ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-982

Strategy Representation and Reasoning in the Situation Calculus

Liping Xiong and **Yongmei Liu**[1]

Abstract. Strategy representation and reasoning has been one of the most active research areas in AI and multi-agent systems. Representative strategic logics are ATL and the more expressive Strategy Logic SL which reasons about strategies explicitly. In this paper, by a simple extension of the situation calculus with a strategy sort, we develop a general framework for strategy representation and reasoning for complete information games. This framework can be used to compactly represent both concurrent and turn-based possibly infinite game structures, specify the internal structure of strategies, reason about strategies explicitly, and reason about strategic abilities of coalitions under commitments to strategy specifications. We show that our framework is strictly more expressive than SL, and inspired by the work of De Giacomo *et al.* on bounded action theories, give a decidable fragment of our framework.

1 INTRODUCTION

Strategy representation and reasoning has been one of the most active research areas in AI and multi-agent systems, and many strategic logics have been proposed recently. From the representation side, the following are some desiderata of a strategic logic: modeling strategic abilities of coalitions; representation and reasoning about strategies explicitly and even the internal structures of strategies; capability to deal with both concurrent games and turn-based games; compact representation of game structures, even infinite ones. Yet hardly any existing strategic logic has all these desiderata.

Most strategic logics are built upon Alternating-time Temporal Logic (ATL) [2] where formula $\langle\langle A\rangle\rangle\varphi$ expresses that coalition A can ensure temporal formula φ holds no matter what the other agents do. However, strategies are treated implicitly in ATL.

To model strategies explicitly, there are mainly two approaches. The first approach is to treat a strategy as an explicit first-order object in which a strategy is a function from states (or sequences of states) to actions [43, 45, 6, 30]. In particular, based on the work of [6] which introduces first-order quantifications over strategies, Mogavero $et\ al.$ [30] propose Strategy Logic SL, which is a very expressive logic for strategic reasoning and strictly contains ATL*, an extension of ATL. Yet this approach cannot model the internal structure of strategies. The second approach is to treat a strategy as a program so that program connectives can be used to obtain combined strategies from simple ones [44, 33, 41].

Nonetheless, the above strategic logics represent game structures with concrete game models which suffer from the state explosion problem. For example, in the Chess game, there are almost 10^{30} states. To model games compactly, the Game Description Language (GDL) has been proposed as a practical language for encoding the rules of arbitrary finite games [16]. Based on GDL, Zhang and Thielscher [47, 48] use propositional modal logic formulas to represent strategies, and introduce prioritised connectives for combining strategies. However, their works can only model turn-based games, and cannot model strategic abilities of coalitions.

Other than modal logics, another main family of logics in AI is action formalisms. A prominent example of action formalisms is the situation calculus [34], which is a first-order language with some second-order ingredients suitable for reasoning about action and change. Based on the situation calculus, a logic programming language Golog [22] has been designed for high-level agent control. There have been a few works [37, 17, 10] studying strategic reasoning in the situation calculus. However, all these works deal with turn-based games. The first work does not support ATL-like reasoning, the second one focuses on the coordination problem, and the third one does not support explicit reasoning about strategies.

In this paper, we propose a framework based on the situation calculus for representation and reasoning about strategies for complete information games with all the four desiderata. We first propose a simple extension of the situation calculus with a strategy sort, which can be used to compactly represent both concurrent and turn-based possibly infinite game structures, and to reason about strategies explicitly. We show that SL can be embedded into the extended situation calculus. Then we use a simple fragment of Golog as a strategy specification language, and define the strategy verification and synthesis problems. We illustrate our logical framework with examples of both turn-based and concurrent games. Finally, inspired by the work of De Giacomo *et al.* on bounded action theories [8], we give a decidable fragment of our extended situation calculus.

2 PRELIMINARIES

In this section, we first introduce the situation calculus, and then review the syntax and semantics of Strategy Logic.

2.1 The situation calculus

The *situation calculus* [34] is a many-sorted first-order logic language (with some second-order elements) specifically designed for representing dynamically changing worlds. There are three disjoint sorts: *situation* for situations, *action* for actions, and *object* for everything else. Intuitively, a situation is a finite sequence of actions. In this language, the constant S_0 is used to denote the initial situation; the binary function $do(a, s)$ is used to denote the successor situation of s resulting from performing action a, and $do([a_1, a_2, ..., a_k], s)$ is used as a shorthand for $do(a_k, \ldots, do(a_2, do(a_1, s)))$; the binary predicate $Poss(a, s)$ means that action a is possible in situation s. Actions can be parameterized, *e.g.*, $repair(r, x)$ represents robot r repairing object x. There are relational and functional fluents whose

[1] Department of Computer Science, Sun Yat-sen University, China, email: xionglp3@mail2.sysu.edu.cn & ymliu@mail.sysu.edu.cn.

values vary from situation to situation. These fluents are denoted by symbols that take a situation term as their last argument. There are also situation-independent predicates and functions. Finally, there is a binary predicate $\sqsubset$ on situations: $s \sqsubset s'$ means that s' is the result of some sequence of actions being performed in s. We use $s \sqsubseteq s'$ as a shorthand for $s \sqsubset s' \vee s = s'$. We say that a situation s is executable if it is possible to perform the actions in s one by one:

$$Exec(s) \doteq \forall a, s'.do(a, s') \sqsubseteq s \supset Poss(a, s').$$

In this language, an application domain is specified by a basic action theory (BAT) $\mathcal{D}$ consisting of five disjoint parts:

1. Σ, the foundational axioms of the situation calculus:
   - $\forall P.P(S_0) \wedge \forall a, s[P(s) \supset P(do(a, s))] \supset (\forall s)P(s)$,
   - $do(a, s) = do(a', s') \supset a = a' \wedge s = s'$,
   - $\neg s \sqsubset S_0 \wedge (s \sqsubset do(a, s') \equiv s \sqsubseteq s')$.
2. $\mathcal{D}_{ap}$, a precondition axiom for each action function specifying when the action can be legally performed.
3. $\mathcal{D}_{ss}$, a successor state axiom for each fluent which describes how fluent values change between situations.
4. $\mathcal{D}_{una}$, unique name axioms for actions.
5. $\mathcal{D}_{S_0}$, axioms describing the initial situation S_0.

2.2 Strategy logic (SL)

Strategy Logic [30] is a very expressive logic for reasoning explicitly about strategies in multi-agent concurrent systems; it strictly contains ATL*, and can express the existence of deterministic multi-player Nash equilibria that cannot be expressed in ATL*.

We fix an SL signature $\langle \text{AP}, \text{AG}, Var \rangle$, here AP is a finite non-empty set of atoms, $\text{AG} = \{1, \ldots, n\}$ is a finite non-empty set of agents, and Var is a countable set of variables.

Definition 1 *SL formulas are built inductively as follows:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathcal{U} \varphi \mid \langle\!\langle x \rangle\!\rangle \varphi \mid (i, x)\varphi,$$

where $p \in$ AP, $i \in$ AG, and $x \in Var$.

Syntactically, SL extends linear-time temporal logic LTL [32] ($\bigcirc$ means *next*, $\mathcal{U}$ means *until*) with two operators. Intuitively, $\langle\!\langle x \rangle\!\rangle$ and (i, x) mean *"there exists a strategy x"*, and *"bind agent i to the strategy associated with variable x"* respectively. We use $\top$ (resp. $\bot$) to represent true (resp. false). We use the universal quantifier $[\![x]\!]$ as the dual of $\langle\!\langle x \rangle\!\rangle$, i.e., $[\![x]\!] = \neg\langle\!\langle x \rangle\!\rangle\neg$, which means *"for all strategies x"*. For a formula φ, we let $free(\varphi)$ denote the set of *free agents and variables* of φ, and we omit its formal definition here. For example, $free(\langle\!\langle x \rangle\!\rangle(1, x)(2, y) \bigcirc p) = \{y\} \cup (\text{AG} - \{1, 2\})$.

Like ATL, the semantics of SL is based on the notion of concurrent game structures.

Definition 2 *A concurrent game structure (CGS) is a tuple $\mathcal{G} = \langle AC, W, \lambda, \tau, w^0 \rangle$, where AC and W are finite non-empty sets of actions and states respectively; $w^0 \in W$ is a designated initial state; $\lambda : W \to 2^{AP}$ is a labeling function; and $\tau : W \times AC^{AG} \to W$ is a transition function mapping a state and a decision (i.e., a function from AG to AC) to a new state. We also denote AC^{AG} as DC.*

A CGS can be viewed as a multi-player game where players perform concurrent actions strategically.

To define the semantics of SL, we begin with some definitions and notations. A track h in a CGS $\mathcal{G}$ is a finite state sequence $w_0 w_1 \ldots w_k$ in W s.t. for all i, $0 \le i < k$, there exists $d \in DC$ s.t. $w_{i+1} = \tau(w_i, d)$. We let $Trk(\mathcal{G})$ denote the set of all tracks in $\mathcal{G}$ beginning from the initial state w^0.

A strategy in $\mathcal{G}$ is a function $f : Trk(\mathcal{G}) \to$ AC. Let $Str(\mathcal{G})$ denote the set of all strategies in $\mathcal{G}$. Intuitively, a strategy is a plan for an agent which contains the choice of action for any track starting from the initial state.

Like a variable assignment in first-order logic, a strategy assignment is a partial function $\chi : \text{AG} \cup \text{Var} \rightharpoonup Str(\mathcal{G})$, mapping agents and variables to strategies. We use $dom(\chi)$ to denote the domain of χ. If $\text{AG} \subseteq dom(\chi)$, χ is called complete. We use $\chi[x/f]$ to denote an assignment exactly like χ except that it maps x to f.

Given a complete strategy assignment χ, it determines a unique infinite state sequence $w_0 w_1 w_2 \ldots$ and a unique infinite decision sequence $d^1 d^2 \ldots$ as follows: $w_0 = w^0$, and for each $j \ge 1$, d^j is the decision associated to the track $w_0 \ldots w_{j-1}$, *i.e.*, for each $i \in$ AG, $d^j(i) = \chi(i)(w_0 \ldots w_{j-1})$, and $w_j = \tau(w_{j-1}, d^j)$. We use $l(\chi)$ to denote this infinite state sequence, and let $l_j(\chi)$ denote the j-th state on the sequence.

Definition 3 *Given a CGS $\mathcal{G} = \langle AC, W, \lambda, \tau, w^0 \rangle$, an SL formula φ, a complete strategy assignment χ with $free(\varphi) \subseteq dom(\chi)$, and a state $w = l_k(\chi)$ for some $k \ge 0$, the relation $\mathcal{G}, w, \chi \models \varphi$ is inductively defined as follows:*

- $\mathcal{G}, w, \chi \models p$ *iff* $p \in \lambda(w)$;
- $\mathcal{G}, w, \chi \models \neg\varphi$ *iff* $\mathcal{G}, w, \chi \not\models \varphi$;
- $\mathcal{G}, w, \chi \models \varphi_1 \wedge \varphi_2$ *iff* $\mathcal{G}, w, \chi \models \varphi_1$ *and* $\mathcal{G}, w, \chi \models \varphi_2$;
- $\mathcal{G}, w, \chi \models \bigcirc\varphi$ *iff* $\mathcal{G}, l_{k+1}(\chi), \chi \models \varphi$;
- $\mathcal{G}, w, \chi \models \varphi_1 \mathcal{U} \varphi_2$ *iff there is an index $k' \in \mathbb{N}$ with $k \le k'$ such that $\mathcal{G}, l_{k'}(\chi), \chi \models \varphi_2$ and, for all indexes $j \in \mathbb{N}$ with $k \le j < k'$, it holds that $\mathcal{G}, l_j(\chi), \chi \models \varphi_1$;*
- $\mathcal{G}, w, \chi \models \langle\!\langle x \rangle\!\rangle \varphi$ *iff there exists a strategy $f \in Str(\mathcal{G})$ such that $\mathcal{G}, w, \chi[x/f] \models \varphi$;*
- $\mathcal{G}, w, \chi \models (i, x)\varphi$ *iff* $\mathcal{G}, w, \chi[i/\chi(x)] \models \varphi$.

An SL formula φ is called a sentence if $free(\varphi)$ is empty. Clearly, for a sentence φ, whether $\mathcal{G}, w, \chi \models \varphi$ does not depend on χ, hence we omit χ. An SL sentence φ is valid if for any CGS $\mathcal{G}$, we have $\mathcal{G}, w^0 \models \varphi$. Denote $\mathcal{G}, w^0 \models \varphi$ as $\mathcal{G} \models \varphi$.

Example 1 The following are some SL sentences:

1. $\varphi_1 : \langle\!\langle x \rangle\!\rangle [\![y]\!](1, x)(2, x)(3, y) \bigcirc p$, which intuitively says that agents 1 and 2 can share a strategy to ensure $\bigcirc p$ no matter what strategy agent 3 takes;
2. $\varphi_2 : \langle\!\langle x \rangle\!\rangle [\![y]\!] \langle\!\langle z \rangle\!\rangle (1, x)(2, y)(3, z) \bigcirc p$;
3. $\varphi_3 : \langle\!\langle x \rangle\!\rangle [\![y]\!](1, x)(2, y) \bigcirc p \supset \langle\!\langle x \rangle\!\rangle \langle\!\langle y \rangle\!\rangle (1, x)(2, y) \bigcirc p$ is valid.

Finally, we introduce an important property of SL that is invariant under local isomorphism, defined as follows:

Definition 4 *[28] Let $\mathcal{G}_1 = \langle AC_1, W_1, \lambda_1, \tau_1, w_1^0 \rangle$ and $\mathcal{G}_2 = \langle AC_2, W_2, \lambda_2, \tau_2, w_2^0 \rangle$ be two CGSs. Then $\mathcal{G}_1$ and $\mathcal{G}_2$ are locally isomorphic iff there is a relation $\sim \subseteq W_1 \times W_2$, and a function $g : \sim \to 2^{AC_1 \times AC_2}$ s.t. the following hold:*

1. $w_1^0 \sim w_2^0$;
2. *for all $w_1 \in W_1$ and $w_2 \in W_2$, if $w_1 \sim w_2$ then*
   (a) $\lambda_1(w_1) = \lambda_2(w_2)$;
   (b) for all $a_1 \in AC_1$, there is $a_2 \in AC_2$ s.t. $(a_1, a_2) \in g(w_1, w_2)$;
   (c) for all $a_2 \in AC_2$, there is $a_1 \in AC_1$ s.t. $(a_1, a_2) \in g(w_1, w_2)$;
   (d) for all $(d^1, d^2) \in \hat{g}(w_1, w_2)$, it holds that $\tau_1(w_1, d^1) \sim \tau_2(w_2, d^2)$, where $\hat{g} :\sim\to 2^{DC_1 \times DC_2}$ mapping pairs of states in $\sim$ to relations between decisions such that $(d^1, d^2) \in \hat{g}(w_1, w_2)$ iff, for all $i \in AG$, it holds that $(d^1(i), d^2(i)) \in g(w_1, w_2)$.
3. *$\sim \cap (\{\tau_1(w_1, d) : d \in DC_1\} \times \{\tau_2(w_2, d) : d \in DC_2\})$ is a bijective function, for all $w_1 \in W_1$ and $w_2 \in W_2$ with $w_1 \sim w_2$.*

Theorem 1 *[28] For SL, it is invariant under local-isomorphism. That is, if two CGSs $\mathcal{G}_1$ and $\mathcal{G}_2$ are locally isomorphic, then for any SL sentence φ, $\mathcal{G}_1 \models \varphi$ iff $\mathcal{G}_2 \models \varphi$.*

3 AN EXTENSION OF THE SITUATION CALCULUS

In this section, we present a simple extension of the situation calculus with a strategy sort, denote as $\mathcal{L}_{ext}$, which can be used to compactly represent both concurrent and turn-based possibly infinite game structures, and to reason about strategies explicitly.

We fix a set of agents AG $= \{1, \ldots, n\}$. We introduce two additional sorts: a sort for *joint actions* and a second-order sort for *strategies*. Intuitively, a joint action is an n-ary vector of actions, one action for each agent. A strategy is a function from situations to actions. Let $A \subseteq$ AG. A collective strategy of coalition A is a function from A to strategies. A joint strategy is a collective strategy of AG. We use variables d, d', ... for joint actions, g, g', ... for strategies, g_A, g'_A, ... for collective strategies of coalition A, and g_{all}, g'_{all}, ... for joint strategies. We treat g_A the same as the set of strategy variables $\{g_i \mid i \in A\}$.

We introduce a function $joint(a_1, \ldots, a_n)$ which maps n actions into a joint action, and n projection functions $pr_i(d)$, $1 \leq i \leq n$, which maps a joint action into its i-th component. For simplicity, we write $joint(a_1, \ldots, a_n)$ as $\langle a_1, \ldots, a_n \rangle$, and write $pr_i(d)$ as d_i. Situations are now sequences of joint actions; so the first argument of the function do and the predicate $Poss$ is of the joint action sort.

The set Σ of foundational axioms for situations is the same as before except that we replace each action variable with a joint action variable. We also add to Σ the following concerning joint actions:

- $\forall d \exists a_1, \ldots, a_n . d = \langle a_1, \ldots, a_n \rangle$;
- $\langle a_1, \ldots, a_n \rangle = \langle a'_1, \ldots, a'_n \rangle \supset a_1 = a'_1 \wedge \ldots \wedge a_n = a'_n$;
- $pr_i(\langle a_1, \ldots, a_n \rangle) = a_i, i = 1, \ldots, n$.

Reiter [34] presents an account of true concurrency where a concurrent action is modeled as a possibly infinite set of simple actions. Our account of concurrent actions as joint actions can be considered as a special case of Reiter's account. ConGolog [7] is an extension of Golog with a rich account of interleaved concurrency. In contrast to interleaved concurrency, our framework is able to deal with the issues of action precondition interaction and concurrent effect specification, which are discussed in [34]. Two simple actions may each be possible, their preconditions may be jointly consistent, yet intuitively they should not be concurrently possible, *e.g.*, each of two robots can walk through a narrow door, but they cannot walk through the door at the same time; also, the effect of concurrently executing two simple actions, *e.g.*, lifting the two ends of a table, might be different from executing the two actions in sequence. Both examples can be specified in our framework.

In the absence of the precondition interaction problem, we introduce n predicates $Poss_i(a, s)$, meaning that it is possible for agent i to perform action a in situation s, and let $Poss(\langle a_1, ..., a_n \rangle, s) \equiv \bigwedge_{i=1}^{n} Poss_i(a_i, s)$. To represent turn-based games, we introduce an action nop meaning doing nothing, and n fluents $turn_i(s)$, $i = 1, \ldots, n$, meaning that it's agent i's turn to make a move. We have $Poss_i(nop, s) \equiv \neg turn_i(s)$.

Let g_A be a collective strategy of coalition A. The abbreviation $s \sqsubseteq_{g_A} s'$ is used to represent the formula

$$s \sqsubseteq s' \wedge \forall s'' \forall d[s \sqsubset do(d, s'') \sqsubseteq s' \supset \textstyle\bigwedge_{i \in A} d_i = g_i(s'')].$$

Intuitively, this means that s is a subhistory of s', and on the way from s to s', each agent i in A performs actions according to strategy g_i. Further, we introduce the abbreviation:

$$s \leq_{g_A} s' \doteq s \sqsubseteq_{g_A} s' \wedge Exec(s').$$

We remark that our idea of extending the situation calculus with joint actions and strategies originates from the literature. To reason about general games, Schiffel and Thielscher [35, 36] introduce joint actions into the situation calculus. To study ability and knowing how in the situation calculus, Lespérance *et al.* [19] introduce strategies, which they call *action choice functions*. Yet De Giacomo *et al.* [11] present another approach to modeling concurrent game structures in the situation calculus. They introduce a subsort $move$ of the $object$ sort, representing agents' moves. They assume only one action function $tick(m_1, ..., m_n)$, representing that each agent i performs move m_i simultaneously.

To illustrate the expressiveness of our language for representing strategic properties, we adapt an example from [30] concerning the existence of deterministic multi-player Nash equilibria:

Example 2 Consider two agents: agent 1 has the temporal goal $\diamond p$ ("eventually p"), and agent 2 has the temporal goal $\diamond q$. Then, we can express the existence of a strategy profile (g_1, g_2) that is a Nash equilibrium for the two agents wrt their goals by the following sentence:

$$\begin{aligned}\exists g_1, g_2.\{(\exists g)(\exists s)[p(s) \wedge S_0 \leq_{(g,g_2)} s] \supset \\ (\exists s)[p(s) \wedge S_0 \leq_{(g_1,g_2)} s]\} \wedge \\ \{(\exists g)(\exists s)[q(s) \wedge S_0 \leq_{(g_1,g)} s] \supset \\ (\exists s)[q(s) \wedge S_0 \leq_{(g_1,g_2)} s]\}\end{aligned}$$

Informally, this asserts that each agent has the "best" strategy once the strategy of the other agent has been fixed.

In the following, we illustrate with examples the expressiveness of our situation calculus language for compactly representing possibly infinite games.

Example 3 Consider the Chomp game [31]. As shown in Figure 1(a), cookies are laid out on a $m \times n$ or $\omega \times \omega$ grid, here ω is the least infinite ordinal. The cookie in the top left position (0,0) is poisoned. Two players take turns making moves: at each move, a player is required to eat a remaining cookie, together with all cookies to the right and/or below it. The loser is the player who has no choice but to eat the poisoned cookie.

Consider representing the Chomp game with concurrent game structures. The ordinal version cannot be represented with a CGS. A CGS representation of an $m \times m$ game would use m^2 actions and at least 2^m states. Hence a CGS representation of the game needs exponential space.

We now formalize this game in the situation calculus. Fluent $ck(j, k, s)$ means that there is a cookie at position (j, k). Action $eat(j, k)$ means eating the cookie at position (j, k) together with all cookies to the right and/or below it. We use $size(m, n, s)$ to denote the formula $\forall j, k.ck(j, k, s) \equiv j < m \wedge k < n$, meaning that the current cookies form an $m \times n$ grid. In addition to the second-order axiomatization of Peano arithmetic, we have the following axioms:

$$\begin{aligned}&Poss_i(eat(j, k), s) \equiv turn_i(s) \wedge ck(j, k, s), i = 1, 2;\\ &turn_i(do(d, s)) \equiv \neg turn_i(s), i = 1, 2;\\ &ck(j, k, do(d, s)) \equiv ck(j, k, s) \wedge \neg \exists j', k'.[(j' \leq j\\ &\qquad \vee k' \leq k) \wedge (d_1 = eat(j', k') \vee d_2 = eat(j', k'))];\\ &turn_1(S_0) \wedge \neg turn_2(S_0);\\ &(\forall j, k) ck(j, k, S_0) \vee (\exists m, n) size(m, n, S_0).\end{aligned}$$

We use $\mathcal{D}_{ch}$ to denote the BAT of Chomp, and we use $\mathcal{D}_{mm}$ (resp. $\mathcal{D}_{2m}$) to denote $\mathcal{D}_{ch}$ with the last axiom replaced by $\exists m.size(m, m, S_0)$ (resp. $\exists m.size(2, m, S_0)$). The abbreviation below says that agent i wins in situation s:

$$Win_i(s) \doteq \neg turn_i(s) \wedge size(1, 1, s).$$

It's easy to prove by a non-constructive proof that for Chomp game, player 1 always has a winning strategy. So we have

$$\mathcal{D}_{ch} \models \exists g_1 \forall g_2 \exists s.S_0 \leq_{(g_1,g_2)} s \wedge Win_1(s).$$

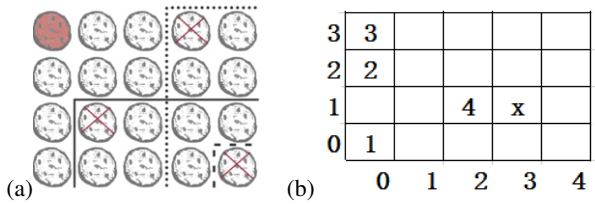

Figure 1. The Chomp and Thieves Games

Finally, we present an example of concurrent games.

Example 4 In a two-dimensional world, three thieves (agents 1-3) attempt to steal a treasure, which is protected by a guard (agent 4), and can only be lifted by at least two agents. As shown in Figure 1(b), the treasure is located at (3,1); initially, the thieves are located at (0,0), (0,2), (0,3) respectively, and the guard is located at (2,1). Any agent can stay at his current position. The thieves can move $right$, up, or $down$ a unit, when they are not caught by the guard. A thief is caught by the guard if they share the same location. The guard can move $left$, up, or $down$ a unit; he can also move $right$ a unit when he is to the left of the treasure. We use fluent $loc_i(p,q,s)$ to represent that in situation s, agent i is located at (p,q); use fluent $cat_i(s)$ to mean that thief i is caught by the guard; and use fluent $win(s)$ to mean that the thieves successfully get the treasure. For illustration purpose, we only present some axioms of the BAT of this game, denoted by $\mathcal{D}_{tg}$:

$Poss_i(right,s) \equiv \neg cat_i(s), i = 1,2,3;$

$Poss_4(right,s) \equiv \exists p,q.loc_4(p,q,s) \wedge p < 2;$

$loc_i(p,q,do(d,s)) \equiv \phi_i(p,q,d,s),$ where $\phi_i(p,q,d,s)$ is

$\quad loc_i(p,q,s) \wedge d_i = stay \vee$

$\quad loc_i(p+1,q,s) \wedge d_i = left \vee \ldots, i = 1,2,3,4;$

$cat_i(do(d,s)) \equiv \exists p,q.\phi_i(p,q,d,s) \wedge \phi_4(p,q,d,s) \vee$

$\quad cat_i(s), i = 1,2,3;$

$win(do(d,s)) \equiv d_1 = lift \wedge d_2 = lift \vee d_2 = lift \wedge$

$\quad d_3 = lift \vee d_1 = lift \wedge d_3 = lift \vee win(s);$

$loc_1(0,0,S_0) \wedge \neg cat_1(S_0).$

Finally, we illustrate that in our situation calculus language, we can specify different notions of strategies, such as memoryless and perfect recall strategies. We first introduce two abbreviations:

1. $Eqst(s,s')$ which says that the states of s and s' are the same[26]:
$Eqst(s,s') \doteq \bigwedge_{i=1}^{n} \forall \vec{x}_i.(F_i(\vec{x}_i,s) \equiv F_i(\vec{x}_i,s')) \wedge$
$\bigwedge_{j=1}^{m} \forall \vec{y}_j.(f_j(\vec{y}_j,s) = f_j(\vec{y}_j,s'))$
Here $F_i(\vec{x}_i,s), 1 \le i \le n$ are the finitely many relational fluents, and $f_j(\vec{y}_j,s), 1 \le j \le m$ are the finitely many functional fluents.
2. $Eqlev(s,s')$ which says that s and s' are situations at the same level, *i.e.*, situations resulting from performing the same number of joint actions: $Eqlev(s,s') \doteq \forall P.P(S_0,S_0) \wedge$
$\forall s_1,s_1',d,d'.[P(s_1,s_1') \supset P(do(d,s_1),do(d',s_1'))] \supset P(s,s').$

Then we can describe different notions of strategies according to different statewise memory abilities (ignoring actions) as follows:

- Statewise memoryless strategy:
$M_0(g) \doteq \forall s,s'.Eqst(s,s') \supset g(s) = g(s').$
- Statewise perfect recall strategy:
$PR(g) \doteq \forall s,s'.[\forall s_1,s_1'.s_1 \sqsubseteq s \wedge s_1' \sqsubseteq s' \wedge Eqlev(s_1,s_1')$
$\supset Eqst(s,s')] \wedge Eqlev(s,s') \supset g(s) = g(s').$
- Statewise perfect recall strategy wrt agent j:
$PR_j(g) \doteq \forall s,s'.[\forall s_1,s_1',d,d'.do(d,s_1) \sqsubseteq s \wedge do(d',s_1') \sqsubseteq s' \wedge$
$Eqlev(s_1,s_1') \wedge d_j = d_j' \supset Eqst(s,s')]$
$\wedge Eqlev(s,s') \supset g(s) = g(s').$
- K-bounded-memory strategy (take the example of $K = 1$):
$M_1(g) \doteq \forall s,s'.Eqst(s,s') \wedge \forall s_1,s_1',d,d'.[do(d,s_1) = s \wedge$
$do(d',s_1') = s' \supset Eqst(s_1,s_1')] \supset g(s) = g(s').$

4 EMBEDDING SL INTO THE SITUATION CALCULUS

In this section, we show that Strategy Logic can be embedded into our extended situation calculus, *i.e.*, our framework is expressive enough to contain SL. With this result, we establish formal connections between our work and existing works.

Firstly, we give an encoding of a concurrent game structure $\mathcal{G}$ into a BAT $\mathcal{D}_{\mathcal{G}}$; then we present a translation function which maps an SL sentence φ into a situation calculus formula φ'; finally, we prove that for any CGS $\mathcal{G}$ and SL sentence φ, $\mathcal{G}, w^0 \models \varphi$ iff $\mathcal{D}_{\mathcal{G}} \models \varphi'$.

We fix an SL signature $\langle \text{AP}, \text{AG}, Var \rangle$. Given a CGS $\mathcal{G} = \langle \text{AC}, W, \lambda, \tau, w^0 \rangle$, where $\text{AC} = \{a^0, \ldots, a^{m-1}\}$ and $W = \{w^0, \ldots, w^{l-1}\}$, the vocabulary of our situation calculus language includes the following: a set of action constants $\{ac_0, \ldots, ac_{m-1}\}$, and a set of state constants $\{sc_0, \ldots, sc_{l-1}\}$; a functional fluent $State(s)$, representing the state of the game in situation s; for each $p \in \text{AP}$, a relational fluent $p(s)$, denoting that p holds in situation s.

We let $\phi(x,d,y)$ denote the following formula, which says y is the state resulting from performing joint action d in state x, *i.e.*, $\phi(x,d,y)$ represents the transition function τ:

$$\bigvee\{x = sc_j \wedge d = \langle ac_{j_1}, \ldots, ac_{j_n} \rangle \wedge y = sc_k \mid j < l, j_1 < m, \ldots, j_n < m, \tau(w^j, \langle a_{j_1}, \ldots, a_{j_n} \rangle) = w^k\}.$$

Definition 5 *Given a CGS $\mathcal{G} = \langle AC, W, \lambda, \tau, w^0 \rangle$, the BAT $\mathcal{D}_{\mathcal{G}}$ includes the following domain-specific axioms:*

(A0) $sc_j \neq sc_k, ac_j \neq ac_k, j \neq k;$

(A1) $Poss_i(ac_j,s) \equiv \top, i = 1,\ldots,n, j < m;$

(A2) $(State(do(d,s)) = x) \equiv \phi(State(s),d,x);$

(A3) $p(do(d,s)) \equiv \bigvee\{\phi(State(s),d,sc_j) \mid p \in \lambda(w^j)\}, p \in AP;$

(A4) $State(S_0) = sc_0;$

(A5) $\bigwedge\{p(S_0) \mid p \in \lambda(w^0)\} \wedge \bigwedge\{\neg p(S_0) \mid p \notin \lambda(w^0)\}.$

Here A0 consists of the unique name axioms for states and actions. A1 says that each action is always possible. A2 is the successor state axiom for fluent $State(s)$, saying that x is the state after executing joint action d in situation s iff x is the state resulting from performing d in the state of s. A3 is the successor state axiom for fluent $p(s)$, saying that p holds after doing d in s iff the new state is one of the states where p holds. A4 states that the initial state is w^0 and A5 specifies the initial truth values of $p \in \text{AP}$ according to $\lambda(w^0)$.

Given an SL formula φ, a joint strategy g_{all}, and a situation s, we define a situation calculus formula $T(\varphi, g_{all}, s)$, which intuitively means that when the agents commit to the joint strategy g_{all}, φ holds in situation s. We use Var as the set of strategy variables in our situation calculus language. Let $x \in Var$. We let $g_{all}[i/x]$ denote the joint strategy exactly like g_{all} except that agent i adopts strategy x. Finally, we use $g_{all}(s)$ to represent the joint action at situation s, *i.e.*, $\langle g_1(s), \ldots, g_n(s) \rangle$.

Definition 6 *Given an SL formula φ, a joint strategy g_{all}, and a situation s, we define a situation calculus formula $T(\varphi, g_{all}, s)$ inductively as follows:*

- $T(p, g_{all}, s) = p(s)$, *for each* $p \in AP$;
- $T(\neg\varphi, g_{all}, s) = \neg T(\varphi, g_{all}, s)$;
- $T(\varphi_1 \wedge \varphi_2, g_{all}, s) = T(\varphi_1, g_{all}, s) \wedge T(\varphi_2, g_{all}, s)$;

- $T(\bigcirc\varphi, g_{all}, s) = T(\varphi, g_{all}, do(g_{all}(s), s))$;
- $T(\varphi_1 \mathcal{U} \varphi_2, g_{all}, s) = \exists s'.s \leq_{g_{all}} s' \wedge T(\varphi_2, g_{all}, s') \wedge \forall s''[s \sqsubseteq s'' \sqsubset s' \supset T(\varphi_1, g_{all}, s'')]$;
- $T(\langle\langle x\rangle\rangle\varphi, g_{all}, s) = \exists x.T(\varphi, g_{all}, s)$;
- $T((i,x)\varphi, g_{all}, s) = T(\varphi, g_{all}[i/x], s)$.

Given the semantics of SL (Definition 3), the above translation is quite straightforward.

Example 5 We illustrate the above translation with the SL sentence $\varphi : \langle\langle x\rangle\rangle[\![y]\!]\langle\langle z\rangle\rangle(1,x)(2,y)(3,z) \bigcirc p$. Let g_{all} be the joint strategy $\langle u, v, w\rangle$. Then we have:

1. $T(\bigcirc p, g_{all}, s) = p(do(\langle u(s), v(s), w(s)\rangle, s))$;
2. $T((1,x)(2,y)(3,z) \bigcirc p, g_{all}, s) = p(do(\langle x(s), y(s), z(s)\rangle, s))$;
3. $T(\varphi, g_{all}, s) = \exists x \forall y \exists z.p(do(\langle x(s), y(s), z(s)\rangle, s))$.

Note that if φ is a sentence, then $T(\varphi, g_{all}, s)$ does not depend on g_{all} and it does not have free strategy variables. In this case, we omit g_{all} and simply write $T(\varphi, s)$.

We now state the embedding theorem as follows:

Theorem 2 *Given a CGS $\mathcal{G}$ and an SL sentence φ, we have $\mathcal{G}, w^0 \models \varphi$ iff $\mathcal{D}_{\mathcal{G}} \models T(\varphi, S_0)$.*

A few remarks are in order before we prove the theorem. First of all, the significance of Theorem 2 lies with that it formally shows that SL can be embedded into the extended situation calculus and it establishes the correctness of our translation function $T(\varphi, g_{all}, s)$. Secondly, indeed, when encoding a CGS as a BAT, the states are represented explicitly, and the size of our encoding is of the same order as the size of the CGS. One may question that this brute-force encoding does not make use of the advantage of the situation calculus in compactly representing large state spaces. However, this is the best we can do given an arbitrary CGS. To represent a specific game structure, we might very well have a more succinct representation in the situation calculus than using a CGS. But given an arbitrary CGS where states are individual objects rather than being factored into atomic features, we cannot restore the original game structure and give a succinct representation. Finally, the extended situation calculus is more expressive than SL, which does not provide a way to compactly represent game structures. In SL, we can only discuss whether a game represented by a CGS has a certain property. In contrast, in the situation calculus, we can discuss properties of a class of games represented by a BAT. For example, $\mathcal{D}_{ch}$ represents the class of Chomp games, either of any finite size or infinite, and $\mathcal{D}_{mm}$ the class of Chomp games with square grids of any size.

To prove the theorem, we first prove a lemma.

A strategy assignment in a model M of $\mathcal{D}_{\mathcal{G}}$ is a partial function from Var to the set of strategies in M. Given a CGS $\mathcal{G} = \langle \text{AC}, W, \lambda, \tau, w^0\rangle$, a joint strategy g_{all}, a complete strategy assignment χ in $\mathcal{G}$, and a state $w = l_k(\chi)$ for some $k \geq 0$, we construct a model $\bar{\mathcal{G}}$ of $\mathcal{D}_{\mathcal{G}}$, a strategy assignment $\bar{\chi}$ in $\bar{\mathcal{G}}$ s.t. $g_{all} \subseteq dom(\bar{\chi})$, and a situation $\bar{w}$ in $\bar{\mathcal{G}}$ as follows.

The action and object domains of $\bar{\mathcal{G}}$ are AC and W respectively. We also call an object a state. The joint action domain of $\mathcal{G}$ is the set of n-ary vectors of actions, and the situation domain of $\mathcal{G}$ is the set of finite sequences of joint actions. We interpret ac_j as a^j, sc_j as w^j, and S_0 as the empty sequence. The $\sqsubset$ predicate and the do, $joint$ and projection functions get their natural interpretations. We interpret $Poss_i(ac_j, s)$, the fluents $State(s)$ and $p(s)$ according to axioms A1-A5 of Definition 5. Clearly, $\bar{\mathcal{G}}$ is a model of $\mathcal{D}_{\mathcal{G}}$. Intuitively, there is a one-to-one correspondence between the situations of $\bar{\mathcal{G}}$ and the tracks of $\mathcal{G}$ starting from the initial state. So there is a one-to-one correspondence between strategies of $\bar{\mathcal{G}}$ and $\mathcal{G}$.

For a strategy κ in $\mathcal{G}$, we define a strategy $\bar{\kappa}$ in $\bar{\mathcal{G}}$ as follows. Let μ be a situation in $\bar{\mathcal{G}}$. Then there exist joint actions $d^1, \ldots, d^k$ s.t. μ is $[d^1, ..., d^k]$. We get a track $h = w_0 \ldots w_k$ in $\mathcal{G}$ s.t. $w_0 = w^0$, $w_j = \tau(w_{j-1}, d^j)$, $1 \leq j \leq k$. Let $\bar{\kappa}(\mu) = \kappa(h)$. We now define $\bar{\chi}$ as follows. For any $i \in$ AG, $\bar{\chi}(g_i) = \bar{\kappa}$, where $\kappa = \chi(i)$; and for any $x \in \text{AG} \cup (Var - g_{all})$, $\bar{\chi}(x) = \bar{\kappa}$, if $\chi(x)$ is defined and $\kappa = \chi(x)$, otherwise $\bar{\chi}(x)$ is undefined. As discussed before Definition 3, χ determines a unique infinite state sequence $w_0 w_1 w_2 \ldots$ and a unique infinite decision sequence $d^1 d^2 \ldots$. Since $w = l_k(\chi)$, *i.e.*, $w = w_k$, we let $\bar{w}$ be the situation $[d^1, ..., d^k]$. By induction on φ:

Lemma 1 *Given an SL formula φ, a CGS $\mathcal{G}$, a joint strategy g_{all}, a complete strategy assignment χ in $\mathcal{G}$, and a state $w = l_k(\chi)$ for some $k \geq 0$, we have $\mathcal{G}, w, \chi \models \varphi$ iff $\bar{\mathcal{G}}, \bar{w}, \bar{\chi} \models T(\varphi, g_{all}, s)$.*

Note that any model of $\mathcal{D}_{\mathcal{G}}$ is isomorphic to $\bar{\mathcal{G}}$. Hence Theorem 2 follows immediately from Lemma 1.

Therefore, this extended situation calculus is at least as expressive as SL. In the following, we show that our extended situation calculus is strictly more expressive than SL.

Example 6 Consider the following sentence φ_0 in our extended situation calculus: $\exists x, y.x(S_0) \neq y(S_0) \wedge \forall z \exists s, s'.S_0 \sqsubseteq_{(x,z)} s \wedge S_0 \sqsubseteq_{(y,z)} s' \wedge p(s) \wedge p(s')$. Intuitively, this sentence means that agent 1 has two different strategies where agent 1 performs different actions at S_0 to ensure that eventually p holds.

Proposition 1 *There exist two CGSs $\mathcal{G}_1$ and $\mathcal{G}_2$ such that no SL sentence can distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$ but φ_0 distinguishes between $\bar{\mathcal{G}}_1$ and $\bar{\mathcal{G}}_2$.*

Proof: Given a signature $\langle\{p\}, \{1,2\}, Var\rangle$, let $\mathcal{G}_1 = \langle AC, W, \lambda, \tau_1, w_0\rangle$ and $\mathcal{G}_2 = \langle AC, W, \lambda, \tau_2, w_0\rangle$, where $AC = \{a_0, a_1, a_2\}$, $W = \{w_0, w_1, w_2\}$. $\lambda(w_1) = \{p\}$, and $\lambda(w_0) = \lambda(w_2) = \emptyset$. $\tau_1(w_0, (a_0, *)) = \tau_1(w_0, (a_1, *)) = \tau_2(w_0, (a_0, *)) = \tau_2(w_0, (a_1, a_1)) = w_1$, $\tau_1(w_0, (a_2, *)) = \tau_2(w_0, (a_2, *)) = \tau_2(w_0, (a_1, a_0)) = \tau_2(w_0, (a_1, a_2)) = w_2$, $\tau_1(w_i, (*, *)) = \tau_2(w_i, (*, *)) = w_i, i = 1, 2$, where $*$ denotes any action. We can easily see that $\bar{\mathcal{G}}_1$ satisfies φ_0, but $\bar{\mathcal{G}}_2$ does not. However, $\mathcal{G}_1$ and $\mathcal{G}_2$ are locally isomorphic, because we can let $\sim = \{(s_i, s_i) : i = 0, 1, 2\}$, function $g : \sim \to 2^{AC \times AC}$, $g(s_i, s_i) = \{(a_0, a_0), (a_1, a_1), (a_2, a_2)\}$. By Theorem 1, no SL sentence can distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$. ■

By Theorem 2 and Proposition 1, we get

Corollary 1 *The extended situation calculus is strictly more expressive than SL.*

5 STRATEGY SPECIFICATION

In this section, we propose a simple strategy specification language based on Golog, and define the strategy verification and synthesis problems.

A situation-suppressed formula φ is a situation calculus formula with all situation arguments suppressed, and $\varphi[s]$ denotes the formula obtained from φ by taking s as the situation arguments of all fluents mentioned in φ.

Definition 7 *Strategy specifications are defined inductively as follows: $\delta ::= (\varphi?; \alpha) \mid (\alpha; \varphi?) \mid (\delta \,|\, \delta) \mid (\pi x.\delta(x))$, where φ is a situation-suppressed formula, and α is an action term.*

Intuitively, $\varphi?; \alpha$ means that if φ holds then performing α, $\alpha; \varphi?$ means performing α so that φ holds, $\delta_1 | \delta_2$ represents nondeterministic choice of two strategies, and $\pi x.\delta(x)$ stands for nondeterministic choice of stratgy arguments.

Note that our strategy specification language does not use the sequence or iteration constructs of Golog. As argued by van Benthem

[42], such a flat language often suffices for the purpose of strategy specifications: a strategy prescribes one move at a time, subject to local conditions; then local iterations make little sense, and local sequences only make sense when a player has consecutive turns.

The formal semantics of strategy specifications is defined by an abbreviation $Does_i(\delta, s, a)$, which intuitively means that action a forms a legal execution of δ by agent i in situation s.

Definition 8 *$Does_i(\delta, s, a)$ is defined inductively as:*

- $Does_i(\varphi?; \alpha, s, a) = \varphi[s] \wedge Poss_i(\alpha, s) \wedge a = \alpha$;
- $Does_i(\alpha; \varphi?, s, a) = \forall d.(Poss(d, s) \wedge d_i = \alpha \supset \varphi[do(d, s)]) \wedge Poss_i(\alpha, s) \wedge a = \alpha$;
- $Does_i(\delta_1 \mid \delta_2, s, a) = Does_i(\delta_1, s, a) \vee Does_i(\delta_2, s, a)$;
- $Does_i(\pi x.\delta(x), s, a) = \exists x.Does_i(\delta(x), s, a)$.

We now formally define the notion that a strategy g satisfies a specification δ for agent i.

Definition 9 *The condition of specification δ for agent i in situation s, denoted $Cond_i(\delta, s)$, is defined inductively as:*

- $Cond_i(\varphi?; \alpha, s) = \varphi[s] \wedge Poss_i(\alpha, s)$;
- $Cond_i(\alpha; \varphi?, s, a) = \forall d.(Poss(d, s) \wedge d_i = \alpha \supset \varphi[do(d, s)]) \wedge Poss_i(\alpha, s)$;
- $Cond_i(\delta_1 \mid \delta_2, s) = Cond_i(\delta_1, s) \vee Cond_i(\delta_2, s)$;
- $Cond_i(\pi x.\delta(x), s) = \exists x.Cond_i(\delta(x), s)$.

Definition 10 *Given a strategy g and a specification δ for agent i,*

$$Sat_i(g, \delta) \doteq \forall s.Cond_i(\delta, s) \supset Does_i(\delta, s, g(s)).$$

Intuitively, $Sat_i(g, \delta)$ means that strategy g satisfies specification δ for agent i, that is, for any situation s, under the condition of δ in s, action $g(s)$ forms a legal execution of δ.

In the following, according to Sat_i, we give two relations between two strategy specifications.

Definition 11 *Given a BAT $\mathcal{D}$, and a strategy constant f for agent i, two strategy specifications δ, δ' are equivalent with respect to f and i, if $\mathcal{D} \models Sat_i(f, \delta) \equiv Sat_i(f, \delta')$ holds. Call δ, δ' are equivalent w.r.t. agent i, if $\mathcal{D} \models \forall g.Sat_i(g, \delta) \equiv Sat_i(g, \delta')$.*

Two strategy specifications δ, δ' for agent i are called disjoint, if $\mathcal{D} \models \forall s.\neg(Cond_i(\delta, s) \wedge Cond_i(\delta', s))$.

Intuitively, two specifications δ, δ' are equivalent with the strategy f for agent i, which means these two specifications satisfy the f at the same time; and two specifications equivalent w.r.t. agent i means that they specify the same set of strategies for agent i. Two specifications for agent i are disjoint, which means the conditions of two specifications for i are inconsistent.

For any disjoint specifications δ_1, δ_2 for agent i, a strategy for i satisfies $\delta_1 \mid \delta_2$ iff the strategy satisfies both δ_1 and δ_2.

Proposition 2 *For disjoint strategy specifications δ_1, δ_2 for agent i, given any strategy constant f for i, we have*

$$\mathcal{D} \models Sat_i(f, \delta_1 \mid \delta_2) \equiv Sat_i(f, \delta_1) \wedge Sat_i(f, \delta_2).$$

In fact, any strategy specification for agent i has an equivalent canonical specification for it.

Definition 12 *A strategy specification δ is called standard if the δ is of the following form, $\delta_1 \mid ... \mid \delta_n$, in which for any $i \in \{1, ..., n\}$, δ_i is like the following $\varphi_1?; a$, $a; \varphi?$ or $\pi\vec{x}.\varphi_2(\vec{x})?; a(\vec{x})$, where φ_1 and $\varphi_2(\vec{x})$ are uniform situation suppressed formulas.*

Proposition 3 *Given a basic action theory $\mathcal{D}$ and any strategy specification δ for agent i, there must exist an equivalent standard specification δ' for agent i.*

A collective strategy specification δ_A for coalition A is a function from A to strategy specifications. Below we formalize the concept of winning strategies. Recall we use g_{all} to denote a joint strategy.

Definition 13 ***(Winning strategy)*** *Given a basic action theory $\mathcal{D}$, a situation-calculus formula $\varphi(g_{all}, s)$, a coalition A, and a collective strategy specification δ_A for A, we say that δ_A is a winning strategy for A wrt φ if $\mathcal{D} \models \forall g_{all}. \bigwedge_{i \in A} Sat_i(g_i, \delta_i) \supset \varphi(g_{all}, S_0)$ holds.*

Here $\varphi(g_{all}, s)$ is a general situation-calculus formula whose free variables are among g_{all} and s. An example of such a formula is $\exists s.S_0 \leq_{g_{all}} s \wedge \phi(s)$, which means that via the joint strategy g_{all}, ϕ eventually holds at some executable situation. Other examples are $\forall s.S_0 \leq_{g_{all}} s \wedge \phi(s)$, meaning that via g_{all}, ϕ always holds, and $\phi(do(g_{all}(s), s))$, meaning that via g_{all}, ϕ holds next.

According to Propositions 2 and 3, when we want to find a winning strategy for A wrt φ, we can try attempt to combine those disjoint specifications of the forms $\varphi?; a$ or $a(\vec{x}); \varphi(\vec{x})?$ to get one standard specification δ_A, then verify whether δ_A is a winning strategy.

We now illustrate strategy specifications with the Chomp, Thieves games, and the Nim game.

Example 3 Cont'd. For the chomp game, although the first player always has a winning strategy, nobody has been able to describe one that applies for all rectangular grids. Yet winning strategies can be specified when the grid is square or only has two finite rows or columns. When the grid is square, player 1's winning strategy is as follows: first do $eat(1, 1)$, and on subsequent moves, if player 2 does $eat(0, k)$ (resp. $eat(k, 0)$), then react with action $eat(k, 0)$ (resp. $eat(0, k)$). When the grid has two rows and m columns, player 1's winning strategy is this: begin with $eat(1, m-1)$, and on subsequent moves, if player 2 does action $eat(0, k)$ (resp. $eat(1, k)$), then respond with $eat(1, k-1)$ (resp. $eat(0, k+1)$). We now present specifications for the above strategies. We first give an abbreviation: $Prev(i, a, s) \doteq \exists s' \exists d.s = do(d, s') \wedge d_i = a$, saying that a is the last action performed by agent i in situation s.

Let $\delta' = \delta_1 \mid \delta_2 \mid \delta_3$, and $\delta'' = \delta_4 \mid \delta_5 \mid \delta_6$, where

- $\delta_1 = \top?; eat(1, 1)$.
- $\delta_2 = \pi k.Prev(2, eat(0, k)) \wedge k > 0?; eat(k, 0)$.
- $\delta_3 = \pi k.Prev(2, eat(k, 0)) \wedge k > 0?; eat(0, k)$.
- $\delta_4 = \pi k.k \geq 1 \wedge size(2, k)?; eat(1, k-1)$.
- $\delta_5 = \pi k.Prev(2, eat(0, k)) \wedge k > 0?; eat(1, k-1)$.
- $\delta_6 = \pi k.Prev(2, eat(1, k))?; eat(0, k+1)$.

Here, these six strategy specifications are pairwise disjoint. Both δ' and δ'' are standard. The following properties say that δ' (resp. δ'') is a winning strategy for player 1 for any $m \times m$ (resp. $2 \times m$) game:

- $\mathcal{D}_{mm} \models \forall g_{all}.Sat(g_1, \delta') \supset \exists s.S_0 \leq_{g_{all}} s \wedge Win_1(s)$;
- $\mathcal{D}_{2m} \models \forall g_{all}.Sat(g_1, \delta'') \supset \exists s.S_0 \leq_{g_{all}} s \wedge Win_1(s)$.

Example 4 Cont'd. The following is a winning strategy for the thieves. Each thief behaves as follows: moves right when he is to the left of the treasure and no matter where the guard is, moves up (resp. down) when he is to the due south (resp. north) of the treasure, and lifts the treasure when he gets to its location. In this way, at least two thieves will escape the guard and seize the treasure.

For $i = 1, 2, 3$, let $\delta_i = \delta_i^1 \mid \delta_i^2 \mid \delta_i^3 \mid \delta_i^4$, where

- $\delta_i^1 = \varphi_i^1?; right$, where $\varphi_i^1 = \exists p, q.loc_i(p, q) \wedge p \leq 2$.
- $\delta_i^2 = \varphi_i^2?; up$, where $\varphi_i^2 = \exists q.loc_i(3, q) \wedge q < 1$.
- $\delta_i^3 = \varphi_i^3?; down$, where $\varphi_i^3 = \exists q.loc_i(3, q) \wedge q > 1$.
- $\delta_i^4 = loc_i(3, 1)?; lift$.

Here for each $i \in \{1, 2, 3\}$, $\{\delta_i^j : j = 1, 2, 3, 4\}$ are pairwise disjoint, and each δ_i is standard. The following says that when each thief i adopts strategy δ_i, they are guaranteed to win:
$\mathcal{D}_{tg} \models \forall g_{all}. \bigwedge_{i=1}^{3} Sat(g_i, \delta_i) \supset \exists s.S_0 \leq_{g_{all}} s \wedge win(s)$.

Example 7 Consider the Nim game [13]. The Nim game is played as follows. There are three piles of chips containing k_1, k_2, and k_3 chips

respectively. Two players take turns moving. Each move consists of selecting one of the three piles and removing one or more chips from it as desired. The winner is the player who removes the last chip.

We now formalize this game in the situation calculus. There is a fluent $rm(k, j, s)$ meaning that there are k chips remaining in the jth pile, $j = 1, 2, 3$. There is an action $take(k, j), k > 0$, meaning taking k chips from the jth piles. Here we also give a situation-independent function $bin(k_1, k_2, k_3)$, which is their addition without carry in base 2 calling it as the nim-sum of them. For example, for $(5, 7, 9)$, $bin(5, 7, 9) = 101 + 111 + 1001 = 1011$.

In [5], Bouton shows that in the Nim game with three piles, the first player has a winning strategy iff for the initial chips (k_1, k_2, k_3), their addition without carry in base 2 is not zero.

In addition to the second-order axiomatization of Peano arithmetic, we have the following axioms:

$Poss_i(take(k, j), s) \equiv turn_i(s) \wedge \exists k'.rm(k', j, s) \wedge k' \geq k,$

$turn_i(do(d, s)) \equiv \neg turn_i(s), \quad i = 1, 2, j = 1, 2, 3;$

$rm(k, j, do(d, s)) \equiv \exists k', k''.rm(k', j, s) \wedge k = k' - k'' \wedge \varphi_0(k'', j) \vee (rm(k, j, s) \wedge \neg\exists k'.\varphi_0(k', j));$

$turn_1(S_0) \wedge \neg turn_2(S_0),$

where $\varphi_0(k, j) \doteq d_1 = take(k, j) \vee d_2 = take(k, j)$. We use $\mathcal{D}_{nim}$ to denote the BAT of the Nim game. Two abbreviations are

- $\varphi_1(s) \doteq \exists k_1, k_2, k_3. \bigwedge_{j=1}^{3} rm(k_j, j, s) \wedge bin(k_1, k_2, k_3) = 0.$
- $\varphi_2(s) \doteq \exists k_1, k_2, k_3. \bigwedge_{j=1}^{3} rm(k_j, j, s) \wedge \neg bin(k_1, k_2, k_3) = 0.$

Intuitively, they mean that in the situation s, the addition of the remaining chips in each pile without carry in base 2 is 0 or not 0.

Denote $\mathcal{D}_{nim} \cup \varphi_1(S_0)$ as $\mathcal{D}_{nim1}$, denote $\mathcal{D}_{nim} \cup \varphi_2(S_0)$ as $\mathcal{D}_{nim2}$. The abbreviation below says that agent i wins in situation s: $Win_i(s) \doteq executable(s) \wedge \neg turn_i(s) \wedge \bigwedge_{j=1}^{3} rm(0, j, s)$.

Then we can give a strategy specification δ_i (i=1,2):

$\pi k, j.take(k, j); \varphi_1?$. Intuitively, δ_i means that if agent i can ensure φ_1 holds next by performing $take(k, j)$, then he does this action.

The following properties say that δ_1 (resp. δ_2) is a winning strategy for player 1 for game with the initial nim-sum is zero (resp. not zero):

- $\mathcal{D}_{nim1} \models \forall g_{all}.Sat(g_1, \delta_1) \supset \exists s.S_0 \leq_{g_{all}} s \wedge Win_1(s);$
- $\mathcal{D}_{nim2} \models \forall g_{all}.Sat(g_2, \delta_2) \supset \exists s.S_0 \leq_{g_{all}} s \wedge Win_2(s).$

For $(5, 7, 9)$, initially, $bin(5, 7, 9) \neq 0$, to make zero, we should take 7 chips away from the 3rd pile, such that the remaining $bin(5, 7, 2) = 0$, and in the next, no matter what player 2 does, the remaining addition will be not zero.

Finally, we formalize the strategy verification and synthesis problems in our framework.

Definition 14 *(**Strategy verification**) Given a BAT $\mathcal{D}$, a situation-calculus formula $\varphi(g_{all}, s)$, a coalition A, and a collective strategy specification δ_A for A, verify if δ_A is a winning strategy for A wrt φ.*

Definition 15 *(**Strategy synthesis**) Given a basic action theory $\mathcal{D}$, a situation-calculus formula $\varphi(g_{all}, s)$, a coalition A, generate a winning strategy for A wrt φ.*

6 A DECIDABLE FRAGMENT

In this section, we give a decidable fragment of our extended situation calculus inspired by the work of De Giacomo *et al*. [8, 9].

In [8], De Giacomo *et al*. define a notion of bounded action theory in the situation calculus, where the theory entails that in all situations, the number of fluent atoms which hold is bounded by a constant, and then prove that verification of an expressive class of first-order μ-calculus temporal properties in such theories is decidable. Roughly, they prove their result by focussing on the active domain of situations, *i.e.*, the set of objects occurring in the extension of some fluent, which is bounded. Essentially, they abstract situations whose active domains are isomorphic into a single state, and by abstracting also actions, they obtain an abstract finite transition system that satisfies exactly the same formulas of their query language.

We strengthen the notion of bounded action theory with the further restriction that the action theory should entail that in all situations, the number of actions executable by any agent is bounded. By similar ideas as in [8], we prove that verification of SL-like situation calculus formulas in such theories is decidable. The further restriction is due to the need of strategic reasoning.

We assume that there are no functions other than constants and there are no situation-independent predicates, there is a finite set $\mathcal{F}$ of relational fluents, and there is a finite set of action functions. Moreover, we assume that the terms of the object sort are in fact a countably infinite set $\mathcal{N}$ of standard names. We restrict our attention to *standard interpretations* of the object sort, where there is a bijection between the set of objects and $\mathcal{N}$. As shown in [20], this restriction can be captured by the set of unique name axioms for constants in $\mathcal{N}$, which we denote by $\mathcal{D}_{uno}$. We use $\mathcal{D} \models_{uno} \phi$ to denote $\mathcal{D} \cup \mathcal{D}_{uno} \models \phi$. We say that $\mathcal{D}_{S_0}$ is a database, if it consists of axioms, one for each fluent F, of the form $\forall \vec{x}.F(\vec{x}, S_0) \equiv \vec{x} = \vec{c}_1 \vee \ldots \vee \vec{x} = \vec{c}_m$, where each $\vec{c}_i$ is a vector of constants from $\mathcal{N}$.

Definition 16 *We say that an action theory $\mathcal{D}$ is strictly bounded by a constant b if $\mathcal{D}$ entails that in any executable situation s, for each fluent F, the number of $F(\vec{x}, s)$ which holds is less than b; and for each agent i and each action type A, the number of $Poss_i(A(\vec{x}), s)$ which holds is less than b.*

We now define the class $\mathcal{SL}$ of SL-like situation calculus formulas. Recall that in Definition 6, we define a translation function $T(\varphi, g_{all}, s)$ which maps an SL formula φ into a situation calculus formula. Firstly, we extend SL to SL^+ so that in the base case, instead of an atom p, we have a situation-suppressed sentence φ, which we call a state sentence, and we let $T(\varphi, g_{all}, s) = \varphi[s]$. Note that a state sentence is a first-order sentence. Now we say ϕ is an SL-like situation calculus formula if it is $T(\varphi, g_{all}, \sigma)$ for some SL^+ formula φ and situation term σ. Note that because of our extension of SL to SL^+, $\mathcal{SL}$ is more expressive than SL. Recall that if φ is a sentence, then $T(\varphi, g_{all}, s)$ does not depend on g_{all} and it does not have free strategy variables. In this case, we simply write $T(\varphi, g_{all}, s)$ as $\varphi[s]$.

Theorem 3 *Verifying if $\mathcal{D} \models_{uno} \phi$, where $\mathcal{D}$ is a strictly bounded action theory with an initial database about S_0, and ϕ is an SL-like situation calculus sentence, is decidable.*

We prove the theorem by showing that the verification problem can be reduced to the model checking problem of SL, which is decidable [29]. Suppose that $\mathcal{D}$ is a strictly bounded action theory with an initial database, and φ is an SL^+ sentence. Then $\mathcal{D}$ has a unique standard model $\mathcal{M}$ whose domain for the object (resp. situation) sort is $\mathcal{N}$ (resp. $\mathcal{S}$). Thus $\mathcal{D} \models_{uno} \varphi[S_0]$ iff $\mathcal{M} \models \varphi[S_0]$. The following are the main steps of the proof.

1. $\mathcal{M}$ induces a concurrent transition system $\mathcal{G}$ with an infinite number of states, each of which is associated with a first-order structure with domain $\mathcal{N}$ s.t. $\mathcal{M}$ satisfies $\varphi[S_0]$ iff $\mathcal{G}$ satisfies φ.
2. $\hat{\varphi}$ is constructed from φ via converting each state sentence of φ into an equivalent domain-independent one.
3. Based on $\mathcal{D}$, a finite domain $\hat{\Delta} \subseteq \mathcal{N}$ can be constructed, and $\mathcal{G}$ can be abstracted into a concurrent transition system $\hat{\mathcal{G}}$ with a finite number of states each of which is associated with a first-order structure with domain $\hat{\Delta}$ s.t. $\mathcal{G}$ satisfies φ iff $\hat{\mathcal{G}}$ satisfies $\hat{\varphi}$. Note that $\hat{\mathcal{G}}$ can be directly constructed from $\mathcal{D}$.

4. By using the domain $\hat{\Delta}$, $\hat{\mathcal{G}}$ can be ground into a CGS, and $\hat{\varphi}$ can be ground into an SL sentence, hence whether $\hat{\mathcal{G}}$ satisfies $\hat{\varphi}$ can be checked via SL model checking.

Firstly, we observe that the evaluation of any $\varphi(\vec{x}, s)$, a uniform FO formula of the situation calculus, does not depend on s, but only on the corresponding interpretation $I_s = \langle \Delta, \cdot^{I_s} \rangle$ of $\mathcal{F}$. For any fluent F in $\mathcal{F}$, $F^{I_s} = \{\vec{u} \mid \mathcal{M} \models F(\vec{u}, s)\}$. By successor state axioms, for any joint action d, if $I_{s_1} = I_{s_2}$, then $I_{do(d,s_1)} = I_{do(d,s_2)}$. For any joint action d, by operating on I_s, we evaluate the situation-suppressed successor states axioms to generate another $I_{do(d,s)}$.

After this, considering all situations in $\mathcal{S}$ and joint actions, we can define a CGS-like system CGS $\mathcal{G} = \langle \Delta, Q, q_0, \{\kappa_i\}_{i \in AG}, \tau, \lambda \rangle$. Here the object domain is $\Delta = \mathcal{N}$; $Q = \mathcal{S}$; q_0 is just S_0; κ_i is a function mapping a state to a set of actions which can be executed by agent i in the state, here $\kappa_i(s)$ is bounded; τ is a transition function $\tau(s, d) = s'$ iff $s' = do^{\mathcal{M}}(d, s)$, and $d_i \in \kappa_i(s)$; and $\lambda : Q \to Int_{\Delta}^{\mathcal{F}}$ is the function $\lambda(s) = I_s$. $\mathcal{G}$ retains all the information necessary to evaluate Φ on $\mathcal{M}$. Next the semantics of SL^+ according to CGSs is defined as in Definition 3 except for the state sentence φ: $\mathcal{G}, \chi, w \models \varphi$ iff $\lambda(w) \models \varphi$.

Lemma 2 *If $\mathcal{G}$ is as above, then for any SL^+ sentence φ, and the corresponding $\mathcal{SL}$ sentence $\varphi[S_0]$, $\mathcal{M} \models \varphi[S_0]$ iff $\mathcal{G} \models \varphi$.*

Thus, to check whether $\mathcal{D}_{uno} \models \varphi[S_0]$, we can check whether $\mathcal{G} \models \varphi$. However, the $\mathcal{G}$ is infinite. In the following, we should abstract this $\mathcal{G}$ to a finite CGS $\hat{\mathcal{G}}$ under the strict boundedness.

Let $adom(I)$ denote the active domain of I, i.e., the set of all objects occurring in some fluent extension F^I. And denote by $\mathcal{C} \subseteq \mathcal{N}$ the finite set of all constants, appearing in $\mathcal{D}$. We say that I and J are ad-isomorphism [8], if there exists a bijection $i : adom(I) \cup \mathcal{C} \to adom(J) \cup \mathcal{C}$, s.t., $i(c) = c$, $c \in \mathcal{C}$, and $\vec{o} \in F^I$ iff $i(\vec{o}) \in F^J$. The notion of A-local-isomorphism $\approx$ is like local-isomorphism in Definition 4, except that if two states are A-local-isomorphism, then their corresponding interpretations should be ad-isomorphism, and in any state, the bisimulation relations g_i on actions are given for each agent i, for instance, for $s_1 \approx s_2$, for any $a_1 \in \kappa_i(s_1)$, there is a $a_2 \in \kappa_i(s_2)$, such that $(a_1, a_2) \in g_i(s_1, s_2)$.

A first-order sentence φ is said to be *domain-independent* [1] if for each interpretation $I = \langle \Delta, \cdot^I \rangle, I \models \varphi$ iff $\hat{I} \models \varphi$, where $\hat{I} = \langle adom(I), \cdot^I \rangle$. And an SL^+ sentence is said *domain-independent* if so are all of its first-order components. Then we can prove that A-local-isomorphism CGSs preserve domain-independent SL^+ sentence as in [28].

Lemma 3 *Given two A-local-isomorphism CGSs $\mathcal{G}, \mathcal{G}'$, and a domain-independent SL^+ sentence φ, $\mathcal{G} \models \varphi$ iff $\mathcal{G}' \models \varphi$.*

In [25], Libkin shows, through syntactic manipulations, any FO sentence φ can be effectively turned into a logically equivalent domain-independent formula φ'. By adapting his approach to each FO component of an $\mathcal{SL}$ sentence, we can generalise this result.

Lemma 4 *For any $\mathcal{SL}$ sentence Φ, there exists a logically equivalent domain-independent $\mathcal{SL}$ sentence Φ'.*

Next, we construct the finite CGS abstraction $\hat{\mathcal{G}}$ of the induced $\mathcal{G}$ as the tuple $\langle \hat{\Delta}, \hat{Q}, \hat{q}_0, \{\hat{\kappa}_i\}_{i \in AG}, \hat{\tau}, \hat{\lambda} \rangle$. Here the finite object domain is $\hat{\Delta} \subset \mathcal{N}$, satisfying $\mathcal{C} \subseteq \hat{\Delta}$ and $\mid \hat{\Delta} \mid > \mid \mathcal{C} \mid + \beta + n\beta_1 + \eta$. Here $\beta = \Sigma_{F \in \mathcal{F}} b \cdot a_F$, with a_F the arity of situation suppressed fluent F, to ensure A-local-isomorphism, each state of $\mathcal{G}$ must have a matching ad-isomorphism state in $\hat{\mathcal{G}}$; $\beta_1 = \Sigma_A b \cdot a_A$, a_A is the arity of action function A; and η is the maximal number of distinct variables occurring in the righthand side of action precondition and successor state axioms to abstract all the possible combinations of the objects used in some action, and mentioned in the axioms. Let $\hat{q}_0 = \langle \star, I_0 \rangle$, here $\star$ is a special symbol, and $I_0 = \langle \hat{\Delta}, \cdot^{I_0} \rangle$ is a FO interpretation over $\hat{\Delta}$. For any $\vec{o} \in \hat{\Delta}^{|\vec{o}|}$, and $F \in \mathcal{F}$, $\vec{o} \in F^{I_0}$ iff $\mathcal{M}_0 \models F(\vec{o})$, then in I_0, by boundness, for each action type $A(\vec{x})$ and agent i, there exist bounded number instances $\vec{o} \in \hat{\Delta}^{|\vec{x}|}$, s.t., $I_0 \models \phi_{A,i}(\vec{o})$, here $Poss_i(A(\vec{x}), s) \equiv \phi_{A,i}(\vec{x}, s)$. So let $A(\vec{o}) \in \kappa_i(\langle \star, I_0 \rangle)$. And there exist bounded executable joint actions for I_0, after executing each such d, we get another interpretation I, then we give another state $\langle d, I \rangle$, and have $\hat{\tau}(\langle \star, I_0 \rangle, d) = \langle d, I \rangle$. By applying this procedure recursively, due to the strictly boundness and $\hat{\Delta}$ finite, we can generate all such finite $\langle d, I \rangle$, give κ_i and the transition function $\hat{\tau}$. Finally, define $\lambda(\langle d, I \rangle) = I$. Then the property below holds.

Lemma 5 *$\mathcal{G}$ and $\hat{\mathcal{G}}$ are A-local-isomorphism.*

In fact, when the initial situation is incomplete, the verification of $\mathcal{SL}$ is also decidable. More details are referred to [8].

7 CONCLUSION

In this paper, by a simple extension of the situation calculus with a strategy sort, we have developed a general framework for strategy representation and reasoning for complete information games. We have shown that Strategy Logic can be embedded into our framework. The framework can be used to compactly represent both concurrent and turn-based possibly infinite game structures, specify the structure of strategies, reason about strategies explicitly, and reason about strategic abilities of coalitions under strategy specifications. In a companion paper [46], we have extended this framework to deal with strategic reasoning for incomplete information games.

Action formalisms like the situation calculus and modal logics have been two main families of logics in AI. Van Benthem [40] proposes the idea that the situation calculus and modal logics meet and merge. Recent works in this direction, *e.g.*, [27, 12], connect dynamic epistemic logic to the situation calculus. Based on earlier works, this paper presents another work in this line connecting strategic logics to the situation calculus. Compared with modal logics for strategic reasoning, the most distinguished feature of our work is expressiveness and compactness in representing game structures and strategies. Many of existing works based on multi-agent situation calculus, *e.g.*, [18, 4], are mainly concerned with epistemic reasoning. Others [37, 14, 17, 10] deal with strategic reasoning, but they either focus on different issues such as the computation of Nash equilibria or the coordination problem or do not support ATL-like or SL-like reasoning about strategies.

In this paper, we focus on the representation side of our strategy formalism. From the reasoning side, our framework is highly undecidable since it is based on an extension of the situation calculus with a second-order strategy sort. Nonetheless, it can serve as the theoretic foundation for strategy verification and synthesis based on first-order theorem proving and fix-point computation techniques. In the software engineering community, there has been considerable progress on program verification via automatic discovery of loop invariants [3, 15, 38]. In recent years, progress has also been made on automated reasoning in the situation calculus and automatic verification of Golog programs [23, 24]. In the planning community, various techniques such as generating and testing [21], and object abstraction [39], have been developed for planning with loops. Inspired by these works, we would like to explore automatic strategy verification and synthesis.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for helpful comments. This work received support from the Natural Science Foundation of China under Grant No. 61572535.

REFERENCES

[1] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*, Addison-Wesley, 1995.

[2] R. Alur, T.A. Henzinger, and O. Kupferman, 'Alternating-time temporal logic', *J. ACM*, **49**(5), 672–713, (2002).

[3] T. Ball, R. Majumdar, T.T. Millstein, and S.K. Rajamani, 'Automatic predicate abstraction of C programs', in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 203–213, (2001).

[4] V. Belle and G. Lakemeyer, 'Reasoning about imperfect information games in the epistemic situation calculus', in *Proc. of AAAI'2010*.

[5] C.L. Bouton, 'Nim, a game with a complete mathematical theory', *Annals of Mathematics*, **3**(1), 35–39, (1901).

[6] K. Chatterjee, T.A. Henzinger, and N. Piterman, 'Strategy logic', *Inf. Comput.*, **208**(6), 677–693, (2010).

[7] G. De Giacomo, Y. Lespérance, and H.J. Levesque, 'Congolog, a concurrent programming language based on the situation calculus', *Artif. Intell.*, **121**(1-2), 109–169, (2000).

[8] G. De Giacomo, Y. Lespérance, and F. Patrizi, 'Bounded situation calculus action theories and decidable verification', in *Proc. of KR'2012*.

[9] G. De Giacomo, Y. Lespérance, and F. Patrizi, 'Bounded situation calculus action theories', *Artif. Intell.*, **237**, 172–203, (2016).

[10] G. De Giacomo, Y. Lespérance, and A.R. Pearce, 'Situation calculus based programs for representing and reasoning about game structures', in *Proc. of KR'2010*.

[11] G. De Giacomo, Y. Lespérance, and A.R. Pearce, 'Synchronous games in the situation calculus', in *Proc. of AAMAS'2015*.

[12] L. Fang and Y. Liu, 'Multiagent knowledge and belief change in the situation calculus', in *Proc. of AAAI'2013*.

[13] T.S. Ferguson, *Game Theory*, UCLA, 2008.

[14] A. Finzi and T. Lukasiewicz, 'Game-theoretic agent programming in golog', in *Proc. of ECAI'2004*.

[15] C. Flanagan and S. Qadeer, 'Predicate abstraction for software verification', in *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages*.

[16] M.R. Genesereth, N Love, and B Pell, 'General game playing: Overview of the AAAI competition', *AI Magazine*, **26**(2), 62–72, (2005).

[17] H Ghaderi, H.J. Levesque, and Y. Lespérance, 'A logical theory of coordination and joint ability', in *Proc. of AAAI'2007*.

[18] R.F. Kelly and A.R. Pearce, 'Asynchronous knowledge with hidden actions in the situation calculus', *Artif. Intell.*, **221**, 1–35, (2015).

[19] Y. Lespérance, H.J. Levesque, F. Lin, and R.B. Scherl, 'Ability and knowing how in the situation calculus', *Studia Logica*, **66**(1), 165–186, (2000).

[20] H.J. Levesque, 'A completeness result for reasoning with incomplete first-order knowledge bases', in *Proc. of KR'1998*.

[21] H.J. Levesque, 'Planning with loops', in *Proc. of IJCAI'2005*.

[22] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R.B. Scherl, 'GOLOG: A logic programming language for dynamic domains', *J. Log. Program.*, **31**(1-3), 59–83, (1997).

[23] N. Li, Y. Fan, and Y. Liu, 'Reasoning about state constraints in the situation calculus', in *Proc. of IJCAI'2013*.

[24] N. Li and Y. Liu, 'Automatic verification of partial correctness of golog programs', in *Proc. of IJCAI'2015*.

[25] L. Libkin, 'Embedded finite models and constraint databases', in *In Finite Model Theory and Its Applications*, 257–338, Springer, (2007).

[26] F. Lin and H.J. Levesque, 'What robots can do: Robot programs and effective achievability', *Artif. Intell.*, **101**(1-2), 201–226, (1998).

[27] Y. Liu and H.J. Levesque, 'Incorporating action models into the situation calculus', in *Johan van Benthem on Logic and Information Dynamics, Outstanding Contributions to Logic*, eds., Alexandru Baltag and Sonja Smets, volume 5, chapter 21, 569–590, Springer, (2014).

[28] F. Mogavero, *Logics in Computer Science - A Study on Extensions of Temporal and Strategic Logics*, volume 3 of *Atlantis Studies in Computing*, Atlantis Press, 2013.

[29] F. Mogavero, A. Murano, G. Perelli, and M.Y. Vardi, 'Reasoning about strategies: On the model-checking problem', *ACM Trans. Comput. Log.*, **15**(4), 34:1–34:47, (2014).

[30] F. Mogavero, A. Murano, and M.Y. Vardi, 'Reasoning about strategies', in *Proc. of FSTTCS'2010*.

[31] R.J. Nowakowski, *Games of No Chance*, Cambridge University Press, 1998.

[32] A. Pnueli, 'The temporal logic of programs', in *Proc. of FOCS'1977*.

[33] R. Ramanujam and S.E. Simon, 'Dynamic logic on games with structured strategies', in *Proc. of KR'2008*.

[34] R. Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*, The MIT Press, 2001.

[35] S. Schiffel and M. Thielscher, 'Reasoning about general games described in GDL-II', in *Proc. of AAAI'2011*.

[36] S. Schiffel and M. Thielscher, 'Representing and reasoning about the rules of general games with imperfect information', *J. Artif. Intell. Res.*, **49**, 171–206, (2014).

[37] O. Schulte and J.P. Delgrande, 'Representing von neumann-morgenstern games in the situation calculus', *Ann. Math. Artif. Intell.*, **42**(1-3), 73–101, (2004).

[38] S. Srivastava and S. Gulwani, 'Program verification using templates over predicate abstraction', in *Proc. of PLDI'2009*.

[39] S. Srivastava, N. Immerman, and S. Zilberstein, 'A new representation and associated algorithms for generalized planning', *Artif. Intell.*, **175**(2), 615–647, (2011).

[40] J. van Benthem, 'Mccarthy variations in a modal key', *Artif. Intell.*, **175**(1), 428–439, (2011).

[41] J. van Benthem, 'Reasoning about strategies', in *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky - Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, pp. 336–347, (2013).

[42] J. van Benthem, *Logic in Games*, The MIT Press, 2014.

[43] W. van der Hoek, W. Jamroga, and M. Wooldridge, 'A logic for strategic reasoning', in *Proc. of AAMAS'2005*.

[44] J. van Eijck, 'PDL as a multi-agent strategy logic', in *Proc. of TARK'2013*.

[45] D Walther, W van der Hoek, and M. Wooldridge, 'Alternating-time temporal logic with explicit strategies', in *Proc. of TARK'2007*.

[46] L. Xiong and Y. Liu, 'Strategy representation and reasoning for incomplete information concurrent games in the situation calculus', in *Proc. of IJCAI'2016*.

[47] D. Zhang and M. Thielscher, 'A logic for reasoning about game strategies', in *Proc. of AAAI'2015*.

[48] D. Zhang and M. Thielscher, 'Representing and reasoning about game strategies', *J. Philosophical Logic*, **44**(2), 203–236, (2015).