



UNSW
THE UNIVERSITY OF NEW SOUTH WALES



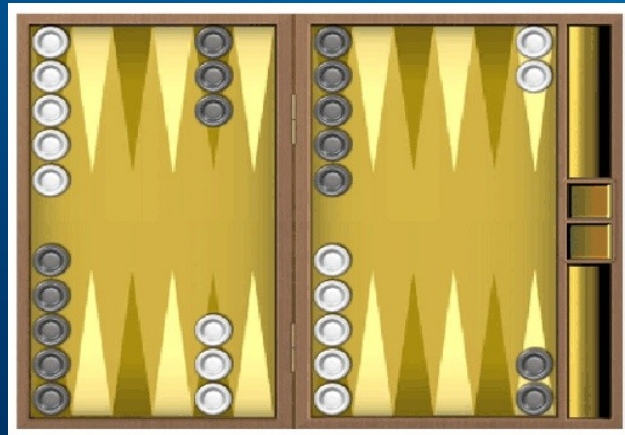
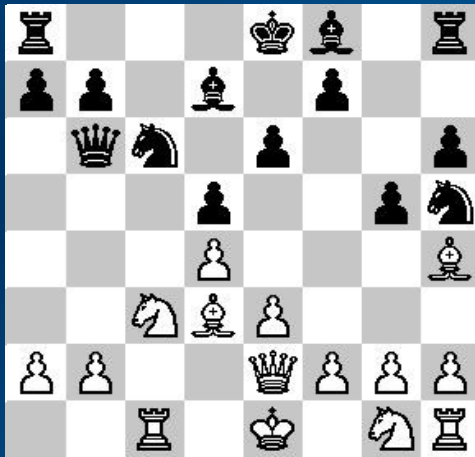
The General Game Playing Description Language is Universal

Michael Thielscher

General Game Playing

A **General Game Player** is a system that

- understands new game descriptions
- plays without human intervention



General Game Playing

New generation of systems that can adapt to new, and possibly radically different, environments.
Allows end users to customize their system.



Outline

- GDL—a general Knowledge Representation language to describe arbitrary n-player games
- Extensive-form games (from game theory)
- Main result: $\text{GDL} \Rightarrow \text{Extensive form}$
 $\text{Extensive form} \Rightarrow \text{GDL}$

Describing Games

Required: universal game description language

Bad:

- Black box as move generator
 - ⇒ doesn't allow systems to **reason** about rules to build their own strategy

Good:

- Purely declarative description

Describing Games

Language **GDL** can describe any game using the syntax of logic programming.



Describing Games

Language **GDL** can describe any game using the syntax of logic programming.

The **execution model** ensures

- all players know the complete rules
- all players know the initial position



Describing Games

Language **GDL** can describe any game using the syntax of logic programming.

The **execution model** ensures

- all players know the complete rules
- all players know the initial position

but

- a designated player moves randomly
- players have individual percepts



Example: Chess

```
role (white) .  
role (black) .  
init (cell (a, 1, white_rook)) .  
init (cell (b, 1, white_knight)) .  
... init (turn (white)) .
```

```
legal (P, castle (Side)) <= true (turn (P)) ,  
                                     can_castle (P, Side) .  
next (cell (g, 1, white_king)) <= does (white,  
                                     castle (king_side)) .  
...  
goal (white, 100) <= checkmate, true (turn (black)) .
```



Example: Chess vs. Kriegspiel

Standard chess requires the rules

```
sees (white,M) <= does (black,M) .  
sees (black,M) <= does (white,M) .
```

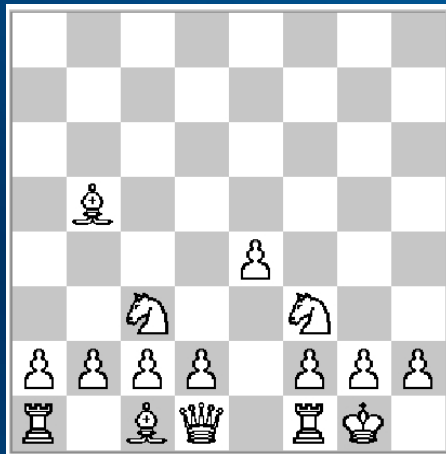


Example: Chess vs. Kriegspiel

Standard chess requires the rules

```
sees (white,M) <= does (black,M) .  
sees (black,M) <= does (white,M) .
```

Omitting these rules gives you **Kriegspiel**



How Expressive is GDL?

GDL can be used to describe **any** (finite) n-player game, including those with

- nondeterministic moves
- information asymmetry



Really?

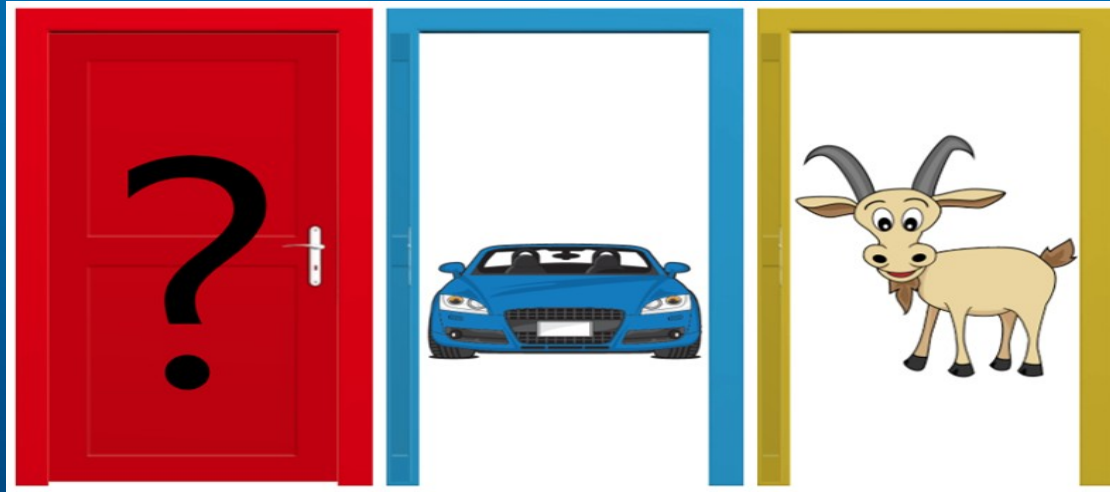
From a review for the AAIL'10 paper:

The paper says that the addition of two keywords suffices to obtain the desired generality. Yet, this claim [...] maybe needs to be softened a bit. For instance, how would it in GDL-II be possible to model a situation where

Agent A knows that p , B knows that A knows p or that A knows not- p , but C considers it possible that A knows nothing about p .



Example 2: Monty Hall



```
role(candidate) .
```

```
role(random) .
```

```
init(closed(1)) .
```

```
init(closed(3)) .
```

```
init(closed(2)) .
```

```
init(step(1)) .
```

Monty Hall (cont'd)

```
% Monty
```

```
legal(random,hide_car(D)) <= true(step(1)),  
                                     true(closed(D)).
```

```
legal(random,open_door(D)) <= true(step(2)),  
                                     true(closed(D)),  
                                     not true(car(D)),  
                                     not true(chosen(D)).
```

```
% Player
```

```
legal(candidate,choose(D)) <= true(step(1)),  
                                     true(closed(D)).
```

```
legal(candidate,noop) <= true(step(3)).
```

```
legal(candidate,switch) <= true(step(3)).
```



Monty Hall (cont'd)

% Percept

```
sees (player, D) <= does (random, open_door (D)) .
```

% Effects

```
next (chosen (D)) <= does (candidate, switch) ,  
                        true (closed (D)) ,  
                        not true (chosen (D)) .
```

```
goal (candidate, 100) <= true (car (D)) ,  
                        true (chosen (D)) .
```

```
goal (candidate, 0) <= true (car (D)) ,  
                        not true (chosen (D)) .
```



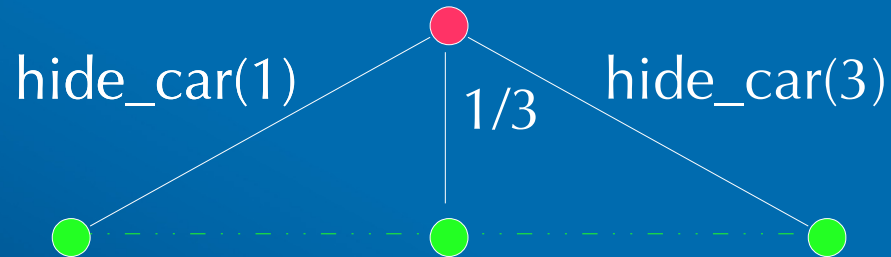
Extensive-form Games

n-player **extensive-form game** consists of

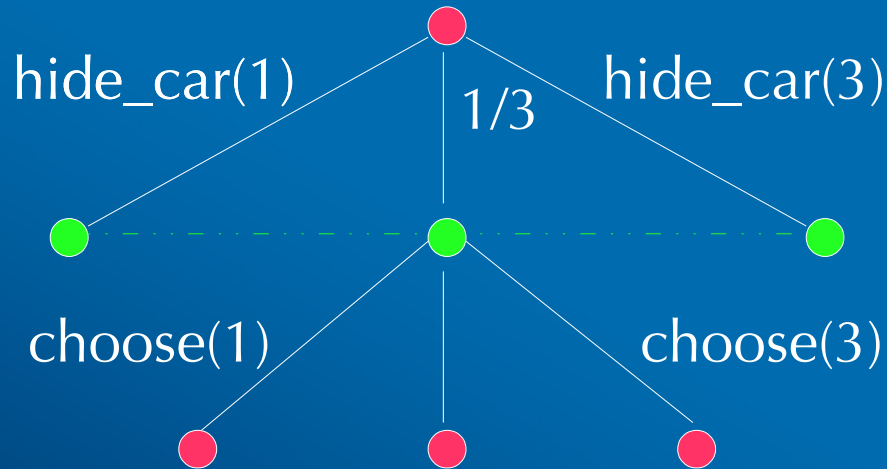
- τ — finite tree
- ι — assignment of nodes to $\{0, \dots, n\}$ (players)
- v — payoff function
- ρ — probability measure for player 0's moves
- η — information partition



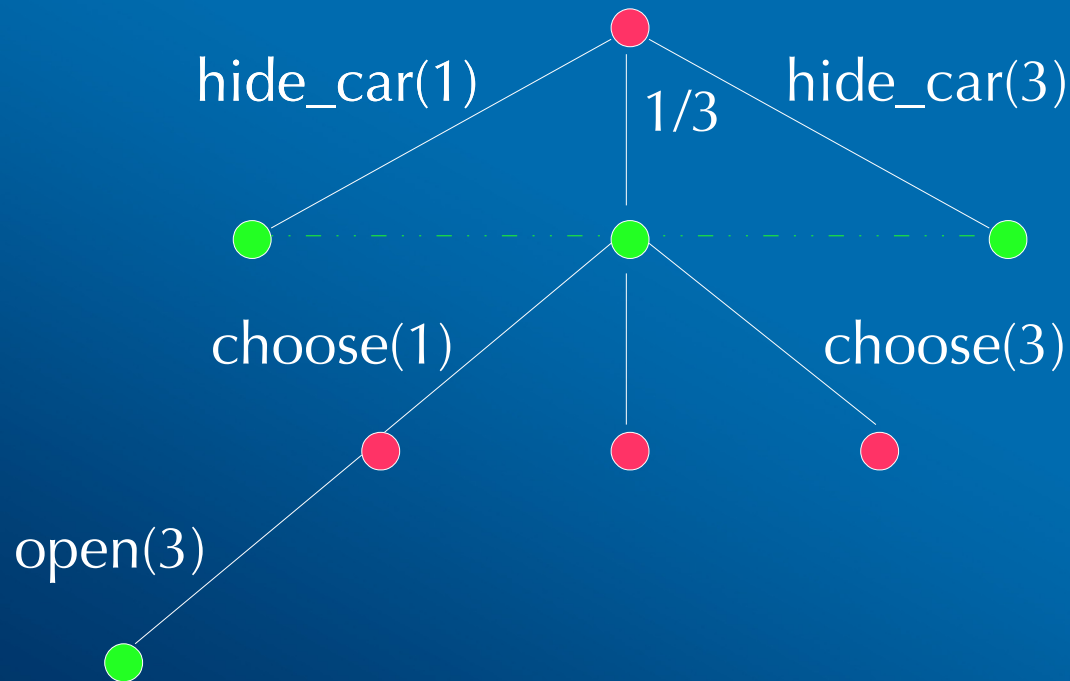
Example: Monty Hall



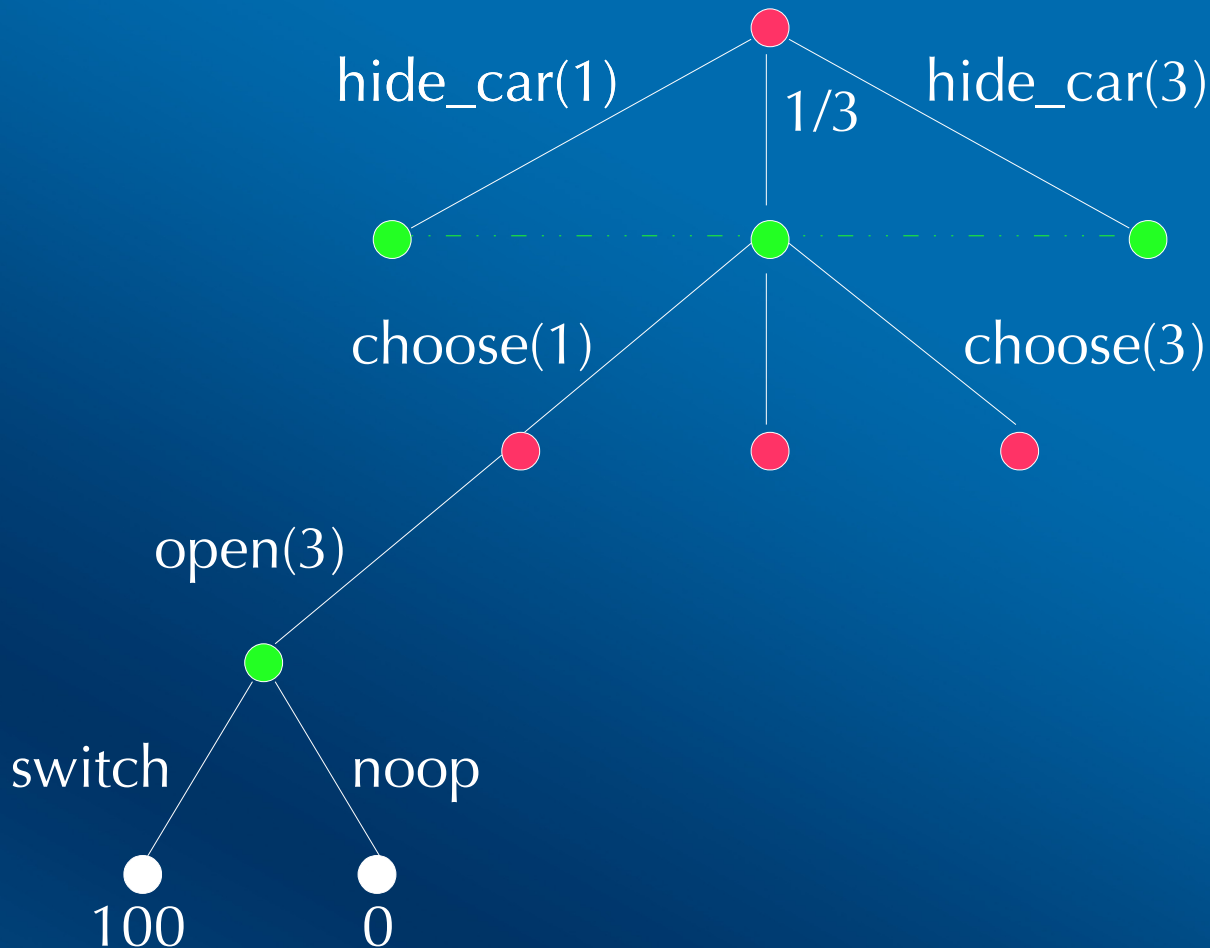
Example: Monty Hall



Example: Monty Hall



Example: Monty Hall



From GDL to Extensive Form

1. Players' perceptions (via the **sees**-predicate) need to be mapped onto information sets.



From GDL to Extensive Form

1. Players' perceptions (via the **sees**-predicate) need to be mapped onto information sets.
2. Simultaneous moves need to be serialised (intermediate states are indistinguishable for players who move "later" in that series).



From GDL to Extensive Form

1. Players' perceptions (via the **sees**-predicate) need to be mapped onto information sets.
2. Simultaneous moves need to be serialised (intermediate states are indistinguishable for players who move "later" in that series).

Theorem.

Any terminating GDL game can be faithfully mapped into an extensive-form game.



From Extensive Form to GDL

1. Information partitions need to be encoded by appropriate **sees**-rules.



From Extensive Form to GDL

1. Information partitions need to be encoded by appropriate **sees**-rules.
2. Non-uniform moves by Nature need to be mapped onto uniform moves for **random**.



From Extensive Form to GDL

1. Information partitions need to be encoded by appropriate **sees**-rules.
2. Non-uniform moves by Nature need to be mapped onto uniform moves for **random**.

Theorem.

Any extensive-form game can be faithfully described in GDL.



Related Work

GALA [Koller & Pfeffer, 97]

- universal game specification language
- coupled with Prolog $\Rightarrow$ operational semantics



Related Work

GALA [Koller & Pfeffer, 97]

- universal game specification language
- coupled with Prolog $\Rightarrow$ operational semantics

Planning languages

- view world from single-agent perspective
- $\Rightarrow$ Opponent Modelling not an issue

Related Work

GALA [Koller & Pfeffer, 97]

- universal game specification language
- coupled with Prolog $\Rightarrow$ operational semantics

Planning languages

- view world from single-agent perspective
- $\Rightarrow$ Opponent Modelling not an issue

Original GDL [Genesereth et al, 05]

- deterministic games w/ complete information

Conclusion

GDL and General Game Playing

- "complete" for the purpose of GGP
- more is needed for open-world games (Scrabble) or robotic systems that play real games

Other Applications of GDL

- Multiagent planning
- Multiagent systems

