The General Game Playing Description Language Is Universal

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

Abstract

The *Game Description Language* is a high-level, rule-based formalisms for communicating the rules of arbitrary games to *general* game-playing systems, whose challenging task is to learn to play previously unknown games without human intervention. Originally designed for deterministic games with complete information about the game state, the language was recently extended to include randomness and imperfect information. However, determining the extent to which this enhancement allows to describe truly arbitrary games was left as an open problem. We provide a positive answer to this question by relating the extended Game Description Language to the universal, mathematical concept of extensive-form games, proving that indeed just any such game can be described faithfully.

1 Introduction

A contemporary Grand AI Challenge, General Game Playing (GGP) aims at building systems that learn to play previously unknown games without human intervention and just by being told the rules of a game [Genesereth *et al.*, 2005]. Broad interest in this challenge was sparked by the inauguration of an annual AAAI GGP Competition in 2005, and in just a few years since then GGP has become an established area in AI: An increasing number of research groups world-wide develop game-playing technology to build actual systems ([Clune, 2007; Schiffel and Thielscher, 2007; Björnsson and Finnsson, 2009], just to mention a few); a series of biennial IJCAI workshops devoted to this topic started in 2009; and AAAI'10 featured the first ever technical session on GGP, contributing to a rapidly growing body of literature.

A main reason for this success of the competition (besides the attractive $10,000 purse!) was the institution of the general *Game Description Language (GDL)* as the foundation for GGP [Genesereth *et al.*, 2005]. A machine-processable language for specifying a wide range of games existed before—GALA (for: Game Language) [Koller and Pfeffer, 1997], which however was never used outside of the context of the GALA system (that for a given game specification created the game tree and then computed optimal strategies). 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 sufficiently system-independent domain specification language.

With GDL the organisers of the AAAI GGP Competition sought a high-level game specification language that admits a purely declarative reading and thus allows general game-playing systems to reason about the rules of a game. In this regard, GDL follows the tradition of AI Planning. Planning languages however describe a problem from the perspective of a single agent, even in case of adversarial planning (see, e.g., [Jensen and Veloso, 2000]). GDL generalises this to the presence of other agents that have their own actions and goals. Reasoning about the intentions of the other players is the basis for Opponent Modelling—one of the crucial aspects in which GGP goes beyond AI Planning [Genesereth *et al.*, 2005]. On the other hand, GDL shares with existing planning languages the compactness of specifications so that, unlike with, say, graph-based, propositionalised encodings (see, e.g., [Mura, 2000]), games of practical interest such as Chess can be fully specified with just a few kilobytes of code.

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. In [Thielscher, 2010] we have defined a notationally simple extension to include both randomness and imperfect information. A number of examples have showed how this enhanced GDL covers games with new features like partial observability, information asymmetry, and communicative actions. However, determining the extent to which this enhancement allows to describe truly arbitrary (discretised, finite) games was left open.

In this paper, we provide a positive answer to this question by relating the extended Game Description Language to the universal, mathematical concept of extensive-form games. As the main result we show the following.

1. The players' information sets determined by a GDL description satisfy all standard requirements of information partitions in extensive-form games.

2. Any extensive-form game can be described faithfully (that is, move-by-move) in GDL.

* The author is the recipient of an Australian Research Council Future Fellowship (project FT 0991348).

First and foremost, these results show that GDL can be considered complete for the purpose of General Game Playing. But we also expect our result to be of interest beyond GGP, because game models arise in many other AI disciplines, such as Multiagent Planning [Larbi *et al.*, 2007] or Multiagent Systems in general. A universal, compact and purely declarative description language for games of any kind is potentially useful wherever there is a need for concise, machine-processable specifications of domains and applications as games.

The rest of the paper is organised as follows. In the next section, we give a brief overview of the extended Game Description Language, followed by recapitulating the formal definition of extensive-form games from a standard textbook [Rasmusen, 2007]. In the section that follows, we formally map GDL game descriptions into games in extensive form. Thereafter we prove that, conversely, every extensive-form game can be described in the language.

2 Preliminaries

2.1 Game Descriptions

The purpose of the Game Description Language is to allow for concise, high-level specifications of the rules of arbitrary finite games. To this end, GDL uses a logic programming-like syntax and is characterised by the following special *keywords*.

`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`
`does(?r,?m)`	player `?r` does move `?m`
`next(?f)`	`?f` holds in the next position
`terminal`	the position is terminal
`goal(?r,?v)`	`?r` gets payoff `?v`
`sees(?r,?p)`	`?r` perceives `?p` in the next position
`random`	the random player (aka. Nature)

As an example, the rules in Fig. 1 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. Lines 1–7 introduce the players' names and the state features that hold initially. The possible moves are specified by the rules with head `legal` (lines 9–21): first, the *random* player must decide where to place the car and, simultaneously, the candidate chooses a door; next, *random* opens a door that is not the one that holds the car nor the chosen one; finally, the candidate can either stick to his earlier choice (*noop*) or switch to the other yet unopened door.

The candidate's only percept throughout the game, viz. the door that gets opened, is defined by the rule with head `sees` (line 22). The remaining clauses specify the position update (rules for `next`), the conditions for the game to end (rule for `terminal`), and the payoff for the player depending on whether he got the door right in the end (rules for `goal`).

Formal Syntax and Semantics

In order to admit an unambiguous interpretation, a set of GDL rules must obey certain general syntactic restrictions. Specifically, to be *valid* a game description must be *stratified* [Apt *et al.*, 1987] and *allowed* [Lloyd and Topor, 1986]. Stratified logic programs are known to admit a unique *standard model*; see [Apt *et al.*, 1987] for details. A further syntactic restriction ensures that only finitely many positive instances are true in this model; for details we must refer to [Love *et al.*, 2006]. Finally, the special keywords are to be used as follows:

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

These restrictions are imposed to ensure that a set of GDL rules can be effectively and unambiguously interpreted by a state transition system as follows. To begin with, any set of clauses G determines an implicit domain-dependent set of ground symbolic expressions Σ, like *candidate*, *hide_car*(1), *chosen*(3), etc. Game positions (i.e., states) are represented by subsets of Σ since they are composed of individual features, as in

$$\{\textit{closed}(1), \textit{closed}(2), \textit{closed}(3), \textit{step}(1)\} \qquad (1)$$

Although Σ itself is usually infinite, the syntactic restrictions in GDL guarantee that the set of roles, each reachable state, and the set of legal moves are always finite subsets of Σ [Love *et al.*, 2006].

Specifically, then, the derivable instances of `role(?r)` define the players. The initial position, (1) in our example game, 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 $S = \{f_1, \ldots, f_k\}$ be a finite state like (1), then G is extended by the k facts

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

Those instances of `legal(?r,?m)` that are derivable from $G \cup S^{\texttt{true}}$, e.g. `legal`(*random*, *hide_car*(3)) given (1), define all legal moves `?m` for role `?r` in state S. In the same way, the clauses for `terminal` and `goal` define termination and goal values *relative* to the encoding of a position.

Determining a position update and the percepts of the players requires the encoding of the current position *and* of a joint move (i.e., one move from each player). Specifically, if players $r_1, \ldots, r_n$ take moves $a_1, \ldots, a_n$, then let

$$M^{\texttt{does}} \stackrel{\text{def}}{=} \{\texttt{does}(r_1, a_1)., \ \ldots, \ \texttt{does}(r_n, a_n).\}$$

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 all percepts `?p` for any particular role `?r` (except *random*) when the given joint move is done in the given position. All this is summarised in the following definition, where "$\models$" means entailment wrt. the aforementioned standard model.

Definition 1. *[Thielscher, 2010] The* semantics *of a valid GDL specification is given by a state transition system composed as follows.*

```
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)).
```

Figure 1: A GDL description of the Monty Hall game [Rosenhouse, 2009]. The host is modelled by the standard role *random*.

- $R = \{r : G \models \texttt{role}(r)\}$ *(player names);*
- $s_0 = \{f : G \models \texttt{init}(f)\}$ *(initial state);*
- $t = \{S : G \cup S^{\texttt{true}} \models \texttt{terminal}\}$ *(terminal states);*
- $l = \{(r, m, S) : G \cup S^{\texttt{true}} \models \texttt{legal}(r, m)\}$ *(legal moves);*
- $u(M, S) = \{f : G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \models \texttt{next}(f)\}$*, for all joint moves* M *and states* S *(position update);*
- $\mathcal{I} = \{(r, M, S, p) : G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \models \texttt{sees}(r, p)\}$*, for all roles* $r \in R \setminus \{$*random*$\}$ *(players' percepts);*
- $g = \{(r, v, S) : G \cup S^{\texttt{true}} \models \texttt{goal}(r, v)\}$ *(goal values).*

This definition provides the basis for the execution model for GDL: Starting with initial state $S \leftarrow s_0$, in each S each player $r \in R$ selects one of his or her legal moves m, that is, which satisfies $l(r, m, S)$. Role *random* chooses a legal move with uniform probability. The update function (synchronously) applies the joint move M to the current position, resulting in the new position $S \leftarrow u(M, S)$. Furthermore, each role $r \in R \setminus \{$*random*$\}$ gets to see any p that satisfies the information relation, $\mathcal{I}(r, M, S, p)$. The game ends when a terminal state $S \in t$ is reached, and then the goal relation $g(r, v, S)$ determines the result for player r.

2.2 Extensive-Form Games

The next definition introduces the mathematical concept of a finite game in extensive form following [Rasmusen, 2007].

Definition 2. *An* n-player extensive-form game *consists in*

- *a finite tree with*
  - *nodes* S *(called* states*),*
  - *initial state* $s_0 \in S$*,*
  - *terminal states* $T \subseteq S$*,*
  - *predecessor function* $f : (S \setminus \{s_0\}) \mapsto S$*;*
- *a function* $\iota : (S \setminus T) \mapsto \{0, \ldots, n\}$*, indicating which state belongs to which player (player no.* 0 *is Nature);*
- *a function* $v : T \mapsto \mathbb{R}^n$ *(called* utility*), defining a payoff value for each player and each terminal state;*
- *a probability measure* ρ_s *over* $f^{-1}(s)$ *for all states with* $\iota(s) = 0$ *(i.e., which belong to Nature);*
- *for all players* $r = 1, \ldots, n$ *a set* $\mathcal{H}_r$ *of* information partitions *such that for each* $H \in \mathcal{H}_r$*:*
  1. *the elements (called* information sets*) in* H *are mutually disjoint;*
  2. *all children of a node* s *such that* $\iota(s) = r$ *are in different information sets;*
  3. *for all branches* β *there is exactly one* $s \in \beta$ *such that* s *occurs in some information set in* H*; and*
  4. *for all* $h \in H$ *the predecessors of all nodes in* h *are in one* $h' \in H$*.*

The rationale behind the conditions on the information partition is as follows [Rasmusen, 2007].

1. If a node s belonged to two information sets h and h', then when the game were in s the player would not know whether a node in h or h' had been reached, hence h and h' were really the same information set.
2. Players know their own moves.
3. For every stage of the game there is an information partition which represents the positions a player is able to distinguish from each other *at this stage*.
4. If a player knows whether the predecessor of s or the predecessor of s' has been reached, then this information suffices to distinguish s from s'.

To see an example at this point, the reader may peek at Fig. 2.

3 From GDL to Extensive-Form Games

The mathematical definition of extensive-form games uses the very general concept of information partitions to model partial observability and information asymmetry. The key to our perhaps surprising result in this paper lies in the fact that the `sees(?r,?p)` predicate in GDL can be used equally generally. The reason is that percepts `?p` are not confined to specific observables (e.g., opponents' moves, state features) but can also be used as abstract identifiers to encode just any

desired information partition. In this section, we will first show that every GDL game, by virtue of the underlying execution model and provided it always terminates, can be understood as a game in extensive form that satisfies all requirements in Definition 2. Two main issues need to be addressed:

1. The perceptive capabilities of the players must be mapped onto appropriate information partitions.
2. Joint moves in GDL need to be serialised in a way that no player is aware of the simultaneous moves by the other players.

While all this can be done for arbitrary n-player games, in the following we will confine ourselves to the notationally simpler case of just two players, one of which is *random*. This covers all relevant aspects of the general case but will considerably ease the notation for the benefit of the reader; the generalisation of the mapping is tedious but straightforward.

We first recall a basic property of GDL games: how they are run and how a player can or cannot distinguish different runs on the basis of his or her percepts. Runs can be described by *developments*, which are sequences of states and moves by each player. The following is adapted from [Thielscher, 2010] to the special case of GDL games $(R, s_0, t, l, u, \mathcal{I}, g)$ (cf. Definition 1) with the two fixed roles $R = \{\mathit{random}, \mathit{player}\}$.

Proposition 1. A *development* δ is a sequence

$$\langle s_0, a_1^{\mathit{random}}, a_1^{\mathit{player}}, s_1, \ldots, s_{d-1}, a_d^{\mathit{random}}, a_d^{\mathit{player}}, s_d \rangle \quad (2)$$

such that $d \geq 0$ and for all $i \in \{1, \ldots, d\}$,

- the selected moves are legal: $(\mathit{random}, a_i^{\mathit{random}}, s_{i-1}) \in l$ and $(\mathit{player}, a_i^{\mathit{player}}, s_{i-1}) \in l$;
- states are updated: $s_i = u((a_i^{\mathit{random}}, a_i^{\mathit{player}}), s_{i-1})$;
- only s_d may be terminal: $\{s_1, \ldots, s_{d-1}\} \cap t = \emptyset$.

Consider developments $\delta = \langle s_0, a_1^{\mathit{random}}, a_1^{\mathit{player}}, s_1, \ldots \rangle$ and $\delta' = \langle s_0, b_1^{\mathit{random}}, b_1^{\mathit{player}}, s_1' \ldots \rangle$. The player *cannot distinguish* δ *from* δ' iff δ, δ' are of the same length d and for all $i \in \{1, \ldots, d-1\}$,

1. the player's percepts are always the same, that is, the two sets $\{p : (\mathit{player}, (a_i^{\mathit{random}}, a_i^{\mathit{player}}), s_i, p) \in \mathcal{I}\}$ and $\{p' : (\mathit{player}, (b_i^{\mathit{random}}, b_i^{\mathit{player}}), s_i', p') \in \mathcal{I}\}$ are equal; and
2. the player takes the same moves, i.e., $a_i^{\mathit{player}} = b_i^{\mathit{player}}$. ■

This characterisation of different executions and how a player can distinguish them provides the basis for the mapping of any GDL game model into an extensive-form game according to Definition 2. Some additional notations are helpful to this end: For a development δ of the form (2) we write $\mathit{length}(\delta) = d$ and $\mathit{end}(\delta) = s_d$. Also,

- $\mathit{prefix}^{\mathit{player}}(\delta) = \langle s_0, a_1^{\mathit{random}}, a_1^{\mathit{player}}, \ldots, s_{d-1}, a_d^{\mathit{random}} \rangle$ (here, a move by *player* would continue the sequence);
- $\mathit{prefix}^{\mathit{random}}(\mathit{prefix}^{\mathit{player}}(\delta)) = \langle s_0, a_1^{\mathit{random}}, a_1^{\mathit{player}}, \ldots, s_{d-1} \rangle$ (here, a move by *random* would follow).

In the following, let Δ be the set of all developments and Δ^{player} all *incomplete* developments of the form $\mathit{prefix}^{\mathit{player}}(\delta)$, that is, where a move by *player* is to follow.

We write $\sim \{\delta : \delta \in \Delta\}$ to denote the partitioning of elements in Δ such that two developments δ, δ' are in the same partition if, and only if, the player cannot distinguish the two. Similarly, we write $\sim \{\delta : \delta \in \Delta^{\mathit{player}}\}$ to denote the partitioning of elements in Δ^{player} such that two developments δ, δ' are in the same partition if, and only if, the player cannot distinguish $\mathit{prefix}^{\mathit{random}}(\delta)$ from $\mathit{prefix}^{\mathit{random}}(\delta')$.

We can now show how to map any terminating GDL game $(\{\mathit{random}, \mathit{player}\}, s_0, t, l, u, \mathcal{I}, g)$ into extensive form:

The nodes are $\Delta \cup \Delta^{\mathit{player}}$.

The root is $\langle s_0 \rangle$.

The leaf nodes are $T = \{\delta \in \Delta : \mathit{end}(\delta) \in t\}$.

The predecessor function is defined as

$$f(\delta) = \begin{cases} \mathit{prefix}^{\mathit{player}}(\delta) & \text{if } \delta \in \Delta \setminus \{\langle s_0 \rangle\} \\ \mathit{prefix}^{\mathit{random}}(\delta) & \text{if } \delta \in \Delta^{\mathit{player}} \end{cases}$$

The nodes are assigned to players by

$$\iota(\delta) = \begin{cases} 0 & \text{if } \delta \in \Delta \setminus T \\ 1 & \text{if } \delta \in \Delta^{\mathit{player}} \end{cases}$$

The utility satisfies $g(\mathit{player}, v(\delta), \mathit{end}(\delta))$ for $\delta \in T$.

The probability measure is given as uniform distribution over $f^{-1}(\delta)$, for all random nodes $\delta \in \Delta$.

The information partitions in $\mathcal{H}_1$ are:

$$\begin{aligned} &\sim \{\delta : \delta \in \Delta, \mathit{length}(\delta) = 0\} \qquad \text{(i.e., } \{\{\langle s_0 \rangle\}\}\text{)} \\ &\sim \{\delta : \delta \in \Delta^{\mathit{player}}, \mathit{length}(\delta) = 1\} \\ &\sim \{\delta : \delta \in \Delta, \mathit{length}(\delta) = 1\} \\ &\sim \{\delta : \delta \in \Delta^{\mathit{player}}, \mathit{length}(\delta) = 2\} \\ &\ldots \\ &\sim \{\delta : \delta \in \Delta, \mathit{length}(\delta) = d\} \end{aligned}$$

where d is the maximal length of developments.

As an example, Fig. 2 depicts the extensive-form game thus obtained from the Monty Hall game description in Fig. 1. The following result shows that we have arrived at a faithful interpretation of GDL games as extensive-form games.

Theorem 2. The construction above satisfies the following.

1. $\mathcal{H}_1$ satisfies the conditions of Definition 2.
2. There is a one-to-one correspondence between the branches (of the extensive-form game) and the developments (of the GDL game), with identical payoffs.
3. The combined probabilities of a branch equals the combined probabilities in the development in the leaf.
4. *player* cannot distinguish two developments iff they are in the same information set of some $H \in \mathcal{H}_1$.

Proof (sketch): Proposition 1, along with construction of $\mathcal{H}_1$ on the basis of the equivalence relation induced by indistinguishable developments, ensure that the information sets are disjoint and that *player* knows his own moves. Also according to the construction of $\mathcal{H}_1$, each information partition $H \in \mathcal{H}_1$ contains developments of the same length k, and hence each branch (i.e., maximal development δ) is represented by exactly one element in each such H, namely, the

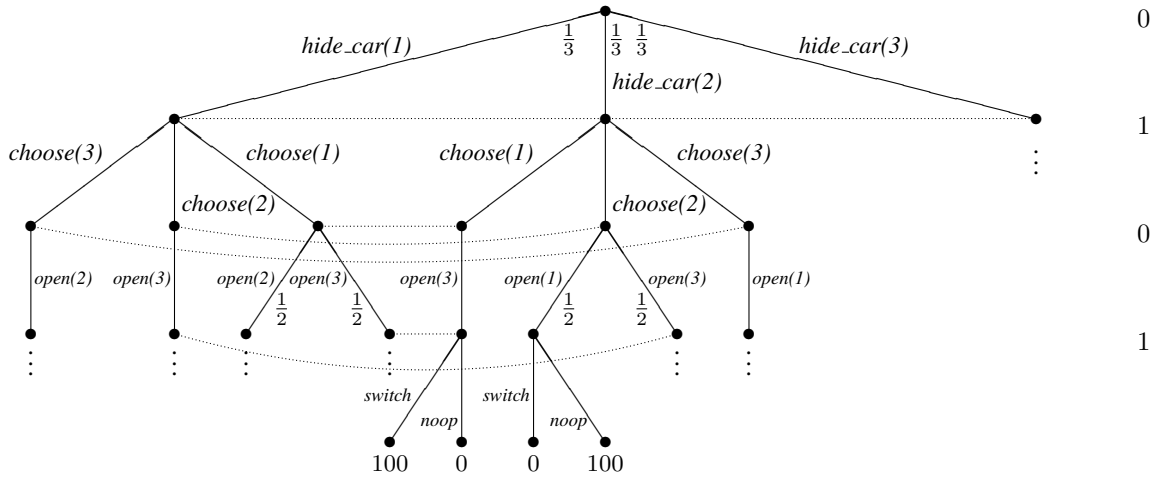

Figure 2: The game of Fig. 1 mapped onto extensive form (with some nodes and branches omitted). The numbers to the very right indicate to which player the nodes of that height belong ($0 =$ *random*, $1 =$ *candidate*). Dotted lines connect nodes in the same information *set* for the candidate. His information *partitions* are the collections of his information sets on each height.

prefix of δ of length k. Finally, if *player* can distinguish two developments then he can distinguish any of their continuations, too, according to Proposition 1. Altogether, this proves 1. Properties 2.–4. can be easily proved from the construction of the game tree, the payoff function, the probability measures, and *player*'s information partitions. □

4 From Extensive-Form Games to GDL

To show that any extensive-form game can be described faithfully (that is, move-by-move) in GDL we need to address two main issues. First, the given information partitions need to be encoded by appropriate `sees`-rules for the individual players. Second, non-uniform probabilities for moves by Nature need to be mapped onto uniform probability distributions for *random*'s moves. This will be achieved by introducing a proportional number of moves that have different names but lead to the same successor state.[1] As before, all this can be done for arbitrary n-player games in extensive form, but for the sake of clarity we again restrict our exposition to the special case of two players, 0 ($=$ *random*) and 1 ($=$ *player*). We also assume, without loss of generality since Nature always chooses among finitely many moves, that each individual probability measure ρ_s is given as a collection of rational numbers, which moreover have the same denominator. Then any extensive-form game $(S, s_0, T, f, \iota, \upsilon, \rho, \mathcal{H})$ (cf. Definition 2) can be described by the following set of GDL rules.

The players and the initial state are defined as

$$\texttt{role}(\texttt{random}). \quad \texttt{role}(\textit{player}). \quad \texttt{init}(s_0).$$

For each leaf node $s \in T$:

$$\begin{aligned}\texttt{terminal} \quad &\texttt{<= true}(s).\\ \texttt{goal}(\textit{player}, \upsilon(s)) &\texttt{<= true}(s).\end{aligned}$$

For non-leafs $s \in S \backslash T$, let their child nodes be given in an arbitrary but fixed order, $f^{-1}(s) = \{s_1, \ldots, s_m\}$ $(m \geq 1)$.

For non-random nodes, i.e. where $\iota(s) \neq 0$:

$$\begin{aligned}&\texttt{legal}(\textit{player}, 1) \quad \texttt{<= true}(s).\\ &\ldots\\ &\texttt{legal}(\textit{player}, m) \quad \texttt{<= true}(s).\\ &\texttt{legal}(\texttt{random}, \textit{noop}) \texttt{<= true}(s).\\ &\texttt{next}(s_1) \texttt{ <= true}(s), \texttt{ does}(\textit{player}, 1).\\ &\ldots\\ &\texttt{next}(s_m) \texttt{<= true}(s), \texttt{ does}(\textit{player}, m).\end{aligned}$$

For random nodes, i.e. where $\iota(s) = 0$, suppose ρ_s over $\{s_1, \ldots, s_m\}$ is given by the probabilities $\{\frac{p_1}{q}, \ldots, \frac{p_m}{q}\}$ such that $(\sum_{k=1}^{m} p_k) = q$. Then

$$\begin{aligned}&\texttt{legal}(\texttt{random}, 1) \texttt{ <= true}(s).\\ &\ldots\\ &\texttt{legal}(\texttt{random}, q) \texttt{ <= true}(s).\\ &\texttt{legal}(\textit{player}, \textit{noop}) \texttt{<= true}(s).\end{aligned}$$

and for each $i = 1, \ldots, m$:

$$\begin{aligned}&\texttt{next}(s_i) \texttt{<= true}(s), \texttt{ does}(\textit{random}, 1 + \textstyle\sum_{k=1}^{i-1} p_k).\\ &\ldots\\ &\texttt{next}(s_i) \texttt{<= true}(s), \texttt{ does}(\textit{random}, \textstyle\sum_{k=1}^{i} p_k).\end{aligned}$$

Hence, for each child s_i there are proportionally many (p_i, to be precise) moves that all lead to s_i. Let $S_1, \ldots, S_l$ be a partition (in arbitrary but fixed order) of $s_1, \ldots, s_m$ such that $s_i, s_{i'}$ are in the same set S_j iff they occur in the same information set of some partition $H \in \mathcal{H}_1$. (In other words, the player cannot distinguish any two states from the same S_j but any two states from different $S_j, S_{j'}$).

For all $j = 1, \ldots, l$ and all $s_i \in S_j$ such that s_i is a non-*player* move, i.e. where $\iota(s_i) \neq 1$:

$$\begin{aligned}\texttt{sees}(\textit{player}, j) \texttt{<= true}(s),&\\ \texttt{does}(\textit{random}, 1 + \textstyle\sum_{k=1}^{i-1} p_k).&\\ \ldots&\\ \texttt{sees}(\textit{player}, j) \texttt{<= true}(s), \texttt{ does}(\textit{random}, \textstyle\sum_{k=1}^{i} p_k).&\end{aligned}$$

Here, index j serves as abstract percept that allows *player* to identify the information set to which s_i belongs but nothing

[1] This follows our example in [Thielscher, 2010] of an unfair coin showing heads with probability $\frac{1}{3}$, modelled by three legal moves for *random*, two of which result in the coin showing tails.

more. Obviously, the size of the resulting game description is 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 description in Fig. 1, say, the construction does not exploit the conciseness of descriptions made possible by having a high-level knowledge representation language. However, this suffices for a proof of principle, and the following result shows that we can give a valid and faithful GDL description for any extensive-form game.

Theorem 3. The construction above satisfies the following.

1. The resulting set of rules meets all syntactic requirements of valid GDL descriptions.
2. There is a one-to-one correspondence between the branches (of the game tree) and the set of all maximal developments (in GDL) in which *random* chooses moves with the same effect, with identical payoffs.
3. The combined probabilities of a branch equals the sum of the combined probabilities of all corresponding maximal developments.
4. *player* cannot distinguish two developments iff their last states are in the same information set of some $H \in \mathcal{H}_1$.

Proof (sketch): It is easy to verify that the use of the keywords complies with all requirements stated in Section 2.1 and that the program is stratified and allowed. The semantics according to Definition 1, the execution model for GDL, and the construction of the rules for `init`, `legal`, `next`, and `terminal` entail the one-to-one correspondence between the branches of the game tree and the maximal developments modulo choices by *random* of different moves that result in the same successor state. The construction of the clauses for `goal` ensure that the payoffs are identical. The construction of the legal moves for *random* ensure that the combined probabilities of a branch equals the sum of the combined probabilities for all corresponding developments. Finally, the construction of the clauses for `sees` ensure that *player* gets identical percepts if, and only if, the states after a move by *random* are in the same information set. This in conjunction with Proposition 1 entails the claim. □

5 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. The recent enhancement of the general Game Description Language aims at remedying this limitation. In this paper we proved that this language extension, notationally simple as it is, indeed suffices to describe just any finite game faithfully. This shows that for the purpose of General Game Playing the language GDL 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 (say, Scrabble) or systems that play real, physical games [Barbu *et al.*, 2010].

Beyond General Game Playing we envision applications of our result in other AI areas in which game models arise, such as Multiagent Planning [Larbi *et al.*, 2007] or Multiagent Systems in general. A universal 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.

References

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

[Barbu *et al.*, 2010] A. Barbu, S. Narayanaswamy, and J. Siskind. Learning physically-instantiated game play through visual observation. In *Proc. of the IEEE Int.'l Conf. on Robotics and Automation*, 1879–1886, 2010.

[Björnsson and Finnsson, 2009] Y. Björnsson and H. Finnsson. CADIAPLAYER: A simulation-based general game player. *IEEE Trans. on CI&AI in Games*, 1(1):4–15, 2009.

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

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

[Jensen and Veloso, 2000] R. Jensen and M. Veloso. OBDD-based universal planning for synchronized agents in non-deterministic domains. *JAIR*, 13:189–226, 2000.

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

[Larbi *et al.*, 2007] R. Ben Larbi, S. Konieczny, and P. Marquis. Extending classical planning to the multi-agent case: A game-theoretic approach. In *Proc. of ECSQARU*, vol. 4724 of *LNCS*, 63–75. Springer, 2007.

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

[Love *et al.*, 2006] N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth. General Game Playing: Game Description Language Specification. Tech. Rep. LG–2006–01, Stanford, 2006. `games.stanford.edu`

[Mura, 2000] P. La Mura. Game networks. In *Proc. of the Conference on Uncertainty in AI*, 335–342, 2000.

[Rasmusen, 2007] E. Rasmusen. *Games and Information: an Introduction to Game Theory*. Blackwell, 4th ed., 2007.

[Rosenhouse, 2009] J. Rosenhouse. *The Monty Hall Problem*. Oxford University Press, 2009.

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

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