Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)

Strategy Representation and Reasoning for Incomplete Information Concurrent Games in the Situation Calculus

Liping Xiong and **Yongmei Liu**
Dept. of Computer Science
Sun Yat-sen University, Guangzhou 510006, China
xionglp3@mail2.sysu.edu.cn, ymliu@mail.sysu.edu.cn

Abstract

Strategy representation and reasoning for incomplete information concurrent games has recently received much attention in multi-agent system and AI communities. However, most of the logical frameworks are based on concrete game models, lack the abilities to reason about strategies explicitly or specify strategies procedurally, and ignore the issue of coordination within a coalition. In this paper, by a simple extension of a variant of multi-agent epistemic situation calculus with a strategy sort, we develop a general framework for strategy representation and reasoning for incomplete information concurrent games. Based on Golog, we propose a strategy programming language which can be conveniently used to specify collective strategies of coalitions at different granularities. We present a formalization of joint abilities of coalitions under commitments to strategy programs. Different kinds of individual strategic abilities can be distinguished in our framework. Both strategic abilities in ATL and joint abilities of Ghaderi et al. can be considered as joint abilities under special programs in our framework. We illustrate our work with a variant of Levesque's Squirrels World.

1 Introduction

Strategy representation and reasoning has recently received much attention in multi-agent system and AI communities. Many strategic logics have been established, and most of them are built upon Alternating-time Temporal Logic (ATL) [Alur *et al.*, 2002] where formula $\langle\langle G\rangle\rangle\phi$ expresses that coalition G has a group strategy to ensure temporal goal ϕ holds no matter what the other agents do.

Many extensions of ATL have been proposed, mainly along two lines. On one hand, in reality, players often have incomplete information about the game states, *e.g.*, in poker games. Although they may have strategies to ensure a goal holds, they may not know what these strategies are or how to execute strategies where the prescribed actions for indistinguishable states are not the same. Extensions of ATL are proposed to deal with such games [van der Hoek and Wooldridge, 2003; Jamroga and van der Hoek, 2004; Jamroga and Ågotnes, 2007]. On the other hand, strategies are treated implicitly in ATL, and extensions of ATL are proposed to reason about strategies explicitly, that is, treat strategies as first-order objects [Walther *et al.*, 2007; Mogavero *et al.*, 2010].

However, the above extensions suffer from other limitations. Firstly, the extensions are mainly propositional, hence lacking expressiveness to compactly specify game structures or indistinguishable states. Secondly, as pointed out by [Ramanujam and Simon, 2008; van Eijck, 2013], these extensions treat strategies as abstract objects rather than considering the internal structure of strategies, that is, the combination of basic strategies to form complex ones. Thirdly, most extensions ignore the coordination problem. As discussed in [Ghaderi *et al.*, 2007], a coalition may have many group strategies to ensure a goal, yet a player may not know other players' choices, hence the coalition may end up with a group strategy which may not ensure the goal.

Other than modal logics, another main family of logics in AI is action formalisms. A prominent example of action formalisms is the situation calculus [Reiter, 2001], which is a first-order language with some second-order ingredients suitable for reasoning about actions and change. Based on the situation calculus, a logic programming language Golog [Levesque *et al.*, 1997] has been designed for high-level agent control. There have been a few works studying strategic reasoning in the situation calculus [Schulte and Delgrande, 2004; Farinelli *et al.*, 2007; De Giacomo *et al.*, 2010; Ghaderi *et al.*, 2007]. The first one deals with von Neumann Morgenstern games, while the second one focuses on Markov games, using Golog to specify agent behavior. The third work studies complete information turn-based games, where ConGolog [De Giacomo *et al.*, 2000], a concurrent version of Golog, is also used to specify game structures. However, these works either focus on solution concepts like Nash equilibria rather than ATL-like properties, or cannot reason about strategies explicitly. Lastly, Ghaderi *et al.* study the coordination problem and present a formalization of joint ability of coalitions based on the idea of iterated elimination of dominated strategies [Osborne and Rubinstein, 1999]. Nonetheless, it is desirable to have a more general account of joint ability of a coalition under constraints, which maybe a rough collective strategy which the coalition commits to, or a protocol that the players must comply with, such as traffic rules.

In this paper, we propose a framework based on the situation calculus for strategy representation and reasoning for incomplete information concurrent games. We first propose a simple extension with a strategy sort of a concurrent variant of multi-agent epistemic situation calculus, which can be used to compactly represent possibly infinite concurrent games, and reason about strategies explicitly. Then we propose a strategy programming language based on Golog, which can be used to specify collective strategies of coalitions. We emphasize that in this paper, by the word "collective strategy", we mean group strategy which is common knowledge of the coalition. Next, we adapt the approach of Ghaderi et al. to formalize joint ability of coalitions under commitments to strategy programs. We illustrate our logical framework with a variant of Levesque's Squirrels World.

2 Preliminaries

In this section, we introduce the situation calculus and Golog.

The *situation calculus* [Reiter, 2001] 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. 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.*, $pick(r, x)$ represents robot r picking up object x. There are relational and functional fluents whose 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 a proper subhistory of s'. We use $s \sqsubseteq s'$ as a shorthand for $s \sqsubset s' \vee s = s'$, and we let $s \leq s'$ abbreviate for $s \sqsubseteq s' \wedge \forall a \forall s^* (s \sqsubset do(a, s^*) \sqsubseteq s' \supset poss(a, s^*))$, meaning that s is a subhistory of s' and every action on the way from s to s' is possible. 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) which describes how the world changes as the result of the available actions. Each BAT $\mathcal{D}$ consists of the following five disjoint parts:

1. Σ, the foundational axioms of the situation calculus;
2. $\mathcal{D}_{ap}$, a precondition axiom for each action specifying when the action can be legally performed;
3. $\mathcal{D}_{ss}$, a successor state axiom (SSA) 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.

Liu and Levesque [2014] propose a multi-agent extension of the situation calculus. They use a special fluent $B(i, s', s)$, which means that agent i considers situation s' accessible from situation s, and introduce a special predicate $A(i, a', a, s)$, meaning that in situation s, agent i considers action a' as a possible alternative of action a. The following is their successor state axiom for the B fluent:

$$\begin{aligned} B(i, s', do(a, s)) \equiv \exists s^* \exists a^*. B(i, s^*, s) \wedge A(i, a^*, a, s) \\ \wedge (Poss(a, s) \supset Poss(a^*, s^*)) \wedge s' = do(a^*, s^*). \end{aligned}$$

Intuitively, for agent i, situation s' is accessible after action a is performed in situation s iff it is the result of doing some alternative a^* of a in some s^* accessible from s, and executability of a in s implies that of a^* in s^*. Then beliefs are defined as follows. Let $\phi(s)$ be a formula with a single situation variable s.

- Agent i believes ϕ:

$$Bel(i, \phi(now), s) \doteq \forall s'. B(i, s', s) \supset \phi(s').$$

- Agent i truly believes ϕ:

$$TBel(i, \phi(now), s) \doteq \phi(s) \wedge Bel(i, \phi(now), s).$$

Based on the situation calculus, Golog [Levesque *et al.*, 1997] has been proposed to represent complex actions obtained by the combinations of primitive actions. Golog programs are defined by the following constructs:

1. α, primitive action;
2. $\delta_1; \delta_2$, action sequence;
3. $\varphi?$, test;
4. $\delta_1 \,|\, \delta_2$, nondeterministic choice of actions;
5. $\pi x.\delta$, nondeterministic choice of arguments;
6. δ^*, nondeterministic iteration.

Conditionals and loops are defined as abbreviations. The formal semantics of Golog is usually specified by an abbreviation $Do(\delta, s, s')$, which intuitively means that executing δ brings us from situation s to s'. Another form of semantics for Golog is the transition semantics, which is based on the important concept of a configuration, denoted as a pair (δ, σ), where δ is a program (that remains to be executed) and σ a situation (of actions that have been performed). As presented in [De Giacomo *et al.*, 2000], the transition semantics for Golog programs is defined by two predicates $Trans(\delta, s, \delta', s')$ and $Final(\delta, s)$.

3 A concurrent epistemic situation calculus

In this section, we present a concurrent epistemic situation calculus with a strategy sort, which can be used to compactly represent incomplete information games where a fixed finite set of agents act simultaneously and instantly at each game step, and to reason about strategies explicitly.

We fix a set of agents AG $= \{1, \ldots, n\}$; we introduce an *agent* sort and n agent constants, for which unique names and domain closure hold. We introduce two additional sorts: an *action profile* sort and a second-order *strategy* sort. Intuitively, an action profile is an n-ary vector of actions, one action for each agent. A strategy is a function from situations to actions. Let $G \subseteq$ AG, a group strategy of coalition G is a function from G to strategies. A *strategy profile* is a group strategy of AG. We use variables d, d', ... for action profiles, f, f', ... for strategies, f_G, f'_G, ... for group strategies of G, and $\overline{G}$ to represent AG$\backslash G$. We treat f_G the same as the set of variables $\{f_i \mid i \in G\}$. When G is a singleton $\{i\}$, we simply write i.

We introduce a function $joint(a_1, \ldots, a_n)$ which maps n actions into an action profile, and n projection functions $pr_i(d)$, $1 \leq i \leq n$, which maps an action profile 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 action profiles; so the first argument of the function do and the predicate $Poss$ is of the action profile sort.

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

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

Reiter [2001] 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 action profiles can be viewed as a special case of Reiter's account.

To specify the precondition axioms for action profiles, we introduce n predicates $Poss_i(a, s)$, meaning that it is possible for agent i to perform action a in situation s. In general, the following holds: $Poss(\langle a_1, ..., a_n \rangle, s) \supset \bigwedge_{i=1}^{n} Poss_i(a_i, s)$. However, the converse does not necessarily hold: two simple actions may each be possible, their preconditions may be jointly consistent, yet intuitively they should not be concurrently possible. This is the precondition interaction problem as discussed in [Reiter, 2001].

To model turn-based games, we introduce an action $wait$, and n fluents $turn_i(s)$, meaning that it is agent i's turn to make a move. We have $Poss_i(wait, s) \equiv \neg turn_i(s)$.

To model agents' observability about action profiles, we introduce a special predicate $A'(i, j, a', a, s)$, meaning that when agent j performs action a in situation s, agent i considers it possible that agent j does action a'. In general, we let $A(i, d', d, s) \equiv \bigwedge_{j=1}^{n} A'(i, j, d'_j, d_j, s)$.

Let f_G be a group strategy of coalition G. The abbreviation $s \sqsubseteq_{f_G} s'$ is used to represent the formula

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

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

$$s \leq_{f_A} s' \doteq s \sqsubseteq_{f_A} s' \wedge s \leq s'.$$

Let $\phi(f_{AG}, s)$ be a situation calculus formula. We introduce the following abbreviations, where we use $f_{AG}(s)$ to represent the action profile at situation s, *i.e.*, $\langle f_1(s), \ldots, f_n(s) \rangle$.

- Next ϕ: $\bigcirc \phi \doteq \phi(f_{AG}, do(f_{AG}(s), s))$;
- Eventually ϕ: $\Diamond \phi \doteq \exists s' . s \sqsubseteq_{f_{AG}} s' \wedge \phi(f_{AG}, s')$.

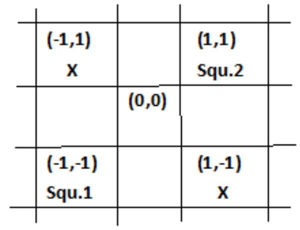

Figure 1: The Squirrels World

We say that a strategy f is an *executable strategy* of agent i, if in any situation, i knows the action required by f and its executability. Formally, we have:

$$EX(i, f) \doteq \forall s \exists a . TBel(i, f(now) = a \wedge Poss_i(a, now), s).$$

For a coalition G, we let $EX(G, f_G) \doteq \bigwedge_{i \in G} EX(i, f_i)$. In fact, the notion of executable strategy coincides with that of *uniform strategy* in the literature, say [Jamroga and van der Hoek, 2004]: a strategy f is uniform if for any situations s and s' indistinguishable for agent i, the values of f at s and s' are the same. Formally, we have:

$$\forall s, s' . B(i, s', s) \supset f(s) = f(s') \wedge Poss_i(f(s), s) \wedge Poss_i(f(s'), s').$$

In the following, we illustrate the use of our situation calculus language with a variant of Levesque's Squirrels World.

Example 1 Squirrels and acorns live in a plane, and there is a fixed set of squirrels acting simultaneously at any time. Each squirrel and acorn is located at a cell (p, q), where $p, q \in \mathbb{Z}$, the set of integers, and each cell can contain any number of acorns and squirrels.

Each squirrel can do actions below: $pick$ up an acorn if he is located at the same cell as this acorn and does not hold any acorn in the current situation; $drop$ an acorn that he is holding; move up, $down$, $right$ and $left$ a cell; also stay at the current location and do nothing (nil). A squirrel can observe the action of another squirrel within a distance of one, but if the action is a sensing action, the result is not observable.

There are only two squirrels 1 and 2. As shown in Figure 1, initially, squirrel 1 is located at the cell $(-1, -1)$ and 2 is located at the cell $(1, 1)$, and any squirrel knows the location of the other one; there is only one acorn in each of the cell $(-1, 1)$ and $(1, -1)$, and in other cells there are no acorns.

We use four ordinary fluents: in situation s, squirrel i is holding an acorn ($hold(i, s)$); squirrel i is at cell (p, q) ($cell(i, p, q, s)$); there are n acorns at cell (p, q) ($acorn(p, q, n, s)$); the situation s will be achieved after m steps from the initial situation S_0 ($step(s) = m$).

For illustration purpose, we only present some axioms of the BAT of this game, denoted by $\mathcal{D}_{sq}$:

$$Poss_i(up, s) \equiv \top, \qquad i = 1, 2;$$

$$Poss_i(pick, s) \equiv \neg hold(i, s) \wedge \exists p, q, n . cell(i, p, q, s) \wedge acorn(p, q, n, s) \wedge n > 0, \quad i = 1, 2;$$

$$Poss(\langle pick, pick \rangle, s) \equiv \textstyle\bigwedge_{i=1}^{2} Poss_i(pick, s) \wedge \neg \exists p, q . \bigwedge_{i=1}^{2} cell(i, p, q, s) \wedge acorn(p, q, 1, s);$$

$$hold(i, do(d, s)) \equiv d_i = pick \vee hold(i, s) \wedge d_i \neq drop$$

$A'(i, j, a, right, s) \equiv \exists p, p', q, q'.cell(i, p, q, s) \wedge$
$cell(j, p', q', s) \wedge (|p-p'|+|q-q'| \leq 1 \supset a = right)$
$\wedge (|p-p'|+|q-q'| > 1 \supset a = nil);$
$TBel(2, cell(1, -1, -1), S_0) \wedge TBel(1, cell(2, 1, 1), S_0);$
$\forall p, q.acorn(p, q, n, S_0) \wedge n > 0 \equiv$
$p = -1 \wedge q = 1 \vee p = 1 \wedge q = -1;$
$acorn(-1, 1, 1, S_0) \wedge acorn(1, -1, 1, S_0).$ ■

Finally, we show that our situation calculus can be used to reason about games described by GDL-II [2014]. For the purpose of General Game Playing, GDL (Game Description Language) has been developed as a high-level language for the specification of complete information games. GDL is based on the standard syntax and semantics of logic programming, characterized by a number of special keywords. GDL-II is an extension of GDL for describing imperfect/incomplete information games. It has two extra keywords: $sees$, which specifies the information that each player gets, and $random$, which denotes a special player who chooses moves randomly.

Schiffel and Thielscher present a full embedding of GDL-II into a multi-agent epistemic situation calculus, and formally prove that this provides a sound and complete reasoning method for players' knowledge about game states as well as about the knowledge of the other players. In fact, the situation calculus they use is essentially our extended situation calculus without the strategy sort. If we let $A(i, d, d', s) \doteq$

$$d_i = d'_i \wedge \forall P.Sees(i, P, d, s) \equiv Sees(i, P, d', s'),$$

which intuitively means that agent i cannot distinguish between two joint actions d and d' if their own actions are the same and the information i gets about d is the same as that about d', then their SSA for the knowledge fluent K coincides with the SSA for our B fluent. Therefore, using their embedding, we get that GDL-II can be embedded into the concurrent epistemic situation calculus.

4 Individual strategic ability

In this section, we show that in our situation calculus, we can distinguish between different kinds of individual strategic abilities, including that studied by [Lespérance *et al.*, 2000].

As discussed in the literature, say [Jamroga and Ågotnes, 2007], by considering whether the strategies of agent i or other agents are executable, and whether agent i knows which strategy to ensure his goal, there are different notions of strategic abilities of agent i. We can formalize these different abilities in our situation calculus as follows. Let $\phi(f_{AG}, s)$ be a situation calculus formula which serves as the goal for agent i. Note that by our notation, $f_{\bar{i}}$ represents a group strategy of agents other than i.

(1) $Can_1(\phi, s) \doteq \exists f_i.Bel(i, \forall f_{\bar{i}}.\phi(f_i, f_{\bar{i}}, now), s);$

(2) $Can_2(\phi, s) \doteq Bel(i, \exists f_i \forall f_{\bar{i}}.\phi(f_i, f_{\bar{i}}, now), s);$

(3) $Can_3(\phi, s) \doteq$
$\exists f_i.EX(i, f_i) \wedge Bel(i, \forall f_{\bar{i}}.\phi(f_i, f_{\bar{i}}, now), s);$

(4) $Can_4(\phi, s) \doteq$
$Bel(i, \exists f_i.EX(i, f_i) \wedge \forall f_{\bar{i}}.\phi(g_i, g_{\bar{i}}, now), s).$

In (1), i knows which strategy to ensure ϕ. In (2), i knows there exists a strategy to ensure ϕ. But in both (1) and (2), the strategy of agent i may not be executable. In (3), agent i knows which strategy to ensure the goal ϕ and this strategy is executable for him, but other agents' strategies may not be executable; in this case, agent i considers the worst case. In (4), i knows there exists an executable strategy to ensure ϕ, but does not know which strategy.

Example 1 Cont'd. We first give an abbreviation meaning that the distance between agents 1 and 2 in s is not more than k:
$dist(1, 2, s) \leq k \doteq \exists p, q, p', q'.cell(1, p, q, s) \wedge$
$cell(2, p', q', s) \wedge |p - p'| + |q - q'| \leq k.$
Also, we let $S_1 = do(\langle right, left\rangle, S_0)$. Since in S_0, any agent cannot see the action of the other one, in S_1, he no longer knows the location of the other one.

1. Let $\phi_1 = \bigcirc dist(1, 2) \leq 4$. Then $\mathcal{D}_{sq} \models Can_3(\phi_1, S_1)$. This is because when agent 1 moves up, no matter what action agent 2 does, agent 1 believes that their distance is no more than 4.
2. Let $\phi_2 = \bigcirc\exists p, q, p', q'.cell(2, p, q) \wedge cell(1, p', q') \wedge [q < 3 \wedge \neg(p = 2 \wedge q = 2) \supset |p-p'| \leq 2 \wedge |q-q'| \leq 2]$. Then $\mathcal{D}_{sq} \models Can_1(\phi_2, S_1) \wedge \neg Can_3(\phi_2, S_1) \wedge Can_4(\phi_2, S_1)$. In S_1, after agent 1 does the same action as agent 2 does in S_0, he believes that ϕ_2 holds. Thus we have $Can_1(\phi_2, S_1)$. However, since agent 1 cannot observe agent 2's action in S_0, this strategy is not executable. In fact, we cannot find an executable strategy of agent 1 which ensures ϕ_2, hence $Can_3(\phi_2, S_1)$ does not hold. On the other hand, for each possible action of agent 2 in S_0, when agent 1 does the same action in S_1, ϕ_2 holds; thus we have $Can_4(\phi_2, S_1)$. ■

Now we show that we can represent in our framework the notion of ability $Can(\varphi, s)$ defined by Lespérance *et al.* (2000) in the single-agent case, where $\varphi(s)$ is a formula about situation s:

- $OnPath(f, s, s') \doteq$
  $s \leq s' \wedge \forall a \forall s^*(s \sqsubset do(a, s^*) \sqsubseteq s' \supset f(s^*) = a);$
  This is the same as our $s \leq_f s'$.
- $CanGet(\varphi, f, s) \doteq \exists s'(OnPath(f, s, s') \wedge Bel(\varphi, s') \wedge$
  $\forall s^*[s \sqsubseteq s^* \sqsubset s' \supset \exists a Bel(f(now) = a, s^*)]);$
  Here the second line requires that g be uniform on the way from s to s'.
- $Can(\varphi, s) \doteq \exists f.Bel(CanGet(\varphi, f, now), s).$

Recall that we use Σ to denote the set of foundational axioms for our extended situation calculus. Then we get

Theorem 1 $\Sigma \models Can(\varphi, s) \equiv Can_3(\Diamond Bel(\varphi, now), s).$

5 A strategy programming language: SGolog

In this section, we propose a strategy programming language SGolog for specifying collective strategies of coalitions.

We first consider a single-step fragment (SSF) of Golog, which is used to specify an agent's possible choice to perform in one step. 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 1 SSF programs are defined inductively as follows: $\delta ::= (\varphi?;\alpha) \mid (\delta_1 \mid \delta_2) \mid (\pi x.\delta(x))$, where φ is a situation-suppressed formula, and α is an action term.

The formal semantics of SSF programs is defined by an abbreviation $Does(\delta, a, s)$, which intuitively means action a forms a legal execution of δ in situation s.

Definition 2 $Does(\delta, a, s)$ is defined inductively as:

- $Does(\varphi?;\alpha, a, s) \doteq \varphi[s] \wedge a = \alpha$;
- $Does(\delta_1 \mid \delta_2, a, s) \doteq Does(\delta_1, a, s) \vee Does(\delta_2, a, s)$;
- $Does(\pi x.\delta(x), a, s) \doteq \exists x.Does(\delta(x), a, s)$.

We use $\top$ (resp. $\bot$) to represent $true$ (resp. $false$). For convenience, we let $*$ be an extra SSF program, which intuitively means taking an arbitrary action, and define $Does(*, s, a) \doteq \top$. Next, we define SGolog programs, which can be used to represent the collective behavior/strategy of a coalition or the protocol of a multi-agent game. A primitive SGolog program is an n-ary vector of SSF programs. We use $\overline{*}$ to denote the n-ary vector of $*$'s. For a primitive program θ, we use θ_i to denote its i-th component.

Definition 3 SGolog programs are defined as follows:
$\rho ::= \theta \mid \varphi? \mid \rho_1;\rho_2 \mid \rho_1 \mid \rho_2 \mid \pi x.\rho(x) \mid \rho^* \mid$
$\quad \textbf{while}\ \varphi\ \textbf{do}\ \ \rho \mid \rho_1 \triangledown \rho_2 \triangledown ... \triangledown \rho_m \mid \rho_1 \vartriangle \rho_2 \vartriangle ... \vartriangle \rho_m$,
where θ is a primitive program, and φ is a situation suppressed formula.

A conditional [$\textbf{if}\ \varphi\ \textbf{then}\ \rho_1\ \textbf{else}\ \rho_2$] is defined as abbreviation for the program $[\varphi?;\rho_1 \mid \neg\varphi?;\rho_2]$. The prioritized disjunction and conjunction operators $\triangledown$ and $\vartriangle$ are inspired by the work of Zhang and Thielscher (2015). Intuitively, $\rho_1 \triangledown \rho_2 \triangledown ... \triangledown \rho_m$ means that if the program ρ_1 can be executed in situation s, then only ρ_1 is executed and the other programs are discarded, else if ρ_2 can be executed, then execute ρ_2, ..., *i.e.*, ρ_i has higher priority than ρ_{i+1}. The program $\rho_1 \vartriangle \rho_2 \vartriangle ... \vartriangle \rho_m$ means that if all of ρ_1, ..., and ρ_m can be applied together in s, then all of them are applied, else we discard ρ_m and consider whether the remaining ρ_i's can be applied together.

The formal semantics of SGolog programs is defined by two predicates: $Trans(\rho, s, \rho', s')$, which holds if executing one step of program ρ in situation s may lead to situation s' with ρ' remaining to be executed; and $Final(\rho, s)$, which holds if program ρ may legally terminate in situation s. The two predicates are inductively defined as follows.

1. $Trans(\theta, s, \rho', s') \equiv \exists d.Poss(d, s) \wedge s' = do(d, s) \wedge [\bigwedge_{i=1}^{n} Does(\theta_i, d_i, s)] \wedge \rho' = \top?$;
2. $Trans(\varphi?, s, \rho', s') \equiv \bot$;
3. $Trans(\rho_1;\rho_2, s, \rho', s') \equiv (\exists \rho_1'.Trans(\rho_1, s, \rho_1', s') \wedge \rho' = \rho_1';\rho_2) \vee Final(\rho_1, s) \wedge Trans(\rho_2, s, \rho', s')$;
4. $Trans(\rho_1 \mid \rho_2, s, \rho', s') \equiv Trans(\rho_1, s, \rho', s') \vee Trans(\rho_2, s, \rho', s')$;
5. $Trans(\pi x.\rho, s, \rho', s') \equiv \exists x.Trans(\rho, s, \rho', s')$;
6. $Trans(\rho^*, s, \rho', s') \equiv \exists \rho''.Trans(\rho, s, \rho'', s') \wedge \rho' = \rho'';\rho^*$;
7. $Trans(\textbf{while}\ \varphi\ \textbf{do}\ \ \rho, s, \rho', s') \equiv \exists \rho''. \varphi[s] \wedge Trans(\rho, s, \rho'', s') \wedge \rho' = \rho''; (\textbf{while}\ \varphi\ \textbf{do}\ \ \rho)$;
8. $Trans(\rho_1 \triangledown ... \triangledown \rho_m, s, \rho', s') \equiv Trans(\rho_1, s, \rho', s') \vee \{\neg\exists \rho_1', s''.Trans(\rho_1, s, \rho_1', s'') \wedge Trans(\rho_2 \triangledown ... \triangledown \rho_m, s, \rho', s')\}$;
9. $Trans(\rho_1 \vartriangle ... \vartriangle \rho_m, s, \rho', s') \equiv (\bigwedge_{k=1}^{m} \exists \rho_k'.Trans(\rho_k, s, \rho_k', s') \wedge \rho' = \rho_1' \vartriangle ... \vartriangle \rho_m') \vee \{\neg[\bigwedge_{k=1}^{m} \exists \rho_k'.Trans(\rho_k, s, \rho_k', s')] \wedge Trans(\rho_1 \vartriangle ... \vartriangle \rho_{m-1}, s, \rho', s')\}$.

1. $Final(\theta, s) \equiv \bot$;
2. $Final(\varphi?, s) \equiv \varphi[s]$;
3. $Final(\rho_1;\rho_2, s) \equiv Final(\rho_1, s) \wedge Final(\rho_2, s)$;
4. $Final(\rho_1 \mid \rho_2, s) \equiv Final(\rho_1, s) \vee Final(\rho_2, s)$;
5. $Final(\pi x.\rho, s) \equiv \exists x.Final(\rho, s)$;
6. $Final(\rho^*, s) \equiv \top$;
7. $Final(\textbf{while}\ \varphi\ \textbf{do}\ \ \rho, s, \rho', s') \equiv \varphi[s] \wedge Final(\rho, s) \vee \neg\varphi[s]$;
8. $Final(\rho_1 \triangledown ... \triangledown \rho_m), s) \equiv \bigwedge_{k=1}^{m} Final(\rho_k, s)$;
9. $Final(\rho_1 \vartriangle ... \vartriangle \rho_m, s) \equiv Final(\rho_1, s)$.

Example 1 Cont'd.

- Let $\rho_1 = (\langle up, * \rangle; \langle up, * \rangle; \langle pick, * \rangle) \triangledown (\langle right, * \rangle; \langle right, * \rangle; \langle pick, * \rangle)$. Then ρ_1 is a strategy for agent 1 in situation S_0 such that he prefers the acorn in position $(-1, 1)$.
- Let $\rho_2 = (\langle up \mid left, * \rangle; \langle up, * \rangle) \vartriangle (\langle up \mid right, * \rangle; \langle down, * \rangle)$. Since this is a prioritized conjunction, when ρ_2 is performed in S_0, agent 1 should move up twice.
- Let $\rho_3 = \langle up, down \rangle; \langle right, left \rangle$. Then ρ_3 is a group strategy for the two agents in situation S_0 to achieve the goal of meeting at $(0, 0)$. ■

We now formally define the notion that a group strategy satisfies an SGolog program. First, we introduce an abbreviation $Sat_1(f_G, \rho, s)$, which intuitively means that group strategy f_G complies with program ρ in one step from s. It is inductively defined as follows:

1. $Sat_1(f_G, \theta, s) \doteq \bigwedge_{i \in G} Does(\theta_i, f_i(s), s)$;
2. $Sat_1(f_G, \varphi?, s) \doteq \bot$;
3. $Sat_1(f_G, \rho_1;\rho_2, s) \doteq Sat_1(f_G, \rho_1, s) \vee Final(\rho_1, s) \wedge Sat_1(f_G, \rho_2, s)$;
4. $Sat_1(f_G, \rho_1 \mid \rho_2, s) \doteq Sat_1(f_G, \rho_1, s) \vee Sat_1(f_G, \rho_2, s)$;
5. $Sat_1(f_G, \pi x.\rho, s) \doteq \exists x.Sat_1(f_G, \rho, s)$;
6. $Sat_1(f_G, \rho^*, s) \doteq Sat_1(f_G, \rho, s)$;
7. $Sat_1(f_G, \textbf{while}\ \varphi\ \textbf{do}\ \ \rho, s) \equiv \varphi[s] \wedge Sat_1(f_G, \rho, s)$;
8. $Sat_1(f_G, \rho_1 \triangledown ... \triangledown \rho_m, s) \doteq \exists \rho', s'.Trans(\rho_1, s, \rho', s') \wedge Sat_1(f_G, \rho_1, s) \vee \forall \rho', s' \neg Trans(\rho_1, s, \rho', s') \wedge Sat_1(f_G, \rho_2 \triangledown ... \triangledown \rho_m, s)$;
9. $Sat_1(f_G, \rho_1 \vartriangle ... \vartriangle \rho_m, s) \doteq \exists s', d.[\bigwedge_{k=1}^{m} \exists \rho_k' Trans(\rho_k, s, \rho_k', s') \wedge s' = do(d, s) \wedge d_G = f_G(s)] \vee \neg[\exists s' \bigwedge_{k=1}^{m} \exists \rho_k' Trans(\rho_k, s, \rho_k', s')] \wedge Sat_1(f_G, \rho_1 \vartriangle ... \vartriangle \rho_{m-1}, s)$.

As for $Sat_1(f_G, \rho, s)$, we only consider one step from situation s. To represent a collective strategy of a coalition by an SGolog program, we specify how a group strategy completely complies with this program from a situation.

Definition 4 $Sat(f_G, \rho, s)$ is defined as:
$\forall s^*.s \leq_{f_G} s^* \wedge s \neq s^* \supset \exists \rho', s'.Trans^+(\rho, s, \rho', s') \wedge$
$(s^* = s' \vee Final(\rho', s') \wedge s' \sqsubset s^*)$.

Here $Trans^+$ is the transitive closure of $Trans$. Intuitively, $Sat(f_G, \rho, s)$ means that any situation s' reachable from s following f_G is either a situation resulting from the execution of ρ, or an situation following a final situation of ρ.

Obviously, we can prove that $\Sigma \models Sat(f_G, \rho, s) \supset Sat_1(f_G, \rho, s)$, that is, if f_G complies with ρ from s on, then it must comply with ρ in one step. In fact, any non-final configuration (ρ', s') which comes from configuration (ρ, s) according to f_G will conform with the remaining ρ' in s' at the next situation. That is,

Proposition 1 $\Sigma \models Sat(f_G, \rho, s) \supset \forall s', \rho'.s \leq_{f_G} s' \wedge Trans^+(\rho, s, \rho', s') \wedge \neg Final(\rho', s') \supset Sat_1(f_G, \rho', s')$.

Therefore, we can say that program ρ completely determines a collective strategy f_G from a situation s, if we have $Sat(f_G, \rho, s) \wedge \forall f'_G, s'.Sat(f'_G, \rho, s) \wedge s \leq_{f_G} s' \supset f_G(s') = f'_G(s')$. According to the above results, we can also have:

Proposition 2

- $\Sigma \models Sat(f_G, \varphi?; \rho, s) \equiv \varphi[s] \wedge Sat(f_G, \rho, s)$.
- $\Sigma \models [\bigwedge_{i \in G} Poss_i(f_i(s), s)] \supset Sat(f_G, \rho, s)$, *where* $\rho = \textbf{while} \top \textbf{ do } \overline{*}$.

6 Joint ability under an SGolog program

In this section, we adapt the approach of [Ghaderi *et al.*, 2007] to formalize joint ability of coalitions under commitments to strategy programs.

We first define the preferred strategies of agent i under an SGolog program ρ in situation s by the approach of iterated elimination of dominated strategies. Given an agent i, a group G, a strategy f_i, a program ρ, a situation s, and a goal ϕ,

$$Pref(i, G, f_i, \rho, \phi, s) \doteq \forall n.Keep(i, G, n, f_i, \rho, \phi, s).$$

So agent i prefers strategy f_i for goal ϕ under the program ρ if it is kept for all levels n, where n is a natural number.

There are two cases for the definition of $Keep$:

(1) $i \notin G$: for any n,
$Keep(i, G, n, f_i, \rho, \phi, s) \doteq EX(i, f_i) \wedge Sat(f_i, \rho, s)$.
So for agents outside the coalition, we keep those executable strategies satisfying ρ.

(2) $i \in G$: $Keep$ is inductively defined as follows:
$Keep(i, G, 0, f_i, \rho, \phi, s) \doteq EX(i, f_i) \wedge Sat(f_i, \rho, s)$;
$Keep(i, G, n+1, f_i, \rho, \phi, s) \doteq Keep(i, G, n, f_i, \rho, \phi, s) \wedge$
$\neg \exists f'_i.[Keep(i, G, n, f'_i, \rho, \phi, s) \wedge GTE(i, G, n, f'_i, f_i, \rho, \phi, s)$
$\wedge \neg GTE(i, G, n, f_i, f'_i, \rho, \phi, s)]$.
For agent i, the strategies kept at level $n+1$ are those kept at level n for which there is not a better one at level n.

GTE is defined as follows:
$GTE(i, G, n, f_i, f'_i, \rho, \phi,) \doteq$
$Bel(i, \forall f_{\bar{i}}. \bigwedge_{j \neq i} Keep(j, G, n, f_j, \rho, \phi, now) \wedge$
$\phi(f_{\bar{i}}, f'_i, now) \supset \phi(f_{\bar{i}}, f_i, now), s)$.

Strategy f_i is as good as f'_i for agent i to achieve the goal ϕ under ρ if i believes that whenever f'_i works with strategies kept by the rest of the agents so does f_i.

If we omit ρ, then $Pref(i, AG, f_i, \phi, s)$ is exactly what is defined by Ghaderi *et al.* If $i \notin G$, then we have $Pref(i, G, f_i, \rho, s) \equiv EX(i, f_i) \wedge Sat(f_i, \rho, s)$.

We now define the joint ability of a coalition to achieve a goal under commitments to an SGolog program.

Definition 5 The joint ability of coalition G to achieve goal ϕ under SGolog program ρ in situation s is defined as:
$SA(G, \rho, \phi, s) \doteq \exists f_G.\{EX(G, f_G) \wedge$
$\forall f_{\overline{G}}.[EX(\overline{G}, f_{\overline{G}}) \wedge Sat(f_G, f_{\overline{G}}, \rho, s) \supset \phi(f_G, f_{\overline{G}}, s)]\} \wedge$
$\forall f_{AG}. \wedge_{i=1}^n Pref(i, G, f_i, \rho, \phi, s) \wedge Sat(f_{AG}, \rho, s) \supset \phi(f_{AG}, s)$.

Intuitively, the meaning of $SA(G, \rho, \phi, s)$ is that first, coalition G has a group strategy that complies with the program, such that no matter which strategies complying with ρ other agents choose, ϕ holds under the strategy profile; second, all combinations of agents' preferred strategies which satisfy ρ can achieve ϕ.

The definition of joint ability depends on the infinite process of iterated elimination. However, for agents with bounded rationality, we can define the concept of bounded joint ability as follows: For $n \geq 0$, if in the above definition of $SA(G, \rho, \phi, s)$, we replace $Pref(i, G, f_i, \rho, \phi, s)$ with $Keep(i, G, n, f_i, \rho, \phi, s)$, we get the definition of the n-th level joint ability, which we denote by $SA_n(G, \rho, \phi, s)$.

Now we give some properties about $SA(G, \rho, \phi, s)$:

Proposition 3 *1. If* $\rho = \textbf{while} \top \textbf{ do } \theta$*, then for any* ρ'*,*
$\Sigma \models SA(G, \rho; \rho', \phi, s) \equiv SA(G, \rho, \phi, s)$*;*

2. $\Sigma \models SA(G, \rho, \bigcirc \top, s) \equiv \exists f_G.EX(G, f_G)$.

In fact, those specifications of abilities defined by Ghaderi et al. (2007) and in ATL can be viewed as joint abilities under some special SGolog programs. First, we formalize them in our framework with minor differences:

- $JCan(\phi, s) \doteq \exists f_G \forall f_{\overline{G}}[EX(AG, f_{AG}) \supset \Diamond \phi(f_{AG}, s)]$
$\wedge \forall f_{AG}[\bigwedge_{i \in AG} Pref(i, AG, f_i, \phi, s) \supset \Diamond \phi(f_{AG}, s)]$;
- $\langle\langle G \rangle\rangle \phi \doteq \exists f_G.EX(G, f_G) \wedge$
$\forall f_{\overline{G}}.EX(\overline{G}, f_{\overline{G}}) \supset \phi(f_{AG}, s)$.

Theorem 2 *1. Let* $\rho = \textbf{while} \top \textbf{ do } \overline{*}$*, then*
$\Sigma \models JCan(\phi, s) \equiv SA(AG, \rho, \Diamond \phi, s)$.

2. Given f_G*, there exists a corresponding SGolog program* $\rho' : \textbf{while} \top \textbf{ do } \theta$*, where* $\theta_i = f_i(now)$ *for each* $i \in G$ *and* $\theta_i = *$ *for* $i \notin G$*. We have*

$$\Sigma \models \langle\langle G \rangle\rangle \phi \equiv \exists f_G.SA(G, \rho', \phi, s).$$

Intuitively, the above two cases are two extremes of SGolog programs. For the former, no agent has any knowledge about the other agents' strategies. For the latter, the group strategy f_G is the common knowledge of group G.

From our definition of joint ability, we can see that SGolog programs serve as the common knowledge of all agents. Thus we can use SGolog programs to specify rough collective strategies, protocols, conventions, or social laws.

Example 1 Cont'd. Suppose the goal of the two squirrels is that each squirrel will hold an acorn in at most 4 steps

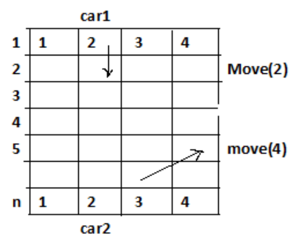

Figure 2: The Traffic Rule Example

from S_0. This can be represented with $\phi_{sq} : \Diamond(step \leq 4 \wedge hold(1) \wedge hold(2))$. We now consider whether the two squirrels have joint abilities to achieve the goal under different SGolog programs:

- Let $\rho_1 = \textbf{while} \top \textbf{do} \langle *, * \rangle$. Then $\mathcal{D}_{sq} \not\models SA(\{1,2\}, \rho_1, \phi_{sq}, S_0)$. Under this program, for squirrel 1 (resp. 2), any of his preferred strategies should move up (resp. down) or right (resp. left) in the first step. However, if squirrel 1 moves up and squirrel 2 moves left in the first step, no matter how they will act in the later steps, their goal cannot be achieved.
- Let $\theta_1 = \langle right, left \rangle$, $\theta_2 = \langle up, down \rangle$, and ρ_2 be θ_1 or θ_2. Then $\mathcal{D}_{sq} \models SA(\{1,2\}, \rho_2, \phi_{sq}, S_0)$. We explain using the example of θ_1. Under this program, the preferred strategy for squirrel 1 must be $right; right; pick$; and for squirrel 2 it must be $left; left; pick$. Therefore, any combination of the preferred strategies of the two squirrels can ensure the goal.
- Let $\rho_3 = \theta_1 \nabla \theta_2$. Then $\mathcal{D}_{sq} \models SA(\{1,2\}, \rho_3, \phi_{sq}, S_0)$. Since θ_1 can be executed in S_0, it will be executed due to higher priority.
- Let $\rho_4 = \langle *, down \rangle; \langle *, down \rangle$. Then $\mathcal{D}_{sq} \models SA_1(\{1,2\}, \rho, \phi_{sq}, S_0) \wedge \neg SA_0(\{1,2\}, \rho, \phi_{sq}, S_0)$. For goal ϕ_{sq}, initially, agent 1 keeps all those executable strategies, and agent 2 keeps those executable strategies whose first two steps both are $down$. When agent 1 moves right in the first step, the two agents cannot achieve their goal. However, at level 1, agent 1 should consider those kept strategies of agent 2 at level 0, then he only keeps those strategies whose first 3 steps are $up, up, pick$. Then any combination of the preferred strategies of the two agents can achieve the goal.

Example 2 We now consider a traffic rule example. As shown in Figure 2, there are two cars (1 and 2) driving in opposite directions on a road in which there are four lanes numbered as $1, 2, 3, 4$. There is only one action $move(k)$ which is always possible, here k is a lane. For simplicity, we assume the velocities of the two cars are the same, that is, when a car executes $move(k)$, then it moves one unit forward. For $i = 1, 2$, there is a fluent $loc_i(m, k, s)$, which means car i is on lane k with coordinate m in situation s. Two cars collide when they are at the same location. In this game, we assume that neither car can see the other car (for instance, when two cars are driving in the night with broken lights).

We have the following axioms:

$$Poss_i(move(k), s) \equiv \top, i = 1, 2;$$
$$loc_1(m, k, do(d, s)) \equiv$$
$$\exists k'.loc_1(m-1, k', s) \wedge d_1 = move(k);$$
$$loc_2(m, k, do(d, s)) \equiv$$
$$\exists k'.loc_2(m+1, k', s) \wedge d_2 = move(k);$$
$$\exists k.loc_1(0, k, S_0) \wedge \exists k'.loc_2(n, k', S_0).$$

We denote the BAT of this game as $\mathcal{D}_{tr}$, and the goal formula is $\varphi = \neg \Diamond \exists m, k.loc_1(m, k) \wedge loc_2(m, k)$, meaning that two cars will never collide.

Let $\rho = \textbf{while} \top \textbf{do} \bar{*}$. Under ρ, for any agent i, any strategy is a preferred strategy. However, if both agents choose to always perform $move(1)$, they must collide. Thus $\mathcal{D}_{tr} \not\models SA(\{1,2\}, \rho, \varphi, S_0)$. However, consider $\rho' = \textbf{while} \top \textbf{do} \langle move(1) \mid move(2), move(3) \mid move(4) \rangle$. Intuitively, ρ' requires that both agents should always drive on the right. It is easy to see that under ρ', the two cars have joint ability to avoid collision. Thus $\mathcal{D}_{tr} \models SA(\{1,2\}, \rho', \varphi, S_0)$.

7 Conclusions

In this paper, by a simple extension of a variant of multi-agent epistemic situation calculus with a strategy sort, we have developed a general framework for strategy representation and reasoning for incomplete information concurrent games. The framework can be used to compactly represent the structure of such games, distinguish between different kinds of individual strategic abilities, specify collective strategies of coalitions, and reason about joint abilities of coalitions under commitments to collective strategy specifications. Both strategic abilities in ATL and joint abilities of Ghaderi et al. can be considered as joint abilities under special strategy specifications in our framework.

In our current framework, a strategy of an agent is a function from situations to actions, thus we have made the assumption that all agents have perfect recall. But in reality, agents may have limited memory, and are even memory-free. In the future, we would like to extend our framework to accommodate bounded-memory agents. Another future work is to identify decidable fragments of our framework based on existing work, *e.g.*, [De Giacomo *et al.*, 2013].

Acknowledgments

We thank the anonymous reviewers for helpful comments. This work received support from the Natural Science Foundation of China under Grant Nos. 61572535 and 61463044.

References

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

[De Giacomo *et al.*, 2000] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1-2):109–169, 2000.

[De Giacomo *et al.*, 2010] Giuseppe De Giacomo, Yves Lespérance, and Adrian R. Pearce. Situation calculus based programs for representing and reasoning about game structures. In *Proc. of KR'10*, pages 445–455, 2010.

[De Giacomo *et al.*, 2013] Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi. Bounded epistemic situation calculus theories. In *Proc. of IJCAI'13*, pages 846–853, 2013.

[Farinelli *et al.*, 2007] Alessandro Farinelli, Alberto Finzi, and Thomas Lukasiewicz. Team programming in golog under partial observability. In *Proc. of IJCAI'07*, pages 2097–2102, 2007.

[Ghaderi *et al.*, 2007] Hojjat Ghaderi, Hector J. Levesque, and Yves Lespérance. A logical theory of coordination and joint ability. In *Proc. of AAAI'07*, pages 421–426, 2007.

[Jamroga and Ågotnes, 2007] Wojciech Jamroga and Thomas Ågotnes. Constructive knowledge: what agents can achieve under imperfect information. *Journal of Applied Non-Classical Logics*, 17(4):423–475, 2007.

[Jamroga and van der Hoek, 2004] Wojciech Jamroga and Wiebe van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2-3):185–219, 2004.

[Lespérance *et al.*, 2000] Yves Lespérance, Hector J. Levesque, Fangzhen Lin, and Richard B. Scherl. Ability and knowing how in the situation calculus. *Studia Logica*, 66(1):165–186, 2000.

[Levesque *et al.*, 1997] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. Golog: A logic programming language for dynamic domains. *J. Log. Program.*, 31(1-3):59–83, 1997.

[Liu and Levesque, 2014] Yongmei Liu and Hector J. Levesque. Incorporating action models into the situation calculus. In Alexandru Baltag and Sonja Smets, editors, *Johan van Benthem on Logic and Information Dynamics, Outstanding Contributions to Logic*, volume 5, chapter 21, pages 569–590. Springer, 2014.

[Mogavero *et al.*, 2010] Fabio Mogavero, Aniello Murano, and Moshe Y. Vardi. Reasoning about strategies. In *Proc. of FSTTCS'10*, pages 133–144, 2010.

[Osborne and Rubinstein, 1999] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, 1999.

[Ramanujam and Simon, 2008] Ramaswamy Ramanujam and Sunil Easaw Simon. Dynamic logic on games with structured strategies. In *Proc. of KR'08*, pages 49–58, 2008.

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

[Schiffel and Thielscher, 2014] Stephan Schiffel and Michael Thielscher. Representing and reasoning about the rules of general games with imperfect information. *J. Artif. Intell. Res. (JAIR)*, 49:171–206, 2014.

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

[van der Hoek and Wooldridge, 2003] Wiebe van der Hoek and Michael Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.

[van Eijck, 2013] Jan van Eijck. PDL as a multi-agent strategy logic. In *Proc. of TARK'13*, pages 206–215, 2013.

[Walther *et al.*, 2007] Dirk Walther, Wiebe van der Hoek, and Michael Wooldridge. Alternating-time temporal logic with explicit strategies. In *Proc. of TARK'07*, pages 269–278, 2007.

[Zhang and Thielscher, 2015] Dongmo Zhang and Michael Thielscher. Representing and reasoning about game strategies. *J. Philosophical Logic*, 44(2):203–236, 2015.