Synchronous Games in the Situation Calculus

(Extended Abstract)

Giuseppe De Giacomo
DIAG, Sapienza
Università di Roma
Roma, Italy
degiacomo@dis.uniroma1.it

Yves Lespérance
EECS
York University
Toronto, Canada
lesperan@cse.yorku.ca

Adrian R. Pearce
Dept. Comp. Sci. & Soft. Eng.
University of Melbourne
Melbourne, Victoria, Australia
adrianrp@unimelb.edu.au

ABSTRACT

We develop a situation calculus-based account of multi-player synchronous games. These are represented as action theories called *situation calculus synchronous game structures (SCSGSs)* that involve a single action *tick* whose effects depend on the combination of *moves* chosen by the players. Properties of games, e.g., winning conditions, playability, weak and strong winnability, etc. can be expressed in a first-order variant of alternating-time mu-calculus. Computationally effective verification can be performed. SCSGSs can be viewed as a variant of the Game Description Language (GDL) where states are represented by first-order theories.

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods.

Keywords

Logics for multi-agent systems; Verification of agent-based systems; General game playing; Reasoning about action.

1. INTRODUCTION

Many types of problems can be viewed as games, e.g., contingent planning, service orchestration, etc. Logics for reasoning about game settings has been an active area, with Alternating-Time Temporal Logic (ATL) [1] a popular choice, and model checking techniques have been used to verify properties of games specified in ATL and to synthesize agent strategies. However, such logics are usually propositional and limited to finite domains. Moreover, the game settings are usually specified using low-level automata-like languages. One exception is the Game Description Language (GDL) [6] developed for the general game playing competition, which is based on logic programming. But, GDL is intended to represent games with finite domains and its semantics based on "negation as failure" is somewhat complex.

[4] proposes an expressive first-order (FO) logical framework for specifying and reasoning about game-like problems based on the situation calculus (SC)[7]. Game settings are specified as a special kind of SC action theory. Legal moves can be specified declaratively, or procedurally in a variant of the SC programming language Golog [7]. Complex temporal properties of games can be expressed in a FO variant of alternating-time μ-calculus (μATL). Methods for verification and synthesis based on fixpoint approximation and regression are also proposed. But [4] assumes only one agent may act in any state, i.e., it deals with *turn-taking* games.

Inspired by [4], we develop a SC-based specification and verification framework that deals with multi-players *synchronous games*, and is similar in spirit to GDL. Games are represented as action theories of a special form, called *situation calculus synchronous game structures (SCSGSs),* in which we have a single action *tick* whose effects depend on the combination of *moves* selected by the players. As in [4] a FO-order variant of μATL [2] is used to specify and verify properties of the game, including winning conditions, playability, weak and strong winnability, etc.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

2. SYNCHRONOUS GAME STRUCTURES

Here, we concentrate on games where there are n players/agents each of whom chooses a move at every time step. All such moves are executed *synchronously* and determine the next state of the game. At each time step, the state of the game is fully observable by all agents, as are all past moves. These assumptions are built into GDL [6]. To represent such multi-player synchronous games, we define a special class of basic action theories [7], called *situation calculus synchronous game structures (SCSGSs)*, defined as follows.

Agents A SCSGS involves a finite set of n agents, and we introduce a subsort *Agents* of *Objects* which includes these finitely many agents $Ag_1, \ldots, Ag_n$, each denoted by a constant, and for which unique names and doman closure hold.

Moves. We also introduce a second subsort *Moves* of *Objects*, representing the possible moves of the agents. These come in finitely many types, represented by function symbols $M_i(\vec{x})$, which are parametrized by objects $\vec{x}$. Given the parameterization, each agent may have an infinite number of possible moves at each time step. We have unique name and domain closure axioms for these functions.

Actions. In SCSGSs, there is only *one action type*, $tick(m_1, \ldots, m_n)$, representing the execution of a joint move by all the agents at a given time step. *tick* has n parameters, one per agent, which are of sort *Moves* and corresponds to the simultaneous choice of the move to perform by the n agents.

Legal moves. SCSGSs include a characterization of the *legal* moves available to each agent in a given situation. This

is specified using a special predicate *LegalM* defined as:

$$LegalM(Ag_i, M_i(\vec{x}), s) \doteq \Phi_{Ag_i,M_i}(\vec{x}, s)$$

meaning that agent Ag_i can legally do move $M_i(\vec{x})$ in situation s iff $\Phi_{Ag_i,M_i}(\vec{x}, s)$ holds. Technically, *LegalM* is an abbreviation for $\Phi_{Ag_i,M_i}(\vec{x}, s)$, which is a uniform formula (i.e., a formula that only refers to a single situation s).

Precondition axioms. The precondition axiom for the action *tick* is fixed and specified in terms of *LegalM* as follows:

$$Poss(tick(m_1, \ldots, m_n), s) \equiv \bigwedge_{i=1,\ldots,n} LegalM(Ag_i, m_i, s)$$

Thus action $tick(m_1, \ldots, m_n)$ denoting the joint move of all agents can be performed iff each selected move m_i is a legal move for agent Ag_i in situation s. Since we only have one action type *tick*, this is the only precondition axiom in $\mathcal{D}_{poss}$.

Successor-state axioms. Then, we have *successor-state axioms* $\mathcal{D}_{ssa}$, specifying the effects and frame conditions of the joint moves $tick(m_1, \ldots, m_n)$ on the fluents. Such axioms, as usual in basic action theories, are domain specific, and characterize the game under consideration. Within such axioms, the agent moves that occur as arguments of *tick* determine how fluents change as a result of the joint move.

Initial situation description. Finally, the initial state of the game is axiomatized in the *initial situation description* $\mathcal{D}_0$ as usual, in a domain specific way.

Example 1. Consider an iterated version of the game Rock-Paper-Scissors. We can specify legal moves as follows:

$$LegalM(ag, play(k), s) \doteq (ag = Ag_1 \vee (ag = Ag_2) \wedge (k = Rock \vee k = Paper \vee k = Scissors)$$

We have a fluent $WinCount$ keeping track of how many rounds each agent has won, with the successor-state axiom:

$$WinCount(ag, i, do(a, s)) \equiv \phi \vee WinCount(ag, i, s) \wedge \neg\phi$$

where $\phi \doteq \exists k, l, j. a = tick(play(k), play(l)) \wedge Beats(k, l) \wedge WinCount(ag, j, do(a, s)) \wedge i = j + 1 \wedge i \leq R$

We can say an agent wins as soon as she has won R rounds. Note that we get an infinite-domain version of the game if we allow move $play(k)$ to take any integer k as argument, and define $Beats(k, l)$, e.g., as $(k - l)\ mod\ 3 = 1$. □

In the full paper [5], we show how one can also specify legal agent moves procedurally in variant of the SC programming language Golog [7]. Our SCSGSs amount to a variant of GDL where states are represented by FO theories. In [5], we show this by giving a translation of GDL specifications into SCSGSs and showing its soundness and completeness.

3. VERIFICATION AND GOAL LANGUAGE

To express properties about SCSGSs, we introduce a logic, inspired by (μ) ATL [1], based on a FO variant of the μ-calculus [2] but that works on games, by suitably considering coalitions acting towards the realization of a temporally extended goal. The key building block in this logic is the so-called *force-next* operator, $\langle\langle G\rangle\rangle \bigcirc \Psi$, meaning that coalition G can ensure that Ψ holds in the next situation. Informally, $\langle\langle G\rangle\rangle \bigcirc \Psi$ holds in a situation if there exist moves for the agents in coalition G such that for all moves by agents not in G, Ψ holds in the situation resulting from such a joint move. The syntax of our logic $\mu\mathcal{L}$ is as follows:

$$\Psi \leftarrow \varphi \mid Z \mid \Psi_1 \wedge \Psi_2 \mid \exists x.\Psi \mid \langle\langle G\rangle\rangle \bigcirc \Psi \mid \mu Z.\Psi$$

Here, φ is a (possibly open) situation-suppressed SC uniform formula, Z is a predicate variable of a given arity, and μ is the *least fixpoint* construct from the μ-calculus. We also use the usual FOL abbreviations, $[[G]] \bigcirc \Psi \doteq \neg\langle\langle G\rangle\rangle \bigcirc \neg\Psi$, and $\nu Z.\Psi(Z) \doteq \neg\mu Z.\neg\Psi(Z)$, i.e., the *greatest fixpoint* of $\Psi(Z)$. The modal μ-calculus [2] is one of the most expressive temporal logic, generalizing of CTL* for instance.

Example 2. Several key properties of games [6] can be expressed: *a) Playability*, i.e., at every step which is not terminal there exists a legal joint move: $\nu Z.Terminal \vee \langle\langle ALL\rangle\rangle \bigcirc Z$. *b) Termination*, i.e., there is a way of playing the game that eventually leads to termination: $\mu Z.Terminal \vee \langle\langle ALL\rangle\rangle \bigcirc Z$. *c) Weak Winnability (by agent Ag)*, i.e., there is a way for agent Ag to win if the others cooperate:

$$\mu Z.Terminal \wedge Goal(Ag, v) \wedge (\bigwedge_{Ag' \neq Ag} Goal(Ag', v') \wedge v' \leq v)) \vee \langle\langle ALL\rangle\rangle \bigcirc Z.$$

d) Strong Winnability (by agent Ag), i.e., there is a way for agent Ag to win no matter what the others do:

$$\mu Z.Terminal \wedge Goal(Ag, v) \wedge (\bigwedge_{Ag' \neq Ag} Goal(Ag', v') \wedge v' \leq v) \vee \langle\langle \{Ag\}\rangle\rangle \bigcirc Z.$$ □

SC reasoning techniques can be used for verifying properties of games, to analyze them and develop better players. These include sound but incomplete techniques that apply to the general setting [4], which is undecidable, and sound and complete techniques for the decidable "bounded fluent extension" setting [3]. See the full paper for details.

4. CONCLUSION

We have defined a logical framework, SCSGSs, for representing synchronous games-like systems and verifying temporal properties over them. GDL games can be represented as SCSGSs. Our framework is truly first-order and can be used to specify games/systems that involve infinite domains and an infinite set of states. Furthermore, when the SCSGS is "bounded", verification of large classes of temporal formulas is decidable. Effective reasoners for our framework can be implemented. A prototype verifier for SC game structures that uses the iterated fixpoint approximation technique [4] has been developed. In future work, we will generalize our account to partially observable game settings.

REFERENCES

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[2] J. Bradfield and C. Stirling. Modal mu-calculi. In *Handbook of Modal Logic*, volume 3, pages 721–756. Elsevier, 2007.

[3] G. De Giacomo, Y. Lespérance, and F. Patrizi. Bounded situation calculus action theories and decidable verification. In *Proc. KR*, 2012.

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

[5] G. De Giacomo, Y. Lespérance, and A. R. Pearce. Synchronous games in the situation calculus. Technical report, La Sapienza Università di Roma, 2015.

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

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