Experiments in Infinite States Verification in Game-Theoretic Logics

Slawomir Kmiec and **Yves Lespérance**
Dept. of Computer Science and Engineering,
York University, Toronto, Canada
skmiec@cse.yorku.ca and lesperan@cse.yorku.ca

Abstract

Many practical problems where the environment is not in the system's control such as service orchestration and contingent and multi-agent planning can be modelled in game-theoretic logics (e.g. ATL). But most work on verification methods for such logics is restricted to finite state cases. [De Giacomo *et al.*, 2010] develops a situation calculus-based logical framework for representing such infinite state game-type problems together with a verification method based on fixpoint approximates and regression. In this paper, we describe some case studies we have done to evaluate this method. We specify some example domains and show that the verification method does allow us to verify various properties. We also find some examples where the method must be extended to exploit information about the initial state and state constraints in order to work. Our example domains can be used to evaluate other infinite state verification methods.

1 Introduction

Many practical problems where the environment is not completely under the system's control, such as service orchestration and contingent and multi-agent planning, can be modeled as games and specified in game-theoretic logics. There has been much work to define such logics (e.g. ATL) and develop verification methods for them, mainly model checking techniques [Alur *et al.*, 2002]. However, most such work is restricted to finite state settings. [De Giacomo *et al.*, 2010] develops an expressive logical framework for specifying such problems within the situation calculus [McCarthy and Hayes, 1969]. In their approach, a game-like problem/setting is represented as a *situation calculus game structure*, a special kind of action theory that specifies who are the players, what the legal moves are, etc. They also define a logic that combines the μ-calculus, game-theoretic path quantifiers as in ATL, and first-order quantification, for specifying properties about such game settings. As well, they propose a procedural language for defining game settings, GameGolog, which is based on ConGolog [De Giacomo *et al.*, 2000]. Finally, they propose a method for verifying temporal properties over infinite state game structures that is based on fixpoint approximates and regression. The method is also adapted for GameGolog-defined settings to exploit a compact "characteristic graph" [Claßen and Lakemeyer, 2008] representation of the program's configuration space.

While [De Giacomo *et al.*, 2010] give examples to illustrate the expressiveness and convenience of their formalism, they recognize that their work is essentially theoretical and call for experimental studies to understand whether these techniques actually work in practice. This is what we begin to address in this paper. We develop several example problems involving infinite state domains and represent them as situation calculus game structures. We then examine whether the [De Giacomo *et al.*, 2010] fixpoint approximates verification method works to verify common temporal properties. In many cases, it does indeed work. So to some extent, our work validates the [De Giacomo *et al.*, 2010] proposal.

We do however find other examples where the [De Giacomo *et al.*, 2010] method does not converge in a finite number of steps. We note that the method uses only the simplest part of the action theory, the unique name and domain closure axioms, to try to show that successive approximates are equivalent (after performing regression). Clearly, using the whole action theory is problematic as it includes a second order axiom to specify the domain of situations. We show that in some cases, adding a few key facts that are entailed by the entire theory (from simple axioms about the initial state to state constraints proven by induction) is sufficient to get convergence in a finite number of steps. This means that the method can be used successfully in a wider range of problems if we can rely on the modeler to identify such facts. Thus, our case studies show that the methods introduced in [De Giacomo *et al.*, 2010] often do work for infinite domains, where very few verification methods are available, and allow reasoning about a range of game problems. Note that in our case studies, the fixpoint approximation method was performed manually. We discuss implementation in the conclusion.

2 Situation Calculus Game Structures

2.1 Situation Calculus and Basic Action Theories

The Situation Calculus (SitCalc) is a many sorted predicate logic language for representing dynamically changing worlds in which all changes are the result of named actions [Mc-

Carthy and Hayes, 1969; Reiter, 2001]. Actions are terms in the language, e.g. $pickup(R, X)$ could represent an action where a robot R picks up an object X. Action terms are denoted by α possibly with subscripts to differentiate different action terms. Action variables are denoted by lower case letters *a* possibly with subscripts. Action types, i.e. actions functions, which may require parameters, are denoted by upper case letters *A* possibly with subscripts. Situations represent possible world histories and are terms in the language. The distinguished constant S_0 denotes the initial situation where no action has yet been performed. The distinguished function symbol do is used to build sequences of actions such that $do(a, s)$ denotes the successor situation that results from performing action a in situation s. Fluents are predicates or functions whose values may vary from situation to situation. They are denoted by symbols that take a situation term as their last argument. A distinguished predicate symbol *Poss(a,s)* is used to state that an action *a* is executable in a situation *s*.

Given this language, one can specify action theories that describe how the world changes as the result of the available actions. We focus on *basic action theories* as proposed in [Reiter, 2001]. We assume that there is a *finite number of action types* in the domains we consider. Thus a basic action theory $\mathcal{D}$ is the union of the following disjoint sets: the foundational, domain independent axioms of the situation calculus (Σ); precondition axioms stating when actions can be legally performed ($\mathcal{D}_{poss}$); successor state axioms describing how fluents change between situations ($\mathcal{D}_{ssa}$); unique name axioms for actions and domain closure on action types ($\mathcal{D}_{ca}$); and axioms describing the initial configuration of the world ($\mathcal{D}_{S_0}$). Successor state axioms specify the value of fluents in situation $do(a, s)$ in terms of the action a and the value of fluents in situation s; they encode the causal laws of the world and provide a solution to the frame problem.

2.2 Situation Calculus Game Structure Definitions

Situation calculus game structures, proposed by [De Giacomo *et al.*, 2010], are a specialization of basic action theories that allow multi-agent game-like settings to be modeled. In SitCalc game structures, every action a has an agent parameter and the distinguished function $agent(a)$ returns the agent of the action. Axioms for the $agent$ function are specified for every action type and by convention the agent parameter is the first argument of any action type. It is assumed that there is a finite set $Agents$ of agents who are denoted by unique names. Actions are divided into two groups: choice actions and standard actions. Choice actions model the decisions of agents and they are assumed to have no effect on any fluent other than $Poss$, $Legal$, and $Control$. $Poss(a, s)$ specifies that an action a is physically possible (i.e. executable) in situation s. Choice actions are always physically possible. Standard actions are the other non-choice actions. There is also a distinguished predicate $Legal(s)$ that is a stronger version of possibility/legality and models the game structure of interest. It specifies what actions an agent may execute and what choices can be made according to the rules of the game. The axioms provided for $Legal$ specify the game of interest. It is required that the axioms for $Legal$ entail 3 properties:

1. $Legal$ implies physically possible
   $Legal(s) \supset s = S_0 \vee \exists a, s'.s = do(a, s') \wedge Poss(a, s')$
2. legal situations are result of an action performed in legal situations
   $Legal(s) \supset s = S_0 \vee \exists a, s'.s = do(a, s') \wedge Legal(s')$
3. only one agent can act in a legal situation
   $Legal(do(a, s)) \wedge Legal(do(a', s)) \supset agent(a) = agent(a')$

$Control(agt, s)$ holds if agent agt is the one that is in control and can act in a legal situation s; it is defined as follows:

$$Control(agt, s) \doteq \exists a.Legal(do(a, s)) \wedge agent(a) = agt$$

As a result of the above constraints on $Legal$, it follows that the predicate $Control$ holds for only one agent in a any given legal situation. As explained in [De Giacomo *et al.*, 2010], games where several agents act simultaneously can be modeled using a round-robin of choice actions; if the result of such simultaneous choices is non-deterministic, a "game master" agent that makes the decision can be introduced. It is worth noting that the state of the game in situation s is captured by the fluents. Finally, [De Giacomo *et al.*, 2010] define a SitCalc game structure to be an action theory $\mathcal{D}_{GS} = \Sigma \cup \mathcal{D}_{poss} \cup \mathcal{D}_{ssa} \cup \mathcal{D}_{ca} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{legal}$ where $\mathcal{D}_{legal}$ contains the axioms for $Legal$ and $Control$ and for the function $agent()$, and the other components are as for standard basic action theories. Note that here, a game structure is a type of situation calculus theory and not a single game model as is often the case.

2.3 Property Language

[De Giacomo *et al.*, 2010] introduces a logical language $\mathcal{L}$ for expressing temporal properties of game structures. It is inspired by ATL [Alur *et al.*, 2002] and based on the μ-calculus [Park, 1976], as used over game structures as in [Bradfield and Stirling, 2007]. The key element of the $\mathcal{L}$-logic is the $\langle\langle G\rangle\rangle \bigcirc \varphi$ operator defined as follows:

$$\begin{aligned}
&\langle\langle G\rangle\rangle \bigcirc \varphi \doteq \\
&\quad (\exists agt \in G.\ Control(agt, now) \wedge \\
&\qquad \exists a.\ agent(a) = agt \wedge \\
&\qquad\quad Legal(do(a, now)) \wedge \varphi[do(a, now)]) \vee \\
&\quad (\exists agt \notin G.\ Control(agt, now) \wedge \\
&\qquad \forall a.\ agent(a) = agt \wedge \\
&\qquad\quad Legal(do(a, now)) \supset \varphi[do(a, now)])
\end{aligned}$$

This operator, in essence, specifies that a coalition G of agents can ensure that ϕ holds next, i.e. after one more action, as follows. If an agent from the coalition G is in control in the current situation, then all we need is that there be some legal action that this agent can perform to make the formula ϕ hold. If the agent in control is not in coalition G, then what we need is that regardless of the action taken by the in-control agent (for all) the formula ϕ holds after the action. The whole logic $\mathcal{L}$ is defined as follows:

$$\begin{aligned}
&\Psi \leftarrow \varphi \mid Z(\vec{x}) \mid \Psi_1 \wedge \Psi_2 \mid \Psi_1 \vee \Psi_2 \mid \exists x.\Psi \mid \forall x.\Psi \mid \\
&\langle\langle G\rangle\rangle \bigcirc \Psi \mid [[G]] \bigcirc \Psi \mid \mu Z(\vec{x}).\Psi(Z(\vec{x})) \mid \nu Z(\vec{x}).\Psi(Z(\vec{x})).
\end{aligned}$$

In the above φ is an arbitrary, possibly open, situation-suppressed situation calculus uniform formula, Z is a predicate variable of a given arity, $\langle\langle G\rangle\rangle \bigcirc \Psi$ is as defined above,

$[[G]] \bigcirc \Psi$ is the dual of $\langle\langle G\rangle\rangle \bigcirc \Psi$ (i.e., $[[G]] \bigcirc \Psi \equiv \neg\langle\langle G\rangle\rangle \bigcirc \neg\Psi$[1]), and μ (resp. ν) is the least (resp. greatest) fixpoint operator from the μ-calculus, where the argument is written as $\Psi(Z(\vec{x}))$ to emphasize that $Z(\vec{x})$ may occur free, i.e., not quantified by μ or ν in Ψ.

The language $\mathcal{L}$ allows one to express arbitrary temporal/dynamic properties. For example, the property that group G can ensure that eventually $\varphi(\vec{x})$ (or has a strategy to achieve $\varphi(\vec{x})$), where $\varphi(\vec{x})$ is a situation suppressed formula with free variables $\vec{x}$, may be expressed by the following least fixpoint construction:

$$\langle\langle G\rangle\rangle\Diamond\varphi(\vec{x}) \doteq \mu Z(\vec{x}).\ \varphi(\vec{x}) \vee \langle\langle G\rangle\rangle \bigcirc Z(\vec{x})$$

Similarly, group G's ability to maintain a property $\varphi(\vec{x})$ can be expressed by the following greatest fixpoint construction:

$$\langle\langle G\rangle\rangle\Box\varphi(\vec{x}) \doteq \nu Z(\vec{x}).\varphi(\vec{x}) \wedge \langle\langle G\rangle\rangle \bigcirc Z(\vec{x})$$

We say that there is a path where $\varphi(\vec{x})$ holds next if the set of all agents can ensure that $\varphi(\vec{x})$ holds next: $\exists \bigcirc \varphi(\vec{x}) \doteq \langle\langle Agents\rangle\rangle \bigcirc \varphi(\vec{x})$. Similarly there is a path where $\varphi(\vec{x})$ eventually holds if the set of all agents has a strategy to achieve $\varphi(\vec{x})$: $\exists\Diamond\varphi(\vec{x}) \doteq \langle\langle Agents\rangle\rangle\Diamond\varphi(\vec{x})$.

2.4 Fixpoint Iteration Verification Method

[De Giacomo *et al.*, 2010] propose a procedure based on regression and fixpoint approximation to verify formulas of logic $\mathcal{L}$ given a SitCalc game structure theory. This recursive procedure $\tau(\Psi)$ tries to compute a first-order formula uniform in current situation *now* that is equivalent to Ψ:

$$\begin{array}{l}
\tau(\varphi) = \varphi \\
\tau(Z) = Z \\
\tau(\Psi_1 \wedge \Psi_2) = \tau(\Psi_1) \wedge\ \tau(\Psi_2) \\
\tau(\Psi_1 \vee \Psi_2) = \tau(\Psi_1) \vee \tau(\Psi_2) \\
\tau(\exists x.\Psi) = \exists x.\tau(\Psi) \\
\tau(\forall x.\Psi) = \forall x.\tau(\Psi) \\
\tau(\langle\langle G\rangle\rangle \bigcirc \Psi) = \mathcal{R}(\langle\langle G\rangle\rangle \bigcirc \tau(\Psi)) \\
\tau([[G]] \bigcirc \Psi) = \neg\mathcal{R}(\langle\langle G\rangle\rangle \bigcirc \tau(\mathrm{NNF}(\neg\Psi))) \\
\tau(\mu Z.\Psi) = \mathit{lfp}Z.\tau(\Psi) \\
\tau(\nu Z.\Psi) = \mathit{gfp}Z.\tau(\Psi)
\end{array}$$

In the above, $\mathcal{R}$ represents the regression operator and $\langle\langle G\rangle\rangle \bigcirc \Psi$ is regressable if Ψ is regressable, $\mathrm{NNF}(\neg\Psi)$ denotes the negation normal form of $\neg\Psi$, and

- $\mathit{lfp}Z.\Psi$ is the formula R resulting from the least fixpoint procedure

$$\begin{array}{l}
R := False; \\
R_{new} := \Psi(False); \\
\textbf{while } (\mathcal{D}_{ca} \not\models R \equiv R_{new})\{ \\
\quad R := R_{new}; \\
\quad R_{new} := \Psi(R)\ \}
\end{array}$$

- $\mathit{gfp}Z.\Psi$ is the formula R resulting from the greatest fixpoint procedure

$$\begin{array}{l}
R := True; \\
R_{new} := \Psi(True)]; \\
\textbf{while } (\mathcal{D}_{ca} \not\models R \equiv R_{new})\{ \\
\quad R := R_{new}; \\
\quad R_{new} := \Psi(R)\ \}
\end{array}$$

[1]Although $\neg\langle\langle G\rangle\rangle \bigcirc \neg\Psi$ is not in $\mathcal{L}$ according to the syntax, the equivalent formula in negation normal form is.

The fixpoint procedures test if $R \equiv R_{new}$ is entailed given only the unique name and domain closure for actions axioms $\mathcal{D}_{ca}$. In general, there is no guarantee that such procedures will ever terminate i.e. that for some i $\mathcal{D}_{ca} \models R_i \equiv R_{i+1}$. But if the *lfp* procedure does terminate, then $\mathcal{D}_{GS} \models R_i[S] \equiv \mu Z.\Psi(Z)[S]$ and R_i is first-order and uniform in S (and similarly *gfp*). In such cases, the task of verifying a fixpoint formula in the situation calculus is reduced to that of verifying a first-order formula. We have the following result:

Theorem 1. *[De Giacomo* et al.*, 2010] Let $\mathcal{D}_{GS}$ be a situation calculus game structure and let Ψ be an $\mathcal{L}$-formula. If the algorithm above terminates, then $\mathcal{D}_{GS} \models \Psi[S_0]$ iff $\mathcal{D}_{S_o} \cup \mathcal{D}_{ca} \models \tau(\Psi)[S_0]$.*

3 Case Studies

3.1 Light World (LW)

Our first example domain is the Light World (LW), a simple game we designed that involves an infinite row of lights, one for each integer. A light can be on or off. Each light has a switch that can be flipped, which will turn the light on (off resp.) if it was off (on resp.). There are 2 players, X and O. Players take turns and initially it is X's turn. The goal of player X is to have lights 1 and 2 on in which case player X wins the game. Initially, only light 5 is on. Note that this is clearly an infinite state domain as the set of lights that can be turned on or off is infinite. Note also that the game may go on forever without the goal being reached (e.g., if player O keeps turning light 1 or 2 off whenever X turns them on).

We will show that the fixpoint approximation method of [De Giacomo *et al.*, 2010] can be used to verify some interesting properties in this domain. We apply the method with one small modification: when checking whether the two successive approximates are equivalent, we use a suitable axiomatization of the integers D_Z in addition to the unique names and domain closure axioms for actions D_{ca}^{LW}, as our game domain involves one light for every integer. [2]

The game structure axiomatization for this domain is:
$\mathcal{D}_{GS}^{LW} = \Sigma \cup \mathcal{D}_{poss}^{LW} \cup \mathcal{D}_{ssa}^{LW} \cup \mathcal{D}_{ca}^{LW} \cup \mathcal{D}_{S_0}^{LW} \cup \mathcal{D}_{Legal}^{LW} \cup \mathcal{D}_Z$.
We have only one action $flip(p,t)$, meaning that player p flips light t, with the precondition axiom (in $\mathcal{D}_{poss}^{LW}$):

$$Poss(flip(p,t), s) \equiv Agent(p)$$

We have the fluents $On(t,s)$, meaning that light t is on in situation s, and $turn(s)$, a function that denotes the agent whose turn it is in s. The successor state axioms (in $\mathcal{D}_{ssa}^{LW}$) are as follows:

$$\begin{array}{l}
On(t, do(a,s)) \equiv \exists p\, a = flip(p,t) \wedge \neg On(t,s) \vee \\
\quad On(t,s) \wedge \forall p.a \neq flip(p,t) \\
\\
turn(do(a,s)) = p \equiv \\
\quad p = O \wedge turn(s) = X \vee p = X \wedge turn(s) = O
\end{array}$$

[2]Our axioms and the properties we attempt to verify only use a very simple part of integer arithmetic. It should be possible to generate the proofs using the decidable theory of Presburger arithmetic [Enderton, 1972] after encoding integers as pairs of natural numbers in the standard way [Hamilton, 1982]. Most theorem proving systems include sophisticated solvers for dealing with formulas involving integer constraints and it should be possible to use these to perform the reasoning about integers that we require.

The rules of the game are specified using the $Legal$ predicate. We have the following axioms in $\mathcal{D}^{LW}_{legal}$:

$$Legal(do(a,s)) \equiv Legal(s) \wedge \\ \exists p,t.\ Agent(p) \wedge turn(s) = p \wedge a = flip(p,t)$$

$$agent(flip(p,t)) = p$$
$$Control(p,s) \doteq \exists a.Legal(do(a,s)) \wedge agent(a) = p$$
$$\forall p.\{Agent(p) \equiv (p = X \vee p = O)\}, \qquad X \neq O$$

Thus legal moves involve the player whose turn it is flipping any switch. We have the following unique name and domain closure axioms for actions in $\mathcal{D}^{LW}_{ca}$:

$$\forall a.\ \{\ \exists p,t.\ a = flip(p,t)\}$$
$$\forall p,p',t,t'.\ \{\ flip(p,t) = flip(p',t') \supset p = p' \wedge t = t'\ \}$$

Finally, the initial state axioms in $\mathcal{D}^{LW}_{S_0}$ are: $turn(S_0) = X$, $\neg On(1,S_0)$, $\neg On(2,S_0)$, $On(5,S_0)$, and $Legal(S_0)$.

For our first verification example, we consider the property that it is possible for X to eventually win (assuming O cooperates), which can be represented by the following formula:

$$\exists \Diamond Wins(X) \doteq \mu Z.Wins(X)\ \vee\ \exists \bigcirc Z,$$

where $Wins(X,s) \doteq Legal(s) \wedge On(1,s) \wedge On(2,s)$. We apply the [De Giacomo *et al.*, 2010] fixpoint iteration verification method to this example. We can show that the regressed approximations simplify as follows (see [Kmiec, 2013] for a more detailed version of all proofs in this paper):

$\mathcal{D}^{\mathbf{LW}}_{\mathbf{ca}} \models \mathbf{R_0(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{False}) \equiv$
$Legal(s) \wedge On(1,s) \wedge On(2,s)$
This approximation evaluates to true if s is such that X is winning in s already (in no steps), i.e., if light 1 and light 2 are on in s.

$\mathcal{D}^{\mathbf{LW}}_{\mathbf{ca}} \cup \mathbf{D_Z} \models \mathbf{R_1(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_0}) \equiv$
$Legal(s) \wedge On(1,s) \wedge On(2,s) \vee$
$Legal(s) \wedge (turn(s) = X \vee turn(s) = O) \wedge On(1,s) \vee$
$Legal(s) \wedge (turn(s) = X \vee turn(s) = O) \wedge On(2,s)$
This approximation evaluates to true if s is such that X can win in at most 1 step; these are legal situations where player X is already winning or where one of lights 1 or 2 is on, as X or O can turn the other light on at the next step.

$\mathcal{D}^{\mathbf{LW}}_{\mathbf{ca}} \cup \mathbf{D_Z} \models \mathbf{R_2(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_1}) \equiv$
$Legal(s) \wedge On(1,s) \wedge On(2,s) \vee$
$Legal(s) \wedge (turn(s) = X \vee turn(s) = O)$
This approximation evaluates to true if s is such that X can win in at most 2 steps; this is the case if X is winning already or if s is any legal situation where it is one of the players' turn, as the controlling player can turn light 1 on at the next step and the other player can and light 2 on at the second step).

$\mathcal{D}^{\mathbf{LW}}_{\mathbf{ca}} \cup \mathbf{D_Z} \models \mathbf{R_3(s)} \equiv \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_2}) \equiv$
$Legal(s) \wedge On(1,s) \wedge On(2,s) \vee$
$Legal(s) \wedge (turn(s) = X \vee turn(s) = O)$
The fixpoint iteration procedure converges at the 4^{th} step as we have: $\mathcal{D}^{LW}_{ca} \cup D_Z \models R_2(s) \equiv R_3(s)$. By the way, note that it can be shown using the entire theory (by induction on situations) that $\mathcal{D}^{LW}_{GS} \models R_2(s) \equiv Legal(s)$, as it is always either X's or O's turn. In essence, it is possible for X to eventually win in any legal situation. It then follows by Theorem 1 of [De Giacomo *et al.*, 2010] that: $\mathcal{D}^{LW}_{GS} \models \exists \Diamond Wins(X)[S_0]$ iff $\mathcal{D}^{LW}_{GS} \models Legal(S_0) \wedge \{On(1,S_0) \wedge On(2,S_0) \vee turn(S_0) = X \vee turn(S_0) = O\}$. By the initial state axioms, the latter holds so $\mathcal{D}^{LW}_{GS} \models \exists \Diamond Wins(X)[S_0]$, i.e., player X can eventually win in the initial situation.

For our second verification example, we look at the property that X can ensure that he/she eventually wins no matter what O does, i.e., the existence of a strategy that ensures $Wins(X)$. This can be represented by the following formula:

$$\langle\langle\{X\}\rangle\rangle \Diamond Wins(X) \doteq \mu Z.\ Wins(X) \vee \langle\langle\{X\}\rangle\rangle \bigcirc Z$$

We apply the [De Giacomo *et al.*, 2010] method to try verify this property. We can show that the regressed approximations simplify as follows:

$\mathcal{D}^{\mathbf{LW}}_{\mathbf{ca}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_0(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\langle\langle\{\mathbf{X}\}\rangle\rangle \bigcirc \mathbf{False})$
$\equiv Legal(s) \wedge On(1,s) \wedge On(2,s)$
This approximation evaluates to true if s is such that X is already winning in s (in no steps); these are situations where lights 1 and 2 are already on.

$\mathcal{D}^{\mathbf{LW}}_{\mathbf{ca}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_1(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\langle\langle\{\mathbf{X}\}\rangle\rangle \bigcirc \mathbf{R_0})$
$\equiv$ $Legal(s) \wedge On(1,s) \wedge On(2,s) \vee$
$Legal(s) \wedge turn(s) = X \wedge On(1,s) \vee$
$Legal(s) \wedge turn(s) = X \wedge On(2,s)$
This approximation evaluates to true if s is such that X can ensure it wins in at most 1 step. This holds if lights 1 and 2 are already on or if either light 1 or 2 is on and it is player X's turn, as X can then turn the other light on at the next step.

$\mathcal{D}^{\mathbf{LW}}_{\mathbf{ca}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_2(s)} \equiv \mathbf{Wins(X,s)} \vee \mathcal{R}(\langle\langle\{\mathbf{X}\}\rangle\rangle \bigcirc \mathbf{R_1})$
$\equiv$ $Legal(s) \wedge On(1,s) \wedge On(2,s) \vee$
$Legal(s) \wedge turn(s) = X \wedge On(1,s) \vee$
$Legal(s) \wedge turn(s) = X \wedge On(2,s)$
Thus the fixpoint iteration procedure converges in the 3^{rd} step as we have: $\mathcal{D}^{LW}_{ca} \cup \mathcal{D}_Z \models R_1(s) \equiv R_2(s)$. Therefore by Theorem 1 of [De Giacomo *et al.*, 2010]: $\mathcal{D}^{LW}_{GS} \models \langle\langle\{X\}\rangle\rangle \Diamond Wins(X)[S_0] \equiv R_1(S_0)$ Since both lights 1 and 2 are off initially, it follows by the initial state axioms that $\mathcal{D}^{LW}_{GS} \models \neg\langle\langle\{X\}\rangle\rangle \Diamond Wins(X)[S_0]$, i.e., there is no winning strategy for X in S_0. However, we also have that $\mathcal{D}^{LW}_{GS} \models \langle\langle\{X\}\rangle\rangle \Diamond Wins(X)[S_1]$, where $S_1 = do(flip(O,3), do(flip(X,1), S_0))$, i.e., X has a winning strategy in the situation S_1 where X first turned light 1 on and then O flipped light 3, as X can turn on light 2 next.

Note that when the fixpoint approximation method is able to show that a coalition can ensure that a property holds eventually, and the theory is complete and we have domain closure, we can always extract a strategy that the coalition can follow to achieve the property: a strategy works if it always selects actions for the coalition that get it from one approximate to a lower approximate (R_i to R_{i-1}).

3.2 Oil Lamp World (OLW)

The [De Giacomo *et al.*, 2010] fixpoint approximation method tries to detect convergence by checking if the i-th approximate is equivalent to the $(i+1)$-th approximate using only the unique name and domain closure axioms for actions D_{ca} (to which we have added the axiomatization of the integers). We now give an example where this method does not converge in a finite number of steps. However, we also show

that if we use some additional facts that are entailed by the entire theory D_{GS}^{OLW}, including the initial state axioms, when checking if successive approximates are equivalent, then we do get convergence in a finite number of steps.

Consider the Oil Lamp World (OLW), a variant of the Light World (LW) domain discussed earlier. It also involves an infinite row of lamps one for each integer, which can be on or off. A lamp has an igniter that can be flipped. When this happens, the lamp will go on provided that the lamp immediately to the right is already on, i.e., flipping the igniter for lamp t will turn it on if lamp $t+1$ is already on. There is only one agent, X. The goal of X is to have lamp 1 on, in which case X wins. Observe that the game may go on indefinitely without the goal being reached, e.g., if X keeps flipping a lamp other than lamp 1 repeatedly.

The game structure axiomatization for this domain is: $\mathcal{D}_{GS}^{OLW} = \Sigma \cup \mathcal{D}_{poss}^{OLW} \cup \mathcal{D}_{ssa}^{OLW} \cup \mathcal{D}_{ca}^{OLW} \cup \mathcal{D}_{S_0}^{OLW} \cup \mathcal{D}_{Legal}^{LW} \cup \mathcal{D}_Z$. As in the previous example, we have only one action, $flip(p,t)$, meaning that p flips the igniter on light t, with the following precondition axiom (in $\mathcal{D}_{poss}^{LW}$):

$$Poss(flip(p,t), s) \equiv Agent(p)$$

But there is no turn taking in this game as there is only one agent X. We have the successor state axiom (in $\mathcal{D}_{ssa}^{LW}$):

$$On(t, do(a,s)) \equiv \\ \exists p\, a = flip(p,t) \wedge On(t+1, s) \vee On(t,s)$$

Note that once a lamp is turned on it remains on. The rules of the game are specified by the axioms in $\mathcal{D}_{legal}^{LW}$ as follows:

$$Legal(do(a,s)) \equiv \\ Legal(s) \wedge \exists p,t.\ Agent(p) \wedge a = flip(p,t)$$

$$agent(flip(p,t)) = p$$
$$Control(p,s) \doteq \exists a. Legal(do(a,s)) \wedge agent(a) = p$$
$$\forall p. \{Agent(p) \equiv p = X\}$$

Thus legal moves involve X flipping any igniter. The unique name and domain closure axioms for actions and the initial state axioms are exactly as in the Light World example.

We are interested in verifying the property that it is possible for X to eventually win $\exists \Diamond Wins(X)$, where $Wins(X,s) \doteq Legal(s) \wedge On(1,s)$. We begin by applying the [De Giacomo *et al.*, 2010] method and try to show that successive approximates are equivalent using only the unique name and domain closure axioms for actions $\mathcal{D}_{ca}^{OLW}$ and the axiomatization of the integers $\mathcal{D}_Z$. We can show that the regressed approximations simplify as follows:

$$\mathcal{D}_{\mathbf{ca}}^{\mathbf{OLW}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_0(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{False}) \equiv \\ Legal(s) \wedge On(1,s)$$

This approximation evaluates to true if s is such that X is already winning (in no steps); these are situations where lamp 1 is on.

$$\mathcal{D}_{\mathbf{ca}}^{\mathbf{OLW}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_1(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_0}) \equiv \\ Legal(s) \wedge (On(1,s) \vee On(2,s))$$

This approximation evaluates to true if s is such that X can win in at most 1 step; these are legal situations where either lamp 1 is on or where lamp 2 is on, and then X can turn lamp 1 on at the next step.

$$\mathcal{D}_{\mathbf{ca}}^{\mathbf{OLW}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_2(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_1}) \equiv \\ Legal(s) \wedge (On(1,s) \vee On(2,s) \vee On(3,s))$$

This approximation evaluates to true if s is such that X can win in at most 2 steps; these are legal situations where either lamp 1 is on, or where lamp 2 is on (and then X can turn lamp 1 on at the next step), or where lamp 3 is on (and then X can turn on lamps 2 and 1 at the next steps).

We can generalize and show that for all natural numbers i

$$D_{ca}^{OLW} \cup D_Z \models R_i \equiv Legal(s) \wedge \bigvee_{1 \le j \le i+1} On(j,s)$$

That is, X can win in at most i steps if some lamp between 1 and $i+1$ is on. It follows that for all i, $D_{ca}^{OLW} \cup D_Z \not\models R_i \equiv R_{i+1}$, since one can always construct a model of $D_{ca}^{OLW} \cup D_Z$ where every light except $i+2$ is off . Thus, the plain [De Giacomo *et al.*, 2010] method fails to converge in a finite number of steps.

Nonetheless, there is a way to beef up the [De Giacomo *et al.*, 2010] method to get convergence in a finite number of steps. The idea is to use some facts that are entailed by the entire theory in addition to the the unique name and domain closure axioms for actions D_{ca}^{OLW} and the integer axioms D_Z. First, we can show by induction on situations that any lamp that is on in the initial situation will remain on forever:

$$\mathcal{D}_{GS}^{OLW} \models \phi_{op} \\ \text{where } \phi_{op} \doteq \forall k \{On(k, S_0) \supset \forall s\, On(k,s)\}$$

Then, it follows that for any natural numbers $i, j, i \le j$,

$$\mathcal{D}_{ca}^{OLW} \cup \mathcal{D}_Z \cup \{On(i+1, S_0), \phi_{op}\} \models R_j \equiv Legal(s)$$

In essence, X can eventually win in any legal situation where some lamp n is known to be on. It follows that:

$$\mathcal{D}_{ca}^{OLW} \cup \mathcal{D}_Z \cup \{On(i+1, S_0), \phi_{op}\} \models R_i \equiv R_{i+1}$$

Thus the fixpoint approximation method converges in a finite number of steps if we use the facts that some lamp n is known to be on initially and that a lamp that is on initially remains on forever. Moreover, our initial state axioms include $On(5, S_0)$. Thus, $\mathcal{D}_{GS}^{OLW} \models \exists \Diamond Wins(X)[S_0]$, i.e., X can eventually win in the initial situation, as it is legal and lamp 5 is on.

We can also show by induction on situations that if all lamps are off initially, they will remain so forever:

$$\mathcal{D}_{GS}^{OLW} - \mathcal{D}_{S_0}^{OLW} \models (\forall k\, \neg On(k, S_0)) \supset (\forall s \forall k\, \neg On(k,s))$$

Then, we can show by a similar argument as above that the fixpoint approximation method converges in a finite number of steps if we use the facts that all lamp are off initially and that if all lamps are off initially, they remain off forever.

3.3 In-Line Tic-Tac-Toe (TTT1D)

Our final example domain is more like a traditional game. It involves a one-dimensional version of the well known tic-tac-toe game that is played on an infinite vector of cells, one for each integer. We show that the [De Giacomo *et al.*, 2010] fixpoint approximation method does work to verify both winnability and the existence of a winning strategy in this game, although in the former case the proof is long and tedious. There are two players, X and O, that take turns,

with X playing first. All cells are initially blank, i.e. marked B. Players can only put their mark at the left or right edge of the already marked area. The functional fluent $curn$ denotes the marking position on the left (negative) side of the marked area and similarly $curp$ denotes the marking position on the right (positive) side of the marked area. Initially, $curn$ refers to cell 0 and $curp$ to cell 1. Player p can put its mark in the cell on the left (negative) side of the marked area, i.e. the cell referred to by $curn$, by doing the action $markn(p)$. This also decreases the value $curn$ by 1 so that afterwards, it points to the next cell on the left. There is an analogous action $markp(p)$ for marking the the cell on the right (positive) side of the marked area denoted by $curp$. A player wins if it succeeds in putting its mark in 3 consecutive cells. For example, if initially we have the following sequence of moves:

$$[markp(X), markn(O), markp(X), markn(O), markp(X)],$$

then in the resulting situation the board is as follows:

$$\ldots, B_{-3}, B_{-2}, O_{-1}, O_0, X_1, X_2, X_3, B_4, B_5, \ldots$$

(with the subscript indicating the cell number) and X wins. Note that the game may go on indefinitely without either player winning, for instance if player O always mimics the last move of player X.

The game structure axiomatization for this domain is: $\mathcal{D}_{GS}^{T^3 1D} = \Sigma \cup \mathcal{D}_{poss}^{T^3 1D} \cup \mathcal{D}_{ssa}^{T^3 1D} \cup \mathcal{D}_{ca}^{T^3 1D} \cup \mathcal{D}_{S_0}^{T^3 1D} \cup \mathcal{D}_{Legal}^{T^3 1D} \cup \mathcal{D}_Z$. The precondition axioms (in $\mathcal{D}_{poss}^{T^3 1D}$) state that the actions $markn(p)$ and $markp(p)$ are always possible if p is an agent. The successor state axioms (in $\mathcal{D}_{ssa}^{LW}$) are as follows:

$$\begin{aligned}
&curn(do(a,s)) = k \equiv \\
&\quad \exists p.\{a = markn(p)\} \wedge curn(s) = k+1 \\
&\quad \vee curn(s) = k \wedge \forall p.\{a \neq markn(p)\} \\
&curp(do(a,s)) = k \equiv \\
&\quad \exists p.\{a = markp(p)\} \wedge curp(s) = k-1 \\
&\quad \vee curp(s) = k \wedge \forall p.\{a \neq markn(p)\} \\
&cell(k, do(a,s)) = p \equiv \\
&\quad a = markp(p) \wedge curp(s) = k \ \vee \\
&\quad a = markn(p) \wedge curn(s) = k \vee \\
&\quad cell(k,s) = p \\
&\qquad \wedge \neg \exists p'.\{a = markp(p') \wedge curp(s) = k\} \\
&\qquad \wedge \neg \exists p'.\{a = markn(p') \wedge curn(s) = k\} \\
&turn(do(a,s)) = p \equiv \\
&\quad agent(a) = X \wedge p = O \wedge turn(s) = X \\
&\quad \vee agent(a) = O \wedge p = X \wedge turn(s) = O
\end{aligned}$$

The rules of the game are specified (in $\mathcal{D}_{legal}^{T^3 1D}$) as follows:

$$\begin{aligned}
&Legal(do(a,s)) \equiv Legal(s) \wedge \\
&\quad \exists p.\{\, turn(s) = p \wedge agent(a) = p \\
&\qquad \wedge (a = markn(p) \ \vee a = markp(p))\ \} \\
&agent(markn(p)) = p, \qquad agent(markp(p)) = p \\
&Control(p,s) \doteq \exists a. Legal(do(a,s)) \wedge agent(a) = p \\
&\forall p.\ \{\ Agent(p) \equiv (p = X \vee p = O)\}, \qquad X \neq O
\end{aligned}$$

The unique name and domain closure axioms for actions are specified in the usual way. Finally, we have the following initial state axioms in $\mathcal{D}_{S_0}^{T^3 1D}$: $curn(S_0) = 0$, $curp(S_0) = 1$, $turn(S_0) = X$, and $Legal(S_0)$.

We first consider the property that it is possible for X to eventually win $\exists \Diamond Wins(X)$, where

$$\begin{aligned}
&Wins(p,s) \doteq \exists k(Legal(s) \wedge \\
&\quad ((curn(s) = k-2 \wedge cell(k-1,s) = p \wedge \\
&\qquad cell(k,s) = p \wedge cell(k+1,s) = p) \vee \\
&\quad (curp(s) = k+2 \wedge cell(k+1,s) = p \wedge \\
&\qquad cell(k,s) = p \wedge cell(k-1,s) = p)))
\end{aligned}$$

(Note that this simple definition allows both players to win.) If we apply the original [De Giacomo *et al.*, 2010] method to this property (using only the unique name and domain closure axioms for actions $\mathcal{D}_{ca}^{T^3 1D}$ and the axiomatization of the integers $\mathcal{D}_Z$ to show that successive approximates are equivalent), the fixpoint approximation procedure does eventually converge, but only after 11 steps. The proof is very long and tedious and there are numerous cases to deal with. The reason for this is that we cannot use the fact that $curn$ is always less than $curp$ and that the cells that are between them are non-blank and that the other cells are blank, which are consequences of the initial state axioms. So our proof has to deal with numerous cases where there are non-blank cells to the left of $curn$ or to the right of $curp$ (if we can rule these out, the proof becomes much simpler). Let us sketch the proof. We can show that the regressed approximations simplify as follows:

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_0(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{False}) \equiv$
$Wins(X,s)$

This approximation evaluates to true if s is such that X is winning in s already (in no steps); these are legal situations where there are 3 X marks in a row on either side.

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_1(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_0}) \equiv$
$R_0(s) \vee XCanPlayToWinNext(s)$

where $XCanPlayToWinNext(s) \doteq$
$Legal(s) \wedge turn(s) = X \wedge \exists k\{$
$(curn(s) = k-1 \wedge cell(k,s) = X \wedge cell(k+1,s) = X) \vee$
$(cell(k-1,s) = X \wedge cell(k,s) = X \wedge curp(s) = k+1) \vee$
$(cell(k-1,s) = X \wedge curn(s) = k \wedge$
$\quad cell(k+1,s) = X \wedge curp(s) = k+2) \vee$
$(cell(k-1,s) = X \wedge cell(k,s) = X \wedge$
$\quad curn(s) = k+1 \wedge curp(s) = k+2) \vee$
$(curn(s) = k-2 \wedge curp(s) = k-1 \wedge$
$\quad cell(k,s) = X \wedge cell(k+1,s) = X) \vee$
$(curn(s) = k-2 \wedge cell(k-1,s) = X \wedge$
$\quad curp(s) = k \wedge cell(k+1,s) = X)\}$

This approximation evaluates to true if s is such that it is possible for X to win in at most 1 step. These are legal situations where there are 3 X marks in a row on either side, i.e. $\uparrow_n XXX$ or $XXX \uparrow_p$ ($\uparrow_n$ representing the position of $curn$ and similarly for $\uparrow_p$ and $curp$), or where it is X's turn and there are 2 X marks already and X can fill in the missing cell to get 3 in a row, i.e. $\uparrow_n XX$ or $XX \uparrow_p$ or $\uparrow_n \uparrow_p XX$ or $XX \uparrow_n \uparrow_p$ or $\uparrow_n X \uparrow_p X$ or $X \uparrow_n X \uparrow_p$.

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_2(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_1}) \equiv$
$R_1(s) \vee$
$Legal(s) \wedge turn(s) = O \wedge \exists k\{$

$(curp(s) < k-1 \wedge curn(s) = k-1 \wedge$
$\quad cell(k,s) = X \wedge cell(k+1,s) = X) \vee$
$(cell(k-1,s) = X \wedge cell(k,s) = X \wedge$
$\quad curp(s) = k+1 \wedge k+1 < curn(s)) \vee$
$(curn(s) < k-1 \wedge cell(k-1,s) = X \wedge$
$\quad cell(k,s) = X \wedge curp(s) = k+1) \vee$
$(curn(s) = k-1 \wedge cell(k,s) = X \wedge$
$\quad cell(k+1,s) = X \wedge k+1 < curp(s)) \vee$
$cell(k-1,s) = X \wedge cell(k,s) = X \wedge$
$\quad k+1 = curn(s) \wedge curp(s) = k+1) \vee$
$(cell(k-1,s) = X \wedge cell(k,s) = X \wedge$
$\quad curn(s) = k+2 \wedge curp(s) = k+2) \vee$
$(curn(s) = k-1 \wedge curp(s) = k-1 \wedge$
$\quad cell(k,s) = X \wedge cell(k+1,s) = X) \vee$
$(curn(s) = k-2 \wedge curp(s) = k-2 \wedge$
$\quad cell(k,s) = X \wedge cell(k+1,s) = X)\}$

This approximation evaluates to true if s is such that it is possible for X to win in at most 2 steps. These are legal situations where X can win in at most 1 step as above, or where it is O's turn, and O can do a move that doesn't interfere and the result is a situation where X can win in at most 1 step; for this we have 8 cases: $\uparrow_n XX$ with $\uparrow_p < \uparrow_n$ or $XX \uparrow_p$ with $\uparrow_n > \uparrow_p$ or $X_kX \uparrow_p$ with $\uparrow_n < k$ or $\uparrow_n XX_k$ with $\uparrow_p > k$ or $XX \uparrow_{n,p}$ or $XX\_ \uparrow_{n,p}$ or $\uparrow_{n,p} XX$ or $\uparrow_{n,p} \_XX$.

We can keep going in this way. As we allow more steps, we have more and more classes of configurations where X can win. We can show that:

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_{10}(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_9}) \equiv$
$\quad Legal(s)$

Thus, it is possible for X to win in at most 10 steps in all legal situations. And we also have that:

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_{11}(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\exists \bigcirc \mathbf{R_{10}}) \equiv$
$\quad Legal(s)$

i.e., it is possible for X to win in at most 11 steps in all legal situations. Thus, the fixpoint approximation procedure converges in the 11^{th} step as we have: $\mathcal{D}_{ca}^{T^3 1D} \cup \mathcal{D}_Z \models R_{10}(s) \equiv R_{11}(s)$. There are situations where it does take at least 10 steps/moves for X to win, for instance if we have $\uparrow_p < \uparrow_n$ with two blank cells in between, i.e., $\uparrow_p BB \uparrow_n$, and it is O's turn. The fact that $\uparrow_p < \uparrow_n$ means that the initial marks that are made will later be overwritten. It is straightforward to check that it takes at least 10 moves for X to have 3 X's in a row and win (O wins as well), for instance if O keeps playing $markn$ and X keeps playing $markp$. It follows from our convergence result by Theorem 1 of [De Giacomo *et al.*, 2010] that: $\mathcal{D}_{GS}^{T^3 1D} \models \exists \Diamond Wins(X)[S_0] \equiv R_{10}(S_0) \equiv Legal(S_0)$. Since we have $Legal(S_0)$ in the initial state axioms, it follows that $\mathcal{D}_{GS}^{T^3 1D} \models \exists \Diamond Wins(X)[S_0]$, i.e., it is possible for X to win in the initial situation.

Finally, we consider the property that X can ensure that it eventually wins $\langle\langle\{X\}\rangle\rangle \Diamond Wins(X)$. We can apply the original [De Giacomo *et al.*, 2010] method to this property (using only the unique name and domain closure axioms for actions $\mathcal{D}_{ca}^{T^3 1D}$ and the axiomatization of the integers $\mathcal{D}_Z$ to show that successive approximates are equivalent), and the fixpoint iteration converges after only 3 steps. We can show that the regressed approximations simplify as follows:

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_0(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\langle\langle\{\mathbf{X}\}\rangle\rangle \bigcirc \mathbf{False})$
$\quad \equiv Wins(X,s)$

This approximation evaluates to true if s is such that X is winning in s already (in no steps); these are situations where there are 3 X marks in a row on either side.

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_1(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\langle\langle\{\mathbf{X}\}\rangle\rangle \bigcirc \mathbf{R_0})$
$\quad \equiv R_0(s) \vee XCanPlayToWinNext(s)$

This approximation evaluates to true if s is such that X can ensure to win in at most 1 step. These are legal situations where there are 3 X marks in a row on either side, or where it is X's turn and there are 2 X marks already and X can fill in the missing cell to get 3 in a row next as discussed in the possibility of winning case.

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_2(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\langle\langle\{\mathbf{X}\}\rangle\rangle \bigcirc \mathbf{R_1})$
$\equiv \quad R_1(s) \vee$
$\quad Legal(s) \wedge turn(s) = O \wedge$
$\quad \exists m.(curn(s) < m-2 \wedge cell(m-2,s) = X \wedge$
$\quad\quad cell(m-1,s) = X \wedge curp(s) = m) \wedge$
$\quad \exists n.(curn(s) = n-1 \wedge cell(n,s) = X \wedge$
$\quad\quad cell(n+1,s) = X \wedge n+1 < curp(s))$

This approximation evaluates to true if s is such that X can ensure to win in at most 2 steps. These are legal situations where X can ensure to win in at most 1 step as above, or where it is O's turn and we have both $X_kX \uparrow_p$ with $\uparrow_n < k$ and $\uparrow_n XX_k$ with $\uparrow_p > k$; then if O plays $markn$ then X can play $markp$ to win afterwards, and if O plays $markp$ then X can play $markn$ to win afterwards.

$\mathcal{D}_{\mathbf{ca}}^{\mathbf{T^3 1D}} \cup \mathcal{D}_{\mathbf{Z}} \models \mathbf{R_3(s)} \doteq \mathbf{Wins(X,s)} \vee \mathcal{R}(\langle\langle\{\mathbf{X}\}\rangle\rangle \bigcirc \mathbf{R_2})$
$\equiv \quad R_1(s) \vee$
$\quad Legal(s) \wedge turn(s) = O \wedge$
$\quad \exists m.(curn(s) < m-2 \wedge cell(m-2,s) = X \wedge$
$\quad\quad cell(m-1,s) = X \wedge curp(s) = m) \wedge$
$\quad \exists n.(curn(s) = n-1 \wedge cell(n,s) = X \wedge$
$\quad\quad cell(n+1,s) = X \wedge n+1 < curp(s))$

This approximation evaluates to true if s is such that X can ensure to win in at most 3 steps. It simplifies to exactly the same formula as $R_2(s)$. Thus the fixpoint iteration procedure converges in the 3^{rd} step as we have: $\mathcal{D}_{GS}^{T^3 1D} \cup \mathcal{D}_Z \models R_2(s) \equiv R_3(s)$. Therefore by Theorem 1 of [De Giacomo *et al.*, 2010]: $\mathcal{D}_{GS}^{T^3 1D} \models \langle\langle\{X\}\rangle\rangle \Diamond Wins(X)[S_0] \equiv R_2(S_0)$. It follows by the initial state axioms that $\mathcal{D}_{GS}^{T^3 1D} \models \neg\langle\langle\{X\}\rangle\rangle \Diamond Wins(X)[S_0]$ i.e., there is no winning strategy for X in S_0. But $\mathcal{D}_{GS}^{T^3 1D} \models \langle\langle\{X\}\rangle\rangle \Diamond Wins(X)[S_1]$ where $S_1 = do([markp(X), markn(O), markp(X), markn(O)], S_0)$ i.e., there is a winning strategy for X in a situation where X has marked twice on the right and O has marked twice on the left.

4 Discussion

In this paper, we described the results of some case studies to evaluate whether the [De Giacomo *et al.*, 2010] verification method actually works. We developed various infinite state game-type domains and applied the method to them. Our example domains are rather simple, but have features present in practical examples (e.g., the TTT1D domain is 1D version of tic-tac-toe on an infinite board). Our experiments do confirm

that the method does work on several non-trivial verification problems with infinite state space. We also identify some examples where the method, which only uses the simplest part of the domain theory, the unique names and domain closure for action axioms, fails to converge in a finite number of steps. We show that in some of these cases, extending the method to use some selected facts about the initial situation and some state constraints does allow us to get convergence in a finite number of steps. Finally, our example domains and properties should be useful for evaluating other approaches to infinite state verification and synthesis. See [Kmiec, 2013] for more details about our verification experiments and proofs. It also develops an evaluation-based Prolog implementation of a version of the method for complete initial state theories with the closed world assumption. It generates successive approximates and checks if they hold in the situation of interest, but does not check if the sequence of approximates converges.

Among related work that deals with verification in infinite-states domains, let us mention [Claßen and Lakemeyer, 2008; 2010], which also uses methods based on fixpoint approximation. There, characteristic graphs are introduced to finitely represent the possible configurations that a Golog program representing a multi-agent interaction may visit. However their specification language is not a game structure logic. Also closely related is [Sardina and De Giacomo, 2009], which uses a fixpoint approximation method to compose a target process expressed as a ConGolog program out of a library of available ConGolog programs. Earlier, [Kelly and Pearce, 2007] proposed a fixpoint approximation method to verify a class of temporal properties in the situation calculus called property persistence formulas. [Shapiro *et al.*, 2002] show how a theorem proving tool can be used to verify properties of multi-agent systems specified in ConGolog and an extended situation calculus with mental states. A leading example of a symbolic model checker for multi-agent systems is MCMAS [Lomuscio *et al.*, 2009]. [Belardinelli *et al.*, 2012] show that model checking of an expressive temporal language on infinite state systems is decidable if the active domain in all states remains bounded. As well, [De Giacomo *et al.*, 2012] show that verification of temporal properties in bounded situation calculus theories where there is a bound on the number of fluent atoms that are true in any situation is decidable.

In future work, we would like to automate the symbolic fixpoint approximation method that we performed manually, perhaps by writing proof tactics in a theorem proving environment. This would require some symbolic manipulation procedures for regression, FOL simplification of the resulting formulas, and checking if two successive approximations are equivalent. It would also be desirable to develop techniques for identifying initial state properties and state constraints that can be used to show finite convergence in cases where these are needed. More generally, we need a better characterization of when this kind of method can be used successfully. Note that the [De Giacomo *et al.*, 2010] framework assumes that every agent has access to all the information specified in the theory. The framework should be generalized to deal with private knowledge and partial observability. Finally, the approach should be evaluated on real practical problems.

References

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

[Belardinelli *et al.*, 2012] Francesco Belardinelli, Alessio Lomuscio, and Fabio Patrizi. An abstraction technique for the verification of artifact-centric systems. In *KR*, 2012.

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

[Claßen and Lakemeyer, 2008] Jens Claßen and Gerhard Lakemeyer. A logic for non-terminating Golog programs. In *Proc. of KR'08*, pages 589–599, 2008.

[Claßen and Lakemeyer, 2010] Jens Claßen and Gerhard Lakemeyer. On the verification of very expressive temporal properties of non-terminating Golog programs. In *Proc. of ECAI'10*, pages 887–892, 2010.

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

[De Giacomo *et al.*, 2010] G. De Giacomo, Y. Lesperance, and A. R. Pearce. Situation calculus-based programs for representing and reasoning about game structures. In *Proc. KR 2010*, pages 445–455, 2010.

[De Giacomo *et al.*, 2012] Giuseppe De Giacomo, Yves Lespérance, and Fabio Patrizi. Bounded Situation Calculus Action Theories and Decidable Verification. In *KR*, 2012.

[Enderton, 1972] Herbert B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[Hamilton, 1982] A.G. Hamilton. *Numbers, Sets and Axioms: The Apparatus of Mathematics*. Cambridge University Press, 1982.

[Kelly and Pearce, 2007] Ryan F. Kelly and Adrian R. Pearce. Property persistence in the situation calculus. In *Proc. IJCAI'07*, pages 1948–1953, 2007.

[Kmiec, 2013] Slawomir Kmiec. Infinite states verification in game-theoretic logics. Master's thesis, Dept. of Computer Science and Engineering, York University, 2013. To appear.

[Lomuscio *et al.*, 2009] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proc. CAV'09*, pages 682–688, 2009.

[McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, volume 4, pages 463–502. 1969.

[Park, 1976] David Park. Finiteness is mu-ineffable. *Theor. Comput. Sci.*, 3(2):173–181, 1976.

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

[Sardina and De Giacomo, 2009] Sebastian Sardina and Giuseppe De Giacomo. Composition of ConGolog programs. In *Proc. IJCAI'09*, pages 904–910, 2009.

[Shapiro *et al.*, 2002] Steven Shapiro, Yves Lespérance, and Hector J. Levesque. The cognitive agents specification language and verification environment for multiagent systems. In *Proc. AAMAS*, pages 19–26. Bologna, Italy, 2002.