Towards a Real-time Game Description Language

Jakub Kowalski[1] and Andrzej Kisielewicz[2]
[1]*Institute of Computer Science, University of Wrocław, Poland*
[2]*Institute of Mathematics, University of Wrocław, Poland*
jakub.kowalski@cs.uni.wroc.pl, andrzej.kisielewicz@math.uni.wroc.pl

Keywords: General Game Playing, Knowledge Representation, Game Description Language, Real-time Games.

Abstract: For the sake of the General Game Playing competition, the Game Description Language (GDL) has been developed as a high-level knowledge representation formalism, able to describe any finite, n-player, turn-based, deterministic, full-information game. The last two restrictions were removed by the later extension called GDL-II. In this paper, we discuss our extension of GDL, called rtGDL, that makes it possible to describe a large variety of games involving a real-time factor. We consider its effectiveness and expressiveness, arguing that this is a promising direction of research in the field of General Game Playing.

1 INTRODUCTION

The aim of *General Game Playing* (GGP) is to develop a system that can play a variety of games with previously unknown rules. Unlike standard AI game playing, where designing an agent requires special knowledge about the game, in GGP the key is to create a universal algorithm performing well in different situations and environments.

The beginning of the GGP field dates from 1968 with the work (Pitrat, 1968) concerning the class of arbitrary chess-like board games. In 2005 General Game Playing was identified as a new Grand Challenge of Artificial Intelligence and from this year the annual International General Game Playing Competition (IGGPC) has taken place to foster and monitor progress in this research area (Genesereth et al., 2005). For a more detailed survey of General Game Playing and International GGP Competition, its history, achievements and actual research challenges, we refer to (Genesereth and Thielscher, 2014).

For the purpose of IGGPC, a new language called the *Game Description Language* (GDL) (Love et al., 2006) was developed. GDL has enough power to describe all turn-based, finite and deterministic n-player games with full information. Playing a game given by such a description requires not only developing a move searching algorithm, but also implementing a reasoning approach to understand the game rules in the sense of computing legal moves, the state update function, and the goal function.

In 2010, GDL-II (for Game Description Language with Incomplete Information), the extension to GDL removing deterministic and full information restrictions was proposed in (Thielscher, 2010), and the claim has been made that GDL is now the complete language, without any need for further extensions.

In our opinion, if GDL is to be a general artificial intelligence language it has still too many restrictions. In particular, one of such restrictions is the requirement that the game has to be turn-based. In (Kowalski and Kisielewicz, 2015), we proposed the extension of GDL called the Real-time Game Description Language (rtGDL) that can represent games involving time-based events, where the exact moment of an action is relevant. This concerns most current video games such as Real-time Strategies, Shooters, Sport Games and Role-playing Games. Research towards playing these games by AI agents did not enter (yet) a phase of generalization, and it is focused on using a specially designed algorithms (Ontanón et al., 2013; Hingston, 2010).

This paper is to argue that the properties of rtGDL make it a potentially useful and interesting direction of research in the GGP domain. No real number arithmetic is required inside the rule engine, which unlike some other proposed extensions, e.g. (Thielscher and Zhang, 2010), preserve the pure declarative character of GDL, and allows to re-use the same reasoning engine as in plain GDL. We show that this extension ensures finite derivability and effectiveness required for practical use. We address the problem of possible lags during the communication, and discuss the expressiveness of the language.

2 REAL-TIME GDL

GDL is a high-level, strictly declarative language using logic programming-like syntax. Every game description contains declarations of player roles, the initial game state, legal moves, state transition function, terminating conditions and the goal function. GDL does not provide any predefined functions: neither arithmetic expressions nor game-domain specific structures like board or card deck. Every function and declaration must be defined explicitly from scratch, and a few provided keywords serve as a minimal set of instructions required to grant to the rules a proper semantic. This ensures that, given the game rules, any additional, background knowledge is not required for the player.

A surprisingly simple, but very powerful GDL-II extension proposed by Thielscher allows nondeterministic, imperfect information games, by adding only two additional keywords, and slightly changing the communication protocol.

The extension of GDL presented in (Kowalski and Kisielewicz, 2015) is intended to remove another important restriction of GDL games that they need to be turn-based. It preserves the purely declarative style, so that the state computing inference engine can remain unchanged. In general, dealing with real-time games may involve real number arithmetic, which cannot be encoded in pure GDL in a simple way. The proposed approach requires real arithmetic on the level of search engine reasoning only. On the level of the rule engine, numbers are treated like standard terms without additional semantics (in a manner similar to natural numbers in the `goal` relation).

Changes between rtGDL and GDL keywords are summarized in Table 1. Keywords that are not listed: `role`, `legal`, `does`, `terminal` and `goal` remain unchanged. Parameter $\mathtt{T} \in \mathbb{R}^+ \cup \{\infty\}$ linked with the fact `F` encodes the *lifetime* of this fact, that is, the time for which this fact will be true. After the time is over, the state update is performed and the additional relation `expired(F)` indicates that `F` have just expired and allows changes based on this knowledge.

Table 1: The rtGDL keywords: the top three are modified GDL keywords, while the last two are the new keywords

`init(T,F)`	`F` lasts for time `T` in the initial state
`true(T,F)`	`F` lasts for time `T` in the current state
`next(T,F)`	`F` lasts for time `T` in the next state
`infinity`	stands for ∞ time value
`expired(F)`	holds when `F` becomes obsolete

State updates are performed after a player makes a move or when some fact becomes obsolete. Before such an update is computed, all times assigned to the set of holding facts are properly updated according to the time since the last update.

Keywords restrictions taking into account their intended meaning remain the same as in standard GDL (Love et al., 2006), with the addition that the new keyword `expired` is restricted in the same way as `true`. Also, we adopt the convention that to be considered as *valid*, rtGDL game description must be *stratified* (Apt et al., 1988), *allowed* (Lloyd and Topor, 1986), and satisfy the general *recursion restriction* (see (Schiffel and Thielscher, 2014, Definition 3.)). These restrictions ensure that game rules, despite introducing time-based values, which can be arbitrary real numbers, can be effectively and unambiguously interpreted by a state transition system, and all relevant derivations remain finite and decidable (see Theorem 1).

2.1 Semantics

We show that rtGDL may be seen as a straightforward extension of GDL. As in GDL, any game description defines a finite set of domain-dependent function symbols and constants, which determines a (usually infinite) set of ground symbolic expressions Σ. These ground terms can represent players, moves, parts of the game state, etc. However, unlike in the standard GDL, the game states are not simply subsets of Σ. Every individual feature of the game state has assigned a *lifetime*, which is a real number value or the infinity symbol, and describes the period of time for which the feature holds (after that, unless other events occur, it ceases to be a part of the state). Moreover, initially declared real numbers which occur in Σ are not the only ones that may appear. Any real number, given by the outside environment as updated lifetime, can appear as a constant later during the game.

Therefore, we define an rtGDL game state to be a finite subset of $\mathbb{R}^\infty_+ \times \Gamma$, where $\mathbb{R}^\infty_+ = (0, \infty]$ (including ∞), and $\Gamma \subseteq \Sigma(\mathbb{R}^\infty_+)$ where $\Sigma(\mathbb{R}^\infty_+)$ is a set of ground terms of the game, with the set of constants extended by $\mathbb{R}^\infty_+$. To define the declarative semantics as a state transition system we follow (Kowalski and Kisielewicz, 2015). First, the elements of the game model are endowed suitably with the component $\mathbb{R}^\infty_+$.

- $R \subseteq \Sigma$ (the *roles*);
- $s_0 \subseteq \mathbb{R}^\infty_+ \times \Sigma$ (the *initial position*);
- $t \subseteq \mathcal{P}_{\text{fin}}(\mathbb{R}^\infty_+ \times \Gamma)$ (the *terminal positions*);
- $l \subseteq R \times \Gamma \times \mathcal{P}_{\text{fin}}(\mathbb{R}^\infty_+ \times \Gamma)$ (the *legality relation*);
- $u : \mathbb{R}^\infty_+ \times (R \nrightarrow \Gamma) \times \mathcal{P}_{\text{fin}}(\mathbb{R}^\infty_+ \times \Gamma) \mapsto \mathcal{P}_{\text{fin}}(\mathbb{R}^\infty_+ \times \Gamma)$ (the *update function*);
- $g \subseteq R \times \mathbb{N} \times \mathcal{P}_{\text{fin}}(\mathbb{R}^\infty_+ \times \Gamma)$ (the *goal relation*);

(where $\mathcal{P}_{\text{fin}}(A)$ denotes the finite subsets of A). The legality relation $l(r,m,S)$, states that in the position S, a player r can perform move m. Given a time Δt since the last state update, and a partial function of the joint moves performed by the players $M : (R \nrightarrow \Gamma)$, the state update function $u(t,M,S)$ determines the updated position. Finally, the relation $g(r,n,S)$ defines the payoff of player r in the game state S.

Let $\mathcal{G}$ be a valid game description of an n-player rtGDL game. The players' roles are defined by the derivable instances of `role(R)`. Other game elements have a real-time factor. A *game state* $S = \{(t_1,f_1),\ldots,(t_k,f_k)\}$ is encoded as a set of facts (ground terms) with assigned lifetimes. At the beginning of the game, the state is composed of the derivable instances of `init(T,F)`. We use the keyword `true` to extend the game description $\mathcal{G}$ by the facts resulting from S, which form the set

$$S^{\texttt{true}} \stackrel{\text{def}}{=} \{\texttt{true}(t_1,f_1),\ldots,\texttt{true}(t_k,f_k)\}.$$

If `terminal` is derivable from $\mathcal{G} \cup S^{\texttt{true}}$, then the state S is a terminal position. Instances of `goal(R,N)` derivable from $\mathcal{G} \cup S^{\texttt{true}}$ assign to every player `R` its reward `N`. Legal actions are derivable instances of `legal(R,M)`, for every role `R` and every move `M`.

The state of the game remains unchanged until an *event* occurs. There are two types of events: *action-based*, caused by players' moves, and *expired-based*, resulting from expiration of some facts in $S^{\texttt{true}}$. Let Δt be the time since the last occurrence of an event. When a subset of players $r_{i_1},\ldots,r_{i_l}$, where $\forall j\ i_j \in \{1,\ldots,n\}$, perform moves $m_{i_1},\ldots,m_{i_l}$, then

$$M^{\texttt{does}} \stackrel{\text{def}}{=} \{\texttt{does}(r_{i_1},m_{i_1}),\ldots,\texttt{does}(r_{i_l},m_{i_l})\}.$$

Let us notice that $M^{\texttt{does}} = \emptyset$ if no player sends a move.

Performing state update requires updating lifetime of every holding fact, and information about facts that expired. These two sets are defined in a following way:

$$\mu(S,\Delta t) = \{\texttt{true}(t_i - \Delta t, f_i) : \texttt{true}(t_i,f_i) \in S^{\texttt{true}} \wedge t_i > \Delta t\}$$

$$S^{\text{exp}}_{\Delta t} \stackrel{\text{def}}{=} \{\texttt{expired}(f_i) : \texttt{true}(t_i,f_i) \in S^{\texttt{true}} \wedge t_i \leq \Delta t\}$$

where $\mu(S,\Delta t)$ updates the facts lifetimes and $S^{\text{exp}}_{\Delta t}$ contains expired facts. The updated position is composed of instances of `next(T,F)` derivable from $\mathcal{G} \cup M^{\texttt{does}} \cup \mu(S,\Delta t) \cup S^{\text{exp}}_{\Delta t}$.

In order to summarize above in a formal definition, as in (Schiffel and Thielscher, 2014), we make use of the fact that any stratified set of clauses $\mathcal{G}$ has a unique *stable* model (Gelfond and Lifschitz, 1988). We denote it $\text{SM}[\mathcal{G}]$. Due to the syntax restriction in rtGDL, it is finite ((Love et al., 2006)).

Definition 1. *Let $\mathcal{G}$ be a valid rtGDL specification, whose signature determines the set of ground terms Σ, and $\Gamma \subseteq \Sigma(\mathbb{R}^{\infty}_{+})$. The semantics of $\mathcal{G}$ is the state transition system (R,s_0,t,l,u,g) given by*

- $R = \{r \in \Sigma : \texttt{role}(r) \in \text{SM}[\mathcal{G}]\}$;
- $s_0 = \{(t,f) \in \mathbb{R}^{\infty}_{+} \times \Sigma : \texttt{init}(t,f) \in \text{SM}[\mathcal{G}]\}$;
- $t = \{S \in 2^{\mathbb{R}^{\infty}_{+} \times \Gamma} : \texttt{terminal} \in \text{SM}[\mathcal{G} \cup S^{\texttt{true}}]\}$;
- $l = \{(r,m,S) : \texttt{legal}(r,m) \in \text{SM}[\mathcal{G} \cup S^{\texttt{true}}]\}$, *for all $r \in R$, $m \in \Gamma$ and $S \in \mathcal{P}_{\text{fin}}(\mathbb{R}^{\infty}_{+} \times \Gamma)$;*
- $u(\Delta t,M,S) = \{(t,f) : \texttt{next}(t,f) \in \text{SM}[\mathcal{G} \cup M^{\texttt{does}} \cup \mu(S,\Delta t) \cup S^{\text{exp}}_{\Delta t}]\}$, *for all $M : (R \nrightarrow \Gamma)$, $S \in \mathcal{P}_{\text{fin}}(\mathbb{R}^{\infty}_{+} \times \Gamma)$, and minimal Δt such that $M^{\texttt{does}} \cup S^{\text{exp}}_{\Delta t} \neq \emptyset$;*
- $g = \{(r,n,S) : \texttt{goal}(r,n) \in \text{SM}[\mathcal{G} \cup S^{\texttt{true}}]\}$, *for all $r \in R$, $n \in \mathbb{N}$ and $S \in \mathcal{P}_{\text{fin}}(\mathbb{R}^{\infty}_{+} \times \Gamma)$.*

The model in the above definition is uncountably infinite, and cannot be used itself as an effective description of the game. Yet, it is *locally finite* in the sense that all stable models involved are finite and effectively computable. This is in spite of that rtGDL games involve a real-time factor and incomplete information (on time when events occur). In other words, our extension preserves the crucial property of "finite derivability", which for rtGDL setting can be specified in the following way. A *state of the game* is the set of all facts (ground terms) holding in a given moment of the game. Each state that can be achieved from the initial state by a finite sequence of moves of the players performed in certain times from the start of the game is called a *reachable state*. The last state occurring before the present state S in this sequence is called the *preceding state* for S (and the given sequence of moves). Note that the same state can be reached by various sequences of moves.

Theorem 1. *Let $\mathcal{G}$ be a valid rtGDL game description. Then, each reachable state is finite and can be effectively computed from any preceding state, given the joint move and the time passed.*

Proof. (Sketch). The proof is similar as in case of GDL and GDL-II, based on results on logic programming. First, since $\mathcal{G}$ is stratified, by (Gelfond and Lifschitz, 1988) it admits a unique stable model $\text{SM}[\mathcal{G}]$ as the declarative semantics, which is the same a the "iterated fixed point" model M_P in (Apt et al., 1988). Moreover, by the allowedness and the general recursion restriction this model is finite (Love et al., 2006; Schiffel and Thielscher, 2010). It can be effectively computed using, e.g., the iterative procedure described in (Apt et al., 1988). This implies, in particular that the initial state is finite and can be effectively computed. Now the proof is by induction. Assuming that a reached state is finite, the joint

move of the players contains a finitely many new real time constants, and therefore the stable model $\mathrm{SM}[\mathcal{G} \cup M^{\mathsf{does}} \cup \mu(S,\Delta t) \cup S^{\mathsf{exp}}_{\Delta t}]$ is finite. It follows that there are finitely many instances of `next(T,F)` derivable from $\mathcal{G} \cup M^{\mathsf{does}} \cup \mu(S,\Delta t) \cup S^{\mathsf{exp}}_{\Delta t}$, and as before they can be effectively computed.

The result above means, in particular, that any finite part of the game tree may be effectively computed on the basis of the moves and times they occur.

3 EXECUTION MODEL

There are various possibilities to handle real-time events in an execution model of rtGDL. The simplest model, we presented in (Kowalski and Kisielewicz, 2015), requires very little deviation from the standard execution model of GDL (Love et al., 2006).

In every state S reached at time t (starting from the initial position s_0), the game waits until the first event occurs. Let t' be the time of that event. Then the game state should be updated according to the function $u(t'-t,M,S)$ and all players should be notified about the update and its time. The process repeats until the game reaches a terminal position, with players' rewards given by the goal relation g.

To decrease the number of messages, and keep the communication protocol similar to standard GDL one, the Game Manager can send to the players only information about actions. Since the players must maintain their own internal clocks anyway, they can use them to keep the track of expiration of facts and update states in connection with expired-based events.

The above simple execution model does not address the problem of possible lags during communication. It may be treated as suitable in the scenario when rtGDL programs compete on the same server, and communication delay may be ignored. Yet, in case of games played through the GGP website, lags may cause practical problems with having up-to-date knowledge about the current state of the game.

One direction to deal with the delay problem is to divide the time axis into small intervals and admit communication only at the end points of these intervals. The intervals should be large enough to provide time for indispensable computation. Another possible approach is to allow for the Game Manager to stop the time of the game in the situations when updating the state requires more time. The choice between models should depend on the possible applications.

Another problem concerns correcting information about the current state of the game in the situation when a message about an action-based event just arrived can make some knowledge obsolete. The point is that an action-based event may change lifetimes of other events.

Consider a message m notifying about an action-based event with timestamp t, received at time $t' > t$. The player, having access to all previous messages, and thus to the move history, has full knowledge about the state s at time t (including lifetimes of all facts). Thus, the player can backtrack to this state and, given m, perform the suitable state update.

Assuming stable time difference between a player and the Game Manager, it is possible for the player, after performing at least one move, to synchronize with the Game Manager and be able to send every later move so that it arrives at the desired time.

Indeed, assume a player sent a move at time t_s. As an answer, he get a `PLAY` message at time t_p, with parameter t_r describing the time when the Game Manager received his move. Now, the player can put $\Delta t = t_r - t_s$, and for every move which should be received by the Game Manager at time t, the player can send it at the time $t - \Delta t$ on its own clock. The estimation Δt can be adjusted to be more reliable as the game continues.

This problem recalls the known practical problem in the GGP competitions of sending moves a little bit earlier to make sure they arrive to the Game Manager at time. Yet, in rtGDL setting, it is more essential to adjust the time as exact as possible.

We may summarize all the above:

Theorem 2. *For each Real-time GDL game taking place through the website, there exists a time delay Δt, such that, for any time t_0, the players are able to compute fully the state of the game at time t_0 with delay not larger than Δt.*

4 EXPRESSIVENESS OF RTGDL

To illustrate how the game rules are constructed, we provide an rtGDL code of a complete *Game of Chicken*. It is a well known game, where two players pretend to be tough, and do not want to swerve from the colliding path waiting for the opponent to do this.

```
 1 role(white).   role(black).
 2 init(infinity,dir(white,straight)).
 3 init(infinity,dir(black,straight)).
 4 init(1.0,timer).
 5
 6 next(T,timer) ⇐ true(T,timer).
 7 next(infinity,dir(R,swerve))
 8   ⇐ does(R,swerve).
 9 next(T,dir(R,D)) ⇐ true(T,dir(R,D))
10   ∧ ¬ does(R,swerve).
11 legal(R,swerve)
12   ⇐ true(T,dir(R,straight)).
```

```
terminal ⇐ expired(timer).
goal(R,0) ⇐ true(T,dir(R,straight))
  ∧ true(S,dir(P,straight))
  ∧ distinct(R,P).
goal(R,80) ⇐ true(T,dir(R,swerve))
  ∧ true(S,dir(P,straight)).
goal(R,90) ⇐ true(T,dir(R,swerve))
  ∧ true(S,dir(P,swerve))
  ∧ distinct(R,P).
goal(P,100) ⇐ true(T,dir(R,swerve))
  ∧ true(S,dir(P,straight)).
```

The game lasts for the one unit of time. Each player starts riding straight ahead, and until the `timer` holds, he player can decide to `swerve`. After swerving, no more legal moves are allowed.

The payoff matrix is defined as follows. In the case of a crash, both players' reward is 0. If only one player decides to swerve his reward is 80, while his opponent's reward is 100. When both players decide to swerve they gain 90 points each.

We now proceed to show that, in principle, all GDL games can be expressed in rtGDL. To this end we need the following definition.

Definition 2. *We say that an rtGDL game $\mathcal{G}'$ has an* equivalent semantics *to a GDL game $\mathcal{G}$ if there is a procedure converting any program for $\mathcal{G}$ to a program for $\mathcal{G}'$, so that the result of any rtGDL competition C' between converted programs is the same as the result of GDL competition C of the original GDL programs.*

The definition could be stated more formally, but it is sufficient for the sketch of the proof provided below.

Theorem 3. *For every GDL game $\mathcal{G}$ there exist real-time GDL game $\mathcal{G}'$ with equivalent semantic.*

Proof. (Sketch). Considering $\mathcal{G}$ as a GDL specification, we convert it, step by step, in an rtGDL specification. First, all occurrences of `true`, `init`, and `next` are modified to fit the rtGDL syntax by endowing each with the lifetime parameter `infinity`. We introduce three new function constants `made(R,M)`, `clock`, and `moved(R)` with intention to postpone the state update until playclock `clock` expires, and to mark down that the player `R` has performed his move. Accordingly, all occurrences of `does(`r`,`m`)` are replaced by `true(infinity, made(`r`,`m`))`, and every rule of the form `next(infinity,`f`)`$\Leftarrow \phi$ is replaced by `next(infinity,`f`)`$\Leftarrow \phi \wedge$ `expired(clock)`. In addition we add a set of new rules:

```
next(infinity,moved(R)) ⇐ does(R,M).
next(infinity,made(R,M))
  ⇐ does(R,M) ∧ ¬ true(infinity,moved(R)).
next(infinity,F)
  ⇐ true(T,clock) ∧ true(infinity,F).
```

The intended meaning of the constant `clock` is ensured by the following:

```
init(1.0,clock).
next(1.0,clock) ⇐ expired(clock).
next(t,clock) ⇐ true(t,clock).
```

The presented code allows the players to perform precisely one move during the unit `clock` interval. The optimal strategy of the players (based on the given strategy for the corresponding GDL game) should be to make use of all the timelimit to compute their single move as in GDL and send it to the Game Manager. Then, the result of the competition will be the same.

Here we need to make a theoretical assumption that, as a result of such strategy, the players send their moves at the same time, at the end of the timelimit, and in consequence, at the same time are informed about their joint move. Another theoretical assumption is that players do not send illegal moves, because the rtGDL Game Manager handles illegal moves in a different way than the GDL one.

In practice, the games $\mathcal{G}$ and $\mathcal{G}'$ would not be fully equivalent, especially when it is essential that players have no knowledge about the moves done by other players during the same turn. Also, the error handling would require a special solution. (The simplest one would be to treat illegal moves as the loss.)

It is natural to compare rtGDL games with Extensive Form Continuous-time games (Simon and Stinchcombe, 1989). Let us recall that in such games the players act on the interval $R = [0,1]$. Let A be the joint set of all possible actions of all players. The *decision node* is a pair $\langle 0, \emptyset \rangle$ or $\langle t, h \rangle$, where $t \in (0,1]$ and h is a function from $[0,t)$ to A, representing the history of the game up to time t. Now, since every rtGDL game has to be finite in terms of match duration, it is always possible to map rtGDL lifetime units to $(0,1]$ such that every match do not go beyond R. The construction is straightforward, and we may see that every rtGDL game can be represented in the form of the Extensive Form Continuous-time game.

However, it should be noted, that rtGDL nodes allows only finite number of legal actions. This restriction adopted to preserve finite derivability property of GDL games is, at the same time, a serious limitation. In particular, it does not allow players to use real numbers in their move messages, which is a natural action in many games.

5 CONCLUSION

With introduction of GDL-II, it was claimed that the GDL language can be considered complete (Thielscher, 2010; Schiffel and Thielscher, 2014), and additional elements can only serve for simplifying description or will be forcing setting extensions far

beyond the concept of General Game Playing (e.g. open-world games or physical games like in General Video Game Playing (Perez et al., 2015)).

Our position is contrary. We argue that our real-time extension (Kowalski and Kisielewicz, 2015) preserves the core idea of GDL – a concise purely logical description of the game and a simple execution model. At the same time it is a larger extension than GDL-II, allowing a game to have an infinite number of states and the players to have an infinite number of actions. By introducing time based events and by giving relevance to the move ordering it makes it possible to describe properly many real world situations. It makes real-time general gaming a very hard, but interesting area of further GGP research. In particular, it allows to model many elements of the popular computer games, which are currently used as a test-bed for dedicated AI players, e.g. Super Mario Bros, Unreal Tournament 2004 (Hingston, 2010), or Starcraft (Ontanón et al., 2013).

Beyond the usage in GGP, the correlations between the GDL and game theory (Thielscher, 2011) and Multiagent Systems (Schiffel and Thielscher, 2010) are often pointed out. A goal for GDL is to become a universal description language which can describe as large class of game-like problems as possible, at the same time remaining compact, high-level and machine-processable. We presented the next step into such a generalization of problems description, which brings the class of games covered by the GDL family closer to real-time game theory (e.g. Extensive Games in Continuous Time, Continuous Time Repeated Games (Bergin and MacLeod, 1993)) and Real Time Multiagent Systems (Julian and Botti, 2004).

Moreover, we think that GDL language family can be further expanded. In particular, it is possible to merge presented Real-time extension with Thielscher's Incomplete Information extension to obtain the broadest general description language so far. The sketch of such construction has been presented in (Kowalski and Kisielewicz, 2015).

ACKNOWLEDGEMENTS

This work was supported by Polish National Science Centre grants No 2014/13/N/ST6/01817 and No 2012/07/B/ST1/03318.

REFERENCES

Apt, K. R., Blair, H. A., and Walker, A. (1988). Foundations of deductive databases and logic programming. chapter Towards a Theory of Declarative Knowledge, pages 89–148.

Bergin, J. and MacLeod, W. B. (1993). Continuous time repeated games. *International Economic Review*, pages 21–37.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080.

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

Genesereth, M. and Thielscher, M. (2014). *General Game Playing*. Morgan & Claypool.

Hingston, P. (2010). A new design for a turing test for bots. In *CIG*, pages 345–350. IEEE.

Julian, V. and Botti, V. (2004). Developing real-time multi-agent systems. *Integrated Computer-Aided Engineering*, 11(2):135–149.

Kowalski, J. and Kisielewicz, A. (2015). Game Description Language for Real-time Games. In *GIGA*, pages 23–30.

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

Love, N., Hinrichs, T., Haley, D., Schkufza, E., and Genesereth, M. (2006). General Game Playing: Game Description Language Specification. Technical Report LG-2006-01, Stanford Logic Group.

Ontanón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., and Preuss, M. (2013). A survey of real-time strategy game AI research and competition in Starcraft. *T-CIAIG*, 5(4):293–311.

Perez, D., Samothrakis, S., Togelius, J., Schaul, T., Lucas, S., Couëtoux, A., Lee, J., Lim, C., and Thompson, T. (2015). The 2014 General Video Game Playing Competition. *T-CIAIG*. To appear.

Pitrat, J. (1968). Realization of a general game-playing program. In *IFIP Congress*, pages 1570–1574.

Schiffel, S. and Thielscher, M. (2010). A Multiagent Semantics for the Game Description Language. In *Agents and Artificial Intelligence*, volume 67 of *CCIS*, pages 44–55.

Schiffel, S. and Thielscher, M. (2014). Representing and Reasoning About the Rules of General Games With Imperfect Information. *JAIR*, 49:171–206.

Simon, L. K. and Stinchcombe, M. B. (1989). Extensive Form Games in Continuous Time: Pure Strategies. *Econometrica*, 57(5):1171–1214.

Thielscher, M. (2010). A General Game Description Language for Incomplete Information Games. In *AAAI*, pages 994–999.

Thielscher, M. (2011). The General Game Playing Description Language is Universal. In *IJCAI*, pages 1107–1112.

Thielscher, M. and Zhang, D. (2010). From general game descriptions to a market specification language for general trading agents. volume 59 of *LNBIP*, pages 259–274.