Towards a Real-time Game Description Language

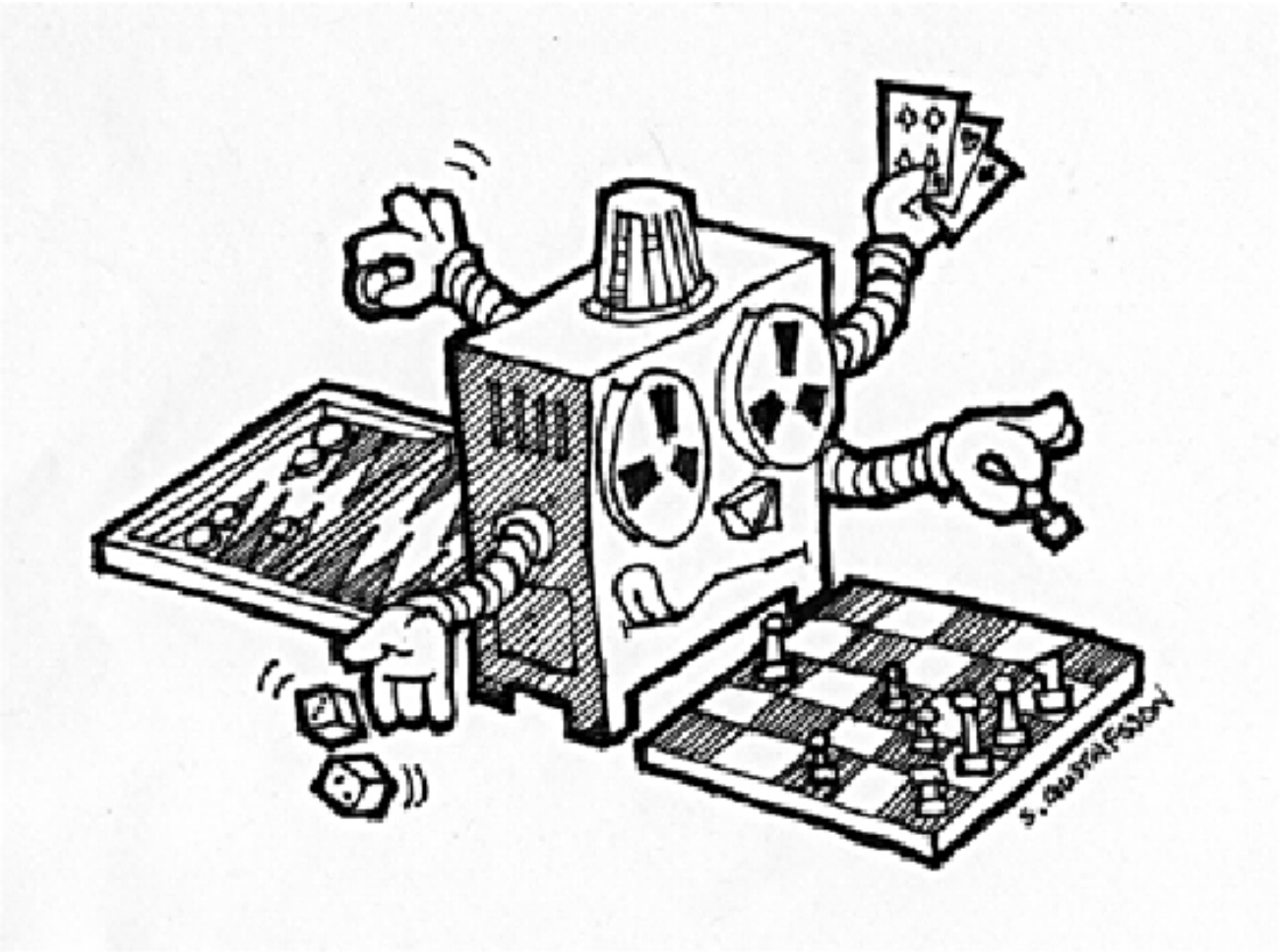
Jakub Kowalski, Andrzej Kisielewicz

ICAART

26 February 2016

General game Playing

GGP robot



What is General Game Playing?

General game playing is the design of artificial intelligence programs to be able to play more than one game successfully.

Wikipedia, The Free Encyclopedia

A General Game Playing System is one that can accept a formal description of a game and play the game effectively without human intervention.

Michael Genesereth, Nathaniel Love; Stanford University

General Game Playing is about playing games that you've never seen before.

Sam Schreiber; ggp.org

Brief history of GGP

- 1968 – Pitrat, J., *Realization of a general game-playing program*

Brief history of GGP

- 1968 – Pitrat, J., *Realization of a general game-playing program*
- 1992 – Pell, B., *METAGAME in Symmetric Chess-Like Games*

Brief history of GGP

- 1968 – Pitrat, J., *Realization of a general game-playing program*
- 1992 – Pell, B., *METAGAME in Symmetric Chess-Like Games*
- 2005 – Genesereth, M.R., Love, N., Pell, B., *General Game Playing: Overview of the AAAI Competition*

Brief history of GGP

- 1968 – Pitrat, J., *Realization of a general game-playing program*
- 1992 – Pell, B., *METAGAME in Symmetric Chess-Like Games*
- 2005 – Genesereth, M.R., Love, N., Pell, B., *General Game Playing: Overview of the AAAI Competition*
- 2014 – Ebner, M., Levine, J., Lucas, S., Schaul, T., Thompson, T., Togelius, J., *Towards a Video Game Description Language*

Game Description Language

Game Description Language (GDL)

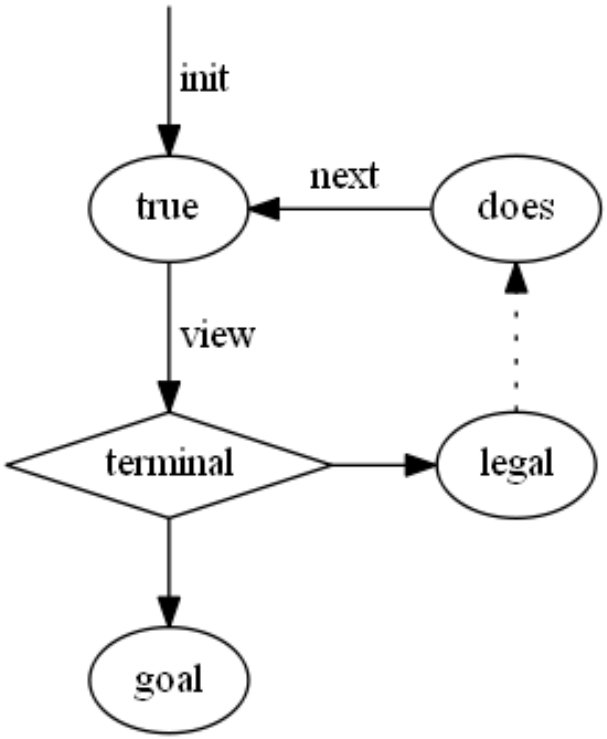
- Strictly declarative, based on the *Knowledge Interchange Format*;
- Finite, deterministic games with full information and simultaneous moves;
- Pure first-order logic: no built-in assumptions, no arithmetic, no physics, no domain specific concepts;
- Prefix notation rules with ? preceded variables.

GDL Keywords

`(role ?r)`	`?r` is a player
`(init ?f)`	fact `?f` is true in the initial state
`(true ?f)`	fact `?f` is true in the current state
`(legal ?r ?a)`	in the current state `?r` can perform action `?a`
`(does ?r ?a)`	`?r` performed action `?a` in the previous state
`(next ?f)`	`?f` will be true in the next state
`terminal`	current state is terminal
`(goal ?r ?n)`	player `?r` score is `?n`

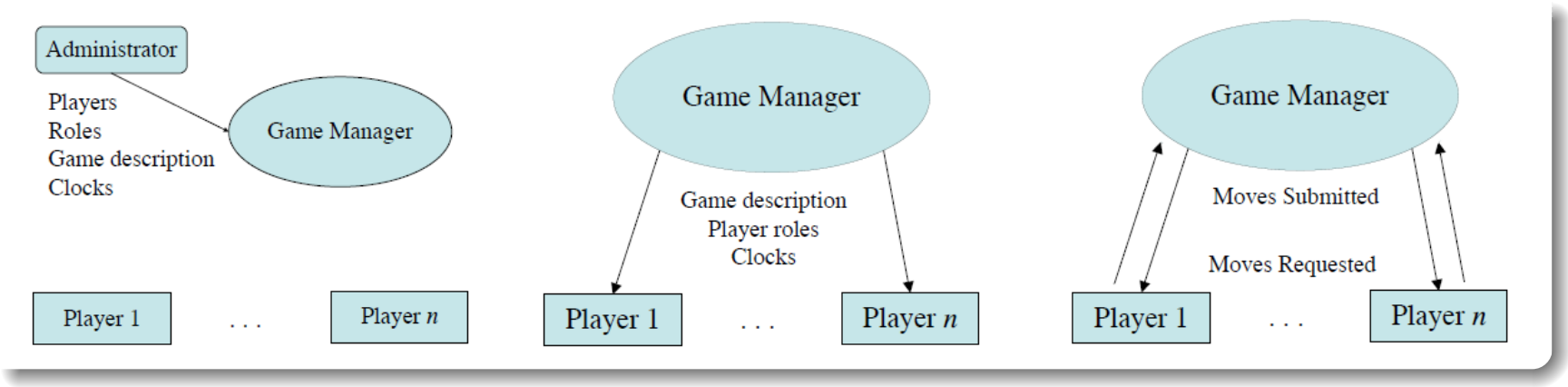
GDL Example: Tic Tac Toe

The execution model



- `(role xplayer)`
- `(init (cell 1 1 b))`
- `(<= (row ?m ?x)`
  `(true (cell ?m 1 ?x))`
  `(true (cell ?m 2 ?x))`
  `(true (cell ?m 3 ?x)))`
- `(<= (legal ?w (mark ?x ?y))`
  `(true (cell ?x ?y b))`
  `(true (control ?w)))`
- `(<= (next (cell ?m ?n x))`
  `(does xplayer (mark ?m ?n))`
  `(true (cell ?m ?n b)))`
- `(<= terminal`
  `(line x))`
- `(<= (goal xplayer 100)`
  `(line x))`

Communication Protocol



Communication

- Players are TCP servers, *Game Manager* is a client application;
- Players connect when they are ready to play;
- Types of messages:
  - `START` – sends game rules, player's role, *startclock* and *playclock*; players can analyze game rules for *startclock* seconds;
  - `PLAY` – sends moves performed by all players; players have to send their moves within *playclock* time limit;
  - `STOP` – informs about the end of the game.

GDL Evolution

GDL (Genesereth, Love, Pell, 2005)

n-player, turn-based, finite, deterministic, full information games

GDL Evolution

GDL (Genesereth, Love, Pell, 2005)

n-player, turn-based, finite, deterministic, full information games

GDL-II (Thielscher, 2010)

n-player, turn-based, finite, ~~deterministic, full information~~ games

GDL Evolution

GDL (Genesereth, Love, Pell, 2005)

n-player, turn-based, finite, deterministic, full information games

GDL-II (Thielscher, 2010)

n-player, turn-based, finite, ~~deterministic,~~ ~~full information~~ games

rtGDL (2015)

n-player, ~~turn-based,~~ finite, deterministic, full information games

GDL Evolution

GDL (Genesereth, Love, Pell, 2005)

n-player, turn-based, finite, deterministic, full information games

GDL-II (Thielscher, 2010)

n-player, turn-based, finite, ~~deterministic,~~ ~~full information~~ games

rtGDL (2015)

n-player, ~~turn-based,~~ finite, deterministic, full information games

rtGDL-II (2015)

n-player, ~~turn-based,~~ finite, ~~deterministic,~~ ~~full information~~ games

Real-time Games Description Language (rtGDL)

What is real-time in games about?

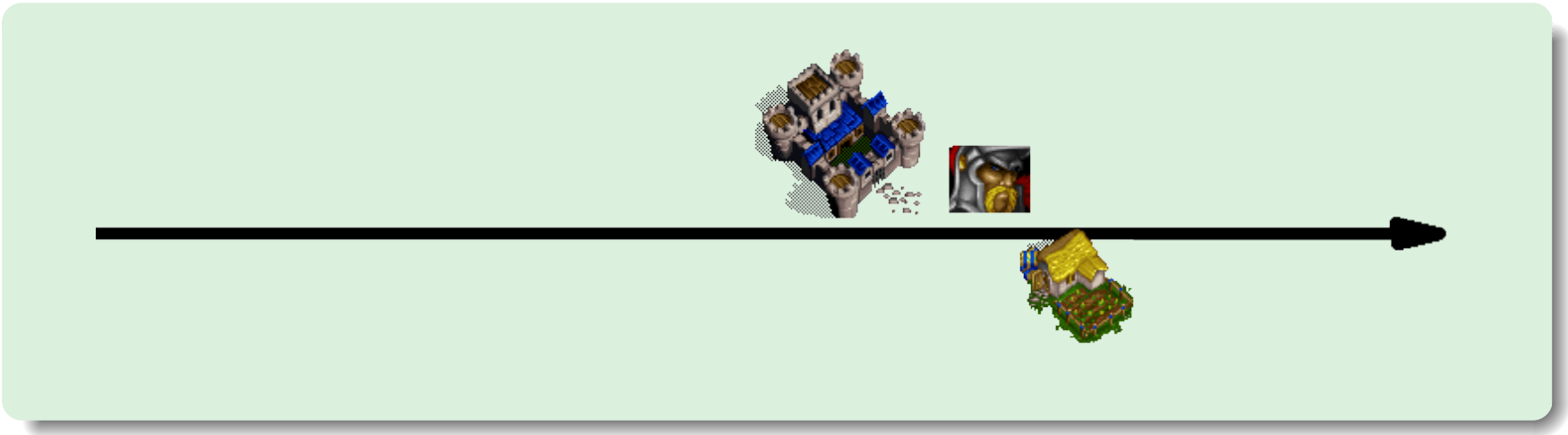
What is real-time in games about?

What is real-time in games about?

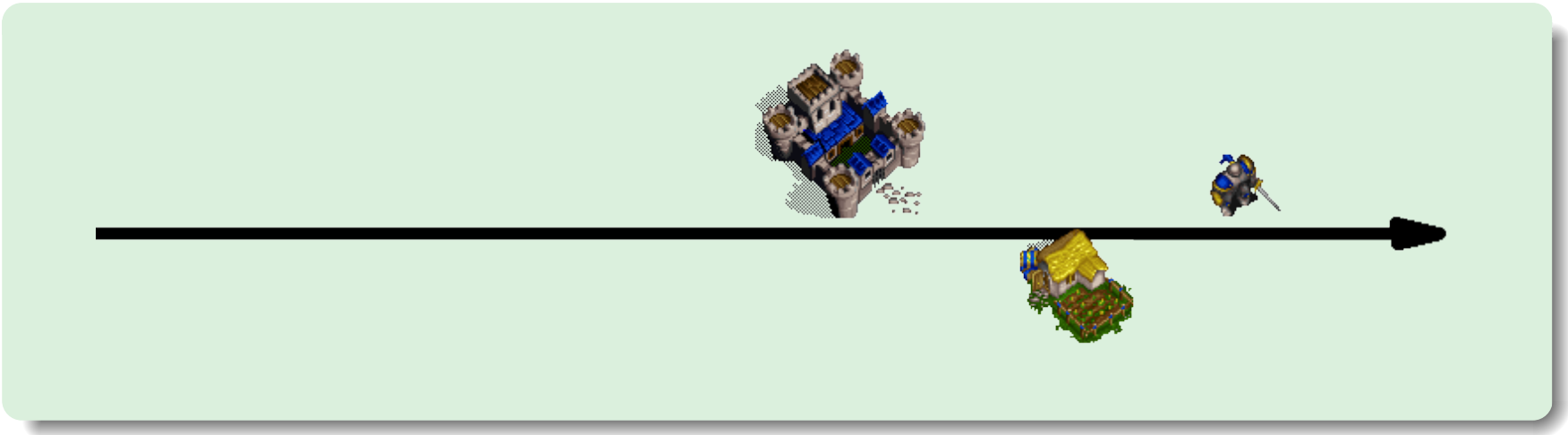
What is real-time in games about?

What is real-time in games about?

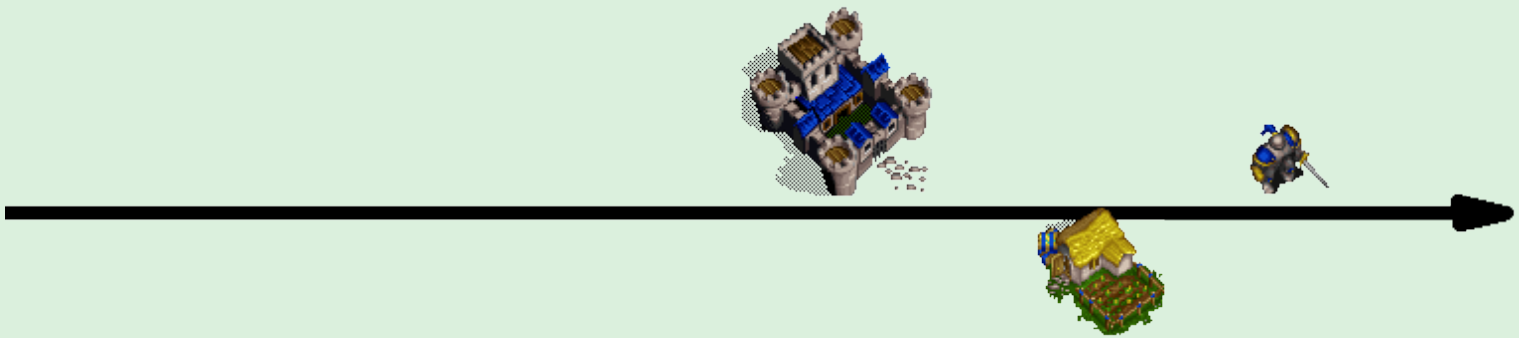
What is real-time in games about?



What is real-time in games about?



What is real-time in games about?



- Players can take actions at any moment. (idealistic)
- Events can have duration time.

Real-time in GDL

Goal

- Preserve purely declarative style
- No arithmetic in rule engine

Real-time in GDL

Goal

- Preserve purely declarative style
- No arithmetic in rule engine
- Reasoners should remain unchanged

Real-time in GDL

Goal

- Preserve purely declarative style
- No arithmetic in rule engine
- Reasoners should remain unchanged

Problem

There have to be real numbers somewhere inside the GDL code.

Real-time in GDL

Goal

- Preserve purely declarative style
- No arithmetic in rule engine
- Reasoners should remain unchanged

Problem

There have to be real numbers somewhere inside the GDL code.

But we already have natural numbers (in goal relation)!

Real-time in GDL

Goal

- Preserve purely declarative style
- No arithmetic in rule engine
- Reasoners should remain unchanged

Problem

There have to be real numbers somewhere inside the GDL code.

But we already have natural numbers (in goal relation)!

Solution

- Treat time as immutable values, only copying is allowed.
- Arithmetic operations are allowed only by the search engine (between state updates).

Keywords

GDL	rtGDL
`role(R)` `init(F)`	

`true(F)`
`legal(R,M)`
`does(R,M)`
`next(F)`	

`terminal` `goal(R,N)`	

Keywords

GDL	rtGDL
role(R)	role(R)
init(F)	init(T,F)
true(F)	true(T,F)
legal(R,M)	legal(R,M)
does(R,M)	does(R,M)
next(F)	next(T,F)
terminal	terminal
goal(R,N)	goal(R,N)
	infinity
	expired(F)

$T \in \mathbb{R}^+ \cup \{\infty\}$
(lifetime)

Sketch of semantics

Event

An *event* occurs when at least one fact becomes obsolete or at least one player performs a move.

Time update

Update of state S after time Δt:

$$\text{Holding facts} : \{\texttt{true}(t_i - \Delta t, f_i) : \texttt{true}(t_i, f_i) \in S \wedge \ t_i > \Delta t\}$$

$$\text{Expired facts} : \{\texttt{expired}(f_i) : \texttt{true}(t_i, f_i) \in S \wedge \ t_i \leqslant \Delta t\}$$

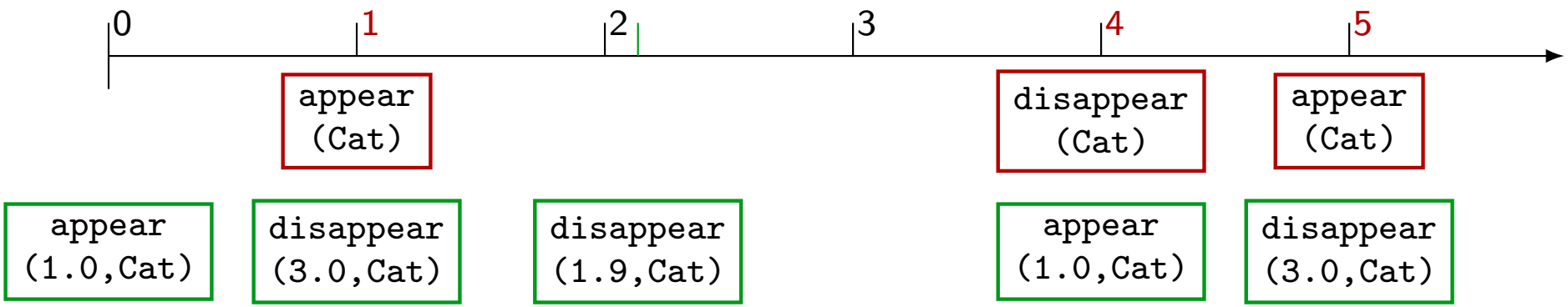
State update

State update is performed with the first occurrence of an event.

Simple example

Self appearing and disappearing object

```
1  init(1.0,appear(cheshireCat))
2
3  next(3.0,disappear(C)) ⇐ expired(appear(C)).
4  next(T,disappear(C))   ⇐ true(T,disappear(C)).
5
6  next(1.0,appear(C)) ⇐ expired(disappear(C)).
7  next(T,appear(C))   ⇐ true(T,appear(C)).
```



rtGDL-II

It is possible to use similar mechanisms as in GDL-II extension.

Nondeterminism

- Add special `random` role which actions are chosen by Game Manager at random.
- If `random`'s action arity > 1 and the first argument is a real number: perform state update after time t' drawn using the $\mathcal{U}(0, t)$ distribution.

Incomplete Information

- Additional `sees` as a predicate for players' percepts.
- Game Manager sends percepts instead of the information about moves.
- We should not give a clue that someone made a move.

Summary and future work

Conclusion

Real-time GDL

- remains in the spirit of GGP,
- preserve pure logical reasoning,
- is the next step into such a generalization of problems description,
- is the largest GDL family language so far,
- opens a new area of GGP research,
- allows to model many elements of the popular computer games,
- can be further extended by merging with GDL-II.

Future work

Practical implementation of rtGDL framework (work in progress)

- Use rationals of determined precision (*lifetime* $\in \mathbb{N}$, floating-point *playclock* as a multiplier)
- Specify communication protocol (silent/verbose expired-based events)
- Reverse client–server architecture in communication model

Formal rtGDL-II model

- Specify communication protocol (player-selected notifications)
- Fit into the rtGDL framework

rtGDL player development

- Extend an MCTS based engine to handle real-time game playing and asynchronous events

Thank you