Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

A General Game Description Language for Incomplete Information Games

Michael Thielscher
School of Computer Science and Engineering
The University of New South Wales, Australia
mit@cse.unsw.edu.au

Abstract

A General Game Player is a system that can play previously unknown games given nothing but their rules. The *Game Description Language (GDL)* has been developed as a high-level knowledge representation formalism for axiomatising the rules of any game, and a basic requirement of a General Game Player is the ability to reason logically about a given game description. In this paper, we address the fundamental limitation of existing GDL to be confined to deterministic games with complete information about the game state. To this end, we develop an extension of GDL that is both simple and elegant yet expressive enough to allow to formalise the rules of arbitrary (discrete and finite) n-player games with randomness and incomplete state knowledge. We also show that this extension 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.

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. Identified as a Grand Challenge for Artificial Intelligence, this endeavour requires to combine methods from a variety of sub-disciplines, including automated reasoning, search, game playing, and learning (Pell 1993). The annual AAAI GGP Competition has been established in 2005 to foster research in this area (Genesereth, Love, and Pell 2005). This has lead to a number of successful approaches, including (Kuhlmann, Dresner, and Stone 2006; Clune 2007; Schiffel and Thielscher 2007; Finnsson and Björnsson 2008). The first international workshop on General Game Playing at IJCAI'09 also serves as a witness for the rapid growth of this research field.

Representing and reasoning about knowledge is a core technique in GGP. A formal Game Description Language is used to represent the rules of n-player games ($n \geq 1$) in such a way that they can be automatically processed by a general game player (Love et al. 2006). The emphasis is on high-level, declarative descriptions. This allows successful players to reason about the rules of an unknown game in order to extract game-specific knowledge (Schiffel and Thielscher 2009a) and to automatically design evaluation functions (Kuhlmann, Dresner, and Stone 2006; Clune 2007; Schiffel and Thielscher 2007).

A fundamental limitation of the existing Game Description Language (GDL), and therefore of contemporary GGP systems, is the restriction to deterministic games with complete information about the game state. While a game may involve simultaneous moves, players are immediately informed about the moves by their opponents and have complete knowledge of the game model and the current position throughout the game (Schiffel and Thielscher 2009b). This applies to a variety of classical games such as Chess, Go, Chinese Checkers, etc. On the other hand, the existing limitation excludes games with elements of chance like Backgammon, games with information asymmetry such as Bridge or Poker, and games which involve private communication among cooperating players like in Bughouse Chess, or in the form of negotiations like in Diplomacy. Moreover, envisaged applications for General Game Playing systems, like automated trading agents, are usually characterised by imperfect information.

In this paper, we lay the foundations for truly General Game Playing by developing a fundamental extension of the existing description language, called GDL-II (for: Game Description Language with Incomplete Information).[1] We show that the addition of just two keywords suffices to obtain the desired generality: One, called `random`, denotes a special player who chooses moves randomly and can thus be used to model dice rolling, card shuffling, etc. The second new keyword, called `sees`, is used to control the information that each player gets. We provide the formal syntax and semantics of this new representation language. We then show that despite its conceptual simplicity and elegance, GDL-II 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

[1]When naming the new language we had to cope with an unfortunate clash of terminology: in AI, an agent who does not know the full state of the environment is said to have *incomplete* information; in Game Theory, when a player does not know the full state when called upon to move, the game is said to be of *imperfect* information. We decided to stick with the standard AI terminology.

`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,N)`	`R` gets `N` points in the current position
`sees(R,P)`	`R` perceives `P` in the next position
`random`	the random player

Table 1: GDL-II keywords: the top eight comprise standard GDL while the last two are added in view of incomplete state knowledge and elements of chance.[3]

player's knowledge.

The rest of the paper is organised as follows. In the next section, we introduce GDL-II as an extension of GDL, show various examples, and give a precise definition of the syntax of this new representation language. In the section that follows, we formally define its semantics. Finally, we show that the language is sufficiently expressive to give rise to an intricate multi-agent epistemic model.

From GDL to GDL-II

General Game Playing requires a formal language for describing the rules of arbitrary games. A complete game description includes: knowledge of the players and the initial position; the legal moves, and how they affect the position; and the terminating and winning criteria. The description language GDL has been developed for this purpose (Genesereth, Love, and Pell 2005; Love et al. 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 hitherto unknown game.

GDL is based on the standard syntax and semantics of Logic Programming (including the principle of negation by failure). A few special *keywords* are used for the different elements of a game description mentioned above. The eight keywords that comprise GDL are shown in the top of Table 1. GDL is suitable for describing any finite, synchronous, and deterministic n-player game.[4] The execution model (Genesereth, Love, and Pell 2005) entails full information symmetry: 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. A single new keyword `sees(R,P)` is needed for specifying the conditions under which player `R` receives information (i.e., "perceives") `P`. This 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. A second 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 to model games with elements of chance, such as rolling dice or shuffling cards. We again refer to Table 1 for the complete list of keywords that comprise the new language GDL-II.

Prior to giving the precise definition of syntax and semantics of GDL-II, let us illustrate the expressiveness of this extended, general Game Description Language with examples.

Example 1. *(Simple Card Game)* Figure 1 depicts a complete GDL-II description of a very simple card game. Two players `jane` and `rick` are accompanied by an anonymous dealer, modelled by the special role `random`. There are eight cards, named `7, 8, ..., king, ace`. The game consists of just two rounds, starting with a `dealingRound`.

The possible moves are specified by the rules with head `legal`: In the first round, the dealer randomly selects two different cards, one for each player. In the second round, indicated by state feature `bettingRound`, both players have the choice between two moves, `allIn` and `fold`.

The clauses with head `sees` specify the conditions under which the players will get some information about the game state. Each of them will know their own card (but not the opponent's card) once it has been dealt to them. After the second round, they will be informed about each other's bet. Finally, they will get to see each other's hand, but only in case both went all-in. These rules are accompanied by the clauses for `next`, which define the effects of the various moves by giving a complete account of the state features that hold after a (joint) move.

The remaining rules say that after betting the game ends as follows. In case both players went all-in, the one with the higher card wins it all (payout 100 vs. 0). If one player went all-in while the opponent folded, then the payout is 75 vs. 25, and if both folded, then the game ends in a draw.

This complete description shows the two new features in GDL-II: The special role `random` is used to model nature, who chooses her moves randomly. The keyword `sees` controls the information that players have about the game state. Hence, despite full initial information, both incomplete knowledge about later states and asymmetry of information result from the individual and partial percepts.

Example 2. *(Kriegspiel)* In standard chess, players see their opponent's move. For GDL-II, this means that the usual rules of chess[5] are augmented by the two clauses

```
sees(white,M) ⇐ does(black,M)
sees(black,M) ⇐ does(white,M)
```

Without these clauses, moves are hidden from the opponent. This chess variant is commonly called Kriegspiel (Pritchard

[3]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).

[4]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.

[5]See *games.stanford.edu* for a GDL axiomatisation of chess.

```
role(jane).    role(rick).    role(random).
card(7).   ...  card(ace).
succ(7,8). ...  succ(king,ace).
init(dealingRound).
---------------------------------------------------------------------------------
legal(random,deal(C,D)) ⇐ true(dealingRound) ∧ card(C) ∧ card(D) ∧ distinct(C,D)
legal(random,noop)      ⇐ true(bettingRound)
legal(R,noop)  ⇐ true(dealingRound) ∧ role(R) ∧ distinct(R,random)
legal(R,allIn) ⇐ true(bettingRound) ∧ role(R) ∧ distinct(R,random)
legal(R,fold)  ⇐ true(bettingRound) ∧ role(R) ∧ distinct(R,random)

sees(jane,yourCard(C))  ⇐ does(random,deal(C,D))
sees(rick,yourCard(D))  ⇐ does(random,deal(C,D))
sees(jane,ricksBid(B))  ⇐ does(rick,B) ∧ true(bettingRound)
sees(rick,janesBid(B))  ⇐ does(jane,B) ∧ true(bettingRound)
sees(jane,ricksCard(C)) ⇐ does(jane,allIn) ∧ does(rick,allIn) ∧ true(hasCard(rick,C))
sees(rick,janesCard(C)) ⇐ does(jane,allIn) ∧ does(rick,allIn) ∧ true(hasCard(jane,C))

next(hasCard(jane,C)) ⇐ does(random,deal(C,D))
next(hasCard(rick,D)) ⇐ does(random,deal(C,D))
next(bet(R,C,allIn))  ⇐ does(R,allIn) ∧ true(hasCard(R,C))
next(bet(R,C,fold))   ⇐ does(R,fold) ∧ true(hasCard(R,C))
next(bettingRound)    ⇐ true(dealingRound)
---------------------------------------------------------------------------------
terminal    ⇐ ¬true(dealingRound) ∧ ¬true(bettingRound)
goal(R,100) ⇐ true(bet(R,C,allIn)) ∧ true(bet(S,D,allIn)) ∧ beats(C,D)
goal(R, 75) ⇐ true(bet(R,C,allIn)) ∧ true(bet(S,D,fold))
goal(R, 50) ⇐ true(bet(R,C,fold)) ∧ true(bet(S,D,fold)) ∧ distinct(R,S)
goal(R, 25) ⇐ true(bet(R,C,fold)) ∧ true(bet(S,D,allIn))
goal(R,  0) ⇐ true(bet(R,C,allIn)) ∧ true(bet(S,C,allIn)) ∧ beats(D,C)
beats(C,D)  ⇐ succ(D,C)
beats(C,D)  ⇐ succ(X,C) ∧ beats(X,D)
```

Figure 1: A complete, formal description in GDL-II of a simple card game.

1994). Actually, in order to play this game effectively, an arbiter is needed who collects all moves and informs the players whenever they intend to make an invalid move. This may be specified in GDL-II as follows:

```
sees(R,badMoveTryAgain) ⇐ does(R,M) ∧
                          ¬validMove(M)
sees(black,yourMoveNow) ⇐ does(white,M) ∧
                          validMove(M)
sees(white,yourMoveNow) ⇐ does(black,M) ∧
                          validMove(M)
```

where `validMove(M)` should be axiomatised as test whether `M` is a correct chess move in the current position.[6]

Example 3. *(Coloured Trails)* This class of games is a popular research test-bed for decision-making and negotiation in a competitive setting (Grosz et al. 2004). Each specific game comes with one or more fixed protocols defining possible interactions among the players. For example, a simple negotiation may consist of player `R` offering player `S` to trade one of its chips `C` for another one, `D`. This may be formalised by these GDL-II clauses:

```
legal(R,propose(S,exchangeChips(C,D))) ⇐
   true(hasChip(R,C)) ∧ true(hasChip(S,D))∧
   distinct(R,S) ∧ distinct(C,D)
sees(S,proposal(R,S,C,D)) ⇐
   does(R,propose(S,exchangeChips(C,D)))
```

Under these rules, the communication is private: only the addressee gets to see the proposal.

Formal Syntax of GDL-II

Game descriptions in GDL-II use the standard syntax of logic programs, including negation. We adopt the Prolog convention of denoting variables by uppercase letters while predicate and function symbols start with a lowercase letter. From its predecessor, GDL-II inherits general restrictions on a set of clauses with the intention to ensure that all relevant derivations are finite. Specifically, a valid game description must be *stratified* (Apt, Blair, and Walker 1987) and *allowed* (Lloyd and Topor 1986); for details we must refer to (Love et al. 2006) for space reasons. Stratified logic programs are known to admit a specific *standard model*; see (Apt, Blair, and Walker 1987) for details. We also imposes restrictions on the use of the keywords in GDL-II:

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

[6]It is important to note the difference between *legal* and *valid* moves in Kriegspiel: each attempt to make a move is considered legal, but only those chess moves that are actually possible in the current position are accepted as valid. For practical play, a rule should be added that does not allow players to resubmit a previously rejected move.

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

These restrictions are imposed in order to ensure that a set of GDL-II rules can be effectively and unambiguously interpreted by a state transition system as a formal game model. This is the topic of the next section.

Semantics: A Game Model for GDL-II

State transition systems are the natural model for (discrete) games. Any game description in GDL-II uses a domain-dependent set of ground symbolic expressions Σ. Specifically, the players, the moves, and the percepts are all represented by individual ground terms like `jane`, `allIn`, `yourCard(queen)`, etc. Game positions (i.e., states) are represented by subsets of Σ since they are composed of individual features like `dealingRound`, `hasCard(jane,ace)`, etc. Accordingly, we define a *game* to be composed of[7]

- $R \subseteq \Sigma$ (the *roles*);
- $s_1 \subseteq \Sigma$ (the *initial position*);
- $t \subseteq 2^\Sigma$ (the *terminal positions*);
- $l \subseteq R \times \Sigma \times 2^\Sigma$ (the *legality relation*);
- $u : (R \mapsto \Sigma) \times 2^\Sigma \mapsto 2^\Sigma$ (the *update function*);
- $\mathcal{I} \subseteq R \times (R \mapsto \Sigma) \times 2^\Sigma \times \Sigma$ (the *information relation*);
- $g \subseteq R \times \mathbb{N} \times 2^\Sigma$ (the *goal relation*);
- $\pi : (R \setminus \{\texttt{random}\} \mapsto \Sigma) \times 2^\Sigma \mapsto \mathcal{P}(2^\Sigma)$.

Some explanatory words: Legality relation $l(r, m, S)$ defines m to be a legal move for player r in position S. The update function takes a move for each player and (synchronously) applies the joint action $M : (R \mapsto \Sigma)$ to a current position S, resulting in the updated position $u(M, S)$. Likewise, relation $\mathcal{I}(r, M, S, p)$ defines p to be a percept for player r when joint move M is taken in position S. Goal relation $g(r, n, S)$ defines natural number n to be the goal value for player r in position S. Finally, $\pi(M, S)$ is a probability measure over the resulting states after joint move M (by all but the randomised player) is taken in S.

Based on the concept of a formal game, a GDL-II specification is to be understood as follows. Any valid game description G contains a finite set of function symbols, including constants, which implicitly determines a (usually infinite) set of ground terms. This set constitutes the symbol base Σ in the game model for G. The syntactic restrictions in GDL-II ensure finite derivability, so that the set of roles, the reachable states, etc. are all finite subsets of Σ.

The derivable instances of `role(R)` define the players. The initial state is composed of the derivable instances of `init(F)`. In order to determine the legal moves of a player in any given state, this state has to be encoded first, using the keyword `true`. More precisely, let $S = \{f_1, \ldots, f_n\}$ be a position (e.g., the derivable instances of `init(F)` at the beginning), then G is extended by the facts

$$\texttt{true}(f_1). \quad \ldots \quad \texttt{true}(f_n).$$

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

Determining a position update and the percepts of the players requires the encoding of the current position along with clauses representing a joint move. Specifically, if players $r_1, \ldots, r_k$ make moves $m_1, \ldots, m_k$, then $G \cup S^{\texttt{true}}$ is further extended by the facts

$$\texttt{does}(r_1, m_1). \quad \ldots \quad \texttt{does}(r_k, m_k).$$

Let these clauses be denoted by $M^{\texttt{does}}$. The instances of `next(F)` that are derivable from $G \cup M^{\texttt{does}} \cup S^{\texttt{true}}$ compose the updated position; likewise, the derivable instances of `sees(R,P)` describe what a player perceives when the given joint move is done in the given position.

Finally, the probability measure is built into the concept of the special `random` role. If this role does not occur in a game description, then the update function determines a unique resulting state of a joint move by all players. Otherwise, the probability is uniformly distributed over all legal moves of `random` in the given position.[8] All of the above is summarised in the following definition, where entailment ($\models$) is via the standard model of a stratified set of clauses.

Definition 1. *Let G be a valid GDL specification, whose signature determines the set of ground terms Σ. The* semantics *of G is the game $(R, s_1, t, l, u, \mathcal{I}, g, \pi)$ given by*

- $R = \{r \in \Sigma : G \models \texttt{role}(r)\}$*;*
- $s_1 = \{f \in \Sigma : G \models \texttt{init}(f)\}$*;*
- $t = \{S \in 2^\Sigma : G \cup S^{\texttt{true}} \models \texttt{terminal}\}$*;*
- $l = \{(r, m, S) : G \cup S^{\texttt{true}} \models \texttt{legal}(r, m)\}$*, for all $r \in R$, $m \in \Sigma$, and $S \in 2^\Sigma$;*
- $u(M, S) = \{f \in \Sigma : G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \models \texttt{next}(f)\}$*, for all $M : (R \mapsto \Sigma)$ and $S \in 2^\Sigma$;*
- $\mathcal{I} = \{(r, M, S, p) : G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \models \texttt{sees}(r, p)\}$*, for all $r \in R \setminus \{\texttt{random}\}$, $M : (R \mapsto \Sigma)$, $S \in 2^\Sigma$, and $p \in \Sigma$;*
- $g = \{(r, n, S) : G \cup S^{\texttt{true}} \models \texttt{goal}(r, n)\}$*, for all $r \in R \setminus \{\texttt{random}\}$, $n \in \mathbb{N}$, and $S \in 2^\Sigma$;*
- *for all $M : (R \setminus \{\texttt{random}\} \mapsto \Sigma)$ and $S, T \in 2^\Sigma$,*
  - *if* $\texttt{random} \notin R$ *then*

$$\pi(M, S)(T) = \begin{cases} 1 & \textit{if } T = u(M, S) \\ 0 & \textit{otherwise} \end{cases}$$

[7]Below, 2^Σ denotes the *finite* subsets of Σ.

[8]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 actions for `random`, two of which have the same effect (the coin showing tails). It should be stressed that basic GDL-II could easily be extended by means for specifying a non-uniform transition probability.

– *else* $\pi(M,S)$ *assigns this probability to state* T:

$$\frac{|\{m \in L : T = u(M \cup \{\texttt{random} \mapsto m\}, S)\}|}{|L|}$$

where $L = \{m : G \cup S^{\texttt{true}} \models l(\texttt{random}, m, S)\}$.

This definition provides a formal semantics for GDL-II in terms of an abstract game model. Finite derivability in valid GDL-II specifications implies that the entailment relation is decidable, which in turn ensures that the definition of the semantics is effective.

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 S each player r chooses a move m that satisfies $l(r,m,S)$. As a consequence the game state changes to $u(M,S)$, where M is the joint move. In contrast to the execution model for GDL (Genesereth, Love, and Pell 2005; Love et al. 2006), 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 $\mathcal{I}(r,M,S,p)$. The game ends as soon as a terminal state $S \in t$ is reached, and then the goal relation $g(r,n,S)$ determines the result for each player (except `random`).

This execution model is simple enough to allow a straightforward implementation of a Game Controller:

1. Send each $r \in R \setminus \{\texttt{random}\}$ the GDL-II description and inform them about their individual roles r (e.g., `jane` or `rick`). Set $S := s_1$.
2. After the appropriate time, collect the individual moves from each player and, in case $\texttt{random} \in R$, choose randomly (with uniform probability) an element from the set $\{m : (\texttt{random}, m, S) \in l\}$. Set $M :=$ 'joint move'.
3. Send to each $r \in R \setminus \{\texttt{random}\}$ the set of percepts $\{p : (r,M,S,p) \in \mathcal{I}\}$. Set $S := u(M,S)$.
4. Repeat 2. and 3. until $S \in t$. Determine the result n for $r \in R \setminus \{\texttt{random}\}$ by $(r,n,S) \in g$.

In this setup the control program knows all moves and hence can always compute all percepts and also determine the end of a match and the resulting goal values for the players.

The Implicit Epistemic Model of GDL-II

The transition-based execution model we have just given for GDL-II is simple and can easily be implemented on a Game Master to control a game. On the other hand, despite its conceptual simplicity our language extension determines an intricate epistemic model. This model can be used by players to reason about their own knowledge, both up front and during game play, as well as about the other player's knowledge *and* about what a player knows about the knowledge of a third player, etc. For example, as soon as player `jane` gets to see the GDL-II rules of Figure 1, she can derive that she does not know up front which card she will be dealt, but that she will know her card after `random`'s first move. She can also conclude that `rick` won't know her card at this point.

In the following, we will formally show what the execution model entails about the knowledge of the individual players before and during game play. To this end, we first define the notion of a *development* as a legal sequence of moves starting in initial position s_1:

$$(s_1, m_1), \ldots, (s_{n-1}, m_{n-1}), s_n$$

where $n \geq 1$ and for all $i \in \{1, \ldots, n-1\}$ we have that $(r, m_i(r), s_i) \in l$ for all $r \in R$ (that is, players make legal moves) and $s_{i+1} = u(m_i, s_i)$ (position update). Furthermore, $\{s_1, \ldots, s_{n-1}\} \cap t = \emptyset$ (only the last state may be terminal). The following theorem characterises precisely what our execution model for GDL-II specified at the end of the previous section entails about the information a player r has at a specific stage in the course of a game.

Theorem 1. Let $\delta = \langle (s_1, m_1), \ldots, (s_{n-1}, m_{n-1}), s_n \rangle$ and $\delta' = \langle (s'_1, m'_1), \ldots, (s'_{n-1}, m'_{n-1}), s'_n \rangle$ be two developments. A player $r \in R \setminus \{\texttt{random}\}$ cannot distinguish δ from δ' at step k if, and only if,

1. $\{p : (r, m_i, s_i, p) \in \mathcal{I}\} = \{p' : (r, m'_i, s'_i, p') \in \mathcal{I}\}$ for all $i \in \{1, \ldots, k-1\}$, and
2. $m_j(r) = m'_j(r)$ for all $j \in \{1, \ldots, n-1\}$.

Proof. (Sketch) According to the execution model, players know the game model as given in Definition 1. In each step they choose their own moves and receive information as given in $\mathcal{I}$. Hence, the two developments are indistinguishable if until step k they entail the same percepts for r (first item) and r always takes the same move (second item). □

Consider, e.g., a development with `random`'s first move being `deal(king,7)` and one with `deal(ace,7)` instead. For `jane` these two are indistinguishable at step 1 but no longer so at step 2. For `rick`, however, they are indistinguishable at step 2, too, as for him they both entail the same percept, `yourCard(7)`.

(In-)distinguishable developments can also be used to ensure that game descriptions obey desirable properties.

Proposition 2. A game description entails that all players always know their legal moves iff for all $r \in R \setminus \{\texttt{random}\}$ there are no two developments δ, δ' leading to states s_n, s'_n such that $\{m : (r, m, s_n) \in l\} \neq \{m' : (r, m', s'_n) \in l\}$ and δ, δ' are indistinguishable for r at n.

Proposition 3. A game description entails that all players know both the end of a game and their own result iff for all $r \in R \setminus \{\texttt{random}\}$ there are no two developments δ, δ' leading to states s_n, s'_n such that

$$s_n \in t,\ s'_n \notin t \quad \text{or} \quad \{s_n, s'_n\} \subseteq t,\ g(r, \_, s_n) \neq g(r, \_, s'_n)$$

and δ, δ' are indistinguishable for r at n.

It is also easy to prove that GDL-II truly extends GDL since games with complete state information can be specified thus:

Theorem 4. Consider a game with $\texttt{random} \notin R$ and

$$\texttt{sees(R1,moves(R,M))} \Leftarrow \texttt{role(R1)} \land \texttt{does(R,M)}$$

the only clause for predicate `sees`. Then there are no two developments δ, δ' leading to states s_n, s'_n such that δ and δ' are indistinguishable for any $r \in R$ at n.

Knowledge at the object level, as we have considered thus far, can be lifted to higher levels so as to determine what a player can know about the *knowledge* of other players (including his or her own). This is made possible by the fact that the GDL-II specification of a game provides all players with the complete rules—so that a player is able to derive which information each other player gets to see under any (hypothetical) development.[9] Formalising this requires to construct a suitable epistemic structure based on possible developments. Consider, to this end, for every δ the function $\mathcal{K}^{\delta}$ that maps every pair (r, i) onto the set of developments that player r at state i cannot distinguish from δ. Meta-level knowledge is then obtained as follows.

Theorem 5. If the game develops according to δ_1, then as far as player r knows at step n, all functions $\mathcal{K}^{\delta_2}$ are possible where δ_2 is indistinguishable from δ_1 for r at n.

Put in words, meta-level knowledge is characterised by a set of possible sets of developments. It follows that a player knows what holds in all $\mathcal{K}$-sets she considers possible. Lack of space prevents us from going into the formal details, but the process can be iterated inductively to determine arbitrary levels of meta-knowledge. The epistemic model for GDL-II thus obtained allows to apply standard formalisms like (Fagin et al. 1995) to be used by players to infer the truth of arbitrary knowledge expressions involving their own as well as the other player's knowledge.

Conclusion

We have presented a conceptually simple yet fundamental extension of the Game Description Language that allows to represent and reason about the rules of general games with information asymmetry and random moves. We have also shown that GDL-II, despite its syntactic elegance and simplicity, 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.

The new language GDL-II is the first extension of the existing GDL that allows to describe games with incomplete state information and elements of chance. The concept of General Game Playing goes back to Pitrat (1968) and Pell (1993). Both define formal languages to describe whole classes of games, but these languages are even less general than basic GDL. The work (Koller and Pfeffer 1997) includes the definition of Gala (for: Game Language) which allows to describe general games with imperfect information. The main differences to GDL-II are: (1) Gala requires an explicit enumeration of the sequence of steps from the beginning to the end of a game; (2) a Gala description is understood by an execution tree, which corresponds to a game tree in so-called extensive form—hence, the semantics of Gala is procedural rather than declarative; (3) understanding a Gala "program" requires to build the entire execution tree for this program, which is unsuitable as a basis for GGP research where the main focus is on learning how to play well games that cannot be searched completely. Finally, it should be mentioned that GDL-II draws from concepts that have been used in Action Languages, e.g. (Lobo, Mendez, and 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.

[9]We remark that common knowledge of the rules does not imply common knowledge of other players. To see this, the interested reader may try a small exercise: construct a 3-player GDL-II game in which a situation may arise where (1) player P_1 knows f and (2) player P_2 knows that P_1 knows about f while player P_3 considers it possible that P_1 knows nothing about f. (We thank an anonymous reviewer for suggesting this scenario.)

References

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

Clune, J. 2007. Heuristic evaluation functions for general game playing. In *AAAI*, 1134–1139.

Fagin, R.; Halpern, J.; Moses, Y.; and Vardi, M. 1995. *Reasoning About Knowledge*. MIT Press.

Finnsson, H., and Björnsson, Y. 2008. Simulation-based approach to general game playing. In *AAAI*, 259–264.

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

Grosz, B.; Kraus, S.; Talman, S.; Stossel, B.; and Havlin, M. 2004. The influence of social dependencies on decision-making: Initial investigations with a new game. In *Proc. of AAMAS*, 782–789.

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

Kuhlmann, G.; Dresner, K.; and Stone, P. 2006. Automatic heuristic construction in a complete general game player. In *AAAI*, 1457–1462.

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

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

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

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

Pitrat, J. 1968. Realization of a general game playing program. In *Proc. of IFIP Congress*, 1570–1574.

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

Schiffel, S., and Thielscher, M. 2007. Fluxplayer: A successful general game player. In *AAAI*, 1191–1196.

Schiffel, S., and Thielscher, M. 2009a. Automated theorem proving for general game playing. In *IJCAI*, 911–916.

Schiffel, S., and Thielscher, M. 2009b. A multiagent semantics for the Game Description Language. In *Agents and Artificial Intelligence: Proc. of ICAART*. Springer.