Noname manuscript No.
(will be inserted by the editor)

GDL-II

Michael Thielscher

Abstract The Game Description Language (GDL) used in the past AAAI competitions allows to tell a system the rules of arbitrary finite games that are characterised by perfect information, but does not extend to games in which players have asymmetric information, e.g. about their own hand of cards, or which involve elements of chance like the roll of dice. Accordingly, contemporary general game-playing systems are not designed to play games such as Backgammon, Poker or Diplomacy. GDL-II (for: GDL with Incomplete/Imperfect Information) is a recent extension of the original description language that makes general game playing truly general, because it allows to describe just any finite game with arbitrary forms of randomness as well as imperfect/incomplete information. This brings along the challenge to build the next generation of truly general game-playing systems that are able to understand any game description given in GDL-II and to learn to master these types of games, too.

Keywords General game playing · Knowledge representation · Imperfect information games

1 Introduction

The execution model underlying the Game Description Language (GDL) assumes that the players are immediately informed about each other's moves and, hence, that all players have complete knowledge of the current position throughout the game. While this is suitable for a variety of classical games such as Chess, Go, Chinese Checkers etc., this excludes games with elements of chance like Backgammon, games with information asymmetry such as Bridge or Poker, and games which involve private communication among cooperating players like in Bughouse Chess, or in the form of negotiations like in Diplomacy. Moreover, envisaged applications for General Game Playing systems, like automated trading agents, are usually characterised by imperfect information.

GDL-II is a recent extension of the original input language by which the in-built restrictions are overcome. The following gives a brief overview of the additional elements in this language and of the modified execution model that these necessitate. Several examples are presented to highlight just how general GDL-II is, and a brief overview is given of the challenges that are raised for contemporary general game-playing systems by the now truly universal game description language.

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

M. Thielscher
School of Computer Science and Engineering, The University of New South Wales, Sydney NSW 2052, Australia
E-mail: mit@cse.unsw.edu.au

2 From GDL to GDL-II

GDL-II differs from GDL by just two additional keywords:

keyword	meaning
`random`	a role that plays randomly
`(sees r p)`	role `r` perceives `p`

Intuitively, `random` denotes a special player (often called "nature" in game theory) of which it is assumed that it always makes a purely random choice among its legal moves in any given position. This allows to model games with elements of chance, such as rolling dice or shuffling cards.

Keyword `(sees r p)` is meant for game rules to specify what information players get under which conditions. There are no limits as to the type of percepts;

it may be a specific detail about the current position, say, or partial knowledge about past moves of the other players, or any other information that the game designer wishes to specify.

Example: Rolling a Die

The following rules specify the roll of a fair die. All players get to observe this move.

```
(role random)

(<= (legal random (roll 1))
    (true (control random)))
(<= (legal random (roll 2))
    (true (control random)))
...
(<= (legal random (roll 6))
    (true (control random)))

(<= (next (die ?n))
    (does random (roll ?n)))

(<= (sees ?p ?m)
    (role ?p)
    (distinct ?p random)
    (does random ?m))
```

By definition, the probability is always uniformly distributed over all legal moves by the random player. This does not necessarily mean, however, that all resulting positions have equal probability. For example, tossing an unfair coin that shows head just with probability $\frac{1}{3}$ can be specified by three legal actions for `random`, two of which have the same effect (the coin showing tails).

Example: Dealing Cards

Most card games include one or more rounds in which cards are randomly dealt to individual players. The following rules specify the dealing of one card each to two players. Both players get to see their own but not their opponent's card.

```
(role alice)
(role bob)
(role random)

(card 2_of_hearts)
...
(card ace_of_spades)

(<= (legal random (deal ?c ?d))
    (true dealing_round)
    (card ?c)
    (card ?d)
    (distinct ?c ?d))

(<= (sees alice ?c)
    (does random (deal ?c ?d)))
(<= (sees bob ?d)
    (does random (deal ?c ?d)))
```

Example: Kriegspiel

Kriegspiel is standard chess without the rules that the players get to see each other's moves [3]. In order to play this game effectively, an arbiter is needed who collects all moves and informs the players whenever they intend to make an invalid move:

```
(<= (sees ?r bad_move_try_again)
    (does ?r ?m)
    (not (valid_move ?m)))

(<= (sees black your_move_now)
    (does white ?m)
    (valid_move ?m))
(<= (sees white your_move_now)
    (does black ?m)
    (valid_move ?m))
```

where the game-specific predicate `(valid_move ?m)` should be specified as test whether `?m` is a correct chess move in the current position. It is important to note the difference between *legal* and *valid* moves here: each attempt to make a move is considered legal, but only those chess moves that are actually possible in the current position are accepted as valid.

Example: Incomplete Information

Game theorists speak of games of incomplete information if players do not know the exact rules of a game, for example, when they do not know the game's payoffs precisely. This can be modelled by an initial, unobservable move in which the random player chooses between different sets of rules, as in the following excerpt from a description of a game inspired by [2].

```
(<= (legal random choose_game1)
    (true (step 1)))
(<= (legal random choose_game2)
    (true (step 1)))

(<= (next defendant_is_liable)
    (does random choose_game1))
(<= (next defendant_is_liable)
    (true defendant_is_liable))

(<= (goal plaintiff 100)
    (true trial)
    (true defendant_is_liable))
(<= (goal plaintiff 20)
    (true trial)
    (not (true defendant_is_liable)))
```

Example: Communication and Negotiation

Coloured Trails is a class of games that is a popular research test-bed for decision-making and negotiation in a competitive setting [1]. Each specific game comes

with one or more fixed protocols defining possible interactions among the players. For example, a simple negotiation may consist of player `?r1` offering player `?r2` to trade one of its chips `?c` for another one, `?d`. This is described by the following rules.

```
(<= (legal ?r1 (send ?r2 (offer ?c ?d)))
    (true (has ?r1 ?c))
    (true (has ?r2 ?d))
    (distinct ?r1 ?r2))

(<= (sees ?recipient (message ?sender ?m))
    (does ?sender (send ?recipient ?m)))
```

Under these rules, the communication is private: only the recipient gets to see the offer in this example.

3 An Execution Model for GDL-II

The additional elements in GDL-II and the modified semantics require an execution model that is suitable for games with arbitrary information asymmetry. Starting in the initial position, in each state each player chooses one of his legal moves. As a consequence the game state is updated according to the rules for keyword `(next ?f)`. In contrast to the execution model for GDL, however, the players are *not* informed about the joint move; rather each role `r` perceives any `?p` for which `(sees r ?p)` is derivable under the current position and the joint move. This continues until a terminal state is reached, and then the goal relation determines the result for each player.

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

1. Send each player the GDL-II description and inform them about their individual roles. Set $S :=$ 'initial_state'.
2. After the appropriate time, collect the individual moves from each player, and if `random` is a role in the game then select a legal move randomly (with uniform probability). Set $M :=$ 'joint move'.
3. Send to each player all percepts that he is entitled to—according to the game rules—when moves M are taken in state S.
4. Set $S :=$ 'update S according to joint move M'.
5. If S is non-terminal, goto step 2. If S is terminal, determine the payoffs for all players `r` by the rules for `(goal r ?n)`.

The Game Master knows all moves and hence can always compute all percepts and also determine the end of a match and the resulting goal values for the players.

4 The General Game Playing-II Challenge

One of the reasons why interest in general game playing has not been growing as rapidly as it could is that it has been restricted to finite, discrete, complete information games. The expectation with the extension to GDL-II is that general game playing will be applicable to a larger set of AI researchers. This also requires to address a fundamentally new challenge for existing general game-playing systems, which comes with the incorporation of nondeterminism and imperfect information: Under the standard execution model, at any state of the game a player is able to infer the complete current position given complete information about both the initial state and all moves. Under imperfect information, however, all that a player can do in general is to maintain an *information set*, which contains all positions that are possible according to what the player knows. But then even basic tasks, such as deriving the legal moves in the current position or updating this information set, become much more intricate reasoning problems. Moreover, in many games, e.g. Kriegspiel, it is practically impossible to maintain an accurate information set after just a few moves, and players have to find feasible ways of storing what they know about the current position and using this knowledge to decide what to do. Extending existing General Game Playing technology so as to be able to master any GDL-II game thus becomes an interesting and ambitious challenge on its own.

References

1. Grosz, B., Kraus, S., Talman, S., Stossel, B., Havlin, M.: The influence of social dependencies on decision-making: Initial investigations with a new game. In: Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 782–789. New York (2004)
2. Png, I.: Strategic behaviour in suit, settlement, and trial. The Bell Journal of Economics **14**(2), 539–550 (1983)
3. Pritchard, D.: The Encycolpedia of Chess Variants. Godalming (1994)

Michael Thielscher is an ARC Future Fellow and a Professor at The University of New South Wales in Sydney since 2010. He received his PhD in 1994 and his Habilitation in 1997 from Darmstadt University. He then joined Dresden University of Technology as an Associate Professor before he moved to Australia. His current research is in Systems with General Intelligence, Knowledge Representation, Agents, Cognitive Robotics, and Constraint Logic Programming.