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

Michael Thielscher
University of New South Wales, Australia
mit@unsw.edu.au

Abstract

We present an extension of the standard game description language for general game playing to include *epistemic games*, which are characterised by rules that depend on the knowledge of players. A single additional keyword suffices to define *GDL-III*, a game description language with *imperfect information* and *introspection*. We define syntax and semantics and present an execution model for this extended language. We develop a provably correct answer set program (ASP) for reasoning about GDL-III games. We also show how the extended language can be used to formalise general epistemic puzzles and how those can be solved using a mere game controller for GDL-III, that is, without requiring any game-playing intelligence beyond the ability to generate legal play sequences.

1 Introduction

The game description language GDL has become the standard for describing the rules of games to general game-playing systems [Love *et al.*, 2006; Genesereth and Thielscher, 2014]. Its syntax is a variation of logic programming with specific keywords for specifying the initial game states, legality and effects of moves, as well as termination and winning conditions. While descriptions of games with simultaneous moves are supported, the language assumes that players have complete knowledge of the game state after every round [Genesereth *et al.*, 2005]. The extension GDL-II has been developed with the aim to include general *imperfect information* games [Schiffel and Thielscher, 2014]. The logical-epistemic foundations of this extended language have been analysed [Ruan and Thielscher, 2014] and general game-playing systems for imperfect information have been developed on the basis of GDL-II [Edelkamp *et al.*, 2012; Schofield and Thielscher, 2015].

Although the extended description languages can be used to describe games with imperfect and incomplete information [Schiffel and Thielscher, 2014], it 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. This is so because while we can mathematically reason about knowledge of players in imperfect-information games, we cannot *refer* to knowledge *within* a GDL-II rule. As an example, consider the game NUMBER-GUESSING from the GDL-II track at the AI'12 general game playing competition,[1] in which the goal for a single player is to repeatedly ask yes/no questions to determine an initially unknown number. However, the player can win merely by guessing correctly. GDL-II is not expressive enough to specify, as a necessary winning condition, that the player actually *knows* the number from the preceding percepts.

Another example of epistemic games are so-called Russian Cards Problems [Cordón-Franco *et al.*, 2013; van Ditmarsch *et al.*, 2006], in which the goal of two cooperating players is to inform each other about their hands through public announcements without a third player being able to learn anything from their communication. GDL-II cannot be used to express these goal rules because the language does not provide means to refer to the fact that a player knows that another player knows their cards, nor that it is common knowledge that a third player does not. Yet other examples beyond the expressiveness of GDL-II are games in which players are obliged, by the official rules, to always truthfully answer questions about what they know. This also requires the conditioning of the legality of a move on players' epistemic states.

The purpose of this paper is to overcome these limitations by developing a formal language suitable for general game playing that supports the specification of game rules which need to refer to the epistemic states of the players. This is a useful addition to GDL-II whenever players' knowledge is not present in the current state but follows implicitly from their past percepts. We will show that a single additional keyword suffices to define *GDL-III*, a general description language for games with *imperfect information* and *introspection*. The new keyword can be used to express both individual, possibly nested knowledge, e.g. that player A knows that her cards are known to player B, as well as common knowledge, e.g. that player C does not know of any card held by another player and everyone knows this (and also that everyone knows that everyone knows etc). We will formally define the syntax and the semantics of GDL-III as an extension of the existing language GDL-II. We will also develop a provably correct answer set program (ASP) for reasoning

[1]see `ai2012.web.cse.unsw.edu.au/ggp.html`

about GDL-III games. This can either be used as the basis for a *legal* player that can handle the extended language, or as a game controller to administer the execution of arbitrary GDL-III games.

While the main purpose of our language extension is to allow for the description of epistemic games for the purpose of general game playing, we will furthermore demonstrate how GDL-III can be used to encode, and automatically solve, epistemic puzzles like Cheryl's Birthday, which recently acquired public fame [Chang, 2015]. Notably, this does not even require an intelligent general game player; a mere game controller that is capable of computing legal playouts suffices to solve these puzzles.

The remainder of the paper is organised as follows. After providing the necessary background in Section 2, we will introduce the syntax of the extended language and its semantics in Section 3. The ASP-based controller for executing GDL-III games will be presented in Section 4. In Section 5, we will use the example of Cheryl's Birthday to show how the extended language can also be used to formalise general epistemic puzzles that can then be automatically solved using the ASP-based implementation of a naive game controller. In Section 6, we will present experimental results to test the scalability of the ASP-based game controller and two of the examples considered in this paper, NUMBERGUESSING and CHERYLSBIRTHDAY. Finally, in Section 7 we compare and contrast GDL-III to other existing languages and conclude.

2 General Game Playing With GDL

The declarative 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 following special keywords.

`(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)

The bottom two keywords have been added in GDL-II to enable the specification of games with randomised moves and imperfect information [Schiffel and Thielscher, 2014].

Example 1 (NUMBERGUESSING) *The rules in Figure 1 formalise a simple number guessing game. Lines 1–2 introduce the roles. Line 7 defines the initial game state. The moves are specified by the rules for* `legal`*: In the first round, a number between 1 and 32 is randomly chosen (rule 12–13). The player can then repeatedly ask yes/no questions (lines 15–16) or attempt a guess (lines 17–18). The guesser's percepts are truthful replies to his questions (rule 21–24),*[2] *using a recursive axiomatisation of* `less` *(rule 19–20). The remaining rules specify the state update (*`next`*), the conditions for the game to end (*`terminal`*), and the payoff for the players (*`goal`*).*

Syntax and Semantics

In order to admit an unambiguous interpretation, so-called *valid* GDL-II game descriptions must obey certain general syntactic restrictions; for details we refer to [Love *et al.*, 2006; Schiffel and Thielscher, 2014]. Under these restrictions, any valid GDL-II game description G determines a state transition system as follows.

To begin with, the derivable instances of `(role R)` define the players, and the initial state consists in the derivable instances of `(init F)`. In order to determine the legal moves of a player in any given game state, this state has to be encoded first, using the keyword `true`: Let $S = \{f_1, \dots, f_n\}$ be a state (more specifically, a finite set of ground terms over the signature of G), then G is extended by the n facts

$$S^{\texttt{true}} \stackrel{\text{def}}{=} \{(\texttt{true}\ f_1)\ \ \dots\ \ (\texttt{true}\ f_n)\} \tag{1}$$

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.

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

$$M^{\texttt{does}} \stackrel{\text{def}}{=} \{(\texttt{does}\ r_1\ m_1)\ \ \dots\ \ (\texttt{does}\ r_k\ m_k)\} \tag{2}$$

All instances of `(next F)` that follow from $G \cup M^{\texttt{does}} \cup S^{\texttt{true}}$ compose the updated position; likewise, the derivable instances of `(sees R P)` describe what a player perceives when the given joint move is done in the given position. All this is summarised below, where "$\models$" denotes entailment wrt. the unique stable model of a stratified set of clauses.

Definition 1 *The* semantics *of a valid GDL-II game description G is given by*

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

GDL-II games are played using the following execution model [Schiffel and Thielscher, 2014].

[2] A rule for a percept `no` can of course be added but is not necessary, because *not* receiving a `yes` suffices as an answer.

```
1 (role guesser)
2 (role random)
3
4 (number 1)  ...  (number 32)
5 (succ 0 1)  ...  (succ 31 32)
6
7 (init (step 0))
8
9 (<= (legal guesser noop) (true (step 0)))
10 (<= (legal random noop) (not (true (step 0))))
11
12 (<= (legal random (choose ?n))
13     (number ?n) (true (step 0)))
14
15 (<= (legal guesser (ask_if_less ?n))
16     (number ?n) (not (true (step 0))))
17 (<= (legal guesser (guess ?n))
18     (number ?n) (not (true (step 0))))
19 (<= (less ?m ?n) (or (succ ?m ?n)
20                      ((succ ?m ?l) (less ?l ?n))))
21 (<= (sees guesser yes)
22     (does guesser (ask_if_less ?n))
23     (true (secret ?m))
24     (less ?m ?n))
25
26 (<= (next (secret ?n)) (does random (choose ?n)))
27 (<= (next (secret ?n)) (true (secret ?n)))
28 (<= (next (step ?n))   (true (step ?m)) (succ ?m ?n))
29 (<= (next right) (does guesser (guess ?n)) (true (secret ?n)))
30 (<= (next wrong) (does guesser (guess ?n)) (not (true (secret ?n))))
31
32 (<= terminal
33     (or right wrong (true (step 12))))
34
35 (<= (goal guesser 100) (true right))
36 (<= (goal guesser   0) (not (true right)))
```

Figure 1: The GDL-II game specification NUMBERGUESSING.

1. All players receive the game description G.
2. Starting with s_0, in each state S each player $r \in R$ selects a legal move from $\{m\colon l(r,m,S)\}$.[3]
3. The update function (synchronously) applies the joint move M to the current position, resulting in the new position $S' = u(M,S)$. Furthermore, each of the roles r receives their individual percepts $\{p\colon \mathcal{I}(r,M,S,p)\}$.
4. This continues until a terminal state is reached, and then the goal relation determines the result for all players.

Based on this protocol, *legal play sequences* are defined [Schiffel and Thielscher, 2014] as sequences $\mu_1, \ldots, \mu_n$ of *joint moves* μ_i, that is a move $\mu_i(r)$ for each $r \in R$, which satisfy the following: There is a sequence of states $s_0, s_1, \ldots, s_n$ such that, for all $i \in \{1, \ldots, n\}$,

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

Furthermore, $\{s_0, \ldots, s_{n-1}\} \cap t = \{\}$, that is, only the last state may be terminal. Two legal play sequences δ, δ' of the same length n are called *indistinguishable* for a player $r \in R$ if r's moves and percepts are the same [Schiffel and Thielscher, 2014], that is, for all $i \in \{1, \ldots, n\}$:

- $\mu_i(r) = \mu_i'(r)$;
- $\{p : (r, \mu_i, s_{i-1}, p) \in \mathcal{I}\} = \{p' : (r, \mu_i', s_{i-1}', p') \in \mathcal{I}\}$.

These definitions based on the *objective* game rules about the percepts of players, as given by a GDL-II game description, assume that players have *perfect recall* [Schiffel and Thielscher, 2014].

3 GDL-II + Introspection

The general game description language with imperfect information is expressive enough to model games that give rise to complex epistemic models including players' knowledge of the knowledge of other players [Ruan and Thielscher, 2014]. However, the language does not support any references to players' knowledge in the game rules themselves, for example, in order to specify knowledge goals or to require players to be truthful when asked about their knowledge. In this section, we will extend the syntax of GDL-II by one additional pre-defined language element to facilitate such specifications.

3.1 GDL-III Syntax

We define the syntax of GDL-III as that of GDL-II augmented by a new keyword for *introspection*:

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

The new keyword comes with the following additional, syntactical restrictions on valid GDL-III game descriptions G:

1. `knows` only occurs in the body of clauses, and neither `role` nor `init` depend on `knows`.
2. There is a total ordering $>$ on all predicate symbols `P` that occur as argument of `knows` in G such that `P` $>$ `Q` whenever `P` itself depends on `(knows R Q)` or `(knows Q)` in G.
3. If `P` occurs as argument of `knows` in G then `P` does not depend on `does` in G.

Formally, the new keyword uses the syntactic concept of *reification*, whereby a defined predicate, `P`, is used as an argument of another predicate. While the basic syntax does not support direct nesting within the new keyword, as in `(knows a (knows b p))`, nested knowledge can be expressed with the help of auxiliary predicates, for example, `(knows a kbp)` along with `(<= kbp (knows b p))`. The syntactical restrictions then ensure that nested knowledge is both hierarchical (condition 2) and confined to state-dependent properties (condition 3). The former simply disallows circular definitions, as in `(<= p (knows q))`, `(<= q (knows p))`, while the latter restriction ensures that knowledge only refers to the current state and not to future actions.

Example 1 (cont'd) *With the help of the new keyword, the objective of the number guessing game can be reformulated as a true knowledge goal. This results in a variant of the game that may appropriately be called* NUMBERDETERMINATION. *This is achieved by replacing the termination and goal conditions (rules 32–36 in Figure 1) as follows:*

[3]The pre-defined role `random`, if present, chooses a legal move with uniform probability.

```
37 (<= (num ?n) (true (secret ?n)))
38 (<= terminal (or (knows guesser (num ?n))
39                   (true (step 12))))
40 (<= (goal guesser 100)
41     (knows guesser (num ?n)))
42 (<= (goal guesser 0)
43     (not (knows guesser (num ?n))))
```

This, furthermore, allows us to simplify gameplay in the original game description by removing the final action of explicitly guessing the number, which is no longer needed (rules 17–18, 29–30 can be deleted).

GDL-III uses derived predicates such as `(num ?n)` as arguments of the knowledge predicate—instead of allowing for the use of state features such as `(secret ?n)`—since this enables specifications of non-atomic knowledge conditions and nested knowledge of different players, as in the following example.

Example 2 *In Russian card games [Cordón-Franco* et al.*, 2013], two players cooperate to learn each other's hands through open communication without revealing enough information for a listening third player to know of any of their cards who holds it. The extended expressiveness of GDL-III can be used to specify this as a complex goal for the two cooperating players, namely, that they both know that they know each other's cards* and *that it is* common knowledge *that the third player does not know of any of their cards. The axiomatisation below uses the following domain-dependent predicates for this purpose:*

`(holds R C)`	*player* R *holds card* C
`(kholdsc R S C)`	*player* R *knows that player* S *holds* C
`(kholds1 R S)`	*player* R *knows* some *card of player* S
`(k_all R S)`	*player* R *knows* all *cards of player* S
`(ignorant R S)`	*player* R *does not know* any *card of* S
`(ignorant1 R S)`	*player* R *does not know* all *cards of* S

The intuitive meaning of these predicates is formalised through the following rules of a GDL-III description suitable to define the goal in Russian card games:

```
(<= (kholdsc ?r ?s ?c)(knows ?r (holds ?s ?c)))
(<= (kholds1 ?r ?s)    (kholdsc ?r ?s ?c))
(<= (ignorant1 ?r ?s)
    (holds ?s ?c) (not (kholdsc ?r ?s ?c)))
(<= (k_all ?r ?s) (not (ignorant1 ?r ?s)))
(<= (ignorant ?r ?s) (not (kholds1 ?r ?s)))
```

This definition can be used to specify the goal of a cooperating player, let us call the two of them `alice` *and* `bob` *and their opponent* `eve`*, as follows:*

```
 (<= (goal alice 100)
     (knows alice (k_all bob alice))
     (knows bob (k_all alice bob))
     (knows (ignorant eve alice))
     (knows (ignorant eve bob)))
```

Put in words, Alice needs to know that Bob knows all her cards and vice versa while it is common knowledge that Eve does not know any of the cards of either Alice or Bob. It is easy to verify that the predicates used as arguments of the knowledge operator are not defined circularly and hence satisfy the requirement of being hierarchical, e.g. by the ordering `ignorant` $>$ `k_all` $>$ `kholdsc` $>$ `holds`.

We refrain from providing a complete formalisation of a specific instance of Russian card games in this paper and just note that this requires formal rules to define the range of communicative actions available to Alice and Bob.

3.2 GDL-III Semantics

The semantics for the new keyword can be defined in two stages. First, the logical interpretation of the rules according to Definition 1 is extended by incorporating the encoding of a given set $K = \{(\texttt{knows}\ r_1\ p_1)\ \ldots\ (\texttt{knows}\ r_n\ p_n)\}$ of instances of the knowledge predicate. The legality of moves, percepts of players, updated states as well as the termination and goal conditions in GDL-III are all evaluated relative to a given set K.

Definition 2 *The* pre-semantics *of a valid GDL-III game description G is given by*

- $R = \{r\colon G \models \texttt{role}(r)\}$;
- $s_1 = \{f\colon G \models \texttt{init}(f)\}$;
- $t = \{(S,K)\colon G \cup S^{\texttt{true}} \cup K \models \texttt{terminal}\}$;
- $l = \{(r,m,S,K)\colon G \cup S^{\texttt{true}} \cup K \models \texttt{legal}(r,m)\}$;
- $u(M,S,K) = \{f\colon G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \cup K \models \texttt{next}(f)\}$;
- $\mathcal{I} = \{(r,M,S,K,p) : G \cup M^{\texttt{does}} \cup S^{\texttt{true}} \cup K \models \texttt{sees}(r,p)\}$;
- $g = \{(r,v,S,K)\colon G \cup S^{\texttt{true}} \cup K \models \texttt{goal}(r,v)\}$.

Example 1 (cont'd) *Consider* NUMBERDETERMINATION *from above, state* $S = \{\texttt{(secret 9)},\texttt{(step 5)}\}$*, and sets* $K = \{\,\}$ *and* $L = \{\texttt{(knows guesser (num 9))}\}$*. Following rule 38–39, we have that* $(S,K) \notin t$ *because there is no known instance of* `(num ?n)` *in* K *nor has step 12 been reached in* S*. On the other hand,* $(S,L) \in t$ *because in* L *the player knows the target. For the same reason,* $(\texttt{guesser}, 0, S, K) \in g$ *and* $(\texttt{guesser}, 100, S, L) \in g$ *by rules 40–43.*

The second step in the definition of the semantics for GDL-III consists in an inductive characterisation of legal play sequences and their resulting knowledge states. *Common* knowledge is defined using the notion of the *transitive closure* $\sim^+$ of a given family of indistinguishability relations $\sim_r$ (one for every $r \in R$ of a set of roles R). More formally, suppose given a set of legal play sequences and individual indistinguishability relations $\sim_r$, then $\sim^+$ 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.*

- *Empty sequence ε is the only legal play sequence of length 0 and satisfies $\varepsilon \sim_r \varepsilon$, for all $r \in R$, and results in state s_0 and knowledge state K_0, the latter being the smallest set that satisfies*

$$K_0 = \{(\texttt{knows}\ r\ p)\colon\ r \in R,\ G \cup s_0^{\texttt{true}} \cup K_0 \models p\} \\ \cup \{(\texttt{knows}\ p)\colon G \cup s_0^{\texttt{true}} \cup K_0 \models p\}$$

- *For the inductive definition, let δ be a legal play sequence of length $n \geq 0$, then δ followed by μ, written δ, μ, is a legal play sequence of length $n+1$ if*
  - *δ is a legal play sequence of length n resulting in (s_n, K_n) and*
  - *$(\mu(r), s_n, K_n) \in l$ for all $r \in R$.*

  Sequence δ, μ results in state $s_{\delta,\mu} = u(\mu, s_n, K_n)$. For the resulting knowledge state, we first define two sequences (of length $n+1$) to satisfy $\delta, \mu \sim_r \delta', \mu'$ for $r \in R$ iff
  - *$\delta \sim_r \delta'$,*
  - *$\mu(r) = \mu'(r)$, and*
  - *$\{p : (r, \mu, s_n, K_n, p) \in \mathcal{I}\} = \{p' : (r, \mu', s'_n, K'_n, p') \in \mathcal{I}\}$.*[4]

  The knowledge state $K_{\delta,\mu}$ resulting from δ, μ is then obtained as the smallest set that satisfies

$$K_{\delta,\mu} = \{\, \texttt{(knows } r\ p\texttt{)} : \ G \cup s^{\texttt{true}}_{\delta',\mu'} \cup K_{\delta,\mu} \models p \ \textit{ for all } (\delta', \mu') \sim_r (\delta, \mu)\} \cup \{\, \texttt{(knows } p\texttt{)} : \ G \cup s^{\texttt{true}}_{\delta',\mu'} \cup K_{\delta,\mu} \models p \ \textit{ for all } (\delta', \mu') \sim^{+} (\delta, \mu)\} \tag{3}$$

This is well-defined for any GDL-III game description G with hierarchically defined knowledge predicates, in which case $K_{\delta,\mu}$ 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.

Definition 3 can be understood as follows: All players have perfect knowledge of all predicates in the initial state (K_0). The legality of moves and the updated states are determined inductively from the preceding (actual and knowledge) state s_n and K_n, respectively. Two legal play sequences δ, μ and δ', μ' cannot be distinguished by a player 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 μ and μ' 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 states $K_{\delta,\mu}$.

4 Automated Reasoning for GDL-III

In order to be able to play games specified in the extended game description language, any general game-playing system needs an automated reasoning component for evaluating the rules to determine legal moves and compute state updates. Several approaches to building reasoners for GDL and GDL-II have been described [Schiffel and Björnsson, 2013; Thielscher, 2013]. In this section, we build on previous uses of Answer Set Programming (ASP) [Gelfond, 2008] in general game playing [Thielscher, 2009; Möller *et al.*, 2011] to develop the foundations for automated reasoners for GDL-III.

[4]cf. the definition at the end of Section 2

Reasoning With Game Rules The game description language and ASP have essentially the same basic semantics given by the unique stable model (a.k.a. answer set) of a stratified set of logic program rules. Hence, the basic reasoning tasks according to Definition 2 can be easily automated by mapping any given game description into ASP clauses. Since the evaluation of knowledge conditions depends on previous moves and percepts, all state-dependent predicates in a game description are augmented by two additional arguments so that a single ASP can be used to reason about different legal play sequences, `seq(S)`, and different time points, `time(T)`. We define $\mathbb{P}(G)$ to be the ASP-encoding thus obtained from any given set of GDL-III rules G. For example, the ASP-encoding of rule 40–41 in NUMBERGUESSING with knowledge (cf. Section 3.1) is

```
goal(guesser,100,S,T) :-
  knows(guesser,num(N),S,T), seq(S), time(T).
```

We follow the convention of using natural numbers for time, so that `(init F)` is replaced by `true(F,S,0)` in $\mathbb{P}(G)$ and `(next F)` by `true(F,S,T+1)`.

Reasoning About Knowledge In accordance with their semantics, knowledge conditions are evaluated on the basis of the (in-)distinguishability of legal play sequences. The definition in the previous section implies that players can distinguish any two sequences in which at least one of their preceding moves or percepts differ. Otherwise, the two sequences are indistinguishable (predicate `ind`):

```
distinguishable(R,S1,S2,N) :- time(N), T<N,
  does(R,M1,S1,T), does(R,M2,S2,T), M1!=M2.
distinguishable(R,S1,S2,N) :- time(T), T<=N,
  sees(R,P,S1,T), not sees(R,P,S2,T).

ind(R,S1,S2,N) :- role(R), seq(S1), seq(S2),
  time(N), not distinguishable(R,S1,S2,N).

indtrans(S1,S1,N) :- seq(S1), time(N).
indtrans(S1,S3,N) :- ind(R,S1,S2,N),
                     indtrans(S2,S3,N).
```

The last two clauses above encode the transitive closure $\sim^{+}$ of the indistinguishability relation over all roles. It is easy to verify that the encoding of `distinguishable(R,S1,S2,N)` corresponds to the semantics of two sequences `S1` and `S2` of length `N` being distinguishable by player `R` according to the inductive characterisation of $\sim_r$ as per Definition 3.

According to the definition of resulting knowledge states, (3), a condition on an individual role's knowledge is true if the property in question holds in all sequences that are indistinguishable by that player. On this basis, every `(knows R` $p(\vec{x})$`)` can be evaluated according to the schema

```
knows(R,p(x⃗),S,T) :- p(x⃗,S,T), not np(R,x⃗,S,T).
np(R,x⃗,S,T) :- ind(R,S,S1,T), not p(x⃗,S1,T).
```

Put in words, if `S` is the actual play sequence at time `T`, then player `R` knows that $p(\vec{x})$ just in case $p(\vec{x})$ actually holds and it is *not* the case that (predicate `np`) there is a sequence `S1` that `R` cannot distinguish from `S` and in which $p(\vec{x})$ does not hold.

Similarly, according to Definition 3, a property $p(\vec{x})$ is common knowledge if $p(\vec{x})$ holds in all sequences that are in the transitive closure of the indistinguishability relation across all players. Hence, every `(knows` $p(\vec{x})$`)` can be evaluated according to the schema

```
knows(p(x⃗),S,T) :- p(x⃗,S,T), not np(x⃗,S,T).
np(x⃗,S,T) :- indtrans(S,S1,T), not p(x⃗,S1,T).
```

Let $\mathbb{K}(G)$ be the ASP clauses thus obtained. Furthermore, for a given set of play sequences $\mathcal{S}$, let $\mathbb{D}(\mathcal{S})$ be its ASP encoding as facts of the form `does(R,M,S,T)` with a unique identifier `S` for each element in $\mathcal{S}$. The correctness of the encoding $\mathbb{P}(G) \cup \mathbb{K}(G) \cup \mathbb{D}(\mathcal{S})$ then follows from the fact that the clauses $\mathbb{K}(G)$ provide a direct encoding of the definition of legal play sequences and their (in-)distinguishability.

Example: A GDL-III Game Controller The ASP-based reasoning technique can be used to design a *game controller* [Genesereth *et al.*, 2005] for automatically controlling the execution of GDL games G with imperfect information and introspection, as follows.

1. Send all players the game description G.
2. Let $\mathcal{S} = \{\varepsilon\}$; $s = \varepsilon$; $t = 0$.
3. Request a move `M` from each player `R`. The legality of the moves can be verified using the predicate `legal(R,M,`s,t`)` against $\mathbb{P}(G) \cup \mathbb{K}(G) \cup \mathbb{D}(\mathcal{S})$. Let m be the joint move.
4. Let $\mathcal{S} = \{\delta, \mu : \delta \in \mathcal{S},\ \mu \text{ legal in } \delta\}$; $s = s, m$; $t = t + 1$.
5. Repeat steps 3 and 4 until $\mathbb{P}(G) \cup \mathbb{K}(G) \cup \mathbb{D}(\mathcal{S})$ entails `terminal(`s,t`)`.

5 Solving Epistemic Puzzles With GDL-III

Epistemic puzzles are characterised by multiple agents starting off with imperfect, and in many cases asymmetric, knowledge. They draw further conclusions by logical reasoning about each other's (lack of) knowledge in the course of a short sequence of actions, which often merely consists in repeated public announcements of an agent's ignorance until everyone has perfect knowledge.

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 the encoding of epistemic puzzles, which can then be automatically solved using basic reasoners, like ASP-based encoding presented above. With the help of an example that recently acquired public fame [Chang, 2015] we will show how puzzles that centre around knowledge of the knowledge of other agents can be formalised using GDL with introspection in such a way that they can be solved by a mere GDL-III legal reasoner such as the ASP-based game controller presented in the preceding section.

Example 3 *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 2 shows how this puzzle can be described in GDL-III with the help of the new keyword* `knows` *in such a way that* every *legal playout corresponds to a solution and vice versa.*

It is worth noting that the description in Figure 2 is not meant to be actually played by general GDL-III players. Rather, the game rules have been designed so as to allow the use of a mere game controller to solve this puzzle. An alternative formalisation of Cheryl's Birthday as a game with the purpose of testing the strategic abilities of general game-playing systems would require the definition of a goal for the three roles; for example, Albert and Bernard may be defined as winners if they end up knowing the birthday while Cheryl may be awarded if she chooses a date that makes it difficult if not impossible for them.[5]

Having used the language elements provided by GDL-III to formalise Cheryl's Birthday as depicted in Figure 2, however, allows it to be solved merely by finding a legal play sequence. To this end, the ASP-based approach to reasoning about (and controlling the execution of) games developed in the previous section can be used with just a small modification: Rather than maintaining an actual sequence s and checking for its termination, we just need to find, in the answer set of $\mathbb{P}(G) \cup \mathbb{K}(G) \cup \mathbb{D}(\mathcal{S})$, a positive instance of `terminal(S,`t`)`. Any such `S`, in conjunction with the corresponding move sequence encoded in $\mathbb{D}(\mathcal{S})$, determines a solution. For CHERYLSBIRTHDAY with the standard 10 choices [Chang, 2015], there is only one such legal play sequence after step 4, and `cheryl`'s initial move gives the unique solution (`jul 16`).

6 Experimental Results

In order to test how an ASP-based controller for GDL-III games scales, we ran two different sets of experiments with the example games in this paper. The experiments were carried out on a 2.8 GHz processor with 8 GB of RAM using an off-the-shelf answer set solver.[6] Times are reported in seconds (CPU time).

NUMBERDETERMINATION (cf. Example 1) is a representative of a class of games in which the goal is for players to uncover hidden knowledge. In these games, players typically start off (after one or more unobserved, random moves) with a maximal information set, which then decreases monotonically as the game unfolds and the players acquire further information. We ran experiments with increased ranges of possible numbers and game lengths (i.e. maximal numbers of questions the player can ask). All moves were chosen randomly. The runtime for an ASP-based game controller, averaged over 1,000 runs for each problem size, are summarised in the table below, including the average size of the final information set.

[5]This would also require to provide Albert and Bernard with more move options to ensure that they have a legal move in every reachable state.

[6]`http://potassco.sourceforge.net/`

```
 1 (role albert)
 2 (role bernard)
 3 (role cheryl)
 4
 5 (date may 15)  (date may 16)  (date may 19)
 6 (date jun 17)  (date jun 18)
 7 (date jul 14)  (date jul 16)
 8 (date aug 14)  (date aug 15)  (date aug 17)
 9 (succ 0 1)  (succ 1 2)  (succ 2 3)  (succ 3 4)
10
11 (init (step 0))
12
13 (<= (legal cheryl (choose ?m ?d))
14     (true (step 0)) (date ?m ?d))
15
16 (<= (sees albert  ?m) (does cheryl (choose ?m ?d)))
17 (<= (sees bernard ?d) (does cheryl (choose ?m ?d)))
18
19 (<= (next (secret ?m ?d)) (does cheryl (choose ?m ?d)))
20 (<= (next (secret ?m ?d)) (true (secret ?m ?d)))
21 (<= (next (step ?n)) (true (step ?m)) (succ ?m ?n))
22
23 (<= (legal albert noop)  (or (true (step 0)) (true (step 2))))
24 (<= (legal bernard noop) (not (true (step 2))))
25 (<= (legal cheryl noop)  (not (true (step 0))))
26 (<= (birthday ?m ?d) (true (secret ?m ?d)))
27
28 (<= (knowsDate ?r)   (knows ?r (birthday ?m ?d)))
29
30 (<= (notKnowsDate ?r)(not (knowsDate ?r)))
31
32 (<= (legal albert sayUnknown)
33     (true (step 1))
34     (not (knowsDate albert))
35     (knows albert (notKnowsDate bernard)))
36
37 (<= (legal bernard sayKnown)
38     (true (step 2))
39     (knowsDate bernard))
40
41 (<= (legal albert sayKnown)
42     (true (step 3))
43     (knowsDate albert))
44
45 (<= (sees ?r ?m)
46     (role ?r)
47     (or (does albert ?m) (does bernard ?m)))
48
49 (<= terminal
50     (true (step 4)))
```

Figure 2: CherylsBirthday: a possible description of this puzzle using the syntax of GDL with introspection. Cheryl begins by picking a date (rule 13–14), of which Albert and Bernard only get to see the month and day, respectively (lines 16–17). Using three defined properties (lines 26–30), announcements are modelled by two moves (`sayUnknown`, `sayKnown`) whose preconditions require players to be truthful (lines 32–43) and which are public (rule 45–47).

numbers	max rounds	avg #models	avg time
100	12	12	0.00
333	25	19	0.01
1000	50	35	0.08
3333	100	55	1.36
10000	200	64	23.36

From the game description in Figure 1 it is clear that a state in NumberGuessing consists of at most two features, independent of the problem size. Still, the results demonstrate that the average time to update a state (i.e. the average overall time divided by the length of a game) increases with the problem size since GDL-III requires game controllers to maintain a knowledge state in addition to the actual state. This is different to GDL-II, where the rules for legality, update and termination depend only on the game state itself and hence do not require reasoning with information sets to run a game.

CherylsBirthday (cf. Example 3) is representative of the class of epistemic puzzles. Their solution with the help of the language GDL-III and a mere game controller typically requires reasoning about players' knowledge of both the game state and the knowledge of other players. We ran experiments with systematically increased overall numbers of possible dates along with different "months" and "days" these dates are composed of. The original problem consists of 10 dates across 4 different months and 6 different days. We kept a similar ratio as we increased the problem size 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 with an ASP-based solver as described in Section 5 are summarised in the table below for each size and averaged over 1,000 random problem instances.

[7]Cheryl's Birthday would of course be less famous if it did not have a unique solution, but to test the runtime behaviour of a controller, problems with zero or multiple solutions are equally relevant.

#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

Again, from the game description in Figure 2 it is clear that a state in CherylsBirthday consists of at most two features, independent of the problem size. Note also that, unlike for the results in NumberGuessing, the length of a game is constant too (4 steps). The results demonstrate how the average time to find a legal play sequence 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.

7 Related Work and Conclusion

The extended general game description language GDL-III shares with its predecessor GDL-II the fact that all game rules are objective in the sense that they specify the observations that players make. Thus the axiomatisation of epistemic games does not require rules that specify how percepts affect the knowledge of players. Rather, this follows implicitly from the execution model. The semantics of the knowledge operator in GDL-III uses the same concept of indistinguishability of play sequences that has been 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 extended expressiveness of the new language is manifest in the definition of its semantics, which is considerably more involved than that for its predecessor. State transition systems that provide the full semantics for GDL-II act merely as the pre-semantics for GDL-III, which then requires the interleaved incorporation of (nested and common) knowledge in order to provide the full semantics. This self-reference, where knowledge feeds back into the definition of legal play sequences, is not present in GDL-II.

The general game description language shares with other logic-based knowledge specification languages for actions and change the use of atomic state features in conjunction with precondition and effect axioms. It has been shown, for example, that the Situation Calculus with knowledge [Scherl and Levesque, 2003] can be used to provide an axiomatisation of the concept of (in-)distinguishable play sequences in GDL-II [Schiffel and Thielscher, 2014]. Since the semantics of the knowledge operator in GDL-III is based on the same concept of indistinguishability, this characterisation can be applied to the extended language as well.

Epistemic puzzles similar to Cheryl's Birthday have been solved using model checking systems for Dynamic Epistemic Logic [van Ditmarsch *et al.*, 2005]. The main difference is that this requires an explicit encoding of an epistemic structure in form of a concrete accessibility relation for the problem in question, whereas it suffices to describe the rules of how the environment evolves in a formal description with the syntax of GDL-III in conjunction with the use of a general game controller (cf. Figure 2); the necessary epistemic structure then follows implicitly from the semantics built into the general reasoner. While the proposed solution of epistemic puzzles uses GDL different from its main purpose in general game playing, because no players actually play the game, it is an interesting by-product of having an ASP-based game controller for the new language element. More generally, the formal relation between GDL-II and epistemic logic has been studied in detail [Ruan and Thielscher, 2014]. These existing results carry over to the extended language.

Our experimental results have shown how the size of the information sets of players influences the runtimes of a game controller, unlike in case of GDL-II. While explicitly maintaining the set of relevant legal play sequences, as in our ASP-based encoding, is practically viable in case of typical epistemic games, in which the goal of the players is to reduce an initially maximal information set, other imperfect-information 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].

GDL-III has applications in general game playing beyond the description of games with explicit knowledge conditions. For example, it could be used by imperfect-information players for the automatic construction of logical *strategy* rules by which they condition their moves on their knowledge.

References

[Ågotnes *et al.*, 2013] T. Ågotnes, Harrenstein, W. van der Hoek, and M. Wooldridge. Boolean games with epistemic goals. In *Logic, Rationality, and Interaction*, vol. 8196 of *Springer LNCS*, 1–14, 2013.

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

[Ciancarini and Favini, 2007] Ciancarini and G. Favini. Representing kriegspiel states with metapositions. In *Proc. of IJCAI*, 2450–2455, 2007.

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

[Edelkamp *et al.*, 2012] S. Edelkamp, T. Federholzner, and Kissmann. Searching with partial belief states in general games with incomplete information. In *Proc. of the German Annual Conference on AI*, vol. 7526 of *Springer LNCS*, 25–36, 2012.

[Gelfond, 2008] M. Gelfond. Answer sets. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, 285–316. Elsevier, 2008.

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

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

[Long *et al.*, 2010] J. Long, N. Sturtevant, M. Buro, and T. Furtak. Understanding the success of perfect information Monte Carlo sampling in game tree search. In *Proc. of AAAI*, 134–140, 2010.

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

[Möller *et al.*, 2011] M. Möller, M. Schneider, M. Wegner, and T. Schaub. Centurio, a general game player: Parallel, Java-, and ASP-based. *KI—Künstliche Intelligenz*, 25:17–24, 2011.

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

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

[Schiffel and Björnsson, 2013] S. Schiffel and Y. Björnsson. Efficiency of gdl reasoners. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(4):343–354, 2013.

[Schiffel and Thielscher, 2014] S. Schiffel and M. Thielscher. Representing and reasoning about the rules of general games with imperfect information. *Journal of AI Research*, 49:171–206, 2014.

[Schofield and Thielscher, 2015] M. Schofield and M. Thielscher. Lifting model sampling for general game playing to incomplete-information models. In *Proc. of AAAI*, 3585–3591, 2015.

[Thielscher, 2009] M. Thielscher. Answer set programming for single-player games in general game playing. In *Proc. of ICLP*, vol. 5649 of *Springer LNCS*, 327–341, 2009.

[Thielscher, 2013] M. Thielscher. Filtering with logic programs and its application to general game playing. In *Proc. of AAAI*, 2013.

[van Ditmarsch *et al.*, 2005] H. van Ditmarsch, J. Ruan, and R. Verbrugge. Model checking Sum and Product. In *Proc. of the Australasian Joint Conference on AI*, vol. 3809 of *Springer LNCS*, 790–795, 2005.

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