# Reasoning About General Games Described in GDL-II

**Stephan Schiffel**
Reykjavík University, Iceland
stephans@ru.is

**Michael Thielscher**
The University of New South Wales, Sydney, Australia
mit@cse.unsw.edu.au

**Abstract**

Recently the general Game Description Language (GDL) has been extended so as to cover arbitrary games with incomplete/imperfect information. Learning—without human intervention—to play such games poses a reasoning challenge for general game-playing systems that is much more intricate than in case of complete information games. Action formalisms like the Situation Calculus have been developed for precisely this purpose. In this paper we present a full embedding of the Game Description Language into the Situation Calculus (with Scherl and Levesque's *knowledge fluent*). We 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.

## Introduction

The goal of general game playing is to create intelligent systems that understand the rules of arbitrary new games and, without human intervention, learn to play these games well (Genesereth, Love, and Pell 2005). The Game Description Language (GDL) has been defined as a formal, machine-processable language for describing the rules of arbitrary games (Love et al. 2006). Originally restricted to complete information games, it has recently been extended so as to cover any $n$-player game with incomplete/imperfect information (Thielscher 2010; 2011a).

Although based on logic, game descriptions in both GDL and its successor GDL-II (for: GDL with *incomplete information*) are minimalistic in the sense that they only provide the bare rules. How to use these rules, e.g. to infer legal moves and to update a game position, is left to the players. In basic GDL, where players are always informed about each other's moves, this is rather straightforward: a simple, Prolog-like inference engine suffices to compute one's own and every one else's legal moves and to maintain a complete state description throughout the match (Genesereth, Love, and Pell 2005). Learning—without human intervention—to play arbitrary GDL-II games, however, poses a reasoning challenge for general game-playing systems that is much more intricate than in case of complete information games. Imperfect information and information asymmetry requires a player to draw conclusions about her own percepts and what they entail about the current position, about her and everybody else's possible moves, as well as about what the other players may know.

Action formalisms like the classical Situation Calculus (McCarthy 1963) have been developed for precisely this purpose. Axiomatisation schemes and inference methods are readily available, but their deployment in general game playing presupposes a proper translation from GDL-II into an existing, suitably expressive action language. In this paper we present such a mapping by which the Game Description Language is fully embedded into the Situation Calculus. More precisely, our target language is based on the Situation Calculus with Scherl and Levesque's (2003) *knowledge fluent*. We suitably extend this variant by multi-agent knowledge and simultaneous moves, and we enrich it by the new concept of *derived action predicates*. We formally prove that our embedding provides for a sound and complete reasoning mechanism that enables players to draw conclusions about their own and the other players' knowledge in past, present, and future game states.

The rest of the paper is organised as follows. In the next section we briefly recapitulate the basic syntax and semantics of GDL-II. Thereafter we present a formal embedding of this language into the Situation Calculus, and in the section that follows we prove the correctness of this translation. We conclude with a brief discussion of related work.

## The Game Description Language GDL-II

The general game description language GDL-II is based on the standard syntax and semantics of logic programming. We follow the Prolog convention of denoting variables by uppercase letters while predicate and function symbols start with a lowercase letter. GDL-II is characterised by the following special *keywords*.

| | |
|---|---|
| `role(R)` | `R` is a player |
| `random` | the random player |
| `init(F)` | `F` holds in the initial position |
| `true(F)` | `F` holds in the current position |
| `legal(R,M)` | `R` can do move `M` in the current position |
| `does(R,M)` | player `R` does move `M` |
| `next(F)` | `F` holds in the next position |
| `sees(R,P)` | `R` perceives `P` in the next position |
| `terminal` | the current position is terminal |
| `goal(R,N)` | `R` gets `N` points in the current position |

The auxiliary pre-defined predicate `distinct(X,Y)` means syntactic inequality of the two arguments. In the following, unbound variables in clauses and other formulas are implicitly assumed to be universally quantified.

As an example consider Fig. 1, which shows a GDL-II description of standard Tic-Tac-Toe but with the interesting twist that the players cannot see the opponent's moves (as in the chess variant *Kriegspiel* (Pritchard 1994), hence the name). The names of the players and the initial position are given in lines 1–6. The moves are specified by the rules with head `legal` (lines 8–12): The player whose turn it is can attempt to mark any cell that he hasn't tried before. The other player can only do *noop*, a move without effect.[1]

The position update is specified by the rules with head `next` (lines 16–27): If the submitted move, $\mathit{mark}(m, n)$, is invalid (cf. line 14)[2] then every feature of the current position continues to hold, and the only change in the overall state is that $\mathit{tried}(m, n)$ becomes true. If, on the other hand, the move is valid, then cell $(m, n)$ receives the player's mark while all other cells retain their contents. Moreover, control goes to the other player. The reader should also note that all instances of $\mathit{tried}(m, n)$ cease to hold as there is no clause with head $\texttt{next}(\mathit{tried}(m, n))$ if *validmove* is true. The clauses with head `sees` (29–34), finally, say that the player whose turn it is will be informed about this.

**Formal Syntax and Semantics** In order to admit an unambiguous interpretation, GDL-II game descriptions must obey certain general syntactic restrictions. Specifically, a valid game description must be *stratified* (Apt, Blair, and Walker 1987) and *allowed* (Lloyd and Topor 1986). Stratified logic programs are known to admit a specific *standard model*; see (Apt, Blair, and Walker 1987) for details. A further syntactic restriction ensures that only finitely many positive instances are true in this model; for details we must refer to (Love et al. 2006) for space reasons. Finally, the special keywords are to be used as follows (Thielscher 2010):

- `role` only appears in the head of facts;
- `init` only appears as head of clauses and does not depend on any of `true`, `legal`, `does`, `next`, `sees`, `terminal`, `goal`;
- `true` only appears in the body of clauses;
- `does` only appears in the body of clauses and does not depend on any of `legal`, `terminal`, `goal`;
- `next` and `sees` only appear as head of clauses.

Under these restrictions, any valid GDL-II game description determines a state transition system as follows. To begin with, any set of rules determines an implicit domain-dependent set of ground terms $\Sigma$, like $\mathit{cell}(1, 1, b)$, $\mathit{mark}(1, 3)$, *yourmove*, etc. Game positions (i.e., states) are represented by subsets of $\Sigma$ since they are composed of individual features. Although $\Sigma$ itself is usually infinite, the syntactic restrictions in GDL-II ensure that the set of roles, the reachable states, and the set of legal moves are always finite subsets of $\Sigma$ (Love et al. 2006). Specifically, the derivable instances of `role(R)` define the players, and the initial state consists in the derivable instances of `init(F)`.

In order to determine the legal moves of a player in any given state, this state has to be encoded first, using the keyword `true`: Let $S = \{f_1, \ldots, f_n\}$ be a state (i.e., a finite set of ground terms), then $G$ is extended by the $n$ facts

$$S^{\texttt{true}} \stackrel{\text{def}}{=} \{\texttt{true}(f_1). \quad \ldots \quad \texttt{true}(f_n).\}$$

Those instances of `legal(R,M)` that are derivable from $G \cup S^{\texttt{true}}$ define all legal moves `M` for player `R` in position $S$. In the same way, the clauses for `terminal` and `goal(R,N)` define termination and goal values *relative* to the encoding of a given position.

Determining a position update and the percepts of the players requires the encoding of both the current position and a *joint move*. Specifically, if $M$ is such that players $r_1, \ldots, r_k$ take moves $m_1, \ldots, m_k$, then let

$$M^{\texttt{does}} \stackrel{\text{def}}{=} \{\texttt{does}(r_1, m_1). \quad \ldots \quad \texttt{does}(r_k, m_k).\}$$

The instances of `next(F)` that are derivable from $G \cup M^{\texttt{does}} \cup S^{\texttt{true}}$ compose the updated position; likewise, the derivable instances of `sees(R,P)` describe what a player perceives when the given joint move is done in the given position. All this is summarised as follows (Thielscher 2010).

**Definition 1.** *The* semantics *of a valid GDL-II game description $G$ is the state transition system given by*

- $R = \{r \colon G \models \texttt{role}(r)\}$ *(player names);*
- $s_1 = \{f \colon G \models \texttt{init}(f)\}$ *(initial state);*
- $t = \{S \colon G \cup S^{\texttt{true}} \models \texttt{terminal}\}$ *(terminal states);*
- $l = \{(r, m, S) \colon G \cup S^{\texttt{true}} \models \texttt{legal}(r, m)\}$ *(legal moves);*
- $u(M, S) = \{f \colon G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \models \texttt{next}(f)\}$*, for all joint moves $M$ and finite states $S$ (position update);*
- $\mathcal{I} = \{(r, M, S, p) \colon G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \models \texttt{sees}(r, p)\}$*, for all roles $r \in R \setminus \{\texttt{random}\}$ (players' percepts);*
- $g = \{(r, n, S) \colon G \cup S^{\texttt{true}} \models \texttt{goal}(r, n)\}$ *(goal values).*

We have omitted the definition of the probability distribution over updated positions when the `random` player is present, as this is irrelevant for the present paper.

## From GDL-II to the Situation Calculus

Recall the game description in Fig. 1. The clauses in line 29–34 specify the players' percepts, indicating that the player who has control next will be informed about this fact. This minimalistic definition suffices since both players should always know whether or not it is their turn. For when a player does *not* perceive *yourmove*, he should be able to conclude that it must be his opponent's turn. Another example of a (strategically very useful!) inference would be to conclude that a cell must carry your opponent's marker if you just tried to mark it yourself and you perceive *yourmove* just again, implying that your attempt must have been unsuccessful.

[1]This is the usual way of modelling turn-taking games in GDL, where in general all players move simultaneously.

[2]It is important to note the difference between *legal* and *valid* moves in Krieg-Tictactoe: each attempt to mark a cell is considered legal, but only those moves are accepted as valid that are actually possible in the current position. Feature $\mathit{tried}(m, n)$ is used to prevent a player from resubmitting a previously rejected move.

```
role(xplayer).
role(oplayer).
init(control(xplayer)).
init(cell(1,1,b)).
...
init(cell(3,3,b)).

legal(R,mark(M,N))   :- true(control(R)),
                         true(cell(M,N,Z)),
                         not true(tried(M,N)).
legal(xplayer,noop) :- true(control(oplayer)).
legal(oplayer,noop) :- true(control(xplayer)).

validmove :- does(R,mark(M,N)), true(cell(M,N,b)).

next(F)            :- not validmove, true(F).
next(tried(M,N))   :- not validmove, does(P,mark(M,N)).
next(cell(M,N,x)):- validmove, does(xplayer,mark(M,N)).
next(cell(M,N,o)):- validmove, does(oplayer,mark(M,N)).
next(cell(M,N,Z)):- validmove, true(cell(M,N,Z)),
                    does(P,mark(I,J)), distinct(M,I).
next(cell(M,N,Z)):- validmove, true(cell(M,N,Z)),
                    does(P,mark(I,J)), distinct(N,J).
next(control(oplayer)):- validmove,
                         true(control(xplayer)).
next(control(xplayer)):- validmove,
                         true(control(oplayer)).

sees(R,yourmove)        :- not validmove,
                           true(control(R)).
sees(xplayer,yourmove) :- validmove,
                           true(control(oplayer)).
sees(oplayer,yourmove) :- validmove,
                           true(control(xplayer)).
```

Figure 1: A GDL-II description of "Krieg-Tictactoe." The game positions are encoded using the three features *control*$(R)$, where $R \in \{xplayer, oplayer\}$; *cell*$(M,N,Z)$, where $M,N \in \{1,2,3\}$ and $Z \in \{x,o,b\}$, with $b$ meaning "blank;" and *tried*$(M,N)$. For the sake of simplicity, we have omitted the (straightforward) specification of termination and goal values.

Drawing conclusions of this sort is the domain of action theories, and the Situation Calculus is the oldest technique for formalising and automating reasoning about actions (McCarthy 1963). To be able to exploit this line of research in the context of General Game Playing requires to fully embed the game description language GDL-II into the Situation Calculus. The mapping that we will develop in this section is based on an extended Situation Calculus with a special fluent to represent the knowledge of agents (Scherl and Levesque 2003). We will have to slightly modify and further extend this approach for our purposes. In general, the Situation Calculus is a predicate logic with a few predefined language elements:

- constant $s_0$, denoting the *initial situation*, and constructor $\text{Do}(\alpha, \sigma)$, denoting the situation resulting from doing action $\alpha$ in situation $\sigma$;
- predicate $\text{Holds}(\varphi, \sigma)$, denoting that *fluent* $\varphi$ (i.e., an atomic state feature) is true in situation $\sigma$;
- predicate $\text{Poss}(\alpha, \sigma)$, denoting that action $\alpha$ is possible in situation $\sigma$.

## Compound actions

In games with two or more players, a position update results from all players moving simultaneously. An adequate formalisation in the Situation Calculus requires to identify the concept of an action $\alpha$ with a vector $\langle m_1, \ldots, m_k\rangle$ containing one move for each player. In a given GDL-II description, the domain of moves is implicitly determined by the arguments of keyword `legal` and the second arguments of `does`; e.g. *mark*$(M,N)$ and *noop* in Krieg-Tictactoe. Assuming an arbitrary but fixed order of the players, as in $(xplayer, oplayer)$, we define a simple axiom that identifies the individual move of a player $r$ in an action vector:

$$\text{Act}(r, \langle m_1, \ldots, m_r, \ldots, m_k\rangle) = m_r \qquad (1)$$

E.g., $\text{Act}(xplayer, \langle mark(1,3), noop\rangle) = mark(1,3)$.

## Derived action predicates

Given a GDL-II game description, we identify as *primitive fluents* those terms that occur in the scope of either of the keywords `init(F)`, `true(F)`, or `next(F)`; in Fig. 1 these are *control*$(R)$, *cell*$(M,N,Z)$, and *tried*$(M,N)$. As *derived fluents* we take all domain-specific predicates that depend on `true` but not on `does`. Following (Davis 1990), derived fluents do not require their own successor state axioms because their truth-values are fully determined by the game rules once the values of all primitive fluents are fixed in a (successor) situation. The keywords `terminal` and `goal(R,N)` are treated as derived fluents, too.

In addition to derived fluents, a mapping of GDL-II into the Situation Calculus requires the introduction of the new concept of a *derived action predicate*. These are the domain-specific predicates that depend on both `true` and `does`. Similar to derived fluents, the truth-value of a derived action predicate is fully determined by the game rules once we have fixed both the values of all primitive fluents in a situation *and* the action that is being taken in that situation. An example of a derived action predicate in Krieg-Tictactoe is *validmove*.

## The mapping

We now show how any GDL-II description $G$ can be mapped in a modular way into a Situation Calculus theory. First, some atoms that occur in $G$ are rewritten as follows.

1. All derived fluents $f(\vec{X})$ are replaced by $f(\vec{X}, S)$ and all derived action predicates $p(\vec{X})$ by $p(\vec{X}, A, S)$, indicating the dependence on a situation $S$ and an action $A$, respectively.
2. Each `init`$(\varphi)$ is replaced by $\text{Holds}(\varphi, s_0)$.
3. Each `true`$(\varphi)$ is replaced by $\text{Holds}(\varphi, S)$.
4. Each `next`$(\varphi)$ is replaced by $\text{Holds}(\varphi, Do(A,S))$.
5. Each `does`$(\varrho, \mu)$ is replaced by $\text{Act}(\varrho, A) = \mu$.

As an example, the clause in line 14 of Fig. 1 becomes

$$\begin{aligned} \mathit{validmove}(A,S) \subset \ & \textsc{Act}(R,A) = \mathit{mark}(M,N) \land \\ & \textsc{Holds}(\mathit{cell}(M,N,b),S) \end{aligned}$$

GDL-II game descriptions are based on the negation-as-failure principle, that is, every proposition that cannot be derived from the game rules is supposed to be false. To reflect this in the Situation Calculus theory, we use the *completion* according to (Clark 1977) of all clauses in the following way: For every predicate $p(\vec{X})$, replace the clauses with this predicate in the head by

$$p(\vec{X}) \equiv \bigvee_{p(\vec{t})\, :\!-\, body \,\in\, G} \vec{X} = \vec{t} \land \mathit{body} \tag{2}$$

**Initial Situation** The transformation defined above yields the following axiomatisation of the initial situation

$$\textsc{Holds}(F, s_0) \equiv \bigvee_{\mathtt{init}(\varphi)\, :\!-\, body \,\in\, G} F = \varphi \land \mathit{body} \tag{3}$$

**Preconditions** Based on the completion of `legal`, we define the precondition axiom for compound actions thus:

$$\textsc{Poss}(A,S) \equiv \forall R.\, \textsc{Legal}(R, \textsc{Act}(R,A), S) \tag{4}$$

**Effects** As a result of the transformation above, we obtain a general successor state axiom (Reiter 1991) as follows:

$$\textsc{Holds}(F, \textsc{Do}(A,S)) \equiv \bigvee_{\mathtt{next}(\varphi)\, :\!-\, body \,\in\, G} F = \varphi \land \mathit{body} \tag{5}$$

**Knowledge** Scherl and Levesque (2003) introduced the special fluent $\mathbf{K}(S', S)$—to be read as: situation $S'$ is accessible from situation $S$—to axiomatise knowledge states (of an agent) in the Situation Calculus. We use a straightforward generalisation for the multi-agent case, where $\mathbf{K}(R, S', S)$ is used to express that player $R$ considers $S'$ a possible situation in $S$. This allows to formalise subjective knowledge similar to (Scherl and Levesque 2003) thus:

$$\textsc{Knows}(\varrho, \Phi, \sigma) \overset{\text{def}}{=} \forall S'.\, \mathbf{K}(\varrho, S', \sigma) \supset \Phi[S']$$

Here, $\Phi$ is a reified formula where the situation argument in all fluents is suppressed; and $\Phi[S']$ means that all situation arguments are reinstated to $S'$. For example, $\textsc{Knows}(\mathit{xplayer}, \forall X,Y.\, \mathit{cell}(X,Y,b), s_0)$ stands for $\forall S'.\, \mathbf{K}(\mathit{xplayer}, S', s_0) \supset \forall X,Y.\, \textsc{Holds}(\mathit{cell}(X,Y,b), S')$. The macro definition can be easily extended to form nested expressions, as in $\textsc{Knows}(\mathit{xplayer}, \textsc{Knows}(\mathit{oplayer}, \mathit{control}(\mathit{oplayer})), \textsc{Do}(\langle \mathit{mark}(1,1), \mathit{noop}\rangle, s_0))$.

In GDL-II all players have complete knowledge of the initial situation. In terms of the Situation Calculus,

$$\mathbf{K}(R, S, s_0) \ \equiv\ S = s_0 \tag{6}$$

The effects of actions and percepts on the knowledge states of the players are defined by the successor state axiom for the special fluent $\mathbf{K}$, for which we adapt the definition from (Scherl and Levesque 2003) as follows:

$$\begin{aligned} & \mathbf{K}(R, S'', \textsc{Do}(A,S)) \equiv \\ & \quad \exists A', S'.\, S'' = \textsc{Do}(A', S') \land \mathbf{K}(R, S', S) \land \\ & \qquad\qquad \textsc{Poss}(A', S') \land \textsc{Act}(R,A) = \textsc{Act}(R,A') \land \\ & \qquad\qquad \forall P.\, \textsc{Sees}(R,P,A,S) \equiv \textsc{Sees}(R,P,A',S') \end{aligned} \tag{7}$$

Put in words, a player considers $S''$ a possible situation after compound action $A$ in $S$ if, and only if, $S''$ is obtained by doing some $A'$ in a situation $S'$ that was conceivable in $S$; $A'$ was executable in $S'$; the player did the same move in $A'$ as in $A$; and the player's sensing result for $A', S'$ is identical to her sensing result for the actual $A, S$ (so that she cannot distinguish the two).

## Completion semantics and stable models

The axiomatisations above applies Clark's completion to a given set of GDL-II game rules. In general, however, the first-order semantics of the completion of a (stratified) logic program is too weak to fully characterise the standard model in the presence of redundant rules like `validmove :- validmove`. The standard model remains the same when such "superfluous" clauses are added, but Clark's completion is weakened by them. This issue is solved by a second-order axiom described in (Ferraris, Lee, and Lifschitz 2011). Denoted by $\mathit{SM}[F]$, the axiom provides a stable model operator for arbitrary first-order formulas $F$. If $F$ is the completion of a stratified logic program, the Herbrand model of $\mathit{SM}[F]$ corresponds to the standard model of the logic program.

As the last step of our translation we therefore add the axiom $\mathit{SM}[F]$ with $F$ being the conjunction of all rules in the transformed game description; we refer to (Ferraris, Lee, and Lifschitz 2011) for details.

## Soundness and Completeness

The theory obtained by the above transformation is indeed a Situation Calculus theory, as we will show now.

**Theorem 1.** Let $G$ be a valid GDL-II game description and $D$ be the axiomatisation obtained from it by the transformation defined above. Then $D$ is a syntactically correct Situation Calculus theory.

*Proof.* As a Situation Calculus theory, $D$ must include a precondition axiom for each action $a(\vec{X})$ of the form

$$\textsc{Poss}(a(\vec{X}), S) \equiv \pi(\vec{X}, S)$$

The formula $\pi(\vec{X}, S)$ must not refer to any situation other than $S$. In our general precondition axiom (4), the variable $A$ can be instantiated with every compound action to obtain an axiom of the form above. The right-hand side of (4) has the only free variables $A$ and $S$ and contains no reference to any other situation besides $S$.

Furthermore, $D$ must contain successor state axioms for each primitive fluent $f(\vec{X})$ of the following form:

$$\textsc{Holds}(f(\vec{X}), \textsc{Do}(A,S)) \equiv \gamma(\vec{X}, A, S)$$

Again, the formula $\gamma(\vec{X}, A, S)$ must not refer to any situation other than $S$. In our general successor state axiom (5) the variable $F$ can be instantiated with every fluent $f(X)$ to obtain an axiom of the form above. The right-hand side of (5) refers to the bodies of rules with head `next`$(\varphi)$. According to the syntactic restrictions of GDL-II these bodies may not depend on `init` or `next` and therefore do indeed never refer to any situation besides $S$. $\square$

We will now show that the transformation from the previous section is sound and complete, that is, a GDL-II game description and the resulting Situation Calculus theory are equivalent in terms of the knowledge that can be inferred from them. For this, we first recall from (Thielscher 2010) the notion of a *development* as a legal sequence of compound actions starting in the initial state $s_1$:

$$s_1 \overset{M_1}{\to} s_2 \overset{M_2}{\to} \ldots s_{n-1} \overset{M_{n-1}}{\to} s_n \tag{8}$$

where $n \geq 1$ and for all $i \in \{1, \ldots, n-1\}$ we have that $l(r, M_i(r), s_i)$ for all $r \in R$ (players make legal moves) and $s_{i+1} = u(M_i, s_i)$ (state updates).

Intuitively, a game development (8) corresponds to the situation $\textsc{Do}(A_{n-1}, \ldots, \textsc{Do}(A_2, \textsc{Do}(A_1, s_0)) \ldots)$ in the Situation Calculus where each joint move $M_i = \{(r_1, m_1), \ldots (r_k, m_k)\}$ corresponds to a compound action $A_i = \langle m_1, \ldots, m_k \rangle$.

Following (Thielscher 2010), two developments are indistinguishable for some role $r$ if $r$ does the same moves in both developments and the sequences of perceptions of $r$ are identical. Formally, two developments

$$s_1 \overset{M_1}{\to} \ldots \overset{M_{n-1}}{\to} s_n \quad \text{and} \quad s_1 \overset{M_1'}{\to} \ldots \overset{M_{n-1}'}{\to} s_n'$$

are indistinguishable for role $r$ if, and only if,

- $\{p : (r, M_i, s_i, p) \in \mathcal{I}\} = \{p' : (r, M_i', s_i', p') \in \mathcal{I}\}$, and
- $M_i(r) = M_i'(r)$ for all $i \in \{1, \ldots, n-1\}$.

Our second theorem states that for a given state and joint move, the GDL-II game rules and the Situation Calculus theory allow to infer the same propositions.

**Theorem 2.** Let $G$ be a valid GDL-II game description and $D$ be the Situation Calculus theory obtained from $G$ with the translation defined in the previous section. Furthermore, let $\delta = s_1 \overset{M_1}{\to} \ldots \overset{M_{n-1}}{\to} s_n$ be a development of the game with corresponding situation $S = \textsc{Do}(A_{n-1}, \ldots, \textsc{Do}(A_1, s_0) \ldots)$, $M$ be a joint move legal in $s_n$, and $A$ be the compound action corresponding to $M$.

For every predicate $p(\vec{X})$ in the GDL-II description, let its translation to the Situation Calculus be denoted by $p^t(\vec{X}, A, S)$; e.g.

- $\texttt{true}^t(F, A, S) = \textsc{Holds}(F, S)$,
- $p^t(\vec{X}, A, S) = p(\vec{X}, S)$ for derived fluents,
- $p^t(\vec{X}, A, S) = p(\vec{X}, A, S)$ for derived action predicates.

Then for every predicate $p(\vec{X})$ of the GDL-II description, $G \cup s_n'^{\texttt{true}} \cup M^{\texttt{does}} \models p(\vec{X})$ iff $D \models p^t(\vec{X}, A, S)$.

*Proof.* The theorem follows from the construction of the Situation Calculus theory and the result from (Ferraris, Lee, and Lifschitz 2011) according to which the standard model of a stratified logic program is equivalent to the Herbrand model of $SM[F]$ if $F$ is the conjunction of the completion of the program. □

Our main theorem states that developments indistinguishable for some player correspond to situations which the player considers mutually possible, and vice versa. This implies that players can use the Situation Calculus theory to reason about their knowledge about past, present, and future positions as well as about the knowledge of other players.

**Theorem 3.** Let $G$ be a valid GDL-II game description with semantics $(R, s_1, t, l, u, \mathcal{I}, g)$ and $D$ be the Situation Calculus theory obtained from $G$ with the translation defined in the previous section. Let there be two developments

$$\begin{aligned} \delta &= s_1 \overset{M_1}{\to} \ldots \overset{M_{n-1}}{\to} s_n, \text{ and} \\ \delta' &= s_1 \overset{M_1'}{\to} \ldots \overset{M_{n-1}'}{\to} s_n' \end{aligned}$$

of the game with the corresponding situations $\sigma, \sigma'$. A role $r \in R$ cannot distinguish $\delta$ and $\delta'$ iff $D \models K(r, \sigma', \sigma)$.

*Proof.* By induction on the length $n$ of the development $\delta$.

For the base case $n = 0$ there is only one development–the initial state $s_1$ with the corresponding initial situation $s_0$. Accordingly, by (6), $D \models K(r, S, s_0)$ iff $S = s_0$.

For the induction step, consider the two developments

$$\delta_+ = \delta \overset{M_n}{\to} s_{n+1} \quad \text{and} \quad \delta_+' = \delta' \overset{M_n'}{\to} s_{n+1}'$$

with the corresponding situations $\textsc{Do}(A_n, \sigma)$ and $\textsc{Do}(A_n', \sigma')$, respectively.

We have to show that $\delta_+, \delta_+'$ are indistinguishable for $r$ if, and only if, $D \models \mathbf{K}(r, \textsc{Do}(A_n', \sigma'), \textsc{Do}(A_n, \sigma))$. According to (7), $\mathbf{K}(r, \textsc{Do}(A_n', \sigma'), \textsc{Do}(A_n, \sigma))$ holds if, and only if, all of the following hold.

(a). $\mathbf{K}(r, \sigma', \sigma)$,
(b). $\textsc{Poss}(A_n', \sigma')$,
(c). $\textsc{Act}(r, A_n) = \textsc{Act}(r, A_n')$, and
(d). $\forall P.\textsc{Sees}(r, P, A_n, \sigma) \equiv \forall P.\textsc{Sees}(r, P, A_n', \sigma')$.

By the induction hypotheses, (a) holds. From the definition of a development it follows that each player's move in $M_n'$ is legal in the state $s_n'$, that is, for all players $p$,

$$G \cup s_n'^{\texttt{true}} \models \texttt{legal}(p, M_n'(p))$$

By Theorem 2 we conclude that this holds exactly if $D \models \textsc{Legal}(p, M_n'(p), \sigma')$ for all players $p$. From (4) and the fact that $M_n'(p) = \textsc{Act}(p, A_n')$ we can conclude (b).

Provided that $\delta$ and $\delta'$ are indistinguishable for role $r$, $\delta_+$ and $\delta_+'$ are indistinguishable for $r$ if, and only if,

$$\{p : (r, M_n, s_n, p) \in \mathcal{I}\} = \{p' : (r, M_n', s_n', p') \in \mathcal{I}\} \tag{9}$$
$$\text{and} \quad M_n(r) = M_n'(r) \tag{10}$$

According to (1), equation (10) holds if and only if (c). From Definition 1 and Theorem 2 it follows that (9) holds just in case (d) holds. Hence, $\mathbf{K}(r, \textsc{Do}(A_n', \sigma'), \textsc{Do}(A_n, \sigma))$ holds if, and only if, $r$ cannot distinguish $\delta_+, \delta_+'$, which concludes the induction step and the proof. □

**Practical considerations** Our completeness result requires the second-order axiom $SM[F]$ in the translated game description. In practice, this can be avoided if we can confine ourselves to finitely many situations. The syntactic restrictions in GDL-II imply that every ground atom only depends on finitely many other ground atoms; this is a consequence of a restricted recursion as defined in (Love et al. 2006). Our mapping extends the GDL rules by situation arguments, but also for these only finitely many ground instances are considered in Theorem 2 because the developments are fixed and hence the situations are depth-restricted. Bonatti (2004) showed that in this case it suffices to consider a finite subset $R(P, F)$ of the grounding of a program $P$ to decide whether a ground atom $F$ is entailed by $P$. Thus, we can consider a finite ground program and replace the second-order axiom by propositional loop formulas (Lin and Zhao 2004) to reconcile the semantics of GDL-II with the Situation Calculus theory. This solution applies whenever we can confine ourselves to finitely many ground situations for the reasoning problem at hand.[3]

## Conclusion

We have presented an embedding of the extended Game Description Language (GDL-II) into a suitable variant of the Situation Calculus that features multi-agent knowledge, simultaneous actions, and action-independent sensing. While a GDL-II description provides general game-playing systems only with the bare rules of a game, the Situation Calculus axiomatisation tells players how to reason about their own percepts and what they entail about the current position, about the possible moves, and about what the other players may know. This is the first full embedding of GDL-II into an action calculus; our recent translation (Thielscher 2011b) into the Action Description Language $\mathcal{C}+$ (Giunchiglia et al. 2004) is restricted to basic GDL and hence covers neither imperfect information games nor knowledge and sensing.

The Situation Calculus has previously been shown to be a viable formalism for representing and reasoning about games. Most notably, Schulte and Delgrande (2004) also use the Situation Calculus variant of (Scherl and Levesque 2003) to axiomatise extensive-form games. The main limitation of their axiomatisation is to restrict the knowledge fluent $\mathbf{K}(S', S)$ to a single agent (namely, the player whose move it is in situation $S$, assuming that players do not move simultaneously). This does not allow for reasoning about the knowledge of any other player in these situations, nor about what one player can conclude about what another player would know. This is remedied in the recent definition of a multi-agent epistemic variant of the Situation Calculus (Belle and Lakemeyer 2010) for axiomatising games. Despite notable differences to GDL-II and our variant of the Situation Calculus, e.g. the restriction to non-simultaneous moves and the necessity to define two domain theories (for objective and subjective knowledge, respectively), we believe that a translation similar to the one presented in this paper can be constructed by which full GDL-II is embedded into a suitably adapted version of (Belle and Lakemeyer 2010).

[3]We also remark that this issue is actually of little relevance to the practice of general game playing; e.g. none of the numerous games played at the past AAAI Competitions featured logically redundant clauses that players had to recognise.

**Acknowledgements.** We thank Joohyung Lee and the anonymous reviewers for helpful comments on the paper. The second author is the recipient of an Australian Research Council Future Fellowship (project FT 0991348).

## References

Apt, K.; Blair, H. A.; and Walker, A. 1987. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*, 89–148.

Belle, V., and Lakemeyer, G. 2010. Reasoning about imperfect information games in the epistemic situation calculus. In *Proceedings of AAAI*, 255–260.

Bonatti, P. A. 2004. Reasoning with infinite stable models. *Artif. Intell.* 156(1):75–111.

Clark, K. L. 1977. Negation as failure. In *Logic and Data Bases*, 293–322.

Davis, E. 1990. *Representations of Commonsense Knowledge*. Morgan Kaufmann.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2011. Nonmonotonic causal theories. *Artif. Intell.* 175(1):236–263.

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

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *Artif. Intell.* 153(1–2):49–104.

Lin, F., and Zhao, Y. 2004. Assat: computing answer sets of a logic program by sat solvers. *Artif. Intell.* 157:115–137.

Lloyd, J., and Topor, R. 1986. A basis for deductive database systems II. *Journal of Logic Programming* 3(1):55–67.

Love, N.; Hinrichs, T.; Haley, D.; Schkufza, E.; and Genesereth, M. 2006. General Game Playing: Game Description Language Specification. Tech. Rep. LG–2006–01, Stanford. Available at: `games.stanford.edu`.

McCarthy, J. 1963. *Situations and Actions and Causal Laws*. Stanford Artif. Intell. Project, Memo 2.

Pritchard, D. 1994. *The Encycolpedia of Chess Variants*.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation*, 359–380.

Scherl, R., and Levesque, H. 2003. Knowledge, action, and the frame problem. *Artif. Intell.* 144(1):1–39.

Schulte, O., and Delgrande, J. 2004. Representing von Neumann-Morgenstern games in the situation calculus. *Annals of Mathematics and Artif. Intell.* 42(1–3):73–101.

Thielscher, M. 2010. A general game description language for incomplete information games. In *Proceedings of AAAI*, 994–999.

Thielscher, M. 2011a. The general game playing description language is universal. In *Proceedings of IJCAI*.

Thielscher, M. 2011b. Translating general game descriptions into an action language. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *LNAI*, 300–314. Springer.