta|(mk)!)$ and the index k' of $\mathcal{P}'$ is $\mathbf{O}(mk)$.*

Alternating Multi-Automata

Definition 7 (Alternating Multi-Automata)**.** *[Bouajjani* et al.*, 1997] Let $\mathcal{P} = (P, \Gamma, \Delta)$ be an APDS. An Alternating Multi-Automaton* (AMA) *is a tuple $\mathcal{M} = (S, \Gamma, \delta, I, S_f)$, where S is a finite set of states with $S \supseteq P$, Γ is an input alphabet, $\delta : (S \times \Gamma) \to \mathcal{B}^+(S)$ is a transition function, $I \subseteq P$ is a finite set of initial states, $S_f \subseteq S$ is a finite set of final states.*

As before, for a set of states $\{s_1, ..., s_n\} \subseteq S$, if $\{s_1, ..., s_n\}$ satisfies $\delta(s, \gamma)$, we will sometimes write $s \xrightarrow{\gamma} \{s_1, ..., s_n\}$ instead. We define the relation $\longrightarrow_\delta \subseteq S \times \Gamma^* \times 2^S$ as the least relation such that the following conditions hold:

- $s \xrightarrow{\epsilon}_\delta \{s\}$ for every $s \in S$;
- $s \xrightarrow{\gamma\omega}_\delta \bigcup_{i\in[n]} S_i$ if $s \xrightarrow{\gamma} \{s_i\}_{i\in[n]}$ and $s_i \xrightarrow{\omega}_\delta S_i$ for every $i \in [n]$.

The AMA $\mathcal{M}$ *accepts* a configuration $\langle p, \omega\rangle$ if there exists $S' \subseteq S_f$ such that $p \xrightarrow{\omega}_\delta S'$ and $p \in I$. Let $\mathcal{L}(\mathcal{M})$ denote the set of all the configurations accepted by $\mathcal{M}$. A set of configurations $C \subseteq \mathcal{C}_\mathcal{P}$ is called *regular* if there exists an AMA $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = C$.

Proposition 3. *[Cachat, 2002] Let $\mathcal{M} = (S, \Gamma, \delta, I, S_f)$ be an AMA. Deciding whether a configuration $\langle p, \omega\rangle$ with $p \in S$ and $\omega \in \Gamma^*$ is accepted by $\mathcal{M}$ or not can be done in $\mathbf{O}(|S| \cdot |\delta| \cdot |\omega|)$ time and $\mathbf{O}(|S|)$ space.*

4.2 SL[CG] and SL[DG] Model Checking

In this section, we consider the problem of model checking SL[CG] and SL[DG]. Thanks to Proposition 1, it is sufficient to consider SL[CG]. Given a PGS $\mathcal{P} = (P, \Gamma, \Delta, \lambda)$ and an SL[CG] formula ϕ, we first deal with the case that ϕ is a *principal sentence* $\psi = \wp(\bigwedge_{i\in[n]} \flat_i\phi_i)$ such that for every $i \in [n]$, ϕ_i is an LTL formula, $\flat_i$ binds all of the agents to some strategy variables, and $\wp$ quantifies all of the strategy variables in $\flat_i$ [Cermák *et al.*, 2015].

To compute $\|\psi\|_\mathcal{P}$, we proceed as follows. First, we construct a DPA $\mathcal{PA}_i = (G_i, \Sigma_i, \delta_i, g_i^0, F_i)$ with index k_i that accepts all the ω-words satisfying ϕ_i, for every $i \in [n]$. Furthermore, for every $q \in AP$, we assume that AMA $\mathcal{M}^q = (S^q, \Gamma, \delta^q, I^q, S^q_f)$ and its complement $\mathcal{M}^{\neg q} = (S^{\neg q}, \Gamma, \delta^{\neg q}, I^{\neg q}, S^{\neg q}_f)$ have been computed. Although the states from P may occur in different AMAs, for our purpose, we assume that each occurrence of $p \in P$ in different $\mathcal{M}^q$ or $\mathcal{M}^{\neg q}$ carries a unique name, for instance, it is decorated by q (resp. $\neg q$), denoted by p^q (resp. $p^{\neg q}$).

Next, we construct an APDS-CP for the SL[CG] principal sentence ψ.

For $\chi \in Ac^{|\wp|}$ and $d \in Dc$, d is said to be *compatible* with χ under $\flat_i$, if for every $a \in Ag$, $\chi(\wp(\flat_i(a))) = d(a)$, where $\wp(\flat_i(a))$ is the position of the variable $\flat_i(a)$ in $\wp$. A *state mapping* f is a *partial* function from $[n]$ to $\bigcup_{i\in[n]} G_i$ such that for every $i \in [n]$, if $f(i)$ is defined, then $f(i) \in G_i$. Let $\mathcal{F}$ be the set of all state mapping functions with $f_0 \in \mathcal{F}$ such that for every $i \in [n]$, $f_0(i) = g_i^0$.

We define APDS-CP $\mathcal{P}_\psi = (P', \Gamma, \Delta', F')$, where

- $P' = (P \times \bigcup_{0\le i\le|\wp|} Ac^i \times \mathcal{F}) \cup \bigcup_{q\in AP} (S^q \cup S^{\neg q})$;
- $F' = \{F'_i\}_{i\in[n]}$, $F'_i : P' \to \{0, ..., k\}$ is the parity objective such that the following conditions hold:
  - $F'_i([p, \chi, f]) = \begin{cases} F_i(f(i)), & \text{if } f(i) \in G_i, \\ 0, & \text{otherwise,} \end{cases}$
  - $F'_i(s) = 0$, for $s \in \bigcup_{q\in AP} S^q \cup S^{\neg q}$;
- Δ' is the smallest transition function satisfying the following constraints: for each $[p, \chi, f] \in P'$, $\gamma \in \Gamma$, and $\alpha \subseteq AP$,

1. $\Delta'([p, \chi, f], \gamma) = \bigvee_{\mathfrak{a}\in Ac} \langle [p, \chi\mathfrak{a}, f], \gamma\rangle$, if $|\chi| < |\wp|$ and $\wp(|\chi| + 1) = \langle x\rangle$ for some $x \in V$;
2. $\Delta'([p, \chi, f], \gamma) = \bigwedge_{\mathfrak{a}\in Ac} \langle [p, \chi\mathfrak{a}, f], \gamma\rangle$, if $|\chi| < |\wp|$ and $\wp(|\chi| + 1) = [x]$ for some $x \in V$;
3. $\Delta'([p, \chi, f], \gamma) =$
$$\bigwedge_{d\in Dc} \langle [p', \epsilon, f'], \omega\rangle \wedge \bigwedge_{q\in\alpha} \langle p^q, \gamma\rangle \wedge \bigwedge_{q\in AP\setminus\alpha} \langle p^{\neg q}, \gamma\rangle,$$
if $|\chi| = |\wp|$, $\Delta(\langle p, \gamma\rangle, d) = \langle p', \omega\rangle$, furthermore, for every $i \in [n]$, $f'(i) = \delta_i(f(i), \alpha)$ if $f(i)$ is defined and d is compatible with χ under $\flat_i$, and $f'(i)$ is undefined otherwise;
4. for every $s \xrightarrow{\gamma} \{s_1, ..., s_m\} \in \bigcup_{q\in AP} \delta^q \cup \delta^{\neg q}$, $\Delta'(\langle s, \gamma\rangle) = \bigwedge_{i\in[m]} \langle s_i, \epsilon\rangle$;
5. for every $s \in \bigcup_{q\in AP} S^q_f \cup S^{\neg q}_f$, $\Delta'(\langle s, \bot\rangle) = \langle s, \bot\rangle$.

Theorem 4. *For every configuration $\langle p, \omega\rangle \in \mathcal{C}_\mathcal{P}$, $\langle p, \omega\rangle \in \|\psi\|_\mathcal{P}$ iff $\langle [p, \epsilon, f_0], \omega\rangle \in \mathcal{L}(\mathcal{P}_\psi)$. The size of $\mathcal{P}_\psi$ is doubly-exponential of ψ and polynomial of $\mathcal{P}$ and AMAs $\mathcal{M}^q$, where $\psi = \wp(\bigwedge_{i\in[n]} \flat_i\phi_i)$.*

Theorem 5. *[Hague and Ong, 2009] For an APDS-P $\mathcal{P} = (P, \Gamma, \Delta, F)$, an AMA $\mathcal{M}$ with $\mathbf{O}(|P|)$ states and $\mathbf{O}(|P| \cdot |\Gamma| \cdot$*

$2^{|P|}$) *transition rules can be computed in* $2^{\mathbf{O}(k|P|)}$ *time such that* $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{P})$*, where* k *is the index of* $\mathcal{P}$.

Applying Theorem 3 and Theorem 5, we get that:

Corollary 1. *For an APDS-CP* $\mathcal{P} = (P, \Gamma, \Delta, \{F_i\}_{i\in[m]})$ *with index* k*, an AMA* $\mathcal{M}$ *with* $\mathbf{O}(|P|(mk)!)$ *states and* $\mathbf{O}((mk)!|P|\cdot|\Gamma|\cdot 2^{|P|(mk)!})$ *transition rules can be computed in* $2^{\mathbf{O}(|P|(mk)!)}$ *time such that* $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{P})$.

For each principal sentence ψ, we can construct an AMA $\mathcal{M}_\psi$ in triply-exponential time of ψ and exponential time of $\mathcal{P}$ and the AMAs $\mathcal{M}^q$ such that $\mathcal{L}(\mathcal{M}_\psi) = \|\psi\|_{\mathcal{P}}$. Moreover, the number of states of $\mathcal{M}_\psi$ is doubly-exponential of the size of the principal sentence and polynomial of the size of the PGS and that of the AMAs $\mathcal{M}^q$.

For any general SL[CG] formula ϕ, a formula ϕ' can be constructed by replacing simultaneously all the subformulae ψ of ϕ that are principal sentences by some fresh atomic propositions q_ψ and extending the labeling function λ by $\lambda(q_\psi) = \mathcal{L}(\mathcal{M}_\psi)$. We then construct AMAs for the principal sentences in ϕ'. Iteratively applying this procedure at most $|\phi|$ times, we can get an AMA $\mathcal{M}_\phi$ such that $\mathcal{L}(\mathcal{M}_\phi) = \|\phi\|_{\mathcal{P}}$.

The lower bound follows from the fact that SL[1G] subsumes ATL* [Laroussinie *et al.*, 2008; Cermák *et al.*, 2015] and the model checking problem for ATL* on PGSs is 3EXPTIME-complete [Chen *et al.*, 2016].

Corollary 2. *The model checking problems for SL[1G], SL[CG] and SL[DG] on PGSs are 3EXPTIME-complete.*

Remark 1. *Our construction relies crucially on the behavioral strategies, which SL[CG]/SL[DG] enjoys according to [Mogavero* et al.*, 2013]. The main reason is that for a fragment of SL, the strategy quantifications can be replaced by action quantifications (which is the case in our construction) only if it admits behavioral strategies. For instance, although it is tempting to think that our construction can be naturally extended to SL[BG], the obvious extension would be incomplete (due to the non-behavioral strategies). This can be illustrated by using the example in Fig. 3 from [Mogavero* et al.*, 2013]. We will make this point explicit in the next version.*

4.3 BSIL Model Checking

In this section, we turn to BSIL. Let us fix a PGS $\mathcal{P} = (P, \Gamma, \Delta, \lambda)$ and a BSIL formula ψ. In case that ψ is a Boolean combination of the formulae of the form $\langle A\rangle\tau$ or $\langle A\rangle\varphi$, an AMA can be computed via Boolean operations on AMAs. Hence, we will focus on the case that $\psi = \langle A\rangle\tau$ or $\langle A\rangle\varphi$ and assume furthermore that each proper sub-state formula of τ or φ has been replaced by a fresh atomic proposition with an associated AMA. Such formulae are called *simple* BSIL formulae.

Recall that Theorem 1 translates a BSIL formula to an equivalent SL[CG] formula. We can apply this translation to a simple BSIL formula ψ, obtaining an SL[CG] formula ψ' which is of the form $\bigvee_{i\in[k]}\langle X_i\rangle[Y_i]\bigwedge_{j\in[l_i]}\flat_{i,j}\psi_{i,j}$, where X_i, Y_i are sets of strategy variables, each $\flat_{i,j}$ is a binding prefix, and each $\psi_{i,j}$ is a $BL(\mathbf{X}, \mathbf{U}, \mathbf{R})$ formula. Moreover, $k = \mathbf{O}(2^{|\psi|})$ and $l_i = \mathbf{O}(|\psi|)$ for every $i \in [k]$.

For each disjunct $\xi = \langle X_i\rangle[Y_i]\bigwedge_{j\in[l_i]}\flat_{i,j}\psi_{i,j}$, we then apply the construction for SL[CG] to obtain an AMA capturing $\|\xi\|_{\mathcal{P}}$. However, we observe that, in this case, since $\psi_{i,j}$ is a $BL(\mathbf{X}, \mathbf{U}, \mathbf{R})$ formula, instead of a fully-fledged LTL formula, we can use 1W-DBA instead of DPA (cf. Section 3.3) which only incurs a *singly exponential* blow-up. As a result, we are able to compute an AMA $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = \|\xi\|_{\mathcal{P}}$, the number of states of $\mathcal{M}$ is *polynomial* in the size of $\mathcal{P}$ and exponential in the size of ξ.

It is then not difficult to obtain an AMA $\mathcal{M}'$ for ψ', i.e., $\mathcal{L}(\mathcal{M}') = \|\psi'\|_{\mathcal{P}}$. Note that here although ψ' may contain exponentially many disjuncts, the number of states of $\mathcal{M}'$ is still polynomial in the size of $\mathcal{P}$ and exponential in the size of ψ'. This result, in conjunction with Proposition 3, yields the main result of this section:

Theorem 6. *The model checking problem for BSIL is 2EXPTIME-complete, and EXPTIME-complete for fixed formulae.*

Since BSIL subsumes ATL, and the model checking problem for ATL on PGSs is EXPTIME-complete for fixed ATL formulae [Chen *et al.*, 2016], we conclude that the model checking problem for BSIL is EXPTIME-hard for fixed formulae. The 2EXPTIME-hardness for non-fixed formulae is obtained by a reduction from the word problem of EXPSPACE-bounded alternating Turing machines $\mathcal{T}$. We can construct, in polynomial time, a PGS $\mathcal{P}$ with two agents E and A simulating the existential moves of $\mathcal{T}$ and the universal moves of $\mathcal{T}$, respectively. The plays of $\mathcal{P}$ have two phases. In the first phase, $\mathcal{P}$ guesses a computation tree of $\mathcal{T}$ over the input word and pushes the guessed symbols in each path of the computation tree into the stack. In the second phase, $\mathcal{P}$ pops up the content of the stack, and checks at the same time that the sequence of symbols stored in the stack is indeed an encoding of a valid computation path of $\mathcal{T}$, with the help of some BSIL formula. (The further details will be provided in the full version of the paper.)

5 Conclusion

In this paper, we have investigated model checking algorithms for variants of strategy logic over PGSs.

We showed that the model checking problems on PGSs for SL[CG], SL[DG] and SL[1G] are 3EXTIME-complete, while for BSIL is 2EXPTIME-complete. Future work includes implementation, extension to SL[AG] and games with imperfect recall or partial information.

6 Acknowledgments

Taolue Chen is partially supported by the ARC Discovery Project (DP160101652), the Singapore Ministry of Education AcRF Tier 2 grant (MOE2015-T2-1-137), and an oversea grant from the State Key Laboratory of Novel Software Technology, Nanjing Unviersity. Fu Song is partially supported by Shanghai Pujiang Program (No. 14PJ1403200), Shanghai ChenGuang Program (No. 13CG21), and NSFC Projects (No. 61402179, 61532019 and 91418203). Zhilin Wu is partially supported by the NSFC projects (No. 61272135, 61472474, and 61572478).

References

[Alur *et al.*, 2002] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[Aminof *et al.*, 2013] Benjamin Aminof, Axel Legay, Aniello Murano, Olivier Serre, and Moshe Y. Vardi. Pushdown module checking with imperfect information. *Inf. Comput.*, 223:1–17, 2013.

[Bouajjani *et al.*, 1997] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability Analysis of Pushdown Automata: Application to Model Checking. In *CONCUR*, 1997.

[Bourahla and Benmohamed, 2005] Mustapha Bourahla and Mohamed Benmohamed. Model checking multi-agent systems. *Informatica (Slovenia)*, 29(2):189–198, 2005.

[Bouyer *et al.*, 2015] Patricia Bouyer, Patrick Gardy, and Nicolas Markey. Weighted strategy logic with boolean goals over one-counter games. In *FSTTCS*, 2015.

[Bozzelli *et al.*, 2010] Laura Bozzelli, Aniello Murano, and Adriano Peron. Pushdown module checking. *Formal Methods in System Design*, 36(1):65–95, 2010.

[Bulling and Jamroga, 2011] Nils Bulling and Wojciech Jamroga. Alternating epistemic mu-calculus. In *IJCAI*, 2011.

[Cachat, 2002] Thierry Cachat. Symbolic strategy synthesis for games on pushdown graphs. In *ICALP*, 2002.

[Carayol and Hague, 2014] Arnaud Carayol and Matthew Hague. Saturation algorithms for model-checking pushdown systems. In *AFL*, 2014.

[Cermák *et al.*, 2015] Petr Cermák, Alessio Lomuscio, and Aniello Murano. Verifying and synthesising multi-agent systems against one-goal strategy logic specifications. In *AAAI*, 2015.

[Chatterjee *et al.*, 2007] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Generalized parity games. In *FOSSACS*, 2007.

[Chatterjee *et al.*, 2010] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Inf. Comput.*, 208(6):677–693, 2010.

[Chen *et al.*, 2016] Taolue Chen, Fu Song, and Zhilin Wu. Global model checking on pushdown multi-agent systems. In *AAAI*, 2016.

[Chockler *et al.*, 2006] Hana Chockler, Orna Kupferman, and Moshe Y. Vardi. Coverage metrics for temporal logic model checking. *Formal Methods in System Design*, 28(3):189–212, 2006.

[Esparza *et al.*, 2003] Javier Esparza, Antonín Kucera, and Stefan Schwoon. Model checking LTL with regular valuations for pushdown systems. *Inf. Comput.*, 186(2):355–376, 2003.

[Gurevich and Harrington, 1982] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, 1982.

[Hague and Ong, 2009] Matthew Hague and C.-H. Luke Ong. Winning regions of pushdown parity games: A saturation method. In *CONCUR*, 2009.

[Jamroga and Murano, 2014] Wojciech Jamroga and Aniello Murano. On module checking and strategies. In *AAMAS*, 2014.

[Jamroga and Murano, 2015] Wojciech Jamroga and Aniello Murano. Module checking of strategic ability. In *AAMAS*, 2015.

[Kupferman and Vardi, 2001] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001.

[Laroussinie *et al.*, 2008] François Laroussinie, Nicolas Markey, and Ghassan Oreiby. On the expressiveness and complexity of ATL. *Logical Methods in Computer Science*, 4(2), 2008.

[Löding *et al.*, 2004] Christof Löding, P. Madhusudan, and Olivier Serre. Visibly pushdown games. In *FSTTCS*, 2004.

[Mogavero *et al.*, 2012] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. What makes ATL* decidable? A decidable fragment of strategy logic. In *CONCUR*, 2012.

[Mogavero *et al.*, 2013] Fabio Mogavero, Aniello Murano, and Luigi Sauro. On the boundary of behavioral strategies. In *LICS*, 2013.

[Mogavero *et al.*, 2014] Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Log.*, 15(4):34:1–34:47, 2014.

[Murano and Perelli, 2015] Aniello Murano and Giuseppe Perelli. Pushdown multi-agent system verification. In *IJCAI*, 2015.

[Piterman and Vardi, 2004] Nir Piterman and Moshe Y. Vardi. Global model-checking of infinite-state systems. In *CAV*, 2004.

[Piterman, 2007] Nir Piterman. From nondeterministic büchi and streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007.

[Schwoon, 2001] Stefan Schwoon. Determinization and complementation of streett automata. In *Automata, Logics, and Infinite Games*, pages 79–91, 2001.

[Serre, 2003] Olivier Serre. Note on winning positions on pushdown games with [omega]-regular conditions. *Inf. Process. Lett.*, 85(6):285–291, 2003.

[Vardi, 1995] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, 1995.

[Walukiewicz, 2001] Igor Walukiewicz. Pushdown processes: Games and model-checking. *Inf. Comput.*, 164(2):234–263, 2001.

[Wang *et al.*, 2015] Farn Wang, Sven Schewe, and Chung-Hao Huang. An extension of ATL with strategy interaction. *ACM Trans. Program. Lang. Syst.*, 37(3):9, 2015.