Journal of Artificial Intelligence Research 49 (2014) 171-206 Submitted 07/13; published 02/14

Representing and Reasoning About the Rules of General Games With Imperfect Information

Stephan Schiffel STEPHANS@RU.IS
School of Computer Science, Reykjavík University,
Menntavegur 1, 101 Reykjavík ICELAND

Michael Thielscher MIT@CSE.UNSW.EDU.AU
School of Computer Science and Engineering,
The University of New South Wales,
Sydney, NSW 2052 AUSTRALIA

Abstract

A *general* game player is a system that can play previously unknown games just by being given their rules. For this purpose, the *Game Description Language (GDL)* has been developed as a high-level knowledge representation formalism to communicate game rules to players. In this paper, we address a fundamental limitation of state-of-the-art methods and systems for General Game Playing, namely, their being confined to deterministic games with complete information about the game state. We develop a simple yet expressive extension of standard GDL that allows for formalising the rules of arbitrary finite, n-player games with randomness and incomplete state knowledge. In the second part of the paper, we address the intricate reasoning challenge for general game-playing systems that comes with the new description language. We develop a full embedding of extended GDL into the Situation Calculus augmented by 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.

1. Introduction

General Game Playing (GGP) is concerned with the development of systems that understand the rules of previously unknown games and learn to play these games well without human intervention. The annual AAAI GGP Competition, which has been established in 2005 to foster research in this area, has led to a number of successful approaches and systems (Kuhlmann, Dresner, & Stone, 2006; Clune, 2007; Schiffel & Thielscher, 2007; Kaiser, 2008; Finnsson & Björnsson, 2008; Kissmann & Edelkamp, 2011; Méhat & Cazenave, 2011; Kirci, Sturtevant, & Schaeffer, 2011). General game-playing programs are a quintessential example of systems that end users can customise for their own specific tasks. This makes GGP an interesting and challenging problem for AI, involving many fundamental issues such as reasoning, learning, planning and decision making (Pell, 1993). Consequently, General Game Playing has broader significance to a variety of AI disciplines beyond conventional computer game playing (Genesereth, Love, & Pell, 2005).

1.1 Representing the Rules of General Games

The first AAAI Competition saw the introduction of the general *Game Description Language (GDL)* as the foundation for general game-playing systems (Genesereth et al., 2005). Other machine-processable languages for the specification of games existed before, most notably Gala (for: "Game Language"). But the latter was never used for any purpose other than as the front-end to a system for computing optimal strategies from game trees (Koller & Pfeffer, 1997). Presumably its tight coupling with a programming language (Prolog) and its operational—rather than declarative—semantics prevented Gala from being adopted by others as a system-independent game specification language.

GDL allows for the description of any game with finitely many players and finitely many legal moves in each state, where all moves are deterministic and fully observable, and where the complete game rules are known to each player. Simultaneous moves are possible, which provides for a restricted form of imperfect information. However, players are always immediately informed about each other's moves and hence always have perfect information about the game state after every round.

With GDL the organisers of the AAAI GGP Competition sought a high-level game specification language that admits a purely declarative reading. Thus, it enables general game-playing systems to reason about the rules of a previously unknown game, for example, in order to extract game-specific knowledge or to automatically design evaluation functions. This proved to be successful as these problems have been studied extensively in the recent past (Kuhlmann et al., 2006; Kaiser, 2007; Clune, 2007; Schiffel & Thielscher, 2007; Edelkamp & Kissmann, 2008; Schiffel & Thielscher, 2009; Ruan, van der Hoek, & Wooldridge, 2009; Schiffel, 2010; Thielscher & Voigt, 2010). GDL is a purely declarative language in the tradition of AI Planning languages and in fact can be seen as a multi-agent extension thereof since existing planning languages always describe a problem from the perspective of a single agent—even in case of adversarial planning (Jensen & Veloso, 2000). The presence of other agents that have their own actions and goals is crucial since reasoning about their intentions is the basis for Opponent Modelling, a central aspect in which GGP goes beyond AI Planning (Genesereth et al., 2005). What GDL inherits from existing planning languages is the compactness of specifications, which contrasts it with other encoding techniques, for instance the use of propositionalised graphs (La Mura, 2000).

Despite steady progress, the current state of the art in General Game Playing is limited to deterministic games with complete information about the game state, owing to the restricted expressiveness of original GDL. This covers a variety of classic games such as Chess, Go, Chinese Checkers etc. but excludes games with elements of chance like Backgammon; games with information asymmetry such as Bridge or Poker; and games that involve private communication among cooperating players like in Bughouse Chess, or negotiations like in Diplomacy. Moreover, envisaged applications for General Game Playing systems, such as automated trading agents (Thielscher & Zhang, 2010), are usually characterised by imperfect information.

In this paper, we lay the foundations for truly general game-playing systems by developing and analysing an extension of the existing description language to GDL-II (for "Game

Description Language with Imperfect/Incomplete Information").[1] GDL-II will allow for the description of any extensive-form game with finitely many players and finitely many legal moves in each state but where moves can be nondeterministic and players can have imperfect and asymmetric information. Although GDL-II will be based on the assumption that the game rules themselves are fully known to each player, incomplete-information games—in the game-theoretic sense (Rasmusen, 2007), that is, where players do not know exactly their own goal or what type the opponents are—can be modelled in our extended language via the standard Harsanyi (1967) transformation of adding an unobserved move by nature at the beginning.

1.2 Reasoning With the Rules of General Games

Game descriptions in GDL-II will include a precise specification of the information that players get throughout a game. But the rules of a game do not imply anything about how players *use* this information, that is, which conclusions they draw from it, how they combine it with what they already know, or whether they memorise it. These are separate questions that concern the *reasoning* that game-playing agents perform on the basis of the rules of a game. The clear separation of these two issues means that, for example, "perfect recall" (Rasmusen, 2007) is a property of individual players and depends on the reasoning mechanism implemented on them, rather than being a property of a game itself.

In basic GDL, where players are always informed about each other's moves, reasoning with the rules of a game is rather straightforward: a simple, Prolog-like inference engine suffices to compute one's own and everyone else's legal moves and to maintain a complete state description throughout the match (Genesereth et al., 2005). In contrast, playing arbitrary games with incomplete state knowledge poses an intricate reasoning challenge for general game-playing systems: Imperfect information and information asymmetry require a player to draw conclusions about his own percepts and what they entail about the current position, about his and everybody else's possible moves, as well as about what the other players may know.

Action formalisms like the classic Situation Calculus (McCarthy, 1963) have been developed for precisely this purpose. They are readily available, but to deploy them in General Game Playing presupposes a proper translation from GDL-II into an existing, suitably expressive action language. Therefore, in the second part of this paper we address the problem of reasoning about GDL-II by presenting a full embedding of our extended general game description language into a suitably extended variant of the Situation Calculus.

1.3 Overview of Results

Our specific contributions in this paper to the foundations for General Game Playing can be summarised as follows:

1. We show that the addition of just two more keywords to GDL suffices to obtain the desired generality of the game description language: The first, called `sees`, is used

1. In an unfortunate clash of terminology, an agent who does not know the full state of the environment is said to have *incomplete* information in AI, whereas in Game Theory, if players do not know the full state when it is their turn, then the game is said to be of *imperfect* information.

to control the information that each player gets. The second, called `random`, denotes a special player who chooses moves randomly.

2. We develop a new execution model for GDL-II and demonstrate how—despite the conceptual simplicity of the representation language—the operational semantics gives rise to an intricate epistemic model, which provides players with sufficient information to enable them to reason about their own knowledge and the knowledge of their opponents, to predict how their knowledge will evolve, and to reason about what players know about other players' knowledge.

3. We develop a new communication protocol for GDL-II to address practical issues that arise in General Game Playing with imperfect information.

4. We investigate the expressive power of our high-level knowledge representation language by relating it to the mathematical concept of extensive-form games (Rasmusen, 2007).

5. We extend the Situation Calculus that includes Scherl and Levesque's *knowledge fluent* (Scherl & Levesque, 2003) with multi-agent knowledge, simultaneous moves, and the new concept of *derived action predicates*.

6. We present a mapping by which GDL-II is fully embedded into this extended Situation Calculus, and we formally prove that this provides for a correct axiomatic account of the semantics of GDL-II, which enables players to draw conclusions about their own and the other players' knowledge in past, present, and future game states.

The paper proceeds as follows. In the next section, we introduce GDL-II as an extension of GDL with randomness and imperfect information, and we use various examples to demonstrate the range of phenomena and games that can be described in this new language. We also develop a modified execution model for GDL-II. In Section 3, we show that the language is sufficiently expressive to give rise to an intricate multi-agent epistemic model, and we relate our language to that of extensive-form games. In Section 4, we present a formal embedding of this language into the Situation Calculus and prove the correctness of this translation. We discuss related work in Section 5 and conclude in Section 6.

2. Describing General Imperfect-Information Games With Randomness

General Game Playing requires a formal language for describing the rules of arbitrary games. Any complete game description needs to provide

- the names of the players,
- the initial position,
- the legal moves and how they affect the position, and
- the terminating and winning criteria.

`role(R)`	`R` is a 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
`terminal`	the current position is terminal
`goal(R,V)`	`R` scores `V` points in the current position
`sees(R,P)`	`R` perceives `P` in the next position
`random`	the random player

Table 1: GDL-II keywords. Standard GDL comprises the top eight. The keywords are accompanied by the auxiliary, pre-defined predicate `distinct(X,Y)`, meaning the syntactic inequality of the two arguments (Love et al., 2006).

The description language GDL has been developed for this purpose (Genesereth et al., 2005; Love, Hinrichs, Haley, Schkufza, & Genesereth, 2006). The emphasis is on high-level, declarative game rules that are easy to understand and maintain. At the same time, GDL has a precise semantics and is fully machine-processable. Moreover, background knowledge is not required—a set of rules is all a player needs to know in order to be able to play a previously unknown game.

GDL is based on the standard syntax and semantics of Logic Programming. A few special *keywords* are used for the different elements of a game description mentioned above. The original game description language GDL uses the first eight keywords shown in Table 1. This version of GDL is suitable for describing any finite, synchronous, and deterministic n-player game (Genesereth et al., 2005).[2] The execution model entails complete state information: The initial position is fully specified and the players are immediately informed about each other's moves, with all (joint) moves being deterministic.

Although GDL was developed for complete-information games only, a surprisingly simple extension to its syntax suffices to generalise it to arbitrary (discrete and finite) games with information asymmetry and random moves.

1. The new keyword `random` is introduced as a special role.

   It is assumed that this "player" always makes a purely random choice among its legal moves in a position. This allows the game designer to describe elements of chance, such as rolling dice or shuffling cards.

2. The second new keyword `sees(R,P)` is introduced for specifying the conditions under which player `R` receives information (i.e., "perceives") `P`.

   To ensure greatest flexibility, arbitrary terms may serve as percepts `P`. Not only does this allow a game designer to have players observe specific state features or actions, but games may also feature rules according to which players are informed about logical

2. Synchronous means that all players move simultaneously. In this setting, turn-taking games are modelled by allowing players only one legal move, without effect, if it is not their turn.

combinations of conditions or receive partial information about a state feature or a move by another player.

The language extension will be accompanied by a modified execution model, in which players are no longer informed about each other's moves by default; rather, they only get to see what the game rules entail about their percepts.

We again refer to Table 1 with a list of all the keywords of the new language GDL-II.

2.1 Examples

Prior to giving the precise definition of syntax and semantics we illustrate the expressiveness of this extended Game Description Language with several examples; the first and third will recur later in the paper.

2.1.1 Example 1 *(Krieg-Tictactoe)*

The rules in Figure 1 describe standard Tic-Tac-Toe but with the twist that the players cannot see their opponent's moves, just like in the chess variant *Kriegspiel* (Pritchard, 1994), hence the name.[3]

The two roles and the initial position are given in lines 1–2 and lines 4–7, respectively. The moves are specified by the rules with head `legal` (lines 9–14): The player whose turn it is can attempt to mark any cell that he does not already own or has tried earlier in the game. The other player can only do *noop*, a move without effect.

The position update is specified by the rules with head `next` (lines 18–29): If the submitted move $mark(M,N)$ by player R is *not* valid (where "valid" means that the targeted cell is still blank; cf. line 16) then every feature of the current position continues to hold, and the only change in the overall state is that $tried(R,M,N)$ becomes true.[4] Otherwise the cell with coordinates (M,N) will be marked while none of the other cells change. Moreover, control goes to the opponent.

The clauses with head `sees` (lines 31–33) say that the player whose turn it is will always be informed about this. This suffices because when a player does *not* perceive *yourmove*, then it follows that it must be his opponent's turn. □

2.1.2 Example 2 *(Coloured Trails)*

This class of games is a popular research test-bed for decision-making and negotiation in a competitive setting (Grosz, Kraus, Talman, Stossel, & Havlin, 2004). Each specific game comes with one or more fixed protocols defining possible interactions among the players.

3. A word on the notation: In this paper we will be largely concerned with the semantics behind GDL game descriptions, which are interpreted as logic programs. For this reason we will use the standard Prolog syntax for GDL, where variables are indicated by uppercase letters. This is in contrast to the more customary KIF-notation of GDL introduced by Genesereth et al. (2005). The KIF-syntax also allows for disjunctions in clause bodies, but these can be easily transformed into normal logic program clauses (Lloyd, 1987). Throughout the paper, we will use "clause" and "(game) rule" interchangeably.

4. 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 $tried(R,M,N)$ 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(1,2,b)).     init(cell(1,3,b)).
init(cell(2,1,b)).     init(cell(2,2,b)).     init(cell(2,3,b)).
init(cell(3,1,b)).     init(cell(3,2,b)).     init(cell(3,3,b)).

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

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 $\mathit{control}(R)$, $\mathit{cell}(M,N,Z)$, and $\mathit{tried}(R,M,N)$, where $R \in \{\mathit{xplayer}, \mathit{oplayer}\}$; $M,N \in \{1,2,3\}$; and $Z \in \{x,o,b\}$, with b meaning "blank." For the sake of simplicity, we have omitted the (straightforward) specification of both the terminating conditions and the goal values.

For example, a simple negotiation may consist of player R offering player S to exchange two of their chips, C and D. This can be formalised by these GDL-II clauses:

```
legal(R,propose(S,tradeChips(C,D))) :-
   true(hasChip(R,C)), true(hasChip(S,D)), distinct(R,S).

sees(S,offer(R,C,D)) :- does(R,propose(S,tradeChips(C,D))).
```

Under these rules the communication is private: only the addressee gets to see the offer. □

2.1.3 Example 3 *(Monty Hall)*

As our second running example, the rules in Figure 2 describe a simple but famous game where a car prize is hidden behind one of three doors and where a candidate is given two chances to pick a door. The host is modelled by the new pre-defined role `random`.[5]

Lines 1–7 introduce the players' names and the state features that hold initially. The possible moves are specified in lines 9–17: First, the `random` player decides where to place the car and, simultaneously, the candidate chooses a door. Next, the host (i.e., `random`) opens a door that is not the one with the car behind it nor the one that the candidate has chosen. Finally, the candidate can either stick to her earlier choice (*noop*) or switch to the other door that is still closed.

The candidate's only percept throughout the game, viz. the door that gets opened, is defined by the rule with head `sees` (line 19). The remaining clauses specify the position update (lines 21–31), the fact that the game ends after the candidate's final decision (line 33), and the payoff for the candidate depending on whether she got the door right in the end (lines 35–36). □

2.1.4 Example 4 *(Poker)*

Together the two new keywords can be used to describe all kinds of card games, which are typically characterised by both randomness (shuffling) and information asymmetry (individual hands). A single card dealt face down to a player, say, can be axiomatised thus:

```
legal(random,deal(P,C)) :- role(P), in_deck(C),
                           distinct(P,random).

next(in_hand(P,C))    :- does(random,deal(P,C)).
sees(P,your_card(C)) :- does(random,deal(P,C)).
```

Here, only the player who is dealt the card can see it. Multiple cards can be handed out in a single move which takes each card as a separate argument and of which players only get to see the argument positions that correspond to their cards. In contrast, a card dealt face-up, as in Texas Hold'em, would be axiomatised as follows.

```
legal(random,deal_river(C)) :- in_deck(C).

next(river(C))    :- does(random,deal_river(C)).
sees(P,river(C)) :- does(random,deal_river(C)), role(P).
```

Here, all players are informed about the river card. □

The example game descriptions illustrate the two new features in GDL-II: The special role `random` is used to model nature, who always moves randomly. The keyword `sees` controls the information that players have about the game state. Although all players have full knowledge of the game rules including the initial state, both imperfect and asymmetric knowledge about later states are a natural consequence of players' individual and limited percepts.

5. Although `random` is a pre-defined constant, it needs to be declared as a role (line 2) with his own goal (line 37). In this way GDL-II supports upward compatibility with the original GDL.

```
role(candidate).
role(random).

init(closed(1)).
init(closed(2)).
init(closed(3)).
init(step(1)).

legal(random,hide_car(D))  :- true(step(1)), true(closed(D)).
legal(random,open_door(D)) :- true(step(2)), true(closed(D)),
                              not true(car(D)), not true(chosen(D)).
legal(random,noop)         :- true(step(3)).

legal(candidate,choose(D)) :- true(step(1)), true(closed(D)).
legal(candidate,noop)      :- true(step(2)).
legal(candidate,noop)      :- true(step(3)).
legal(candidate,switch)    :- true(step(3)).

sees(candidate,D) :- does(random,open_door(D)).

next(car(D))    :- does(random,hide_car(D)).
next(car(D))    :- true(car(D)).
next(closed(D)) :- true(closed(D)), not does(random,open_door(D)).
next(chosen(D)) :- does(candidate,choose(D)).
next(chosen(D)) :- true(chosen(D)), not does(candidate,switch).
next(chosen(D)) :- does(candidate,switch),
                   true(closed(D)), not true(chosen(D)).

next(step(2)) :- true(step(1)).
next(step(3)) :- true(step(2)).
next(step(4)) :- true(step(3)).

terminal :- true(step(4)).

goal(candidate,100) :- true(chosen(D)), true(car(D)).
goal(candidate,  0) :- true(chosen(D)), not true(car(D)).
goal(random,     0).
```

Figure 2: A GDL-II description of the Monty Hall game (Rosenhouse, 2009).

2.2 Formal Syntax

GDL-II is based on the standard syntax of logic programs, including negation.

Definition 1
Consider a signature with function symbols (including constants) and variables.

- *A* term *is either a variable, or a function symbol with terms as arguments.*
- *An* atom *is a predicate symbol with terms as arguments.*
- *A* literal *is an atom or its negation.*

- *A* clause *is of the form* h **:-** $b_1, \ldots, b_n$**.**, *where* h *(the* head*) is an atom and* $b_1, \ldots, b_n$ *(the* body*) are literals* ($n \geq 0$)*, with the meaning that* $b_1, \ldots, b_n$ *together imply* h*.*

GDL-II imposes the following restrictions on the use of the pre-defined keywords in Table 1.

Definition 2
The dependency graph *for a set* G *of clauses is a directed, labeled graph whose nodes are the predicate symbols that occur in* G *and where there is a* positive *edge* $p \overset{+}{\rightarrow} q$ *if* G *contains a clause* $p(\vec{s})$ **:-** $\ldots, q(\vec{t}), \ldots$**.**, *and a* negative *edge* $p \overset{-}{\rightarrow} q$ *if* G *contains a clause* $p(\vec{s})$ **:-** $\ldots, \neg q(\vec{t}), \ldots$ **.** *We say that* p depends on q *in* G *if there is a path from* p *to* q *in the dependency graph of* G*.*

A GDL-II game description *is a finite set of clauses where*

- `role` *only appears in facts (i.e., clauses with empty body) or in the body of clauses;*
- `init` *only appears in the head of clauses and does not depend on any of* `true`*,* `legal`*,* `does`*,* `next`*,* `terminal`*, or* `goal`*;*
- `true` *only appears in the body of clauses;*
- `does` *only appears in the body of clauses, and none of* `legal`*,* `terminal`*, or* `goal` *depends on* `does`*;*
- `next` *and* `sees` *only appear in the head of clauses;*
- `distinct` *only appears in the body of clauses.*[6]

2.3 Valid Game Descriptions

In order to admit an unambiguous interpretation as a game, GDL-II descriptions must obey further general syntactic restrictions, all of which are inherited from its predecessor GDL.

Definition 3
To constitute a valid *GDL-II description, a set of clauses* G *must satisfy the following.*

1. G *is* stratified*, that is, there are no cycles involving a negative edge in the dependency graph for* G *(Apt, Blair, & Walker, 1987; Gelder, 1989).*
2. G *is* allowed*, that is, each variable in a clause occurs in at least one positive atom in the body (Lloyd & Topor, 1986).*
3. *If* p *and* q *occur in a cycle in the dependency graph and* G *contains a clause*

$$p(\vec{s}) \;\textbf{:-}\; q_1(\vec{t_1}), \ldots, q(v_1, \ldots, v_k), \ldots, q_n(\vec{t_n})\textbf{.}$$

then for every $v_i \in \{v_1, \ldots, v_k\}$,

- v_i *is ground, or*

6. The formal meaning of this predicate is given by tacitly assuming the unary clause `distinct`(s, t)., for every pair s, t of syntactically different, ground (i.e., variable-free) terms.

- *v_i is an element of $\vec{s}$, or*
- *v_i is an element of some $\vec{t}_j$ $(1 \leq j \leq n)$ such that q_j does not appear in a cycle with p.*

The last condition imposes a restriction on the combination of function symbols and recursion to ensure finiteness and decidability of all relevant derivations (Love et al., 2006).

The reader is invited to verify that our example game descriptions in Figure 1 and 2 satisfy all requirements of a valid GDL-II description. The syntactic restrictions ensure that a set of game rules can be effectively and unambiguously interpreted by a state transition system as a formal game model, which we will describe next.

2.4 Semantics

A unique game model is obtained from a valid GDL-II game description using the concept of a *stable model* of a logic program with negation (Gelfond & Lifschitz, 1988).

Definition 4
For a set of clauses G and a set of ground atoms $\mathcal{M}$, let $G^{\mathcal{M}}$ be the set of negation-free implications $h \subset b_1 \wedge \ldots \wedge b_k$ obtained by taking all ground instances of clauses in G and

- *deleting all clauses with a negative body literal $\neg b_i$ such that $b_i \in \mathcal{M}$,*
- *deleting all negative body literals from the remaining clauses.*

Then $\mathcal{M}$ is a stable model *for G if and only if $\mathcal{M}$ is the least model for $G^{\mathcal{M}}$.*

A useful property of stable models is to provide a unique model whenever the underlying set of clauses is stratified (Gelfond & Lifschitz, 1988), which is a requirement of valid GDL-II specifications according to Definition 3. Moreover, the syntactic restrictions in GDL-II ensure that this model is *finite* for all logic programs we consider—a property inherited from original GDL (Love et al., 2006). Hence, for the following game model for GDL-II we can assume a finite set of players, finite states, and finitely many legal moves in each state. We shall denote by $SM[G]$ the unique stable model for stratified set of clauses G.

Specifically, then, the derivable instances of keyword `role(R)` from a given game description define the **players**. The **initial state** is composed of the derivable instances of `init(F)`. In order to determine the legal moves in any given state, this state has to be encoded first, using the keyword `true`. Let, to this end, $\sigma = \{f_1, \ldots, f_n\}$ be a state (i.e., a finite set of ground terms, a.k.a. *fluents*), then the game description G is augmented by the n facts

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

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

Example 2 *(continued)*
The rules in Figure 1 entail that Krieg-Tictactoe features the two roles *xplayer* and *oplayer*.

The initial position is $\sigma_0 = \{\mathit{control}(\mathit{xplayer}), \mathit{cell}(1,1,b), \ldots, \mathit{cell}(3,3,b)\}$. With ${\sigma_0}^{\texttt{true}}$ added to the rules we can infer $\texttt{legal}(\mathit{xplayer}, \mathit{mark}(M,N))$ for each combination of coordinates $M, N \in \{1,2,3\}$, while *oplayer* only has the one move *noop*. □

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

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

Taken together, all instances of `next(F)` that are derivable from $G \cup \mu^{\texttt{does}} \cup \sigma^{\texttt{true}}$ compose the **updated position**. Similarly, the derivable instances of keyword `sees(R,P)` describe what a player **perceives** when joint move μ is played in position σ.

Example 4 *(continued)*
According to the rules in Figure 2, the initial position in the Monty Hall game is given by $\sigma_0 = \{\mathit{closed}(1), \mathit{closed}(2), \mathit{closed}(3), \mathit{step}(1)\}$. There are three legal moves for each player in this state, viz. $\mathit{hide\_car}(D)$ for `random` and $\mathit{choose}(D)$ for *candidate*, where $D \in \{1,2,3\}$. Consider, say, $\mu_1 = \{\texttt{random} \mapsto \mathit{hide\_car}(1), \mathit{candidate} \mapsto \mathit{choose}(3)\}$, then with ${\sigma_0}^{\texttt{true}}$ and $\mu_1^{\texttt{does}}$ added to the game rules, the clauses for keyword `next` determine the updated state $\sigma_1 = \{\mathit{car}(1), \mathit{chosen}(3), \mathit{closed}(1), \mathit{closed}(2), \mathit{closed}(3), \mathit{step}(2)\}$. According to the only clause for keyword `sees` the candidate perceives nothing at this stage and hence cannot know that the car is hidden behind door 1. □

All of the above is summarised in the following definition.

Definition 5
Let G be a valid GDL-II description. The game semantics *of G is the state transition system $(R, \sigma_0, T, l, u, \mathcal{I}, g)$ given by*

- *roles* $R = \{r : \texttt{role}(r) \in \mathrm{SM}[G]\}$;
- *initial position* $\sigma_0 = \{f : \texttt{init}(f) \in \mathrm{SM}[G]\}$;
- *terminal positions* $T = \{\sigma : \texttt{terminal} \in \mathrm{SM}[G \cup \sigma^{\texttt{true}}]\}$;
- *legal moves* $l = \{(r, m, \sigma) : \texttt{legal}(r, m) \in \mathrm{SM}[G \cup \sigma^{\texttt{true}}]\}$;
- *state update function* $u(\mu, \sigma) = \{f : \texttt{next}(f) \in \mathrm{SM}[G \cup \sigma^{\texttt{true}} \cup \mu^{\texttt{does}}]\}$*, for all joint moves μ and states σ;*
- *information relation* $\mathcal{I} = \{(r, \mu, \sigma, p) : \texttt{sees}(r, p) \in \mathrm{SM}[G \cup \sigma^{\texttt{true}} \cup \mu^{\texttt{does}}]\}$;
- *goal relation* $g = \{(r, v, \sigma) : \texttt{goal}(r, v) \in \mathrm{SM}[G \cup \sigma^{\texttt{true}}]\}$.

2.5 A New Execution Model

The additional elements in GDL-II and the modified semantics require a new execution model for games with incomplete state information and randomness: Starting in the initial position, in each state σ each player chooses a legal move. The special `random` role is assumed to choose randomly with uniform probability among its legal moves.[7] Given a joint move μ the game state changes to $u(\mu, \sigma)$. In contrast to the execution model for GDL (Genesereth et al., 2005; Love et al., 2006; Schiffel & Thielscher, 2010), the players are *not* informed about the joint move; rather each role $r \in R \setminus \{\texttt{random}\}$ gets to see any p that satisfies $\mathcal{I}(r, \mu, \sigma, p)$. The game ends as soon as a terminal state is reached, and then the goal relation determines the result for the players. This modified execution model is spelled out in Figure 3. It is straightforwardly implemented on a Game Master, which runs a game by collecting all moves, which allows it to maintain the actual game state and thus to compute all percepts and also to determine the end of a match and the resulting goal values for the players.

1. Send each $r \in R \setminus \{\texttt{random}\}$ the GDL-II description and inform them about their individual roles r (e.g., *xplayer* or *oplayer*). Set $\sigma := \sigma_0$.
2. After the appropriate time, collect a move m_r from each player $r \in R \setminus \{\texttt{random}\}$ and, in case $\texttt{random} \in R$, choose with uniform probability an element $m_{\texttt{random}}$ from the set $\{m : (\texttt{random}, m, \sigma) \in l\}$. Set $\mu := \{r_1 \mapsto m_{r_1}, \ldots, r_k \mapsto m_{r_k}\}$.
3. Send each $r \in R \setminus \{\texttt{random}\}$ the set of percepts $\{p : (r, \mu, \sigma, p) \in \mathcal{I}\}$. Set $\sigma := u(\mu, \sigma)$.
4. Repeat 2. and 3. until $\sigma \in T$. Determine the result v for $r \in R$ by $(r, v, \sigma) \in g$.

Figure 3: Game play in GDL-II given a game $(R, \sigma_0, T, l, u, \mathcal{I}, g)$.

2.6 A New Communication Protocol

The changes in the execution model from GDL to GDL-II require some modifications to the communication protocol between the Game Master program and the general game players. First of all, the Game Master has to inform the players about their individual percepts instead of the joint move from the previous step. For practical purposes, the Game Master should also confirm back each individual move as well as the current step in the game. This information is useful for players to become aware of communication problems (such as dropped messages or timeouts) and to be able to recover from those problems at least in some cases. In order to make the transition from GDL to GDL-II easier, we keep most of the communication protocol as defined by Love et al. (2006) and only apply the necessary changes.

7. Note that this does not necessarily mean that all resulting states have equal probability. For example, tossing an unfair coin that shows head just with probability $\frac{1}{3}$ may be axiomatised in GDL-II by three legal moves for `random`, two of which have the same effect (the coin showing tails). We stress that basic GDL-II could easily be extended by syntactic means for specifying non-uniform transition probabilities.

The communication between the Game Master and the players happens through HTTP messages, where the players take the role of HTTP servers that "serve" moves to the Game Master. The body of the HTTP messages are commands that are encoded using the Knowledge Interchange Format (KIF) (Genesereth, 1991). We use the messages `START`, `PLAY` and `STOP` as follows.

`(START <MATCHID> <ROLE> <DESCRIPTION> <STARTCLOCK> <PLAYCLOCK>)`

This is the first message sent to each player. It contains a unique identifier for the match (`<MATCHID>`) and informs the player about his role in the game (`<ROLE>`), the game rules (`<DESCRIPTION>`) as well as the time constraints for the start-up phase (`<STARTCLOCK>`) and for submitting moves (`<PLAYCLOCK>`). Players are supposed to reply to this message with the string `READY` within `<STARTCLOCK>` seconds.

This message is identical to the start message in the original GDL communication protocol.

`(PLAY <MATCHID> <TURN> <LASTMOVE> <PERCEPTS>)`

This message is sent to a player at each step of the match. It informs the player about the number of moves so far in the game (`<TURN>`), his last move (`<LASTMOVE>`), and his percepts (`<PERCEPTS>`) according to the information relation $\mathcal{I}$.

`<TURN>` is 0 for the first play message and increased by one for every subsequent step of the game. For the first play message, when there is no previous move, `<LASTMOVE>` is `NIL`.

Players are supposed to reply within `<PLAYCLOCK>` seconds with their next move. If they do not submit a legal move on time, the Game Master will make a random selection on their behalf. In this case the `<LASTMOVE>` argument of the next message, which informs the player about that move, is vital knowledge to be able to continue.

`(STOP <MATCHID> <TURN> <LASTMOVE> <PERCEPTS>)`

This message is sent to each player for the last step of the match, that is, when a terminal state has been reached. It has the same structure as the play message above. Players should reply to this message with the string `DONE`.

Note that the parameter `<LASTMOVE>` in the play and stop messages is necessary if we want to guarantee that players always know their own move in case of dropped messages or timeouts. It is worth stressing that this information cannot be provided through a game rule like `sees(R,lastmove(M)) :- does(R,M).` This clause is redundant in the light of the execution model for GDL-II in Figure 3, which implies that players always choose, and hence know, their own moves anyway.

3. The Expressive Power of GDL-II

Percepts in GDL-II can be arbitrary terms and are triggered by arbitrary conditions on the current state and joint move. This provides greater flexibility than the standard definition of a sensing action in planning domain description languages, by which an agent learns about the truth values of one or more fluents (Golden & Weld, 1996; Petrick & Bacchus, 2004). Observations in GDL-II need not be related to one's own move, which is especially appropriate in a multi-agent setting, where often players see something as a side-effect of someone else's actions. A point in case are the rules 32 and 33 in Figure 1. This follows the spirit behind general effect axioms in GDL, that is, rules for `next`, which also may be independent of a specific move. A simple example are the rules 29–31 in Figure 2.

GDL-II descriptions are solely concerned with providing the players with an objective description of the rules of a game. This puts it in contrast also to standard axiomatisations in action formalisms (Lakemeyer & Levesque, 1998; Thielscher, 2000; Scherl & Levesque, 2003; Forth & Shanahan, 2004), where observations are described in terms of how they affect the knowledge of an agent. Our game description language is simpler in this regard because it is agnostic about how players use a percept to draw inferences, combine it with what they already know, and whether they remember it.

The conceptual simplicity of describing observations in GDL-II notwithstanding, our language extension gives rise to an intricate epistemic model about what a player can, in principle, know of a position and of the other players' knowledge. For example, the GDL-II rules for Krieg-Tictactoe shown in Figure 1 imply that one cell will carry *xplayer*'s mark after the first round but *oplayer* will be unable to determine which one. Moreover, provided that they have perfect recall (Rasmusen, 2007) and are capable of drawing the right conclusions, both players should know about each other's knowledge and lack of knowledge, respectively. In the following, we will analyse formally what the semantics of a GDL-II description and the execution model of Figure 3 entail about the knowledge that a player with perfect reasoning capabilities can have at any point during game play.

3.1 Legal Play Sequences

We begin by defining the set of all possible ways in which a game can develop.

Definition 6
Consider a game $(R, \sigma_0, T, l, u, \mathcal{I}, g)$. *A* legal play sequence *is*

$$\sigma_0 \stackrel{\mu_1}{\to} \sigma_1 \stackrel{\mu_2}{\to} \ldots \sigma_{n-1} \stackrel{\mu_n}{\to} \sigma_n$$

where $n \geq 0$ *and for all* $i \in \{1, \ldots, n\}$,

- μ_i *is a joint move, i.e., a mapping from players* r *to their individual move,* $\mu_i(r)$;
- $(r, \mu_i(r), \sigma_{i-1}) \in l$ *for all* $r \in R$ *(that is, players make legal moves); and*
- $\sigma_i = u(\mu_i, \sigma_{i-1})$ *(position update).*

Furthermore, $\{\sigma_0, \ldots, \sigma_{n-1}\} \cap T = \emptyset$, *that is, only the last state may be terminal.*

The following definition characterises precisely what a player with perfect reasoning capabilities can in principle know at a specific stage in the course of a game.

Definition 7
Let $\delta = \sigma_0 \stackrel{\mu_1}{\to} \sigma_1 \stackrel{\mu_2}{\to} \ldots \sigma_{n-1} \stackrel{\mu_n}{\to} \sigma_n$ *and* $\delta' = \sigma_0 \stackrel{\mu'_1}{\to} \sigma'_1 \stackrel{\mu'_2}{\to} \ldots \sigma'_{n-1} \stackrel{\mu'_n}{\to} \sigma'_n$ *be two legal play sequences in a game with roles* R, *initial state* σ_0, *and information relation* $\mathcal{I}$. *A player* $r \in R \setminus \{\texttt{random}\}$ cannot distinguish δ from δ' *if, and only if, the following holds for all* $i \in \{1, \ldots, n\}$.

1. $\{p : (r, \mu_i, \sigma_{i-1}, p) \in \mathcal{I}\} = \{p' : (r, \mu_i', \sigma_{i-1}', p') \in \mathcal{I}\}$ *and*
2. $\mu_i(r) = \mu_i'(r)$.

Example 2 *(continued)*
Let σ_1 be the initial state in Krieg-Tictactoe, where all cells are empty, and consider these two legal play sequences:

$$\delta \;=\; \sigma_0 \overset{\{xplayer \mapsto mark(1,3), oplayer \mapsto noop\}}{\rightarrow} \sigma_1 \overset{\{xplayer \mapsto noop, oplayer \mapsto mark(2,2)\}}{\rightarrow} \sigma_2 \tag{1}$$
$$\delta' \;=\; \sigma_0 \overset{\{xplayer \mapsto mark(2,2), oplayer \mapsto noop\}}{\rightarrow} \sigma_1' \overset{\{xplayer \mapsto noop, oplayer \mapsto mark(2,2)\}}{\rightarrow} \sigma_2' \tag{2}$$

Role *xplayer* can distinguish δ from δ' already after the first round because of his different moves. In contrast, *oplayer* finds the two sequences indistinguishable at this step since he took the same move, *noop*, and got the same percept $\{yourmove\}$ by rule 30 in Figure 1. Ultimately, however, *oplayer* also is able to distinguish (1) from (2) because, according to the game rules, after his second move he perceives $\{\,\}$ in δ but $\{yourmove\}$ in δ' (where he attempted to mark a non-blank cell). □

(In-)distinguishable play sequences can be used also to ensure that game descriptions obey desirable properties such as the following two, which are a straightforward consequence of Definition 7.

1. A game description with roles R and legality relation l entails that all players can always know their legal moves iff for all $r \in R \setminus \{\texttt{random}\}$ there are no two legal play sequences δ, δ' leading to states σ_n, σ_n' such that

$$\{m : (r, m, \sigma_n) \in l\} \neq \{m' : (r, m', \sigma_n') \in l\}$$

   and δ, δ' are indistinguishable for r.

2. A game description with roles R, terminal states T, and goal relation g entails that all players can know both the end of a game and their own result iff for all $r \in R \setminus \{\texttt{random}\}$ there are no two legal play sequences δ, δ' leading to states σ_n, σ_n' such that

$$\sigma_n \in T,\ \sigma_n' \notin T \quad \text{or} \quad \{\sigma_n, \sigma_n'\} \subseteq T,\ \{v : (r, v, \sigma_n) \in g\} \neq \{v' : (r, v', \sigma_n') \in g\}$$

   and δ, δ' are indistinguishable for r.

While a "fair" game should always satisfy both of these properties, proving them for a new game may be difficult in practice since in general this requires checking all possible legal play sequences. There is an easy way around this, namely, by adding `sees`-rules that always inform players explicitly about their legal moves, termination, and goal values. On the other hand, some games may be especially designed to test the ability of players to logically infer their legal moves under highly incomplete knowledge.

Example 4 *(continued)*
The candidate in our axiomatisation of the Monty Hall game (cf. Figure 2) can always derive her legal moves and when the game ends. This is easily seen from the fact that she can determine the value of the *step* fluent at all times. On the other hand, the candidate never receives enough information to be able to know her result. For example, she cannot distinguish these two legal play sequences:

$$\begin{aligned} \delta &= \sigma_0 \overset{\{candidate \mapsto choose\_door(1), random \mapsto hide\_car(1)\}}{\to} \sigma_1 \overset{\{candidate \mapsto noop, random \mapsto open\_door(3)\}}{\to} \sigma_2 \\ \delta' &= \sigma_0 \overset{\{candidate \mapsto choose\_door(1), random \mapsto hide\_car(2)\}}{\to} \sigma_1' \overset{\{candidate \mapsto noop, random \mapsto open\_door(3)\}}{\to} \sigma_2' \end{aligned}$$

But no matter which action the candidate chooses in the end, δ will result in a different goal value for her than δ'. □

Knowledge at the object level, as we have considered thus far, can be lifted to higher levels so as to determine what the players can know about each other's knowledge. This is possible because the GDL-II description of a game provides the complete rules to all players, so that each of them is able to derive which information the other roles get to see under any (hypothetical) legal play sequence. Formalising this requires constructing a suitable epistemic structure based on possible play sequences. Consider, to this end, for every δ the function $\mathcal{K}^\delta$ that maps every pair (r, i) onto the set of legal play sequences that player r at state i cannot distinguish from δ. Meta-level knowledge is then obtained as follows.

Definition 8
If the game develops according to δ, then as far as player r knows, all functions $\mathcal{K}^{\delta'}$ are possible where δ' is indistinguishable from δ for r.

Put in words, meta-level knowledge is characterised by a set of possible sets of legal play sequences. It follows that a player knows what holds in all $\mathcal{K}$-sets he considers possible. If, say, a Krieg-Tictactoe match unfolds according to the example sequence δ' from above (cf. (2)) then *xplayer* knows that *oplayer* knows that cell $(2, 2)$ is marked.

This process can be iterated inductively to determine arbitrary levels of meta-knowledge, which shows that common knowledge of the game rules does not necessarily mean common knowledge of other players. For example, it is possible to obtain a situation in a 3-player game where player A knows about a property f and where player B knows that A knows about f, while player C considers it possible that A knows nothing about f: Just let A make a move by which he learns about f in a game where B (but not C) is always informed about A's moves.

We conclude our analysis of (in-)distinguishable play sequences by proving that GDL-II provides a true extension of GDL since any game in which agents can derive the complete state after every round can be easily specified in GDL-II.

Proposition 1 Consider a game with roles R such that `random` $\notin R$ and where the only clause for keyword `sees` is

```
sees(R1,moves(R,M)) :- role(R1), does(R,M).
```

Then there are no two legal play sequences δ, δ' of any length $n \geq 0$ such that δ and δ' are indistinguishable for any $r \in R$.

Proof: By induction on n. For $n = 0$ there is only one legal play sequence, viz. the initial game state σ_0, which establishes the claim. For the induction step, suppose there are two legal play sequences δ, δ' leading to states $\sigma_{n+1} \neq \sigma'_{n+1}$ such that δ and δ' are indistinguishable for some role $r \in R$. From Definition 7 it follows that δ, δ' shortened by one step are indistinguishable for this player. By the induction hypothesis, this is only possible if δ and δ' are identical up to step n. Hence, the two legal play sequences are of the form

$$\begin{aligned} \delta &= \sigma_0 \overset{\mu_1}{\to} \ldots \overset{\mu_n}{\to} \sigma_n \overset{\mu_{n+1}}{\to} \sigma_{n+1} \\ \delta' &= \sigma_0 \overset{\mu_1}{\to} \ldots \overset{\mu_n}{\to} \sigma_n \overset{\mu'_{n+1}}{\to} \sigma'_{n+1} \end{aligned}$$

From Definition 5 it follows that state update is deterministic, hence $\sigma_{n+1} \neq \sigma'_{n+1}$ implies $\mu_{n+1} \neq \mu'_{n+1}$. The latter in conjunction with the above clause for `sees` implies that the percepts for all roles differ when in state σ_n joint move μ'_{n+1} is taken instead of μ_{n+1}. This contradicts the assumption that δ and δ' are indistinguishable for role r, which establishes the claim. □

3.2 GDL-II vs. Extensive Form

By virtue of its state-transition semantics and the concept of indistinguishable play sequences, every terminating GDL-II game can be understood as a so-called *extensive-form* game. For the following, we assume the reader to be familiar with the basic notion of a mathematical game tree with imperfect information (Rasmusen, 2007; Leyton-Brown & Shoham, 2008), which is based on the very general concept of *information sets* as a model of partial observability and information asymmetry. The `sees(R,P)` predicate in GDL-II can be used equally generally. The main reason is that percepts are not confined to specific observables (e.g., state features or opponents' moves) but can be arbitrary symbols used as identifiers for any desired information partition.

3.2.1 From GDL-II to Extensive-Form Games

For a given GDL-II game, a corresponding extensive-form game can be obtained in two steps.

1. Using an arbitrary but fixed order of the roles, joint moves in GDL-II are serialised in such a way that no player is aware of the simultaneous moves by the other players (Rasmusen, 2007).

2. The information sets for a player r are identified with the set of legal play sequences that this player cannot distinguish.

As an example, Figure 4 depicts the extensive-form game thus obtained from the Monty Hall game description in Figure 2.

It is important to realise that percepts in GDL-II may sometimes provide information that is not contained in a state but that nonetheless needs to be taken into account when partitioning nodes into information sets. The following example illustrates that the ability to solve a game may depend on such information.

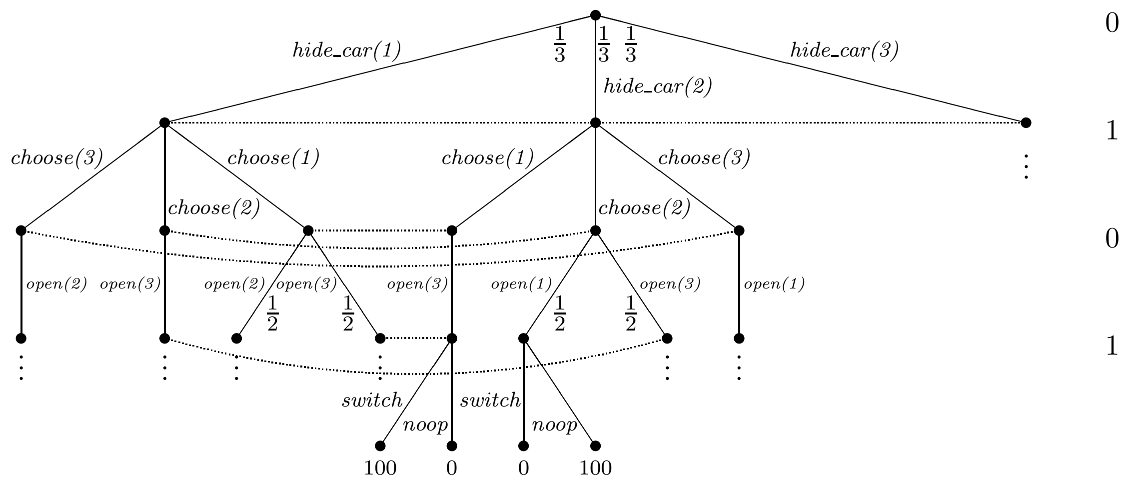

Figure 4: The Monty Hall game of Figure 2 mapped onto extensive form (with some nodes and branches omitted). The numbers to the very right indicate which player owns the nodes of that height (0 = `random`, 1 = *candidate*). Dotted lines connect nodes in the same *information set* for the candidate.

3.2.2 Example 5 *(Spy & Spy)*

The GDL-II in Figure 5 describes a game in which one spy sees which of three coloured wires is used to arm a bomb. He can signal the name of a colour to a second spy, who then tries to disarm the bomb. Both win if the second player cuts the right wire and lose otherwise. Hence, the first spy has every incentive to help the second spy, and to do so there seems to be one obvious rational move, namely, to signal the colour of the wire.

The crux, however, is that in the game rules, the colour being signalled is logically independent of the colour of the wire used to arm the bomb. This is illustrated in the game tree in Figure 6: If the second player were to distinguish only the possible states themselves after round 2, that is, $\{armed(red), step(3)\}$, $\{armed(blue), step(3)\}$, and $\{armed(green), step(3)\}$, then they all would belong to the same information set, no matter which colour has been signalled. This would leave both players with no better a choice than moving randomly. □

3.2.3 From Extensive-Form Games to GDL-II

To describe faithfully in GDL-II (that is, move-by-move) a game given in extensive form, two issues need to be addressed.

1. The given information sets need to be encoded by appropriate `sees`-rules for the individual players. This can be achieved by indexing these sets and always letting their owner see the index but nothing else.

2. Non-uniform probabilities for moves by nature need to be mapped onto uniform probability distributions for `random`'s moves. This can be achieved by introducing a

```
role(spy1).
role(spy2).
role(random).

colour(red).
colour(blue).
colour(green).

init(step(1)).

legal(random,arm_bomb(C)) :- colour(C), true(step(1)).
legal(spy1,signal(C))     :- colour(C), true(step(2)).
legal(spy2,cut_wire(C))   :- colour(C), true(step(3)).

legal(random,noop) :- not true(step(1))
legal(spy1,noop)   :- not true(step(2))
legal(spy2,noop)   :- not true(step(3))

sees(spy1,C) :- does(random,arm_bomb(C)).
sees(spy2,C) :- does(spy1,signal(C)).

next(armed(C))  :- does(random,arm_bomb(C)).
next(armed(C))  :- true(armed(C)).
next(explosion) :- does(spy2,cut_wire(C)), not true(armed(C)).
next(step(2))   :- true(step(1)).
next(step(3))   :- true(step(2)).
next(step(4))   :- true(step(3)).

terminal :- true(step(4)).

goal(spy1,100) :- not true(explosion).
goal(spy2,100) :- not true(explosion).
goal(spy1,  0) :-     true(explosion).
goal(spy2,  0) :-     true(explosion).
goal(random,0).
```

Figure 5: GDL-II description of a cooperative Spy & Spy game.

proportional number of moves that have different names but lead to the same successor state.

A finite extensive-form game can then be described in GDL-II by using a single fluent to indicate which node the game is at and by specifying each arc in the tree as a state transition. Using the methods of Schulte and Delgrande (2004), it can be shown that the resulting GDL-II description is correct for any finite n-player game in extensive form with perfect recall. The size of a description resulting from this transformation is obviously of the same order as the original game tree since moves are not parallelised and states are encoded as individual objects rather than being factored into atomic features. Hence, unlike the Monty Hall rules in Figure 2, say, this direct construction does not exploit the conciseness of descriptions made possible by having a high-level knowledge representation language.

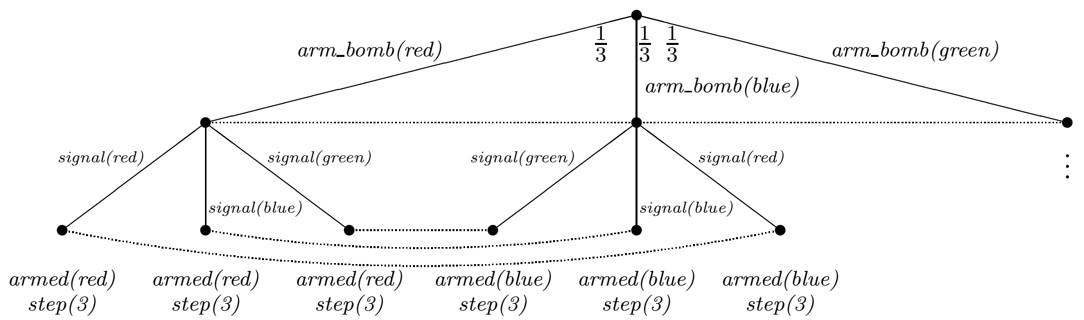

Figure 6: The game of Figure 5 mapped onto extensive form (with some nodes and branches omitted). Dotted lines connect nodes that *spy2* cannot distinguish. If the nodes at depth 3 with identical states were collapsed onto a single node, then they all would be in the same information set of this player.

4. A Logic for Reasoning About GDL-II Games

Building a basic general game player, e.g. one that only knows how to move legally, is a relatively simple task under the restriction to perfect-information games. In the general case, however, even basic game play is a much more intricate problem. Recall, for example, the description of the Krieg-Tictactoe game in Figure 1. The rules in lines 31–33 specify the players' percepts, indicating that the player who has control next will be informed about this fact. This suffices since both players should be able to always derive whether or not it is their turn. But to do so they must be capable of inferring that when they do *not* perceive *yourmove*, then it must be their opponent's turn. Another example of a (strategically 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* again, implying that your attempt must have been unsuccessful.

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). In this second part of the paper, we will lay the foundations for building general game-playing systems that are capable of reasoning with imperfect information. Using existing techniques like the Situation Calculus for this purpose requires a full embedding of the game description language GDL-II into these formalisms.

In the following, we will develop such a mapping based on the Situation Calculus variant that uses a special fluent to represent the knowledge of agents (Scherl & Levesque, 2003). We will have to slightly modify and further extend this formalism for our purposes. Generally speaking, the Situation Calculus is a predicate logic with a few pre-defined language elements:

- constant s_0, which denotes the *initial situation*, along with constructor $\text{Do}(A, S)$ to denote the situation resulting from doing *action* A in situation S;

- predicate $\text{Holds}(F, S)$, which denotes that *fluent* F (i.e., an atomic state feature) is true in situation S;

- predicate $\text{Poss}(A, S)$, which denotes that action A is possible in situation S.

4.1 Compound Actions

In games with two or more players, positions are updated as a consequence of all players moving simultaneously. For an adequate formalisation in the Situation Calculus we therefore need to identify the concept of an action 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 (second) arguments of the keywords `legal` and `does`; e.g. $mark(M, N)$ and *noop* in Krieg-Tictactoe. Assuming an arbitrary but fixed order of the players, say as in $(xplayer, oplayer)$, we define a simple axiom that identifies the individual move of a player r in an action vector:

$$\text{Act}(r_i, \langle M_1, \ldots, M_i, \ldots, M_k \rangle) = M_i \ . \tag{3}$$

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

4.2 Derived Action Predicates

Given a GDL-II game description G, 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 Figure 1 these are $control(R)$, $cell(M, N, Z)$, and $tried(R, M, N)$. As *derived fluents* we take all domain-specific predicates that depend on `true` but not on `does` in G. Derived fluents (Davis, 1990) 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,V)` are treated as derived fluents too.

In addition, 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` in G. An example is *validmove* in our description of Krieg-Tictactoe (cf. line 16 in Figure 1). 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 (compound) action that is being taken in that situation.

4.3 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.[8]
2. Each `init`(F) is replaced by $\text{Holds}(F, s_0)$.
3. Each `true`(F) is replaced by $\text{Holds}(F, S)$.
4. Each `next`(F) is replaced by $\text{Holds}(F, \text{Do}(A, S))$.

8. This mapping also applies to the derived fluents `legal`(R, M), `goal`(R, V), and `terminal`.

5. Each $\texttt{does}(R, M)$ is replaced by $\text{Act}(R, A) = M$.

As an example, the clause in line 16 of Figure 1 becomes

$$\mathit{validmove}(A, S) \subset \text{Act}(R, A) = \mathit{mark}(M, N) \wedge \text{Holds}(\mathit{cell}(M, N, b), S) \ .$$

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* (Clark, 1978) of all clauses in the following way: For every predicate $p(\vec{X})$, replace the clauses in G with p in the head by

$$p(\vec{X}) \ \equiv \bigvee_{p(\vec{t})\,\texttt{:-}\,\mathit{body}\,\in G} (\exists)\vec{X} = \vec{t} \wedge \mathit{body} \ . \tag{4}$$

In this context we use $(\exists)$ as abbreviation for $\exists \vec{Y}$, where $\vec{Y}$ are all variables that occur in either $\vec{t}$ or *body* but not in $\vec{X}$.

4.3.1 Initial Situation

The transformation defined above yields the following axiomatisation of the initial situation:

$$\text{Holds}(F, s_0) \ \equiv \bigvee_{\texttt{init}(t)\,\texttt{:-}\,\mathit{body}\,\in G} (\exists)F = t \wedge \mathit{body} \ .$$

4.3.2 Preconditions

Based on the completion of `legal` according to (4), i.e.,

$$\text{Legal}(R, M, S) \ \equiv \bigvee_{\texttt{legal}(r,m)\,\texttt{:-}\,\mathit{body}\,\in G} (\exists)R = r \wedge M = m \wedge \mathit{body} \ ,$$

we define the precondition axiom for compound actions A in situations S thus:

$$\text{Poss}(A, S) \ \equiv \forall R.\, \text{Legal}(R, \text{Act}(R, A), S) \ . \tag{5}$$

4.3.3 Effects

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

$$\text{Holds}(F, \text{Do}(A, S)) \ \equiv \bigvee_{\texttt{next}(t)\,\texttt{:-}\,\mathit{body}\,\in G} (\exists)F = t \wedge \mathit{body} \ . \tag{6}$$

4.3.4 Knowledge

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

$$\text{Knows}(R, \Phi, S) \ \stackrel{\text{def}}{=} \ \forall S'.\, \mathbf{K}(R, S', S) \supset \Phi[S'] \ . \tag{7}$$

Here, Φ 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 $\forall S'.\, \mathbf{K}(\mathit{xplayer}, S', s_0) \supset \forall X, Y.\, \textsc{Holds}(\mathit{cell}(X, Y, b), S')$. To express the knowledge of a player about another player's knowledge, macro definition (7) can be easily extended to form nested expressions, as in

$$\textsc{Knows}(\mathit{xplayer}, \textsc{Knows}(\mathit{oplayer}, \mathit{control}(\mathit{oplayer})), \textsc{Do}(\langle \mathit{mark}(1,1), \mathit{noop}\rangle, s_0)) \ .$$

It is also possible to define *common knowledge* of a group of players as any property that is shared by all situations belonging to the reflexive and transitive closure of the combined accessibility relations.

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

$$\mathbf{K}(R, S, s_0) \ \equiv \ S = s_0 \ . \tag{8}$$

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 Scherl and Levesque's definition as follows:

$$\begin{aligned} \mathbf{K}(R, S'', \textsc{Do}(A, S)) \ \equiv \ \exists A', S'.\ & S'' = \textsc{Do}(A', S') \wedge \mathbf{K}(R, S', S) \wedge \\ & \textsc{Poss}(A', S') \wedge \textsc{Act}(R, A) = \textsc{Act}(R, A') \wedge \\ & \forall P.\ \textsc{Sees}(R, P, A, S) \equiv \textsc{Sees}(R, P, A', S') \ . \end{aligned} \tag{9}$$

Put in words, a player considers S'' a possible situation after joint move 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 his sensing result for the actual A, S (so that he cannot distinguish the two).

It should be noted that axioms (8) and (9) postulate both perfect recall and common knowledge of the game rules among all players: From any actual situation $\textsc{Do}(\ldots, S_0)$ only situations that are consistent with all of a player's previous information are accessible. Moreover, all these accessible situations are of the form $\textsc{Do}(\ldots, S_0)$ too, which implies that every situation belonging to the reflexive and transitive closure of the combined players' accessibility relations are governed by the same precondition and effect rules described by axioms (5)–(6).

4.4 Completion Semantics and Stable Models

The axiomatisation 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 its (unique) stable model in the presence of redundant rules like `validmove :- validmove`. The stable model remains the same when such "superfluous" clauses are added, but Clark's completion is weakened by them. For example, for the mentioned rule (assuming no further clauses) the only stable model is the empty set, while the first-order semantics of the completion admits to two models—one in which `validmode` holds and one where it is false.

This issue can be resolved by a second-order axiom (Ferraris, Lee, & Lifschitz, 2011). Denoted by $SM[F]$, the axiom provides a stable model operator for arbitrary first-order formulas F. The operator $SM[F]$ is defined in the following way:

Definition 9
Let $\vec{P} = (P_1, \ldots, P_n)$ be a list of predicates and $\vec{U} = (U_1, \ldots, U_n)$ be a list of distinct predicate variables of the same length as $\vec{P}$. Furthermore, let $\vec{U} = \vec{P}$ denote the conjunction of the formulas $\forall \vec{X}. U_i(\vec{X}) \equiv P_i(\vec{X})$ for all $i = 1, \ldots, n$. By $\vec{U} \leq \vec{P}$ we denote the conjunction of the formulas $\forall \vec{X}. U_i(\vec{X}) \supset P_i(\vec{X})$ for all $i = 1, \ldots, n$, and $\vec{U} < \vec{P}$ stands for $(\vec{U} \leq \vec{P}) \land \neg(\vec{U} = \vec{P})$.

The expression $\mathrm{SM}[F; \vec{P}]$ stands for the second-order sentence

$$F \land \neg \exists \vec{U}. (\vec{U} < \vec{P}) \land F^*(\vec{U}) \ ,$$

where $F^*(\vec{U})$ is defined recursively:

- $P_i(\vec{t})^* = U_i(\vec{t})$ *for any list $\vec{t}$ of terms;*
- $F^* = F$ *for any atomic formula F that does not contain members of $\vec{P}$;*
- $(F \land G)^* = F^* \land G^*$ *;*
- $(F \lor G)^* = F^* \lor G^*$ *;*
- $(F \supset G)^* = (F^* \supset G^*) \land (F \supset G)$ *;*
- $(\neg F)^* = (F \supset \bot)^*$ *;*
- $(F \equiv G)^* = (F \supset G)^* \land (G \supset F)^*$ *;*
- $(\forall \vec{X}. F)^* = \forall \vec{X}. F^*$ *;*
- $(\exists \vec{X}. F)^* = \exists \vec{X}. F^*$ *.*

Expression $\mathrm{SM}[F]$ is shorthand for $\mathrm{SM}[F; \vec{P}]$ where $\vec{P}$ is the list of all predicates in F.

The models of $SM[F]$ are the stable models of F. Specifically, if F is the completion of a stratified logic program, $SM[F]$ has a unique Herbrand model that corresponds to the unique stable model of the logic program.

As the last step of our translation we therefore add the axiom $SM[F]$, where F is the conjunction of all rules in the transformed game description.

It should be noted that the Situation Calculus typically uses a second-order induction axiom to limit the set of situations to the smallest set containing s_0 that is closed under the Do operator. However, different from $SM[F]$, the induction axiom is not sufficient to enforce a unique model in the presence of redundant rules.

4.5 Soundness and Completeness

The theory obtained by the transformation developed in the previous section is indeed a Situation Calculus theory, as we will show now.

Proposition 2 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 (5), the variable A can be instantiated with every compound action to obtain an axiom of the form above. The right-hand side of (5) 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 (6) the variable F can be instantiated with every primitive fluent to obtain an axiom of the form above. The right-hand side of (6) refers to the bodies of rules with head `next`(F'). 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. □

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 Definition 6 the notion of a legal play sequence:

$$\sigma_0 \overset{\mu_1}{\to} \sigma_1 \overset{\mu_2}{\to} \ldots \sigma_{n-1} \overset{\mu_n}{\to} \sigma_n \ .$$

Intuitively, any such sequence can be interpreted as the situation

$$\textsc{Do}(A_n, \ldots, \textsc{Do}(A_2, \textsc{Do}(A_1, s_0)) \ldots)$$

in the Situation Calculus, where each joint move $\mu_i = \{(r_1, m_1), \ldots, \ (r_k, m_k)\}$ corresponds to a compound action $A_i = \langle m_1, \ldots, m_k \rangle$. The following theorem states that for a given state and joint move, the GDL-II game rules and the Situation Calculus theory entail the same propositions.

Theorem 3 Let G be a valid GDL-II game description and D be the Situation Calculus theory that is obtained from G with the translation defined above. Furthermore, let $\delta = \sigma_0 \overset{\mu_1}{\to} \ldots \overset{\mu_n}{\to} \sigma_n$ be a legal play sequence of the game with corresponding situation $S = \textsc{Do}(A_n, \ldots, \textsc{Do}(A_1, s_0) \ldots)$; μ be a joint move legal in σ_n; and A be the compound action corresponding to μ.

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

- $\texttt{true}^t(F, A, S) = \textsc{Holds}(F, S)$,
- $p^t(\vec{X}, A, S) = p(\vec{X}, S)$ 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, $p(\vec{X}) \in SM[G \cup \sigma_n^{\texttt{true}} \cup \mu^{\texttt{does}}]$ iff $D \models p^t(\vec{X}, A, S)$.

Proof: The theorem follows from the construction of the Situation Calculus theory in Section 4.3 and a result by Ferraris et al. (2011) according to which the unique stable 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. $\square$

Our main proposition states that indistinguishable legal play sequences for some player correspond to situations that 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.

Proposition 4 Let G be a valid GDL-II description with semantics $(R, \sigma_0, T, l, u, \mathcal{I}, g)$ and D be the Situation Calculus theory obtained from G with the translation defined above. Let there be two legal play sequences

$$\begin{aligned} \delta \ &= \sigma_0 \overset{\mu_1}{\to} \ldots \overset{\mu_n}{\to} \sigma_n, \text{ and} \\ \delta' \ &= \sigma_0 \overset{\mu'_1}{\to} \ldots \overset{\mu'_n}{\to} \sigma'_n \end{aligned}$$

of the game with the corresponding situations $S = \textsc{Do}(A_n, \ldots, \textsc{Do}(A_1, s_0)\ldots)$ and $S' = \textsc{Do}(A'_n, \ldots, \textsc{Do}(A'_1, s_0)\ldots)$. A role $r \in R$ cannot distinguish δ and δ' iff $D \models \mathbf{K}(r, S', S)$.

Proof: We prove the proposition by induction on the length n of sequence δ.

For the base case $n = 0$ there is only one legal play sequence—the initial state σ_0 with the corresponding initial situation s_0. Accordingly, by (8), $D \models \mathbf{K}(r, S, s_0)$ iff $S = s_0$, which proves the base case.

For the induction step, consider the two legal play sequences

$$\delta_+ = \delta \overset{\mu_{n+1}}{\to} \sigma_{n+1} \quad \text{and} \quad \delta'_+ = \delta' \overset{\mu'_{n+1}}{\to} \sigma'_{n+1}$$

with the corresponding situations $\textsc{Do}(A_{n+1}, S)$ and $\textsc{Do}(A'_{n+1}, S')$, respectively.

We have to show that δ_+, δ'_+ are indistinguishable for r if, and only if,

$$D \models \mathbf{K}(r, \textsc{Do}(A'_{n+1}, S'), \textsc{Do}(A_{n+1}, S)) \ .$$

From the successor state axiom for the special fluent $\mathbf{K}$ (cf. axiom (9)), we know that $\mathbf{K}(r, \textsc{Do}(A'_{n+1}, S'), \textsc{Do}(A_{n+1}, S))$ if, and only if, all of the following hold.

(a) $\mathbf{K}(r, S', S)$;

(b) $\textsc{Poss}(A'_{n+1}, S')$;

(c) $\textsc{Act}(r, A_{n+1}) = \textsc{Act}(r, A'_{n+1})$; and

(d) $\forall P.\, \textsc{Sees}(r, P, A_{n+1}, S) \equiv \textsc{Sees}(r, P, A'_{n+1}, S')$.

We now show that each one of these holds.

By the induction hypotheses, (a) holds.

From the definition of a legal play sequence (Definition 6) it follows that each player's move in μ'_{n+1} is legal in the state σ'_n, that is, for all players r,

$$\texttt{legal}(r, \mu'_{n+1}(r)) \in SM[G \cup \sigma'^{\texttt{true}}_n] \,.$$

By Theorem 3 we conclude that this holds exactly if $D \models \textsc{Legal}(r, \mu'_{n+1}(r), S')$ for all players r. Because of $\mu'_{n+1}(r) = \textsc{Act}(r, A'_{n+1})$, this is equivalent to the right hand side of our precondition axiom (5) and hence equivalent to $\textsc{Poss}(A'_{n+1}, S')$. Thus (b) follows.

According to Definition 7, provided that δ and δ' are indistinguishable for role r, δ_+ and δ'_+ are indistinguishable for r if, and only if,

$$\{p : (r, \mu_{n+1}, \sigma_n, p) \in \mathcal{I}\} = \{p' : (r, \mu'_{n+1}, \sigma'_n, p') \in \mathcal{I}\} \tag{10}$$

$$\text{and} \quad \mu_{n+1}(r) = \mu'_{n+1}(r) \tag{11}$$

By the definition of $Act(r, A)$ (cf. (3)), $Act(r, A_{n+1}) = \mu_{n+1}(r)$ and $Act(r, A'_{n+1}) = \mu'_{n+1}(r)$. Thus, equation (11) holds if and only if (c).

By Definition 5, $\mathcal{I} = \{(r, \mu, \sigma, p) : \texttt{sees}(r, p) \in SM[G \cup \sigma^{\texttt{true}} \cup \mu^{\texttt{does}}]\}$. Thus, $(r, \mu'_{n+1}, \sigma_n, p) \in \mathcal{I}$ if, and only if, $\texttt{sees}(r, p) \in SM[G \cup \sigma^{\texttt{true}}_n \cup \mu^{\texttt{does}}_{n+1}]$. By Theorem 3, this is equivalent to $D \models \textsc{Sees}(r, p, A_{n+1}, S)$. The same reasoning holds for the right hand side of (10). Thus, (10) holds just in case (d) holds.

This proves that (a) to (d) are consequences of the induction hypothesis. Hence, $\mathbf{K}(r, \textsc{Do}(A'_{n+1}, S'), \textsc{Do}(A_{n+1}, S))$ holds if, and only if, r cannot distinguish δ_+ and δ'_+, which concludes the induction step and the proof. □

4.6 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 the restricted recursion according to Definition 3. Our mapping extends the GDL rules by situation arguments, but also for these only finitely many ground instances are considered in Theorem 3 because the legal play sequences are fixed and hence the situations are depth-restricted. 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 (Bonatti, 2001). Thus, we can consider a finite ground program and replace the second-order axiom by propositional loop formulas (Lin & 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.[9]

9. 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.

Although with these points in mind decidability of reasoning is not an issue, tractability still is. It is possible to construct game rules where computing all legal moves alone is too expensive to do any kind of planning ahead in a reasonable amount of time. A practically viable approach is the use of sound but incomplete proof methods (Haufe & Thielscher, 2012; Thielscher, 2013).

5. Related Work

The new language GDL-II is the first extension of the existing GDL for nondeterministic games with imperfect information. The idea behind General Game Playing itself goes back to early work by Pitrat (1968) and, later, Pell (1993). Both define formal languages to describe whole classes of games, but these languages are even less general than basic GDL for perfect-information games. Earlier work also includes the definition of Gala for describing general games with imperfect information (Koller & Pfeffer, 1997). The main difference to GDL-II is that Gala is tightly coupled with a programming language (Prolog) and therefore has an operational—rather than declarative—semantics. It should be mentioned also that GDL-II draws from concepts that have been used in Action and Planning Languages (Lobo, Mendez, & Taylor, 2001) to represent effects of actions in the presence of incomplete knowledge. GDL-II can be seen as generalising this line of work to multi-agent and competitive settings.

The keyword `sees` that we introduced into the Game Description Language allows us to describe a player's percepts as side-effects of moves. This is different from the traditional modelling in reasoning about actions of observations as direct effects of sensing actions (Scherl & Levesque, 2003; Thielscher, 2000; Lobo et al., 2001) and is especially appropriate for multi-player games where agents may see something without having acted, e.g., when they observe other players' moves or cards they are being dealt.

Our translation from GDL-II into the Situation Calculus is the first full embedding of GDL-II into an action formalism; a recent mapping (Thielscher, 2011) into the Action Description Language $\mathcal{C}+$ (Giunchiglia, Lee, Lifschitz, McCain, & Turner, 2004) is restricted to basic GDL and hence covers neither imperfect information games nor knowledge and sensing. But we expect our translation to provide the basis for mappings of GDL-II into other existing action languages, for example, the one developed by Lobo et al. (2001). This opens up the road to the full range of applicable methods and systems that exist in reasoning about actions and AI planning.

The Situation Calculus has previously been shown to be a viable formalism for representing and reasoning about games by Schulte and Delgrande (2004), who also use the Situation Calculus variant with knowledge (Scherl & Levesque, 2003) to axiomatise extensive-form games. On the one hand, their approach is more general then ours in that it can deal with infinite games, does not assume perfect recall and is not restricted to uniform probabilities for nature's moves. On the other hand, it is these restrictions that make GDL-II a compact axiomatisation language which focuses on the rules of games rather than on how to reason about them. In addition, GDL-II has a state update semantics, which is in general preferred over the usual regression semantics of the Situation Calculus for performance reasons.

A limitation of Schulte and Delgrande's axiomatisation is the restriction of 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 agent can conclude about what another agent would know. This is remedied in a recent definition of a multi-agent epistemic variant of the Situation Calculus for axiomatising games (Belle & Lakemeyer, 2010). Despite notable differences to GDL-II and our variant of the Situation Calculus, e.g. the restriction to non-simultaneous moves and the augmentation of a domain theory with an epistemic theory about subjective knowledge, 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's approach.

There are other formalisms beside the Situation Calculus for reasoning about knowledge and actions in multi-agent environments, for example, the epistemic logic developed by Belardinelli and Lomuscio (2009). A translation of GDL-II into this formalism seems possible and might be interesting because model checkers for this logic are available (Lomuscio, Qu, & Raimondi, 2009).

Several General Game Playing systems exist that are able to play games encoded in GDL-II (Schofield, Cerexhe, & Thielscher, 2012; Edelkamp, Federholzner, & Kissmann, 2012). Although these systems have to reason about a game as well, the focus lies on solving specific reasoning tasks (such as computing legal moves and possible continuations of the game) under time constraints.

6. Conclusion

One of the reasons why General Game Playing has not yet found as many applications outside the game-playing area as it could, is that the current state of the art is restricted to deterministic games with complete state information. Aiming at overcoming this limitation, we have defined a conceptually simple yet powerful extension of the Game Description Language for representing the rules of general games with information asymmetry and random moves. We have shown that GDL-II, notationally simple as it is, gives rise to an intricate epistemic model and thus suffices to provide players with all information they need to reason about their own knowledge as well as that of the other players up front and during game play.

We have argued that our language extension suffices to describe any finite n-player game in extensive form with perfect recall. This shows that for the purpose of General Game Playing the language GDL-II can be considered complete; additional elements can only serve to obtain more succinct descriptions (e.g., by allowing explicit specifications of non-uniform probabilities for moves by `random`) or will be needed when the very concept of General Game Playing itself is extended beyond the current setting, e.g. to open-world games such as Scrabble (Sheppard, 2002) or systems that play real, physical games (Barbu, Narayanaswamy, & Siskind, 2010).

In the second part of the paper, we have presented an embedding of our extended game description language into a suitable variant of the Situation Calculus that features multi-agent knowledge, simultaneous actions, and action-independent sensing. While a GDL-II game description itself defines what observations players make throughout a game but not how they use this information, 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.

A central aspect of our work is the distinction between objective information on the one hand and how it affects the epistemic state of an agent on the other hand. The difference manifests itself in the two key predicates of this paper: GDL-II keyword `sees(R,P)` describes objective information while $\textsc{Knows}(R, \Phi, S)$ in the Situation Calculus axiomatises subjective knowledge. This distinction is also present in the two semantics given in this paper: state transition systems to model the rules of a game environment and Situation Calculus axiomatisations to model the information processing of players. This draws a connection of our work to the more general distinction often made in AI between the functionality of agents and properties of their task environments (Russell & Norvig, 2010). Seen from this perspective, GDL-II extends GDL in that it enables us to describe a larger class of game environments in a rule-based language. Meanwhile, our Situation Calculus axiomatisation can be viewed as a model for game-playing agents whose functionality includes perfect recall and reasoning abilities. An interesting direction for future work is to develop alternative Situation Calculus axiomatisations of reasoning about GDL-II games and to use these to study other types of agents, e.g. which are memoryless and purely reactive, as well as aspects of bounded rationality in game-playing programs (Russell & Wefald, 1991).

Beyond General Game Playing we therefore envision applications of GDL-II in other AI areas in which game models arise, such as automated trading agents (Thielscher & Zhang, 2010), Multiagent Planning (Larbi, Konieczny, & Marquis, 2007), or Multiagent Systems in general. A general description language for games of any kind is potentially useful wherever there is a need for compact and high-level yet machine-processable specifications of problem domains and applications in the form of games. Also there is potential for using the available General Game Playing infrastructure and systems for developing and testing new games or for modelling, simulating, and reasoning about dynamic environments like, for example, economic systems. However, some additions to the language, such as arithmetic, might be necessary to allow for concise descriptions of some of these domains. In addition, GDL-II and the General Game Playing infrastructure have been used as tools for teaching AI and the development of agent programs for dynamic environments in several university courses.[10]

Acknowledgments

We thank Joohyung Lee and the anonymous reviewers of an earlier version of this paper for their thoughtful and very constructive comments. This research was supported under Australian Research Council's (ARC) *Discovery Projects* funding scheme (project number DP 120102023) and by the Icelandic Centre for Research (RANNIS, grant number 210019). The second author is the recipient of an ARC Future Fellowship (project number FT 0991348). He is also affiliated with the University of Western Sydney.

10. Some material is available at `www.general-game-playing.de/teaching/teaching.html`

References

Apt, K., Blair, H., & Walker, A. (1987). Towards a theory of declarative knowledge. In Minker, J. (Ed.), *Foundations of Deductive Databases and Logic Programming*, chap. 2, pp. 89–148. Morgan Kaufmann.

Barbu, A., Narayanaswamy, S., & Siskind, J. (2010). Learning physically-instantiated game play through visual observation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1879–1886, Anchorage. IEEE Press.

Belardinelli, F., & Lomuscio, A. (2009). Quantified epistemic logics for reasoning about knowledge in multi-agent systems. *Artificial Intelligence*, *173*(9–10), 982–1013.

Belle, V., & Lakemeyer, G. (2010). Reasoning about imperfect information games in the epistemic situation calculus. In Fox, M., & Poole, D. (Eds.), *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 255–260, Atlanta. AAAI Press.

Bonatti, P. (2001). Reasoning with open logic programs. In Eiter, T., Faber, W., & Trusczynski, M. (Eds.), *Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, Vol. 2173 of *LNCS*, pp. 147–159, Vienna. Springer.

Clark, K. (1978). Negation as failure. In Gallaire, H., & Minker, J. (Eds.), *Logic and Data Bases*, pp. 293–322. Plenum Press.

Clune, J. (2007). Heuristic evaluation functions for general game playing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1134–1139, Vancouver. AAAI Press.

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

Edelkamp, S., Federholzner, T., & Kissmann, P. (2012). Searching with partial belief states in general games with incomplete information. In Glimm, B., & Krüger, A. (Eds.), *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, Vol. 7526 of *LNCS*, pp. 25–36, Saarbrücken, Germany. Springer.

Edelkamp, S., & Kissmann, P. (2008). Symbolic classification of general two-player games. In Dengel, A., Berns, K., Breuel, T., Bomarius, F., & Roth-Berghofer, T. (Eds.), *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, Vol. 5243 of *LNCS*, pp. 185–192, Kaiserslautern. Springer.

Ferraris, P., Lee, J., & Lifschitz, V. (2011). Stable models and circumscription. *Artificial Intelligence*, *175*(1), 236–263.

Finnsson, H., & Björnsson, Y. (2008). Simulation-based approach to general game playing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 259–264, Chicago. AAAI Press.

Forth, J., & Shanahan, M. (2004). Indirect and conditional sensing in the event calculus. In de Mántras, R. L., & Saitta, L. (Eds.), *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 900–904, Valencia, Spain. IOS Press.

Gelder, A. V. (1989). The alternating fixpoint of logic programs with negation. In *Proceedings of the 8th Symposium on Principles of Database Systems*, pp. 1–10. ACM SIGACT-SIGMOD.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In Kowalski, R., & Bowen, K. (Eds.), *Proceedings of the International Joint Conference and Symposium on Logic Programming (IJCSLP)*, pp. 1070–1080, Seattle. MIT Press.

Genesereth, M. (1991). Knowledge interchange format. In Allen, J. F., Fikes, R., & Sandewall, E. (Eds.), *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 599–600, Cambridge. Morgan Kaufmann.

Genesereth, M., Love, N., & 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., & Turner, H. (2004). Nonmonotonic causal theories. *Artificial Intelligence*, *153*(1–2), 49–104.

Golden, K., & Weld, D. (1996). Representing sensing actions: The middle ground revisited. In Aiello, L. C., Doyle, J., & Shapiro, S. (Eds.), *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 174–185, Cambridge, MA. Morgan Kaufmann.

Grosz, B., Kraus, S., Talman, S., Stossel, B., & Havlin, M. (2004). The influence of social dependencies on decision-making: Initial investigations with a new game. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 782–789, New York.

Harsanyi, J. (1967). Games with incomplete information played by "Bayesian" players, I–III: Part I. the basic model. *Management Science*, *14*(3), 159–182.

Haufe, S., & Thielscher, M. (2012). Automated verification of epistemic properties for general game playing. In Brewka, G., Eiter, T., & McIlraith, S. (Eds.), *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 339–349, Rome. AAAI Press.

Jensen, R., & Veloso, M. (2000). OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research*, *13*, 189–226.

Kaiser, D. (2007). Automatic feature extraction for autonomous general game playing agents. In Durfee, E., Yokoo, M., Huhns, M., & Shehory, O. (Eds.), *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Honolulu.

Kaiser, D. (2008). The design and implementation of a successful general game playing agent. In Wilson, D., & Sutcliffe, G. (Eds.), *Proceedings of the International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pp. 110–115, Key West. AAAI Press.

Kirci, M., Sturtevant, N., & Schaeffer, J. (2011). A GGP feature learning algorithm. *KI—Künstliche Intelligenz*, *25*, 35–42. Springer.

Kissmann, P., & Edelkamp, S. (2011). Gamer, a general game playing agent. *KI—Künstliche Intelligenz*, *25*, 49–52. Springer.

Koller, D., & Pfeffer, A. (1997). Representations and solutions for game-theoretic problems. *Artificial Intelligence*, *94*(1), 167–215.

Kuhlmann, G., Dresner, K., & Stone, P. (2006). Automatic heuristic construction in a complete general game player. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1457–1462, Boston. AAAI Press.

La Mura, P. (2000). Game networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 335–342, Stanford. Morgan Kaufmann.

Lakemeyer, G., & Levesque, H. (1998). $\mathcal{AOL}$: A logic of acting, sensing, knowing, and only knowing. In Cohn, A. G., Schubert, L. K., & Shapiro, S. C. (Eds.), *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 316–327, Trento. AAAI Press.

Larbi, R. B., Konieczny, S., & Marquis, P. (2007). Extending classical planning to the multi-agent case: A game-theoretic approach. In Mellouli, K. (Ed.), *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, Vol. 4724 of *LNCS*, pp. 63–75, Hammamet. Springer.

Leyton-Brown, K., & Shoham, Y. (2008). *Essentials of Game Theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool.

Lin, F., & Zhao, Y. (2004). ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence*, *157*, 115–137.

Lloyd, J. (1987). *Foundations of Logic Programming* (second, extended edition). Series Symbolic Computation. Springer.

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

Lobo, J., Mendez, G., & Taylor, S. R. (2001). Knowledge and the action description language $\mathcal{A}$. *Theory and Practice of Logic Programming*, *1*(2), 129–184.

Lomuscio, A., Qu, H., & Raimondi, F. (2009). MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of the International Conference on Computer Aided Verification (CAV)*, Vol. 5643 of *LNCS*, pp. 682–688, Grenoble, France. Springer.

Love, N., Hinrichs, T., Haley, D., Schkufza, E., & Genesereth, M. (2006). General Game Playing: Game Description Language Specification. Tech. rep. LG–2006–01, Stanford Logic Group, Computer Science Department, Stanford University, 353 Serra Mall, Stanford, CA 94305. Available at: `games.stanford.edu`.

McCarthy, J. (1963). *Situations and Actions and Causal Laws*. Stanford Artificial Intelligence Project, Memo 2, Stanford University, CA.

Méhat, J., & Cazenave, T. (2011). A parallel general game player. *KI—Künstliche Intelligenz*, *25*, 43–47. Springer.

Pell, B. (1993). *Strategy Generation and Evaluation for Meta-Game Playing*. Ph.D. thesis, Trinity College, University of Cambridge.

Petrick, R., & Bacchus, F. (2004). Extending the knowledge-based approach to planning with incomplete information and sensing. In Dubois, D., Welty, C., & Williams, M.-A. (Eds.), *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 613–622, Whistler. AAAI Press.

Pitrat, J. (1968). Realization of a general game playing program. In Morrell, A. (Ed.), *Proceedings of IFIP Congress*, pp. 1570–1574, Edinburgh.

Pritchard, D. (1994). *The Encycolpedia of Chess Variants*. Godalming.

Rasmusen, E. (2007). *Games and Information: an Introduction to Game Theory* (4th edition). Blackwell Publishing.

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*, pp. 359–380. Academic Press.

Rosenhouse, J. (2009). *The Monty Hall Problem*. Oxford University Press.

Ruan, J., van der Hoek, W., & Wooldridge, M. (2009). Verification of games in the game description language. *Journal of Logic and Computation*, *19*(6), 1127–1156.

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd edition). Prentice Hall.

Russell, S., & Wefald, E. (1991). *Do the Right Thing: Studies in Limited Rationality*. MIT Press.

Scherl, R., & Levesque, H. (2003). Knowledge, action, and the frame problem. *Artificial Intelligence*, *144*(1), 1–39.

Schiffel, S. (2010). Symmetry detection in general game playing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 980–985, Atlanta. AAAI Press.

Schiffel, S., & Thielscher, M. (2007). Fluxplayer: A successful general game player. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1191–1196, Vancouver. AAAI Press.

Schiffel, S., & Thielscher, M. (2009). Automated theorem proving for general game playing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 911–916, Pasadena.

Schiffel, S., & Thielscher, M. (2010). A multiagent semantics for the Game Description Language. In Filipe, J., Fred, A., & Sharp, B. (Eds.), *Agents and Artificial Intelligence*, Vol. 67 of *Communications in Computer and Information Science*, pp. 44–55. Springer.

Schofield, M., Cerexhe, T., & Thielscher, M. (2012). HyperPlay: A solution to general game playing with imperfect information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1606–1612, Toronto. AAAI Press.

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

Sheppard, B. (2002). World-championship-caliber Scrabble. *Artificial Intelligence*, *134*(1–2), 241–275.

Thielscher, M. (2000). Representing the knowledge of a robot. In Cohn, A., Giunchiglia, F., & Selman, B. (Eds.), *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 109–120, Breckenridge. Morgan Kaufmann.

Thielscher, M. (2011). Translating general game descriptions into an action language. In Balduccinin, M., & Son, T. (Eds.), *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in Honor of Michael Gelfond*, Vol. 6565 of *LNAI*, pp. 300–314. Springer.

Thielscher, M. (2013). Filtering with logic programs and its application to general game playing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Bellevue. AAAI Press.

Thielscher, M., & Voigt, S. (2010). A temporal proof system for general game playing. In Fox, M., & Poole, D. (Eds.), *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1000–1005, Atlanta. AAAI Press.

Thielscher, M., & Zhang, D. (2010). From general game descriptions to a market specification language for general trading agents. In David, E., Gerding, E., Sarne, D., & Shehory, O. (Eds.), *Agent-Mediated Electronic Commerce: Designing Trading Strategies and Mechanisms for Electronic Markets*, Vol. 59 of *LNBIP*, pp. 259–274. Springer.