

Hannes Strass

Faculty of Computer Science, Institute of Artificial Intelligence, Computational Logic Group

# Game Description Language

Lecture 11, 29th Jun 2026 // Algorithmic Game Theory, SS 2026

# Previously ...

- In a **finite repeated game**, a two-player normal-form game is repeated for a fixed number of times; cooperation cannot be expected in this case.
- In a **random repeated game**, the end of interaction can not be predicted for sure; cooperation can emerge for large enough continuation probabilities, but equilibria make no specific predictions.
- A **noisy repeated game** may have implementation/perception errors.
- An **evolutionarily stable strategy** is a Nash equilibrium that performs better against “mutants” than the “mutants” against themselves.
- Deciding whether a game has an ESS is NP-hard and coNP-hard.

(1, 2)	Hawk	Dove
Hawk	$\frac{V-C}{2}$	$V$
Dove	$0$	$\frac{V}{2}$

- If  $V > C$ , then  $\frac{V-C}{2} > 0$  and **Hawk** is an ESS.
- If  $V \leq C$  and  $C > 0$ , then  $\pi = \left\{ \text{Hawk} \mapsto \frac{V}{C}, \text{Dove} \mapsto 1 - \frac{V}{C} \right\}$  is an ESS.

# Motivation: General Game Playing

- Game playing agents are a testbed for AI approaches and techniques.
- Programs playing specific games have limited value (for AI):
  - very narrow: can play the game(s) they are programmed for, but may not be able to learn to play other (not even simpler) games
  - most analysis and design work is done in advance by human programmers
- **General Game Playing** (GGP) systems use given descriptions of arbitrary games to play these games effectively without human intervention.
- A formal **game description language** (GDL) is used to compactly represent (state-based models of) games.
- Success of the general game player also depends on the “intelligence” of the system itself and not just the human programmer(s).

# Overview

Game Description Language

Playing Games

Incomplete Information

# Game Description Language

# Game Description Language: Ideas

- Main idea: **games** can be described **declaratively** using **logic**
- Game **rules** are described by a **set of formulas** (a normal logic program)
- A **state** in the game is represented by a logical **interpretation**
- GDL uses **simultaneous moves** (sequentiality is modelled via “no-ops”)
- GDL’s **payoffs** are scaled to values from  $[0, 100]$
- During play, information is obtained from descriptions via reasoning:
  - Which moves are legal in a state
  - What the next state looks like after a joint move in a state
  - Which states are terminal
  - Players’ payoffs in terminal states
- But logical reasoning can in principle also be used to analyse the game.
- Thus GDL is also relevant for knowledge representation and reasoning.

# Background: First-Order Logic (Syntax)

We start out from a logical vocabulary  $(\mathcal{P}, \mathcal{F}, \mathcal{V})$  with

- $\mathcal{P}$  a set of **predicate** symbols  $p, q, p_1, p_2, \dots$ , each with an **arity**  $n \in \mathbb{N}$ ,
- $\mathcal{F}$  a set of **function** symbols  $f, g, f_1, f_2, \dots$ , each with an arity  $n \in \mathbb{N}$ , and
- $\mathcal{V}$  a set of **variables**  $x, y, x_1, x_2, \dots$

The set  $T_{\mathcal{P}, \mathcal{F}, \mathcal{V}}$  of **terms** over  $(\mathcal{P}, \mathcal{F}, \mathcal{V})$  is the smallest set such that:

- every variable  $v \in \mathcal{V}$  is a term, and
- if  $t_1, \dots, t_n$  are terms and  $f \in \mathcal{F}$  is a function symbol of arity  $n$ , then  $f(t_1, \dots, t_n)$  is a term.

The set  $A_{\mathcal{P}, \mathcal{F}, \mathcal{V}}$  of **atoms** over  $(\mathcal{P}, \mathcal{F}, \mathcal{V})$  contains all expressions of the form  $p(t_1, \dots, t_n)$  where  $p$  is a predicate symbol of arity  $n$  and  $t_1, \dots, t_n \in T_{\mathcal{P}, \mathcal{F}, \mathcal{V}}$ .

- The **Herbrand universe** is  $T_{\mathcal{P}, \mathcal{F}, \emptyset}$ , the set of all variable-free terms.
- The **Herbrand base** is  $A_{\mathcal{P}, \mathcal{F}, \emptyset}$ , the set of all variable-free atoms.

# Background: Logic Programs (Syntax)

## Definition

Let  $(\mathcal{P}, \mathcal{F}, \mathcal{V})$  be a logical vocabulary.

- A **definite** clause is an expression of the form  $H \leftarrow (B_1 \wedge \dots \wedge B_m)$  (implicitly universally quantified)

$$H \leftarrow (B_1 \wedge \dots \wedge B_m)$$

where  $H, B_1, \dots, B_m \in A_{\mathcal{P}, \mathcal{F}, \mathcal{V}}$ ;  $H$  is called the **head** and each  $B_i$  a **body** atom.

- A **normal** clause is an expression of the form

$$H \leftarrow (B_1 \wedge \dots \wedge B_m \wedge \sim B_{m+1} \wedge \dots \wedge \sim B_{m+n})$$

where  $H, B_1, \dots, B_{m+n} \in A_{\mathcal{P}, \mathcal{F}, \mathcal{V}}$  and  $0 \leq m, n$ ; the symbol  $\sim$  is read as “not”.

- A (normal) **logic program** is a set of (normal) logic program clauses.
- A logic program  $D$  over vocabulary  $(\mathcal{P}, \mathcal{F}, \mathcal{V})$  **defines** a predicate  $p \in \mathcal{P}$  iff  $D$  contains some clause(s) with head  $p(t_1, \dots, t_n)$  for some  $t_1, \dots, t_n \in T_{\mathcal{P}, \mathcal{F}, \mathcal{V}}$ .

**Intuition:** A clause is a logical implication “body implies head”.

# GDL by Example: Tic-Tac-Toe (1)

The **Game Description Language** uses logic programs to define games by requiring a number of **special predicate symbols** to be used in a special way.

- Implication  $\leftarrow$  is written as `:-` and conjunction  $\wedge$  is written as `&`.
- Variables in terms are indicated by upper case identifiers.

There are two roles (players), `X` and `O`:

```
role(x)
role(o)
```

Cells are addressed by indices and can be either blank or marked:

```
base(cell(X, Y, M)) :- index(X) & index(Y) & marker(M)
index(1)      index(2)      index(3)
marker(P) :- role(P)
marker(b)
```

- Predicate `base` declares the **fluents** for describing game states.
- A **fluent** is a (reified) proposition whose truth value can change over time.

## GDL by Example: Tic-Tac-Toe (2)

Available moves are “marking a cell” and “doing nothing”:

```
base(control(P)) :- role(P)
input(P, mark(X, Y)) :- role(P) & index(X) & index(Y)
input(P, noop) :- role(P)
```

Initially, all cells are blank and it is X's turn:

```
init(cell(X, Y, b)) :- index(X) & index(Y)
init(control(x))
```

A player is allowed to mark a cell if that cell is blank and it is the player's turn:

```
legal(P, mark(X, Y)) :- true(cell(X, Y, b)) & true(control(P))
```

If it is not the player's turn, the only legal action is doing nothing:

```
legal(x, noop) :- true(control(o))
legal(o, noop) :- true(control(x))
```

# GDL by Example: Tic-Tac-Toe (3)

If a player marks a cell, the cell gets that mark:

```
next(cell(X, Y, P)) :- does(P, mark(X, Y)) & true(cell(X, Y, b))
```

Attention: We need to explicitly axiomatise persistence:

Any marked cell retains its mark for the rest of the game:

```
next(cell(X, Y, M)) :- true(cell(X, Y, M)) & distinct(M, b)
```

Blank cells stay blank if not marked:

```
next(cell(X, Y, b)) :-  
    does(P, mark(I, J)) & true(cell(X, Y, b)) & distinct(X, I)  
next(cell(X, Y, b)) :-  
    does(P, mark(I, J)) & true(cell(X, Y, b)) & distinct(Y, J)
```

# GDL by Example: Tic-Tac-Toe (4)

Control alternates between the players:

```
next(control(o)) :- true(control(x))
next(control(x)) :- true(control(o))
```

The game terminates when one player has won or every cell is marked:

```
terminal :- line(P)
terminal :- ~open
open :- true(cell(X, Y, b))
```

Player  $X$ 's payoffs in terminal states are as expected (similarly for  $O$ ):

```
goal(x, 100) :- line(x) & ~line(o)
goal(x, 50) :- ~line(x) & ~line(o)
goal(x, 0) :- ~line(x) & line(o)
```

**Exercise:** Define the predicate `line`, possibly using further auxiliary predicates.

# GDL Special Predicates: Overview

The **special predicates** of GDL are the following:

- `role( $r$ )` ...  $r$  is a role (player) in the game
- `input( $r, m$ )` ... player  $r$  has feasible move  $m$  in the game
- `base( $p$ )` ...  $p$  is a base proposition (fluent) in the game
- `init( $p$ )` ...  $p$  is true in the initial state
- `true( $p$ )` ...  $p$  is true in the current state
- `does( $r, m$ )` ... player  $r$  makes move  $m$  in the current state
- `next( $p$ )` ...  $p$  is true in the next state
- `legal( $r, m$ )` ... it is legal for player  $r$  to make move  $m$  in the current state
- `goal( $r, u$ )` ... the current state has utility  $u$  for player  $r$
- `terminal` ... the current state is a terminal state

The pre-defined auxiliary predicate `distinct` defines syntactic inequality.

# GDL Game Descriptions: Definition

## Definition

A GDL **game description** is a logic program  $D$  over a vocabulary  $(\mathcal{P}, \mathcal{F}, \mathcal{V})$  where  $\mathcal{P}$  includes the special predicates of GDL. Furthermore:

1.  $D$  must give complete definitions for **role**, **base**, **input**, and **init**.
2.  $D$  must define **legal**, **terminal**, and **goal** in terms of **true**.
3.  $D$  must define **next** in terms of **true** and **does**.
4.  $D$  must not define **true** and **does**.

“Defining  $p$  in terms of  $q_1, \dots, q_n$ ” means:

For every clause with head predicate  $p$ , its body only contains:

- atoms with predicates among  $q_1, \dots, q_n$ , or
- auxiliary predicates (in turn defined in terms of  $q_1, \dots, q_n$ ). (e.g. line)

# Background: First-Order Logic (Semantics)

- An **interpretation** is a pair  $\mathcal{J} = (\Delta, \cdot^{\mathcal{J}})$  where  $\Delta \neq \emptyset$  and  $\cdot^{\mathcal{J}}$  assigns:
  - to each predicate symbol  $p \in \mathcal{P}$  of arity  $n$  a relation  $p^{\mathcal{J}} \subseteq \Delta^n$ , and
  - to each function symbol  $f \in \mathcal{P}$  of arity  $n$  a function  $f^{\mathcal{J}}: \Delta^n \rightarrow \Delta$ .
- A **variable valuation** is a function  $v: \mathcal{V} \rightarrow \Delta$ .
- An **Herbrand interpretation** is an interpretation  $(\Delta, \cdot^{\mathcal{J}})$  with  $\Delta = T_{\mathcal{P}, \mathcal{F}, \emptyset}$  where every ground term  $t \in T_{\mathcal{P}, \mathcal{F}, \emptyset}$  is interpreted by itself,  $t^{\mathcal{J}} = t$ .
- The value of a term  $t \in T_{\mathcal{P}, \mathcal{F}, \mathcal{V}}$  under an interpretation  $\mathcal{J}$  and variable valuation  $v$  is

$$t^{\mathcal{J}, v} := \begin{cases} v(x) & \text{if } t = x \in \mathcal{V}, \\ f^{\mathcal{J}}(t_1^{\mathcal{J}, v}, \dots, t_n^{\mathcal{J}, v}) & \text{if } t = f(t_1, \dots, t_n). \end{cases}$$

- An interpretation  $\mathcal{J}$  with variable valuation  $v$  **satisfies** an atom  $p(t_1, \dots, t_n)$ , written  $\mathcal{J} \models p(t_1, \dots, t_n)$ , iff  $(t_1^{\mathcal{J}, v}, \dots, t_n^{\mathcal{J}, v}) \in p^{\mathcal{J}}$ .

# Background: Logic Programs (Semantics I)

## Definition

Let  $D$  be a logic program under vocabulary  $(\mathcal{P}, \mathcal{F}, \mathcal{V})$  and  $\mathcal{J}$  be an interpretation for the vocabulary.

- $\mathcal{J}$  **satisfies** a clause  $H \leftarrow (B_1 \wedge \dots \wedge B_m \wedge \sim B_{m+1} \wedge \dots \wedge \sim B_{m+n})$  iff if  $\mathcal{J} \models B_i$  for  $1 \leq i \leq m$  and  $\mathcal{J} \not\models B_{m+j}$  for  $1 \leq j \leq n$ , then  $\mathcal{J} \models H$ .
  - $\mathcal{J}$  is a **model** of a logic program  $D$  iff  $\mathcal{J}$  satisfies all clauses in  $D$ .
  - An atom  $A \in A_{\mathcal{P}, \mathcal{F}, \emptyset}$  is **entailed** by a logic program  $D$ , written  $D \models A$ , iff for every model  $\mathcal{J}$  of  $D$ , we have  $\mathcal{J} \models A$ .
- 
- Herbrand interpretations can be represented as sets  $I \subseteq A_{\mathcal{P}, \mathcal{F}, \emptyset}$  of atoms.
  - Definite logic programs (containing only definite clauses) have a unique  $\subseteq$ -least Herbrand model capturing the set of all atoms entailed by it.
  - For normal logic programs, a (unique) model need not exist in general.

# Background: Logic Programs (Semantics II)

For **normal** logic programs (using negation  $\sim$ ), a unique least Herbrand model exists only under special circumstances.

In one particular set of restrictions, the program must be:

- **safe** (in every clause, every variable occurring in the head or in a negated body atom must also occur in a positive body atom)
- **stratified** (there must be no recursion through negation)
- **recursion-restricted** (positive recursion must be range-restricted)

Then, the intended semantics of the program is given by its **standard model**:

- We first consider the least model  $M_0$  of the subset of rules for predicates  $\mathcal{P}_0 \subseteq \mathcal{P}$  that do not depend negatively on another predicate.
- We next extend  $M_0$  by all ground atoms derivable by clauses for predicates  $\mathcal{P}_1 \subseteq \mathcal{P} \setminus \mathcal{P}_0$  that depend negatively only on predicates from  $\mathcal{P}_0$ .
- ... For more details, see the lecture *Foundations of Logic Programming* (WS).

We typically assume our game descriptions to satisfy these restrictions.

# Game Description Language: Semantics

## Definition

Given a GDL game description  $D$ , the resulting state-based game model is obtained as follows: (where  $\models$  is w.r.t. the standard model)

- The players are  $P = \{r \mid D \models \text{role}(r)\}$ . (Denote  $n = |P|$ .)
- The moves of each player  $r \in P$  are  $M_r = \{m \mid D \models \text{input}(r, m)\}$ .
- The set of states is given by  $2^Q$  with  $Q = \{\text{true}(q) \mid D \models \text{base}(q)\}$ .
- The initial state is given by  $S_0 = \{\text{true}(q) \mid D \models \text{init}(q)\}$ .
- The legal moves of  $r \in P$  in state  $S \subseteq Q$  are  $\{m \mid D \cup S \models \text{legal}(r, m)\}$ .
- Given a state  $S \subseteq Q$  and a joint move  $(m_1, \dots, m_n)$ , the next state is given by  $\{\text{true}(q) \mid D \cup S \cup \{\text{does}(r_1, m_1), \dots, \text{does}(r_n, m_n)\} \models \text{next}(q)\}$ .
- A state  $S \subseteq Q$  is terminal iff  $D \cup S \models \text{terminal}$ .
- The utility of player  $r \in P$  in terminal state  $S \subseteq Q$  is  $u$  for  $D \cup S \models \text{goal}(r, u)$ .

There are further technical requirements (playability, winnability) that we will not delve into.

# Playing Games

# Playing GDL Games

- A **game manager** coordinates the individual players (agents) via network using the **game communication language**.
- In the beginning, a **start**(*id, role, D, startclock, playclock*) message from the game manager to an agent signals that:
  - the match with *id* starts after *startclock* seconds have elapsed,
  - the agent receiving the message will play *role*, and
  - the agent will have *playclock* seconds to choose each move.
- Agents use the *startclock* time to understand the game rules, analyse the game and possibly start searching.
- For each subsequent round of the match, a **play**(*id, move*) message from the game manager to an agent indicates that:
  - the agent is supposed to submit a move for match *id*,
  - where the previous joint move (for non-initial states) is given in *move*.
- When the game is over, the game manager sends a **stop**(*id, move*) message to all agents, informing them about the last *move*.

# Playing GDL Games: Example

Denote by  $D$  the GDL game description of Tic-Tac-Toe considered earlier.

- By description and definition, the initial state is

$$S_0 = \{\text{true}(\text{cell}(1, 1, b)), \text{true}(\text{cell}(1, 2, b)), \dots, \text{true}(\text{cell}(3, 3, b)), \text{true}(\text{control}(x))\}$$

- The legal moves of  $X$  in  $S_0$  are

$$\text{mark}(1, 1, x), \text{mark}(1, 2, x), \dots, \text{mark}(3, 3, x)$$

- The only legal move of  $O$  in  $S_0$  is *noop*.
- After the joint move  $(\text{mark}(2, 2, x), \text{noop})$ , the next state is

$$S_1 = \{\text{true}(\text{cell}(2, 2, x)), \text{true}(\text{cell}(1, 1, b)), \dots, \text{true}(\text{cell}(3, 3, b)), \text{true}(\text{control}(o))\}$$

- State  $S_1$  is not yet terminal, as  $D \cup S_1 \not\models \text{terminal}$  because  $D \cup S_1 \models \text{open}$ .

# Playing GDL Games: Move Selection

## Challenge

The GDL game model has **simultaneous** moves of all players.

## Approach:

- Assume (pessimistically) that all others “collude” and select their joint move as to minimise one’s own payoff.
  - ↪ Treat  $n$ -player general-sum simultaneous move game as two-player zero-sum turn-taking game. (Every opponent joint move is considered to be a single move of the “virtual” opponent; we move second.)
- Implement Monte Carlo or Alpha Beta Tree Search on this sequential game.
- Heuristics for depth-limited game tree search:
  - Analyse **goal** rules for goal proximity heuristics.
  - Analyse **legal** moves in states for mobility heuristics.
- Analyse **next** rules to find persistent fluents (e.g. markers in Tic-Tac-Toe).

# Incomplete Information

# GDL-II: GDL with Incomplete Information

Both imperfect information and incomplete information can be modelled using only two additional keywords:

- `percept`( $r, q$ ) ... player  $r$  has possible percept  $q$  in the game
- `sees`( $r, q$ ) ... player  $r$  perceives  $q$  in the next state

To model chance nodes (moves by `Nature`), a new role name is introduced:

- `random` ... special role that chooses a legal move uniformly at random

## Definition

A **GDL-II game description** is a logic program  $D$  over vocabulary  $(\mathcal{P}, \mathcal{F}, \mathcal{V})$  where  $\mathcal{P}$  includes the GDL-II keywords and  $\mathcal{F}$  includes the constant symbol `random`. Furthermore,  $D$  must obey the syntactic restrictions of GDL game descriptions where additionally predicate `sees` only appears as head of clauses and must be defined in terms of `true` and `does`.

# GDL-II by Example: Simplified Poker (1)

There are three cards, two players, and the game begins with dealing:

```
card(1)      card(2)      card(3)
beats(3,2)   beats(3,1)   beats(2,1)
role(ann)    role(bob)     init(control(random))
```

Nature moves first and deals the cards (otherwise does nothing):

```
legal(random, deal(C, D)) :-
    true(control(random)) & card(C) & card(D) & distinct(C, D)
legal(random, noop) :- ~true(control(random))
```

Dealing has the expected effects and percepts:

```
next(hasCard(ann, C)) :- does(random, deal(C, D))
next(hasCard(bob, D)) :- does(random, deal(C, D))
sees(ann, yourCard(C)) :- does(random, deal(C, D))
sees(bob, yourCard(D)) :- does(random, deal(C, D))
```

## GDL-II by Example: Simplified Poker (2)

Next comes Ann's turn to choose a move:

```
next(control(ann)) :- true(control(random))
legal(ann, check) :- true(control(ann))
legal(ann, raise) :- true(control(ann))
```

Bob can see Ann's decision and can move iff Ann did a raise:

```
sees(bob, annsMove(M)) :- does(ann, M)
next(control(bob)) :- true(control(ann)) & does(ann, raise)
next(showdown) :- does(ann, check)
next(hasCard(P, C)) :- true(hasCard(P, C))
```

Bob's moves are fold and call, with a showdown happening after call:

```
legal(bob, fold) :- true(control(bob))
legal(bob, call) :- true(control(bob))
next(showdown) :- does(bob, call)
```

# GDL-II by Example: Simplified Poker (3)

If Bob folds, the game is over and Ann wins:

```
next(annWins) :- true(control(bob)) & does(bob, fold)
terminal      :- true(annWins)
goal(bob, 0)  :- true(annWins)
goal(ann, 100) :- true(annWins)
```

In a showdown, cards are revealed and the higher card wins:

```
sees(P, hasCard(O, C)) :-
  does(ann, check) & true(hasCard(O, C)) & role(P) & distinct(P, O)
sees(P, hasCard(O, C)) :-
  does(bob, call) & true(hasCard(O, C)) & role(P) & distinct(P, O)
terminal :- true(showdown)
goal(P, 100) :-
  true(hasCard(P, C)) & true(hasCard(O, D)) & beats(C, D)
goal(O, 0) :-
  true(hasCard(P, C)) & true(hasCard(O, D)) & beats(C, D)
```

# GDL-II: Semantics via State Transitions

For a GDL-II game description  $D$ , the resulting state-based game model is:

- Players, (legal) moves, and initial/terminal state(s) are obtained as in GDL.
- The next state after joint move  $\mathbf{m} := (m_1, \dots, m_n)$  is obtained as usual:

$$n(\mathbf{m}, S) := \{\text{true}(q) \mid D \cup S \cup \{\text{does}(r_1, m_1), \dots, \text{does}(r_n, m_n)\} \models \text{next}(q)\}$$

- An **information relation**  $\mathcal{I} \subseteq P \times M^n \times 2^Q \times Q$  models players' incomplete information:  $(r, \mathbf{m}, S, q)$  indicates that player  $r$  perceives  $q$  after joint move  $\mathbf{m}$  happens in state  $S$ .
- A probability distribution over possible resulting states models uncertainty induced by **random**'s moves: After (non-**random** players') joint move  $\mathbf{m}$  in state  $S \subseteq Q$ , the probability of  $T \subseteq Q$  being the resulting state is

$$\frac{|\{m \in L \mid n((m; \mathbf{m}), S) = T\}|}{|L|}$$

where  $L = \{m \in M_{\text{random}} \mid D \cup S \models \text{legal}(\text{random}, m, S)\}$ ,

and  $(m; \mathbf{m}) := (m, m_1, \dots, m_n)$  extends  $\mathbf{m}$  by **random**'s move  $m$ .

# Playing GDL-II Games

Game management can be adjusted to the incomplete information setting:

1. Send each agent the game description and inform them about their role.
2. Set  $S$  to the initial game state.
3. For every subsequent state  $S$  of the game:
  - (a) Collect moves from all agents and (if applicable) choose a legal move for **random** with uniform probability.
  - (b) To every agent  $r \in P$ , send percepts  $\{q \in Q \mid (r, \mathbf{m}, S, q) \in \mathcal{J}\}$  for joint move  $\mathbf{m}$  in  $S$ .
  - (c) Update current state  $S$  to next state  $n(\mathbf{m}, S)$ .
4. Repeat until  $S$  is terminal, then send utilities to agents.

Since the game manager has complete knowledge about the game state, it can compute all percepts and resulting states.

# GDL-II: Developments

For a GDL-II game description  $D$ , it is also possible to define an extensive form game  $G_D$ . A first necessary ingredient is that of a development.

## Definition

Consider the state-based game model of a GDL-II game description.

- A **development** is a finite sequence  $\delta = \langle S_0, \mathbf{m}_1, S_1, \dots, \mathbf{m}_d, S_d \rangle$  where
  - $d \geq 0$ ,
  - $S_0, \dots, S_d \subseteq Q$  are states, in particular  $S_0$  is the initial state,
  - $\mathbf{m}_j = (m_0, m_1, \dots, m_n)$  is a joint move (possibly including a move  $m_0$  for **random**),
  - every move in  $\mathbf{m}_j$  is legal (for its player) in state  $S_{j-1}$ , for all  $1 \leq j \leq d$ ,
  - the sequence obeys state update, i.e.  $n(\mathbf{m}_j, S_{j-1}) = S_j$  for all  $1 \leq j \leq d$ , and
  - only  $S_d$  may be terminal.
- Two developments  $\delta, \delta'$  of length  $d$  are **indistinguishable** for player  $i \geq 1$  iff:
  - player  $i$  makes the same move in  $\mathbf{m}_j$  and  $\mathbf{m}'_j$ , for all  $1 \leq j \leq d$ , and
  - $\{q \in Q \mid (i, \mathbf{m}_j, S_{j-1}, q) \in \mathcal{J}\} = \{q \in Q \mid (i, \mathbf{m}'_j, S'_{j-1}, q) \in \mathcal{J}\}$  for all  $1 \leq j \leq d$ .

# GDL-II: Quasi-Developments

- **Main Idea:** Sequentialise joint moves and keep individual moves private until a joint move (and thus the next state with its resulting percepts) is complete.

## Definition

Consider the state-based game model of a GDL-II game description.

- A **partial joint move** is a tuple  $\mathbf{m}^{(i)} = (m_0, m_1, \dots, m_i)$  with  $0 \leq i < n$ .
- A **quasi-development** is of the form  $\gamma = \langle \delta, \mathbf{m}^{(i)} \rangle$  where  $\delta$  is a development and  $\mathbf{m}^{(i)}$  is a partial joint move.
- **Intuition:** A partial joint move  $\mathbf{m}^{(i)}$  serves to model the sequentialisation of a joint move where players  $\{i + 1, \dots, n\}$  are yet to move.
- The **history arising from a development**  $\delta = \langle S_0, \mathbf{m}_1, S_1, \dots, \mathbf{m}_d, S_d \rangle$  is then  $h_\delta := [(\mathbf{m}_1)_0, (\mathbf{m}_1)_1, \dots, (\mathbf{m}_1)_n, (\mathbf{m}_2)_0, \dots, (\mathbf{m}_d)_n]$ ;
- the **history arising from a quasi-development**  $\langle \delta, \mathbf{m}^{(i)} \rangle$  is then  $h_{\langle \delta, \mathbf{m}^{(i)} \rangle} := [h_\delta; (\mathbf{m}^{(i)})_0, \dots, (\mathbf{m}^{(i)})_i]$ .

For a tuple  $\mathbf{m} = (m_0, \dots, m_n)$  we denote  $(\mathbf{m})_i := m_i$  for  $0 \leq i \leq n$ .

# GDL-II: Semantics via Extensive-Form Games

## Definition

For a GDL-II game description  $D$ , its **associated extensive-form game**  $G_D$  is:

- Its players are  $P = \{0, 1, \dots, n\}$ , where 0 denotes **Nature** (random).
- Its moves and utilities are as in the state-based game model.
- Its histories  $H$  are all those that arise from (quasi-)developments of  $D$ .
- Its terminal histories arise from developments  $\delta$  with  $S_d$  terminal.
- Its player function assigns  $p(h_\delta) = 0$  and  $p(h_{\langle \delta, \mathbf{m}^{(i)} \rangle}) = i + 1$ .
- Its probability distributions for chance nodes are always uniform.
- The indistinguishability relation  $\sim_i$  of each player  $i \geq 1$  is as follows:

$$h_{\langle \delta, \mathbf{m}^{(i-1)} \rangle} \sim_i h_{\langle \delta', \mathbf{m}'^{(i-1)} \rangle} \quad \text{iff} \quad \delta \text{ and } \delta' \text{ are indistinguishable for player } i$$

- The information sets of player  $i$  are  $\{h \in H \mid p(h) = i\}$  partitioned by  $\sim_i$ .

# Properties of GDL-II: Extension of GDL

## Proposition

GDL-II is a proper extension of GDL.

## Proof.

- Let  $D$  be a game description in GDL.
- To express the same game in GDL-II, we add one rule:  
`sees(P, move(0, M)) :- role(P) & does(0, M)`
- Thus, every player knows every move of every other player. □

# Properties of GDL-II: Universality (1)

Theorem (Thielscher, 2011)

GDL-II is universal, i.e. for every finite extensive-form game  $G$  there is a GDL-II game description  $D_G$  that formalises  $G$ .

Proof (Sketch, 1/3).

- We assume a game  $G$  given in extensive form (i.e. as explicit tree).
- Players  $P = \{1, \dots, n\}$  are defined through `role(random)`, `role(1)`,  $\dots$ , `role(n)`.
- Histories  $h \in H$  are encoded as terms  $t_h$  via
$$t_{[]} := \text{nil} \quad \text{and} \quad t_{[h;m]} := \text{cons}(t_h, m).$$
- The initial state is encoded via `init(nil)`.
- Terminal states are expressed via `terminal :- true(t_h)` for all  $h \in Z$ .
- We declare utilities via `goal(i, u_i(h)) :- true(t_h)` for  $h \in Z$ .  
(Utilities are scaled to  $[0, 100]$  using  $\min/\max \{u_i(h) \mid h \in Z, 1 \leq i \leq n\}$ .)

# Properties of GDL-II: Universality (2)

Proof (Sketch, 2/3).

- Legality and state update are defined as expected:

$$\text{legal}(i, m) \text{ :- true}(t_h)$$
$$\text{next}(t_{[h;m]}) \text{ :- true}(t_h) \ \& \ \text{does}(i, m)$$
$$\text{legal}(j, \text{noop}) \text{ :- true}(t_h)$$

for all  $[h; m] \in H$ ,  $p(h) = i$  with  $1 \leq i \leq n$ ,  $m \in M_i$ , and  $0 \leq j \leq n$  with  $j \neq i$ .

- Information sets of the game lead to abstract percepts:

$$\text{sees}(j, k) \text{ :- true}(t_h) \ \& \ \text{does}(i, m)$$
$$\text{member}(t_{[h;m]}, k)$$

for  $[h; m] \in \mathcal{J}_k$ ,  $p(h) = i$ , and  $p(\mathcal{J}_k) = j$ , for  $0 \leq i, j \leq n$ .

# Properties of GDL-II: Universality (3)

Proof (Sketch, 3/3).

- For moves of **Nature** (**random**), we assume the probability distribution over moves is  $\{m_1 \mapsto \frac{p_1}{q}, \dots, m_\ell \mapsto \frac{p_\ell}{q}\}$  for some  $h \in H$  with  $p(h) = \mathbf{Nature}$ .
- For every  $1 \leq k \leq \ell$ , we now create  $p_k$  many copies of  $m_k$  and specify

`legal(random,  $m_k^{(1)}$ ) :- true( $t_h$ )`

`next( $t_{[h;m_k]}$ ) :- true( $t_h$ ) & does(random,  $m_k^{(1)}$ )`

`⋮`

`legal(random,  $m_k^{(p_k)}$ ) :- true( $t_h$ )`

`next( $t_{[h;m_k]}$ ) :- true( $t_h$ ) & does(random,  $m_k^{(p_k)}$ )`

to express proportionality of probabilities. □

# Playing GDL-II Games: Move Selection

Schofield, Cerexhe, & Thielscher [2012] propose a method called **HyperPlay**:

- Estimate the true history by a list of samples from the information set.
- Each sample is a complete history that is consistent with what is known.
- Initialise the list of samples as  $\langle [], \dots, [] \rangle$ .
- Use “conventional” techniques to select a move for each complete history.
- An overall move is selected based on its expected utility weighted by the probability that its history  $h$  is the true match history given percepts  $Q$ :

$$P(h|Q) = \frac{P(Q|h) \cdot P(h)}{P(Q)}$$

- After each own move and received percepts, update the samples:
  - Randomly sample from other players’ legal moves to obtain a full joint move.
  - Compute the next state and expected own percepts.
  - Remove those samples where received and expected percepts disagree.

# Conclusion

## Summary

- **General Game Playing** is concerned with computers learning to play previously unknown games without human intervention.
- The **game description language** (GDL) is used to declaratively specify (deterministic) games (with complete information about game states).
- The syntax of GDL game descriptions is that of **normal logic programs**; various restrictions apply to obtain a finite, unique interpretation.
- The semantics of GDL is given through a state transition system.
- GDL-II allows to represent moves by **Nature** and information sets.
- The semantics of GDL-II can be given through extensive-form games.
- Conversely, GDL-II can express any finite extensive-form game.

**Exercise:** Adapt the payoffs in the GDL model of simplified poker.