GDL-III: A Description Language for Epistemic General Game Playing

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

Abstract

GDL-III, a description language for general game playing with *imperfect information* and *introspection*, supports the specification of *epistemic* games. These are characterised by rules that depend on the knowledge of players. GDL-III provides a simpler language for representing actions and knowledge than existing formalisms: domain descriptions require neither explicit axioms about the epistemic effects of actions, nor explicit specifications of accessibility relations. We develop a formal semantics for GDL-III and demonstrate that this language, despite its syntactic simplicity, is expressive enough to model the famous Muddy Children domain. We also show that it significantly enhances the expressiveness of its predecessor GDL-II by formally proving that termination of games becomes undecidable, and we present experimental results with a reasoner for GDL-III applied to general epistemic puzzles.

1 Introduction

General game-playing (GGP) systems can understand the rules of new strategy games at runtime and learn to play these games effectively without human intervention [Genesereth *et al.*, 2005]. The interest for AI lies in the fact that general game-playing expertise requires algorithms beyond those designed in advance for specific games [Genesereth and Thielscher, 2014], exemplified through the annual international GGP competition since 2005 [Genesereth and Björnsson, 2013].

The language GDL, which uses the syntax and semantics of logic programming, has become the standard for describing the rules of games to general game-playing systems [Genesereth *et al.*, 2005]. The extension GDL-II has been developed with the aim to include general *imperfect-information* games [Schiffel and Thielscher, 2014]. Several general game-playing systems have recently been built to play this broader class of games too [Edelkamp *et al.*, 2012; Schofield and Thielscher, 2015]. However, even the extended language does not support the specification of games with *epistemic* goals [Ågotnes *et al.*, 2013] or, more generally, with rules that depend on the epistemic state of players. For even though we can mathematically reason about knowledge of players in imperfect-information games, it is not possible to refer to knowledge *within* a GDL-II rule. An example requiring this extended expressiveness are the so-called Russian Cards Problems [van Ditmarsch *et al.*, 2006; Cordón-Franco *et al.*, 2013], in which two cooperating players want to inform each other about their hands without a third player being able to learn anything from their (public) communication. GDL-II lacks the means to express such games since goal rules cannot refer to the knowledge of players.

With the addition of a single new keyword, the recently proposed *GDL-III* (for: GDL with *imperfect information* and *introspection* [Thielscher, 2016]) provides a description language suitable for *epistemic* general game playing. The additional keyword can be used to express individual, possibly nested knowledge, e.g. that player A knows that her cards are known to player B, as well as common knowledge. More generally, the extension to epistemic game rules is necessary in all applications of general game playing for domains where the goal of multiple agents is to gain, share and hide knowledge [Cooper *et al.*, 2016]. GDL-III provides a simpler language for representing actions and knowledge than existing formalisms like, for example, DEL (Dynamic Epistemic Logic [Bolander and Andersen, 2011]), in that neither explicit axioms about the epistemic effects of actions are required, nor explicit specifications of accessibility relations.

In this paper, we will formally define the semantics of GDL-III. It highlights the added expressiveness that players' knowledge can influence the state transition system for the game, a feature that is not available in GDL-II. We will formally prove that termination of games becomes undecidable in GDL-III, which shows that this language goes significantly beyond the expressiveness of its predecessor.

While the main motivation for the extended language is to describe of epistemic games for the purpose of general game playing, we will also present experimental results where GDL-III is used to encode, and automatically solve, epistemic puzzles such as "Cheryl's Birthday" [Chang, 2015]. Notably, this does not require any game-playing intelligence beyond the ability to compute legal playouts based on the game rules.

2 General Game Playing With GDL

The Game Description Language (GDL) is a formal language for specifying the rules of strategy games to a general game-playing system [Genesereth *et al.*, 2005]. It uses a prefix-variant of the syntax of logic programs along with the follow-

```
1  (role ann) (role bob) ... (role ken) (role random)
2  (init (round 0))
3
4  (yesno 0) (yesno 1) (yes 1)
5  (<= (legal random (muddy ?a ?b ... ?k))
6      (true (round 0))
7      (yesno ?a) (yesno ?b) ... (yesno ?k)
8      (or (yes ?a) ... (yes ?k)))
9  (<= (legal ?c noop)
10     (role ?c) (distinct ?c random) (true (round 0)))
11 (<= (legal random noop) (not (true (round 0))))
12 (<= (next (has ann mud)) (does random (muddy 1 ?b ... ?k)))
13 (<= (next (has bob mud)) (does random (muddy ?a 1 ... ?k)))
14     ...
15 (<= (next (has ?c mud)) (true (has ?c mud)))
16 (<= (sees ?c (dirt ?d)) (true (has ?d mud)) (distinct ?c ?d))
```

Figure 1: First part of a GDL-description of the MUDDYCHILDREN problem (schematically shown for $k \geq 1$ players).

ing special keywords (the last two were added in GDL-II for imperfect-information games [Schiffel and Thielscher, 2014]):

`(role R)`	`R` is a player
`(init F)`	feature `F` holds in the initial position
`(true F)`	feature `F` holds in the current position
`(legal R M)`	`R` has move `M` in the current position
`(does R M)`	player `R` does move `M`
`(next F)`	feature `F` holds in the next position
`terminal`	the current position is terminal
`(goal R V)`	player `R` gets payoff `V`
`(sees R P)`	player `R` is told `P` in the next position
`random`	the random player (aka. Nature)

GDL-II can and has been used to describe a variety of commonly played imperfect-information games (see, for example, `ggpserver.general-game-playing.de`). Here, we will describe one of the most commonly known epistemic domains as a game in order to demonstrate the expressiveness of GDL-III, beginning with the non-epistemic part.

Example 1 *For a GDL-description of the famous Muddy Children Puzzle, Figure 1 shows the rules of an "initialisation" round by which the players are randomly muddied. Lines 1–2 introduce the roles and the initial game state. The moves are specified by the rules for* `legal`*: Lines 5–8 defines the random muddying of the children (at least one, according to line 8). Lines 9–10 prescribe a no-op to the children in the first round and to* `random` *for the rest of the game. Lines 12–15 specify the effect of the random move on each of the children's foreheads and the persistence of mud. Line 16 implies that each child can see everyone else's forehead except his or her own.*

Any game description G that obeys certain general syntactic restrictions—for details we refer to Love *et al.* [2006]—determines a state transition system as follows. The derivable instances of `(role R)` define the players. The initial state consists in the derivable instances of `(init F)`. In order to determine the legal moves of a player in any given game state, this state has to be encoded using the keyword `true`: Let $S = \{f_1, \ldots, f_n\}$ be a state (more specifically, a finite set of ground terms over the signature of G), then G is extended by the n facts $S^{\texttt{true}} = \{ (\texttt{true}\ f_1) \ \ldots\ (\texttt{true}\ f_n)\}$. Those instances of `(legal R M)` that follow from $G \cup S^{\texttt{true}}$ define all legal moves `M` for player `R` in position S. In the same way, the clauses with `terminal` and `(goal R N)` in the head define, respectively, termination and goal values *relative* to the encoding of a given position. Finally, to determine a position update and the percepts of the players after a *joint move*, let M denote that players $r_1, \ldots, r_k$ take moves $m_1, \ldots, m_k$ and define $M^{\texttt{does}} = \{ (\texttt{does}\ r_1\ m_1) \ \ldots\ (\texttt{does}\ r_k\ m_k)\}$. All instances of `(next F)` that follow from $G \cup M^{\texttt{does}} \cup S^{\texttt{true}}$ comprise the updated position; likewise, the derivable instances of `(sees R P)` describe what a player perceives after joint move M in a given position S. All this is summarised below.

Definition 1 ([Schiffel and Thielscher, 2014]) *The semantics of a valid GDL-II game description G is given by*

- $R = \{r \colon G \models (\texttt{role}\ r)\}$
- $s_0 = \{f \colon G \models (\texttt{init}\ f)\}$
- $t = \{S \colon G \cup S^{\texttt{true}} \models \texttt{terminal}\}$
- $l = \{(r, m, S) \colon G \cup S^{\texttt{true}} \models (\texttt{legal}\ r\ m)\}$
- $u(M, S) = \{f \colon G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \models (\texttt{next}\ f)\}$
- $\mathcal{I} = \{(r, M, S, p) \colon G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \models (\texttt{sees}\ r\ p)\}$
- $g = \{(r, v, S) \colon G \cup S^{\texttt{true}} \models (\texttt{goal}\ r\ v)\}$

Schiffel and Thielscher [2014] define *legal play sequences* $M_1, \ldots, M_n$ of *joint* moves M_i (determining a move $M_i(r)$ for each $r \in R$) such that there are states $s_0, s_1, \ldots, s_n$ with

- $(r, M_i(r), s_{i-1}) \in l$ for all $r \in R$ (legality of moves);
- $s_i = u(M_i, s_{i-1})$ (position update).

Two legal sequences δ, δ' of the same length n are called *indistinguishable* for a player $r \in R$, written $\delta \sim_r \delta'$, if r's moves and percepts are the same throughout, that is, for all $i \in \{1, \ldots, n\}$:

- $M_i(r) = M'_i(r)$;
- $\{p \colon (r, M_i, s_{i-1}, p) \in \mathcal{I}\} = \{p' \colon (r, M'_i, s'_{i-1}, p') \in \mathcal{I}\}$.

It is worth noting that these definitions, which are based on the *objective* game rules about the percepts of players as given by a GDL-II description, imply standard properties about the relations of actions and knowledge, notably *synchronicity*, *perfect recall* and *no miracles* [van Benthem, 2014].

3 GDL-III: Semantics

The general game description language GDL-II is expressive enough to model games that give rise to complex epistemic models for players' knowledge [Ruan and Thielscher, 2014]. But the language does not support any references to players' knowledge in the game rules themselves, in order to specify knowledge goals, for example, or to require players to be truthful about their knowledge. We have therefore proposed to extend the syntax of GDL-II to GDL-*III* by one additional keyword for ***I**ntrospection* [Thielscher, 2016]:

`(knows R P)`	player `R` knows `P` in the current position
`(knows P)`	`P` is common knowledge

```
17 (<= (isMuddy ?c) (true (has ?c mud)))
18 (<= (legal ?c (say Yes)) (knows ?c (isMuddy ?c)))
19 (<= (legal ?c (say No))  (role ?c) (distinct ?c random)
20                          (not (knows ?c (isMuddy ?c))))
21 (<= (sees ?c (said ?d ?x)) (does ?d (say ?x)))
```

Figure 2: The remaining GDL-III game rules for MUDDYCHILDREN (without termination or goal rules).

Example 1 (cont'd) *The new keyword is used in Figure 2 to complete the description of* MUDDYCHILDREN*: Every child must* `say Yes` *if, and only if, they know that they themselves are muddy (lines 17–20); and the children hear each other's announcement (rule 21).*

The example nicely illustrate a unique feature of GDL-III as a knowledge representation language for epistemic domains: Only *objective* rules need to be given, which specify what agents observe and can do. While these rules *implicitly* determine what agents can know in principle, as we will show in this this section, it is left entirely to general game-playing systems themselves to reason about what the game rules imply about their and the other players' knowledge.

The new keyword uses *reification*, whereby a defined predicate, `P`, is used as an argument of another predicate.[1] A simple syntactic requirement ensures that nesting of knowledge is not circular: There must be some ordering $>$ on all predicate symbols `P` that occur as argument of `knows` such that `P` $>$ `Q` whenever `P` itself depends on `(knows R Q)` or `(knows Q)` [Thielscher, 2016].

The extended language GDL-III can be used to describe common epistemic imperfect-information games to general game-playing systems. An example are Russian Card Problems, where a winning condition for player Alice is that it is common knowledge that player Cath does not know anyone else's cards [Cordón-Franco *et al.*, 2013]. In GDL-III,

```
(<= (knowsSomeCard ?r)
    (knows ?r (has ?s ?c)) (distinct ?r ?s))
(<= (blind ?r) (not (knowsSomeCard ?r)))
(<= (goal alice 100) (knows (blind cath))..)
```

The interested reader can find the complete GDL-III description of a Russian Cards game in an accompanying technical report [Thielscher, 2017], along with a Public Announcement game [Ågotnes and van Ditmarsch, 2011] as another example.

We now define a formal semantics for GDL-III in two stages. First, we extend the logical interpretation of a set of game rules according to Definition 1 by incorporating the encoding of a given set $K = \{(\texttt{knows}\ r_1\ p_1),\ \ldots,\ (\texttt{knows}\ r_n\ p_m), (\texttt{knows}\ q_1), \ldots, (\texttt{knows}\ q_n)\}$ of instances of the knowledge predicate.[2] Most components of a game in GDL-III are evaluated *relative to a given set* K.

[1]Nested knowledge can be expressed with the help of auxiliary predicates; for example, `(knows a knows_b_p)` along with `(<= knows_b_p (knows b p))` says that `a` knows that `b` knows `p`. Syntactic sugar could of course be used to allow for actual nesting of `knows` handled by preprocessing.

[2]Here and in the following, we assume that knowledge sets K are always restricted to the relevant instances of `knows`-expressions, that is, which occur in some rule in the underlying GDL-description.

Definition 2 *The* pre-semantics *of a GDL-III game description G is given by sets R, s_0 as in Definition 1 along with*

- $t = \{(S,K)\colon\ G \cup S^{\texttt{true}} \cup K \models \texttt{terminal}\}$
- $l = \{(r,m,S,K)\colon\ G \cup S^{\texttt{true}} \cup K \models \texttt{legal}(r,m)\}$
- $u(M,S,K) = \{f\colon\ G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \cup K \models \texttt{next}(f)\}$
- $\mathcal{I} = \{(r,M,S,K,p)\colon G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \cup K \models \texttt{sees}(r,p)\}$
- $g = \{(r,v,S,K)\colon\ G \cup S^{\texttt{true}} \cup K \models \texttt{goal}(r,v)\}$

Example 1 (cont'd) *Suppose the current game state is given as* $S = \{\texttt{(has bob mud)}\}$.

If $K = \{\ \}$*, then* $(\texttt{bob}, \texttt{(say No)}, S, K) \in l$ *according to rule 19–20. This would also be Bob's only legal move when* $K = \{\texttt{(knows ann (isMuddy bob))}, \texttt{(knows ken (isMuddy bob))}\}$*, by rules 18–20.*

But if, say, $K' = \{\texttt{(knows bob (isMuddy bob))}\}$*, then game rule 18 implies that* $(\texttt{bob}, \texttt{(say Yes)}, S, K') \in l$*.*

It is important to note that the pre-semantics merely tells us how to determine all game-specific predicates *relative* to some knowledge state. Determining the correct K is the purpose of the second step in the definition of the semantics for GDL-III.

This second step requires an inductive characterisation of legal play sequences and their resulting knowledge states. *Common* knowledge is defined using the notion of the *reflexive, transitive closure* $\sim^+$ of a given family of indistinguishability relations $\sim_r$ (one for every role r in R). Formally, $\sim^+$ is the smallest relation such that for all $\delta, \delta'\delta''$:

- $\delta \sim^+ \delta$ and
- if $\delta \sim^+ \delta'$ and $\delta' \sim_r \delta''$ for some $r \in R$ then $\delta \sim^+ \delta''$.

Definition 3 *Let G be a game description along with all the sets and relations it describes according to Definition 2.*

- *The play sequence of length* 0*, denoted by* ε*, is* legal *and satisfies* $\varepsilon \sim_r \varepsilon$*, for all* $r \in R$*. It results in* state s_0 *and* knowledge state K_ε *as the smallest set that satisfies*
$$K_\varepsilon = \{(\texttt{knows}\ r\ p)\colon\ r \in R,\ G \cup s_0^{\texttt{true}} \cup K_\varepsilon \models p\} \cup \{(\texttt{knows}\ p)\colon G \cup s_0^{\texttt{true}} \cup K_\varepsilon \models p\}^{[3]}$$
- *For the inductive definition, let δ be a legal play sequence of length $n \geq 0$ resulting in (s_n, K_n).*
Sequence δ followed by M, written δM, is a legal *play sequence of length* $n+1$ *if* $(M(r), s_n, K_n) \in l$ *for all* $r \in R$*. It results in* state $s_{\delta M} = u(M, s_n, K_n)$ *and, as the* knowledge state*, the smallest* $K_{\delta M}$ *that satisfies*
$$\begin{aligned} K_{\delta M} = \{\ \texttt{(knows}\ r\ p\texttt{)} :\ & G \cup s^{\texttt{true}}_{\delta' M'} \cup K_{\delta M} \models p \\ & \textit{for all } \delta' M' \sim_r \delta M\} \\ \cup \{\ \texttt{(knows}\ p\texttt{)} :\ & G \cup s^{\texttt{true}}_{\delta' M'} \cup K_{\delta M} \models p \\ & \textit{for all } \delta' M' \sim^+ \delta M\} \end{aligned} \tag{1}$$
This definition uses a straightforward generalisation from GDL-II of the notion of (in-)distinguishable sequences (of length $n+1$): Relation $\delta M \sim_r \delta' M'$ holds if role r cannot distinguish δ from δ' and takes the same move and obtains the same percepts in M. Formally,

[3]Hence, as in GDL-II there is no uncertainty about the initial state. However, any desired initial epistemic model can be obtained by just one `random` move [Ruan and Thielscher, 2014].

– $\delta \sim_r \delta'$
– $M(r) = M'(r)$
– $\{p : (r, M, s_n, K_n, p) \in \mathcal{I}\}$
$= \{p' : (r, M', s'_n, K'_n, p') \in \mathcal{I}\}$

This is well-defined for any GDL-III game description G with acyclically defined knowledge predicates: $K_{\delta,M}$ can be "constructed" by, first, evaluating all `(knows` r p`)` and `(knows` p`)` instances for which p itself does not depend on `knows` and, then, evaluating the other instances in accordance with the hierarchy. This implies that knowledge states are always consistent. It is also easy to verify that Definitions 2 and 3 always produce a single (knowledge) state transition system.

Definition 3 can be understood as follows: Players know everything that follows from the initial state (K_ε). Legality and effects of moves are determined inductively from actual state s_n and knowledge state K_n. As in GDL-II, two legal play sequences δM and $\delta' M'$ cannot be distinguished by role r after the last move if they were indistinguishable beforehand (i.e., $\delta \sim_r \delta'$) and if the player made the same legal move in both M and M' and obtained the same percepts. This (in-)distinguishability relation in turn determines the evaluation of the knowledge predicates, including common knowledge, for the resulting knowledge state $K_{\delta M}$.

Example 1 (cont'd) *For the sake of brevity, we only show the (non-elementary) 2-children variant of our example game (with* `ann` *and* `bob`*). Let the three possible joint actions in the initial round (cf. 5–8 in Figure 1) be denoted by* `01,11,10`*. Since the game rules are common knowledge, all children know that at least one of them is muddy after the first round. Nothing is observed as a direct consequence of this choice by* `random`*, however. Hence the knowledge state, which initially is empty (no one is muddy and ignoring other knowledge), does not change in any of the cases:*

$$K_\epsilon = \{\} = K_{01} = K_{11} = K_{10}$$

According to the rules in Figure 2, the two children can only say `No`*, denoted by* `NN` *below. However, by game rule 16 they get to see each other's forehead, which determines the following (in-)distinguishability after the next round:*

$$\texttt{01}\cdot\texttt{NN} \sim_{\texttt{ann}} \texttt{11}\cdot\texttt{NN} \sim_{\texttt{bob}} \texttt{10}\cdot\texttt{NN} \qquad (2)$$

In particular, Bob can distinguish $\texttt{01}\cdot\texttt{NN}$ *from* $\texttt{11}\cdot\texttt{NN}$ *after receiving an information token about Ann's being muddy; similarly for Ann. From (1) in Definition 3 and relation (2), along with game rules 12–15 and 17, it follows that*

$$\begin{aligned}
K_{\texttt{01}\cdot\texttt{NN}} &= \{\, \texttt{(knows ann (isMuddy bob))}, \\
&\qquad \texttt{(knows bob (isMuddy bob))}\,\} \\
K_{\texttt{11}\cdot\texttt{NN}} &= \{\, \texttt{(knows ann (isMuddy bob))}, \\
&\qquad \texttt{(knows bob (isMuddy ann))}\,\} \\
K_{\texttt{10}\cdot\texttt{NN}} &= \{\, \texttt{(knows ann (isMuddy ann))}, \\
&\qquad \texttt{(knows bob (isMuddy ann))}\,\}
\end{aligned}$$

Hence, without having seen his own forehead, Bob can (in fact, must) say `Yes` *in* $\texttt{01}\cdot\texttt{NN}$*; the same for Ann in* $\texttt{10}\cdot\texttt{NN}$*; and both can only say* `No` *in* $\texttt{11}\cdot\texttt{NN}$*. Now, according to rule 21, Ann can hear what Bob says. She can thus distinguish* $\texttt{01}\cdot\texttt{NN}\cdot\texttt{NY}$ *from* $\texttt{11}\cdot\texttt{NN}\cdot\texttt{NN}$*, where* `Y` *abbreviates* `(say Yes)`*. Likewise, Bob can distinguish* $\texttt{10}\cdot\texttt{NN}\cdot\texttt{YN}$ *from* $\texttt{11}\cdot\texttt{NN}\cdot\texttt{NN}$*. All play sequences thus become distinguishable by both players, hence:*

$$\begin{aligned}
K_{\texttt{11}\cdot\texttt{NN}\cdot\texttt{NN}} = \{\, &\texttt{(knows ann (isMuddy ann))}, \\
&\texttt{(knows ann (isMuddy bob))}, \\
&\texttt{(knows bob (isMuddy ann))}, \\
&\texttt{(knows bob (isMuddy bob))}\,\}
\end{aligned}$$

To summarise, if only one of the children is muddied by the initial random move, then he or she will know after the next round. If both are muddy, they will know one round later.

The argument can be easily generalised to prove that if $l \geq 1$ of k children are muddied, then it takes $l + 1$ rounds for all children to know who is muddy, for any $l \leq k$ and $k \in \mathbb{N}$.

4 On the Expressiveness of GDL-III

The two-stage, inductive semantics for GDL-III is considerably different from the state-transition systems for its two predecessor languages, which can be defined without induction. This indicates that the addition of the epistemic keyword significantly enhances the expressiveness of the game description language. Indeed, as we will show in the following, extending GDL-II to GDL-III has the somewhat surprising effect that termination of games becomes undecidable even under the usual syntactic restrictions that guarantee finiteness of the state space.

The intuitive reason for this is that even over a finite game state space, the knowledge that players have of each other, and of each other's knowledge, can grow arbitrarily. This is of course true for GDL-II games as well. But game rules in that language can only be conditioned on the actual game state and not the knowledge of the players, which suffices to guarantee decidability of termination. In contrast, in GDL-III the end of a game may be conditioned on the epistemic structure. In order to formally prove that this leads to the undecidability of termination, we adapt a technique introduced by Bolander and Andersen [2011] of representing a Turing machine and its tape with the help of an *epistemic* structure.

Let an arbitrary Turing machine (TM) be given, with finite set of states Q, binary alphabet $\{\texttt{blank}, \texttt{marked}\}$ and initially blank tape. We construct a GDL-III game with `random` and two players, called `ann` and `bob`, as follows. The game states are built from the finite set of features $Q \cup \{\texttt{blank}, \texttt{marked}, (\texttt{right ann}), (\texttt{right bob})\}$. The initial game state is encoded by rule 2 in Figure 3, where $\texttt{q0} \in Q$ is the starting state of the TM.

Each instruction of the Turing machine is translated to a small set of legal moves for `random`. Moves by `random` are only partially observed by the two players, thus leading to a new knowledge structure that corresponds to the next state of the Turing machine and its tape. As an example, Figure 3 shows the encoding for transitions of the form

$$(q, \texttt{blank}) \;\rightarrow\; (q', \texttt{marked}, \textit{right}) \qquad (3)$$

Put in words, if the TM is in state q and over a blank cell, then the cell gets marked, the TM transitions to state q' and the head moves to the right. This instruction gets translated into a set of eight possible moves, abbreviated as `(i ?r ?x)` in Figure 3 with $\texttt{?r} \in \{\texttt{ann}, \texttt{bob}\}$ and $\texttt{?x} \in \{\texttt{1...4}\}$.

```
1 (role ann) (role bob) (role random)
2 (init q0) (init blank) (init (right ann))
3
4 (legal ann noop) (legal bob noop)
5 (<= (legal random (i ann ?x))
6     (not (true (right bob))) (pre ?r ?x))
7 (<= (legal random (i bob ?x))
8     (not (true (right ann))) (pre ?r ?x))
9 (<= notQ (not (true q)))
10 (<= (pre ?r 1) (knows ?r notQ))
11 (<= (pre ?r 2) (true q) (true (right ?r)) (true blank))
12 (<= (pre ?r 3) (knows ?r q) (true (right ?r)) (true blank))
13 (<= (pre ?r 4) (not (true q)) (not (knows ?r notQ)))
14
15 (<= (sees ?r1 i) (role ?r1) (does random (i ?r ?x)))
16 (<= (sees ann 3) (does random (i bob 3)))
17 (<= (sees ann 4) (does random (i bob 4)))
18 (<= (sees bob 3) (does random (i ann 3)))
19 (<= (sees bob 4) (does random (i ann 4)))
20 (<= (next ?f) (true ?f) (does random (i ?r 1)))
21 (<= (next marked) (does random (i ?r 2)))
22 (<= (next blank) (does random (i ?r 3)))
23 (<= (next q')
24     (or (does random (i ?r 3)) (does random (i ?r 4))))
25 (<= (next (right bob))
26     (or (does random (i ann 3))(does random (i ann 4))))
27 (<= (next (right ann))
28     (or (does random (i bob 3))(does random (i bob 4))))
29 (<= (next blank)  (true blank) (does random (i ?r 4)))
30 (<= (next marked) (true marked)(does random (i ?r 4)))
```

Figure 3: Encoding a Turing Machine in GDL-III, with the specific type of instruction (3) as example.

These arguments are used to determine which of the possible moves are actually legal depending on the game state and the knowledge of the players according to rules 5–13.

Suppose, as an example, instruction (3) occurs in the TM-program with $q = \texttt{q0}$ and $q' = \texttt{q1}$. From the initial state given by line 2 and from rules 5–13 it follows that only `(i ann 2)` and `(i ann 3)` are actual legal moves for `random` in this state. Both players learn that an `i`-action happens according to rule 15. But only Bob knows which of the two, because he gets to see `3` in one case (rule 18) but not the other. Following rules 20–30, the states resulting from `(i ann 2)` and `(i ann 3)` are as follows, where it is also indicated that Ann cannot distinguish between them:[4]

$$\texttt{marked} \sim_{\text{ann}} \texttt{q1,blank,(right bob)} \quad (4)$$

This epistemic structure corresponds to the state of the TM after marking the initial cell and moving the head to the right cell, which is blank. The reader is invited to verify that if the TM-program contains another instance of instruction (3) with $q = \texttt{q1}$ and $q' = \texttt{q2}$, then the game rules in Figure 3 entail the following epistemic structure after the next move:[5]

$$\texttt{marked} \sim_{\text{ann}} \texttt{marked} \sim_{\text{bob}} \texttt{q2,blank,(right ann)} \quad (5)$$

This shows how, despite a finite state space, the epistemic state can grow arbitrarily and thus model an unbounded TM tape.

To summarise, the instance of action `(i ?r ?x)` with `?x` $=$ `1` applies to all "possible worlds" in the knowledge structure except for the one that corresponds to the cell currently under the head of the TM and the one immediately to the right; alternative `?x` $=$ `2` is for the cell under the head; and alternative `?x` $=$ `3` is for the next cell to the right in case it has not been seen before (and hence is set to `blank`). Alternative `?x` $=$ `4` handles the cell to the right in case it has been seen before (which, in terms of the epistemic structure, means there is a possible state that player `?r` cannot distinguish from the state in which q holds, cf. rule 13).

Lack of space does not allow us to provide game rules for the other types of TM-instructions, which can be defined similar to those shown in Figure 3. To prove the main result, let `qf` $\in Q$ be the (only) final state of the TM. The following rules say that the game terminates when it is no longer common knowledge that `qf` is false:

```
(<= notQF (not (true qf)))
(<= terminal (not (knows notQF)))
```

Hence the game ends if and when the epistemic structure for the knowledge of the players contains an accessible state where `qf` became true; e.g., if `q2` is the final TM state, then `terminal` would be true under knowledge state (5). This leads to the following result.

Theorem 1 *The GDL-III game that encodes a Turing machine* TM *with binary alphabet, initially blank tape and a single final state terminates if, and only if,* TM *halts.*

5 Solving Epistemic Puzzles With GDL-III

Epistemic puzzles are characterised by multiple agents that reason about each other's (lack of) knowledge in the course of a sequence of actions. A common approach to axiomatising epistemic puzzles is the use of (multi-)modal logics, e.g. Dynamic Epistemic Logic [van Ditmarsch *et al.*, 2005]. With the addition of introspection, the game description language provides an alternative, general formalism for encoding of epistemic puzzles, which can then be automatically solved by a mere "legal reasoner" as proposed in [Thielscher, 2016].

Example 2 *[Chang, 2015] Albert and Bernard want to know Cheryl's birthday. She draws a list with a number of possible dates and then tells them separately the correct month and day, respectively. A dialogue follows in which Albert first says that he doesn't know the birthday* and *that he knows that Bernard doesn't know it either, then Bernard says that he now knows the date, and after that Albert announces that finally he does so too. Figure 4 shows how this puzzle can be described in GDL-III in such a way that* every *legal playout corresponds to a solution and vice versa.*[6]

To test the scalability of a GDL-III legal reasoner, we ran experiments on a 2.8 GHz processor with 8 GB of RAM

[4]More precisely, Ann cannot distinguish the two legal play sequences that lead to the two states.

[5]To see why, note in particular that `(i bob 1)` is legal in the first state in (4) and leaves it unchanged according to rule 20.

[6]It is worth noting that the description in Figure 4 is not meant to be actually played by general GDL-III players but rather to be used by a mere "legal" player to solve this puzzle. An alternative formalisation, to test the strategic abilities of general game-playing systems, would require the definition of a goal for the three roles etc.

```
 1 (role albert) (role bernard) (role cheryl)
 2 (date may 15)  (date may 16)  (date may 19)
 3 (date jun 17)  (date jun 18)  (date jul 14)  (date jul 16)
 4 (date aug 14)  (date aug 15)  (date aug 17)
 5 (succ 0 1)  (succ 1 2)  (succ 2 3)  (succ 3 4)  (init (step 0))
 6 (<= (legal cheryl (choose ?m ?d)) (true (step 0)) (date ?m ?d))
 7 (<= (sees albert  ?m) (does cheryl (choose ?m ?d)))
 8 (<= (sees bernard ?d) (does cheryl (choose ?m ?d)))
 9 (<= (next (secret ?m ?d)) (does cheryl (choose ?m ?d)))
10 (<= (next (secret ?m ?d)) (true (secret ?m ?d)))
11 (<= (next (step ?n)) (true (step ?m)) (succ ?m ?n))
12 (<= (legal albert noop)  (or (true (step 0)) (true (step 2))))
13 (<= (legal bernard noop) (not (true (step 2))))
14 (<= (legal cheryl noop)  (not (true (step 0))))
15 (<= (birthday ?m ?d) (true (secret ?m ?d)))
16 (<= (knowsDate ?r)   (knows ?r (birthday ?m ?d)))
17 (<= (notKnowsDate ?r)(not (knowsDate ?r)))
18 (<= (legal albert sayUnknown)
19     (true (step 1)) (not (knowsDate albert))
20     (knows albert (notKnowsDate bernard)))
21 (<= (legal bernard sayKnown)
22     (true (step 2)) (knowsDate bernard))
23 (<= (legal albert sayKnown)
24     (true (step 3)) (knowsDate albert))
25 (<= (sees ?r ?m)
26     (role ?r)
27     (or (does albert ?m) (does bernard ?m)))
28 (<= terminal (true (step 4)))
```

Figure 4: CHERYLSBIRTHDAY: a possible description of this puzzle using the syntax of GDL with introspection. Cheryl begins by picking a date (rule 6), of which Albert and Bernard only get to see the month and day, respectively (lines 7–8). Using three defined properties (lines 15–17), announcements are modelled by two moves (`sayUnknown`, `sayKnown`) whose preconditions assume players to be truthful (lines 18–24) and which are public (rule 25–27).

with `http://potassco.sourceforge.net`, an off-the-shelf answer set solver, for interpreting GDL-III rules. Times are reported in seconds (CPU time). The original problem consists of 10 dates across 4 different months and 6 different days. We kept a similar ratio of different "months" and "days" as we increased the problem size (# of dates) in order to ensure that the randomly chosen instances are equally difficult in that the number of solutions averages to approximately one.[7] The results are summarised in the table below for each size and averaged over 1,000 random problem instances. They demonstrate how the average time to compute legal play sequences increases since this game requires maintaining and evaluating the knowledge state of both roles, including what they entail about one player's knowledge of the other player.

#dates	#months	#days	avg #solutions	avg time
10	5	5	0.57	0.00
100	30	30	1.37	0.01
1000	200	200	1.07	0.20
10000	1500	1500	0.68	3.79

6 Related Work and Conclusion

The extended general game description language GDL-III shares with its predecessor GDL-II the unique feature that game designers have to specify only the objective rules about what players can see and do. Rules about how moves and percepts affect the knowledge of players are not required. Our semantics of the knowledge operator in GDL-III inherits the concept of indistinguishability of play sequences used to characterise the evolution of knowledge in GDL-II, assuming players with perfect recall [Schiffel and Thielscher, 2014]. Knowledge preconditions and knowledge goals in GDL-III therefore refer to what players can and cannot know in principle given the observations they make throughout a game.

The added expressiveness necessitates a semantics where state transition systems, which provide the full semantics for GDL-II, act merely as a pre-semantics. The complete semantics for GDL-III accounts for knowledge feeding back into the definition of legal play sequences. This significantly enhances the expressiveness of game descriptions, but it also leads to the undecidability of termination of games in general. This does not affect the practicality of the language for general game playing, however. Since all reachable (epistemic) states are finite, all reasoning problems, such as determining legal moves, remain decidable. A game designer can easily guarantee termination of a game, e.g. through the standard use of a step counter [Genesereth and Björnsson, 2013].

Our experiments have shown how the size of the information sets of players influences the runtimes of a basic reasoner, unlike in case of GDL-II. Explicitly maintaining the set of relevant legal play sequences is practically viable for short epistemic games or puzzles. However, many games will require compact representations of information sets, e.g. as has been proposed for Kriegspiel [Ciancarini and Favini, 2007], or approximations via state sampling [Long *et al.*, 2010].

Several languages for epistemic multi-agent domains have been proposed recently, including [Kominis and Geffner, 2015; Muise *et al.*, 2015; Jiang *et al.*, 2016]. These differ from GDL-III in that they all require explicit specifications of what an agent knows (respectively, believes) as the result of an action or sensing. To axiomatise MUDDYCHILDREN in any of these languages, for instance, you would need to provide explicit axioms on the effect of a child (not) saying "yes" on everyone else's knowledge. Likewise, epistemic puzzles similar to Cheryl's Birthday have been solved using model checking systems for Dynamic Epistemic Logic [van Ditmarsch *et al.*, 2005]. Again, the main difference is that DEL requires an explicit encoding of an epistemic structure in form of a concrete accessibility relation for the problem in question.

The proposed encoding of epistemic puzzles uses GDL different from its main purpose in general game playing, because no players actually play the game. Yet it is an interesting by-product of having a basic reasoner for the new language element. GDL-III also has applications in general game playing beyond the description of epistemic games. For example, it could be used by GDL-II players to generate logical *strategy* rules by which they condition their moves on their knowledge. Beyond general game playing, GDL-III can be used for general multi-agent epistemic problems [Cooper *et al.*, 2016].

[7]Cheryl's Birthday would of course be less famous if it did not have a unique solution, but problems with no or multiple solutions are equally relevant for testing the runtime behaviour of a controller.

References

[Ågotnes and van Ditmarsch, 2011] Thomas Ågotnes and Hans van Ditmarsch. What will they say?—Public announcement games. *Synthese*, 179:57–85, 2011.

[Ågotnes *et al.*, 2013] Thomas Ågotnes, Paul Harrenstein, Wiebe van der Hoek, and Michael Wooldridge. Boolean games with epistemic goals. In *Logic, Rationality, and Interaction*, volume 8196 of *LNCS*, pages 1–14, Hangzhou, China, October 2013. Springer.

[Bolander and Andersen, 2011] Thomas Bolander and Mikkel Andersen. Epistemic planning for single- and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011.

[Chang, 2015] Kenneth Chang. A math problem from Singapore goes viral: When is Cheryl's birthday? *The New York Times*, 14 Apr 2015.

[Ciancarini and Favini, 2007] Paolo Ciancarini and Gian Piero Favini. Representing kriegspiel states with metapositions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2450–2455, Hyderabad, India, January 2007.

[Cooper *et al.*, 2016] Martin Cooper, Andreas Herzig, Faustine Maffre, Frédeéric Maris, and Pierre Régnier. A simple account of multi-agent epistemic planning. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 193–201, The Hague, August 2016.

[Cordón-Franco *et al.*, 2013] Andrés Cordón-Franco, Hans van Ditmarsch, David Fernández Duque, and Fernando Soler-Toscano. A colouring protocol for the generalized Russian cards problem. *Journal of Theoretical Computer Science*, 495:81–95, 2013.

[Edelkamp *et al.*, 2012] Stefan Edelkamp, Tim Federholzner, and Peter Kissmann. Searching with partial belief states in general games with incomplete information. In *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, volume 7526 of *LNCS*, pages 25–36, Saarbrücken, Germany, September 2012. Springer.

[Genesereth and Björnsson, 2013] Michael Genesereth and Yngvi Björnsson. The international general game playing competition. *AI Magazine*, 34(2):107–111, 2013.

[Genesereth and Thielscher, 2014] Michael Genesereth and Michael Thielscher. *General Game Playing*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2014.

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

[Jiang *et al.*, 2016] Guifei Jiang, Dongmo Zhang, Laurent Perrussel, and Heng Zhang. Epistemic GDL: A logic for representing and reasoning about imperfect information games. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1138–1144, New York, 2016.

[Kominis and Geffner, 2015] Filippos Kominis and Hector Geffner. Beliefs in multiagent planning: From one agent to many. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 147–155, Jerusalem, Israel, June 2015.

[Long *et al.*, 2010] Jeffrey Long, Nathan Sturtevant, Michael Buro, and Timothy Furtak. Understanding the success of perfect information Monte Carlo sampling in game tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 134–140, Atlanta, July 2010.

[Love *et al.*, 2006] Nathaniel Love, Timothy Hinrichs, David Haley, Eric Schkufza, and Michael Genesereth. General Game Playing: Game Description Language Specification. Technical Report LG–2006–01, Stanford University, 2006. Available at: `games.stanford.edu`.

[Muise *et al.*, 2015] Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3327–3334, Austin, January 2015.

[Ruan and Thielscher, 2014] Ji Ruan and Michael Thielscher. Logical-epistemic foundations of general game descriptions. *Studia Logica*, 102(2):321–338, 2014.

[Schiffel and Thielscher, 2014] Stephan Schiffel and Michael Thielscher. Representing and reasoning about the rules of general games with imperfect information. *Journal of Artificial Intelligence Research*, 49:171–206, 2014.

[Schofield and Thielscher, 2015] Michael Schofield and Michael Thielscher. Lifting model sampling for general game playing to incomplete-information models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3585–3591, Austin, January 2015.

[Thielscher, 2016] Michael Thielscher. GDL-III: A proposal to extend the game description language to general epistemic games. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 1630–1631, The Hague, August 2016.

[Thielscher, 2017] Michael Thielscher. A formal description language for general epistemic games. Technical Report UNSW-CSE-TR-201708, School of Computer Science and Engineering, University of New South Wales, May 2017. Available at: `www.cse.unsw.edu.au/~reports/`

[van Benthem, 2014] Johan van Benthem. *Logic in Games*. MIT Press, 2014.

[van Ditmarsch *et al.*, 2005] Hans van Ditmarsch, Ji Ruan, and Rineke Verbrugge. Model checking Sum and Product. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, volume 3809 of *LNCS*, pages 790–795, Sydney, Australia, December 2005. Springer.

[van Ditmarsch *et al.*, 2006] Hans van Ditmarsch, Wiebe van der Hoek, Ron van der Meyden, and Ji Ruan. Model checking russian cards. *Electronic Notes in Theoretical Computer Science*, 149(2):105–123, 2006.