8. Behind the Scenes of FluxPlayer
## Outline

<table>
<thead>
<tr>
<th>Date</th>
<th>What will we do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.10.2013</td>
<td>Introduction, Repetition propositional logic and FOL</td>
</tr>
<tr>
<td>29.10.2013</td>
<td>Repetition FOL / Datalog and Prolog</td>
</tr>
<tr>
<td>05.11.2013</td>
<td>Game Description Language</td>
</tr>
<tr>
<td>12.11.2013</td>
<td>Design of GDL games</td>
</tr>
<tr>
<td>19.11.2013</td>
<td>Search Algorithms 1</td>
</tr>
<tr>
<td>26.11.2013</td>
<td>No lecture</td>
</tr>
<tr>
<td>03.12.2013</td>
<td>Search Algorithms 2</td>
</tr>
<tr>
<td>10.12.2013</td>
<td>Real GGP 1</td>
</tr>
<tr>
<td>17.12.2013</td>
<td><strong>Midterm competition</strong></td>
</tr>
<tr>
<td>07.01.2014</td>
<td>GIGA 1 – State of the art 2009</td>
</tr>
<tr>
<td>14.01.2014</td>
<td>GIGA 2 – State of the art 2011</td>
</tr>
<tr>
<td>21.01.2014</td>
<td>GIGA 3 – State of the art 2013</td>
</tr>
<tr>
<td>28.01.2014</td>
<td>Summary and Outlook</td>
</tr>
<tr>
<td>04.02.2014</td>
<td><strong>Final Competition</strong></td>
</tr>
<tr>
<td>11.02.2014</td>
<td><strong>Exam</strong></td>
</tr>
</tbody>
</table>
Competition

• Single-player:
  – Three types of games (easy, medium, hard)
  – Each player plays all three games
  – Total score of a player is the weighted sum of points (goal score):
    • \( \text{TOTAL} = \text{points(easy)} + 2 \times \text{points(medium)} + 3 \times \text{points(hard)} \)
  – The player with maximum number of points wins
Competition

- Two-player: depends on number of teams
- If 9 teams:
  - Three groups with 3 player each
  - All against all in a group (2 games for each player)
  - Players are ranked by the total score
  - First team from each group goes to the final round
  - Final round:
    - one group with three teams
    - All against all in the final group (again 2 games for each player)
    - Player with highest score wins competition
- If 8/10 team:
  - Double knockout
Oral examination

- Available appointments can be found on course web page
Repetition

- What is the difference between heuristic search and Monte-Carlo tree search?
Repetition

• What is the difference between heuristic search and Monte-Carlo tree search?
FluxPlayer
Overview

• Developed at Dresden University of Technology
• Quite successful
  – Winner 2006
  – Several times runners-up
• Outline for today
  – Generating state evaluation functions
  – Identifying syntactic structures
  – Distance estimation for fluents and states
  – Proving properties of games
• Uses fuzzy logic …
Fuzzy logic
Fuzzy logic

**Definition 2.15** (Fuzzy set). Let $X$ be a set of objects. A fuzzy set $\mu$ of $X$ is a function $\mu : X \rightarrow [0, 1]$, which associates each object in $X$ a real number in the interval $[0, 1]$ representing the “grade of membership” of $x$ in the fuzzy set.

**Complement** The complement $\neg \mu$ of a fuzzy set $\mu$ is defined elementwise using a negation function $n$:

$$(\neg \mu)(x) = n(x)$$

**Definition 2.16** (Negation function). A negation function $n$ is a function $n : [0, 1] \rightarrow [0, 1]$ with the following properties:

- $n(0) = 1$
- $n(1) = 0$
- $a \leq b \Rightarrow n(a) \geq n(b)$
**Fuzzy logic**

**Intersection** The intersection of two fuzzy sets \( \mu_1, \mu_2 \) is a fuzzy set \( \mu \), i.e., a function mapping each object \( x \) to a value in \([0, 1]\) representing the grade of membership of \( x \) in both sets \( \mu_1, \mu_2 \). Such a function is called a t-norm and defined as follows:

**Definition 2.17** (T-norm). A function \( \sqcap : [0, 1]^2 \to [0, 1] \) is called t-norm, if

- \( \sqcap(a, 1) = a \) (neutral element),
- \( a \leq b \supset \sqcap(a, c) \leq \sqcap(b, c) \) (monotonicity),
- \( \sqcap(a, b) = \sqcap(b, a) \) (commutativity), and
- \( \sqcap(a, \sqcap(b, c)) = \sqcap(\sqcap(a, b), c) \) (associativity).

**Definition 2.18** (Continuous T-norm). A t-norm \( \sqcap \) is called continuous if

\[
a < b \land c > 0 \supset \sqcap(a, c) < \sqcap(b, c)
\]
Fuzzy logic

Union  The union of two fuzzy sets $\mu_1$ and $\mu_2$ is defined similarly to the intersection as

$$(\mu_1 \lor \mu_2)(x) = \bot(\mu_1(x), \mu_2(x)).$$

The function $\bot$ is some t-conorm:

Definition 2.19 (T-conorm). A function $\bot : [0, 1]^2 \to [0, 1]$ is called t-conorm, if

- $\bot(a, 0) = a$ (neutral element),
- $a \leq b \Rightarrow \bot(a, c) \leq \bot(b, c)$ (monotonicity),
- $\bot(a, b) = \bot(b, a)$ (commutativity), and
- $\bot(a, \bot(b, c)) = \bot(\bot(a, b), c)$ (associativity).

Each t-norm $\top$ defines a dual t-conorm $\bot$ in the following way:

$$\bot(a, b) = 1 - \top(1 - a, 1 - b)$$
Fuzzy logic: motivation

• One can interpret a propositional formula as a fuzzy set and provide a fuzzy evaluation of a propositional formula. Thus, the degree of truth of the sentence “the state s is near to a goal state” can be computed by a function

\[
f = \text{true}(\text{cell}(a,1,x)) \land \text{true}(\text{cell}(b,1,x)) \land \text{true}(\text{cell}(c,1,x)) \lor \\
\text{true}(\text{cell}(a,2,x)) \land \text{true}(\text{cell}(b,2,x)) \land \text{true}(\text{cell}(c,2,x)) \lor \\
... \\
\text{true}(\text{cell}(a,3,x)) \land \text{true}(\text{cell}(b,2,x)) \land \text{true}(\text{cell}(c,1,x))
\]
Generating State Evaluation Functions
Idea

- Idea: define the fuzzy evaluation $\text{eval}(f, s)$ of arbitrary formulas $f$ with respect to state $s$

Definition 5.1 (GDL Formula). A GDL formula $f$ of a game description $D$ is a first order formula with the usual connectives for conjunction $\land$, disjunction $\lor$, negation $\neg$, and existential quantifiers $\exists$. The atoms of $f$ are atoms over the signature (relation symbols and function symbols) of $D$. 
Fuzzy Formula evaluation

**Definition 5.2 (Fuzzy Formula Evaluation).** Let $D$ be a set of GDL rules and parameter $p$ be a real value with $0.5 < p \leq 1$. Furthermore, let $a$ denote GDL atoms, $f$ and $g$ denote arbitrary GDL formulas, and $\top$ denote an arbitrary $t$-norm with dual $t$-conorm $\bot$. We define a fuzzy evaluation function for GDL formulas wrt. a game state $s$ as follows:

- **conjunction** If $f$ and $g$ contain no common variables:

  $eval(f \land g, s) = \top(eval(f, s), eval(g, s))$ \hspace{1cm} (5.1)

- **disjunction** If $f$ and $g$ contain no common variables:

  $eval(f \lor g, s) = \bot(eval(f, s), eval(g, s))$ \hspace{1cm} (5.2)

- **negation**

  $eval(\neg f, s) = 1 - eval(f, s)$ \hspace{1cm} (5.3)

- **atoms defined by rules** For every atom $a$ except distinct($t_1, t_2$), true($t$), and does($r, m$), let $a_1 : = b_1, \ldots, a_n : = b_n$ be all rules in $D$ such that $a$ unifies with the head $a_i$ of each rule with unifier $\sigma_i$, that is, $a_i\sigma_i = a$ for all $i \in 1 \ldots n$. In this case,

  $eval(a, s) = eval(b_1\sigma_1 \lor b_2\sigma_2 \lor \ldots \lor b_n\sigma_n, s)$ \hspace{1cm} (5.4)

- **other** For all GDL formulas $f$ that do not match any of the rules above:

  $eval(f, s) = \begin{cases} 
P & \text{if } D \cup s^{true} \models (\exists) f \\ 
1 - P & \text{otherwise} \end{cases}$ \hspace{1cm} (5.5)
Example: Winning Tic-Tac-Toe with x-player

line(P) :- true(cell(a,Y,P)),
  true(cell(b,Y,P)), true(cell(c,Y,P)).
line(P) :- true(cell(X,1,P)),
  true(cell(X,2,P)), true(cell(X,3,P)).
line(P) :- true(cell(a,1,P)),
  true(cell(b,2,P)), true(cell(c,3,P)).
line(P) :- true(cell(a,3,P)),
  true(cell(b,2,P)), true(cell(c,1,P)).
goal(xplayer,100) :- line(x).
\texttt{eval(goal(xplayer, 100), s) ?}
eval(goal(xplayer, 100), s)

According to Equation 5.4:

\[
\text{eval}(\text{goal}(\text{xplayer}, 100), s) = \text{eval}(\text{line}(x), s)
\]

\[
= \text{eval}(
    [(\exists Y)\text{true(cell}(a, Y, x)) \land \text{true(cell}(b, Y, x)) \land \text{true(cell}(c, Y, x))] \lor
    [(\exists X)\text{true(cell}(X, 1, x)) \land \text{true(cell}(X, 2, x)) \land \text{true(cell}(X, 3, x))] \lor
    [\text{true(cell}(a, 1, x)) \land \text{true(cell}(b, 2, x)) \land \text{true(cell}(c, 3, x))] \lor
    [\text{true(cell}(a, 3, x)) \land \text{true(cell}(b, 2, x)) \land \text{true(cell}(c, 1, x))], s
\]
eval(goal(xplayer, 100), s)

Conjunctions and disjunctions in the formula are evaluated with a t-norm and t-conorm respectively (see Equation 5.1 and 5.2). Thus, Equation 5.6 above is equal to

\[ \bot \left( \text{eval}([[\forall Y] \text{true}(\text{cell}(a, Y, x)) \land \ldots \land \text{true}(\text{cell}(c, Y, x))], s), \right. \]

\[ \bot \left( \text{eval}([[\exists X] \text{true}(\text{cell}(X, 1, x)) \land \ldots \land \text{true}(\text{cell}(X, 3, x))], s), \right. \]

\[ \ldots \]

\[ \top(\right. \]

\[ \text{eval}(\text{true}(\text{cell}(b, 2, x)), s), \]

\[ \text{eval}(\text{true}(\text{cell}(c, 1, x)), s) \]

\[ \right) \]

\[ \ldots \]

\[ \right) \]
The quantified formulas

$$(\exists Y) \text{true}(\text{cell}(a, Y, x)) \land \text{true}(\text{cell}(b, Y, x)) \land \text{true}(\text{cell}(c, Y, x))$$

and

$$(\exists X) \text{true}(\text{cell}(X, 1, x)) \land \text{true}(\text{cell}(X, 2, x)) \land \text{true}(\text{cell}(X, 3, x))$$

as well as the atoms

$$\text{true}(\text{cell}(a, 1, x)), \text{true}(\text{cell}(b, 2, x)), \text{true}(\text{cell}(c, 3, x)), \text{true}(\text{cell}(a, 3, x)), \text{and } \text{true}(\text{cell}(c, 1, x))$$

are evaluated as $p$ or $1 - p$ depending on whether or not they are entailed by the game description in combination with the state $s$ according to Equation 5.5.
Two degrees of freedom

- Value for \( p \)
- Actual t-norm and t-conorm
The value of $p$

- How about $p=1$?
The value of $p$

- How about $p=1$?
- Problem:

```prolog
1 goal(player, 100) :-
  true(on(a,b)), true(on(b,c)), true(ontable(c)).
```

- Just let $p<1$
For example, consider the state \( s = \{\text{ontable}(a), \text{on}(b,c), \text{ontable}(c)\} \) in which two of the three atoms of the goal condition above hold. The evaluation of the goal condition yields:

\[
\text{eval}(\text{goal}(\text{player}, 100), s) = \text{eval}(\text{true}(\text{on}(a, b)) \land \\
\text{true}(\text{on}(b, c)) \land \\
\text{true}(\text{ontable}(c)))
\]

\[
= \top(\text{eval}(\text{true}(\text{on}(a, b)), s), \\
\top(\text{eval}(\text{true}(\text{on}(b, c)), s), \\
\text{eval}(\text{true}(\text{ontable}(c)), s)))
\]

\[
= \top(1 - p, \top(p, p))
\]
The choice of T-norm and T-conorm

• Difference between min/max and others?
• There is another problem:

\[ eval(a_1 \land \ldots \land a_n, s) \to 0 \text{ for } n \to \infty. \]
The choice of T-norm and T-conorm

\[ eval(a_1 \land \ldots \land a_n, s) \rightarrow 0 \text{ for } n \rightarrow \infty. \]

1. goal(player, 100) :- % goal A
2. true(on(a,b)), true(on(b,c)), true(ontable(c)).
3. goal(player, 100) :- % goal B
4. true(on(block1,block2)),
5. true(on(block2,block3)),
6. ...
7. true(on(block999,block1000)),
8. true(ontable(block1000)).

- \( s_1 = \{ \text{ontable}(a), \text{on}(b, c), \text{ontable}(c), \text{ontable}(\text{block1}), \ldots, \text{ontable}(\text{block1000}) \} \) That is, 2 of the 3 atoms of “goal A” are fulfilled but only one of the 1000 atoms of “goal B”.
- \( s_2 = \{ \text{ontable}(a), \text{ontable}(b), \text{ontable}(c), \text{on}(\text{block1,block2}), \text{on}(\text{block2,block3}), \ldots, \text{on}(\text{block999,block1000}) \} \) That is, only one of the 3 atoms of “goal A” are fulfilled but all of the 1000 atoms of “goal B”.
The choice of T-norm and T-conorm

• With t-norm a*b:

\[
\begin{align*}
\text{eval}(\text{goal}(\text{player}, 100), s_1) &= \perp(p^2 * (1 - p), p * (1 - p)^{999}) \\
&= p^2 * (1 - p) + p * (1 - p)^{999} - p^3 * (1 - p)^{1000} \\
&\approx p^2 * (1 - p) \\
\text{eval}(\text{goal}(\text{player}, 100), s_2) &= \perp(p * (1 - p)^2, p^{1000}) \\
&= p * (1 - p)^2 + p^{1000} - p^{1001} * (1 - p)^2 \\
&\approx p * (1 - p)^2 \\
\text{eval}(\text{goal}(\text{player}, 100), s_1) &> \text{eval}(\text{goal}(\text{player}, 100), s_2) \text{ although the goal is already fulfilled in } s_2 \text{ but not in } s_1.
\end{align*}
\]

The approximate values \( p^2 * (1 - p) \) and \( p * (1 - p)^2 \) result from the fact that \( p * (1 - p)^{999} - p^3 * (1 - p)^{1000} \) and \( p^{1000} - p^{1001} * (1 - p)^2 \) are nearly zero. Thus, \( \text{eval}(\text{goal}(\text{player}, 100), s_1) > \text{eval}(\text{goal}(\text{player}, 100), s_2) \) although the goal is already fulfilled in \( s_2 \) but not in \( s_1 \).

\[
\begin{align*}
\top(a, b) &= \begin{cases} 
\max(\top'(a, b), t) & \text{if } \min(a, b) > 0.5 \\
\top'(a, b) & \text{otherwise}
\end{cases} \\
\perp(a, b) &= 1 - \top(1 - a, 1 - b)
\end{align*}
\]
State evaluation function

- So far, the degree of truth of a formula can be measured
- How to evaluate a state (with respect to multiple goals)?
- For a role $r$ in state $s$, $h(r,s)$ is defined as:

\[
h(r, s) = \frac{100}{\sum_{gv \in GV} gv} \cdot \sum_{gv \in GV} gv \cdot h(r, gv, s)
\]

\[
h(r, gv, s) = \begin{cases} 
\text{eval}\left(\text{goal}(r, gv) \lor \text{terminal}, s\right) & \text{if } D \cup s^{\text{true}} \models \text{goal}(r, gv) \\
\text{eval}\left(\text{goal}(r, gv) \land \neg \text{terminal}, s\right) & \text{else}
\end{cases}
\]
Identifying syntactic structures
• So far, only binary fluents
• Cannot distinguish pos(1) from pos(7), if the goal is pos(8)
• Evaluation function should reflect the distance to the goal
Computing domains

- Computation of a domain graph

```
1 succ(0, 1).
2 succ(1, 2).
3 succ(2, 3).
4 init(step(0)).
5 next(step(X)) :-
6    true(step(Y)),
7    succ(Y, X)).

dom(step, 1) = dom(succ, 1) = dom(succ, 2) = \{0, 1, 2, 3\}
dom(init, 1) = dom(next, 1) = dom(true, 1) = \{step/1\}
```
Static Structures: Successor/Order relations

• For every binary relation:

1. Compute the domains $\text{dom}(r, 1)$ and $\text{dom}(r, 2)$ of both arguments of $r$ using the algorithm described in Section 5.2.1.
2. If $\text{dom}(r, 1) \neq \text{dom}(r, 2)$, we do not consider $r$ as successor or order relation, otherwise.
3. If for all $x, y, z \in \text{dom}(r, 1)$, the two properties

$$D \models r(x, y) \land r(y, x) \supset x = y$$ (antisymmetry) and

$$D \models r(x, y) \land r(y, z) \supset r(x, z)$$ (transitivity)

hold, then $r$ is an order relation.

4. If for all $x, y, z \in \text{dom}(r, 1)$, the three properties

$$D \models r(x, y) \land r(y, x) \supset x = y$$ (antisymmetry),

$$D \models r(x, y) \land r(x, z) \supset y = z$$ (functional), and

$$D \models r(y, x) \land r(z, x) \supset y = z$$ (injective)

hold, then $r$ is a successor relation.
Dynamic Structures

• Game boards
  – Criteria:
    • some arguments of a predicate are ordered
    • Distinguished input/output arguments

• Quantities
  – Fluent that has an ordered output argument
  – (step counter are quantities as well)
Distance Estimates for fluents and states
Disadvantages of previous approaches

- Distances are computed based on predefined metrics on the board
  - Do not take into account the type of piece to be moved
New approach

• Compute an admissible heuristic for the number of steps necessary to make a fluent true

• Advantages
  – Does not depend on syntactic pattern
  – No internal simulation of games needed
  – Not limited to board-like structures
Example game

```
role(xplayer). role(oplayer).

init(cell(1,1,b)). init(cell(1,2,b)).
init(cell(1,3,b)). ...
init(cell(1,3,b)). init(control(xplayer)).

legal(P, mark(X, Y)) :-
  true(control(P)), true(cell(X, Y, b)).
legal(P,noop) :-
  role(P), not true(control(P)).

next(cell(X,Y,x)) :- does(xplayer,mark(X,Y)).
next(cell(X,Y,o)) :- does(oplayer,mark(X,Y)).
next(cell(X,Y,C)) :-
  true(cell(X,Y,C)), distinct(C, b).
next(cell(X,Y,b)) :- true(cell(X,Y,b)),
  does(P, mark(M, N)),
  (distinct(X, M) ; distinct(Y, N)).

goal(xplayer, 100) :- line(x).
...
terminal :- line(x) ; line(o) ; not open.

line(P) :- true(cell(X, 1, P)),
  true(cell(X, 2, P)), true(cell(X, 3, P)).
...
open :- true(cell(X, Y, b)).
```
**Definition 1 (Game).** Let $\Sigma$ be a set of ground terms and $2^\Sigma$ denote the set of finite subsets of $\Sigma$. A game over this set of ground terms $\Sigma$ is a state transition system $\Gamma = (R, s_0, T, l, u, g)$ over sets of states $S \subseteq 2^\Sigma$ and actions $A \subseteq \Sigma$ with

- $R \subseteq \Sigma$, a finite set of roles;
- $s_0 \in S$, the initial state of the game;
- $T \subseteq S$, the set of terminal states;
- $l : R \times A \times S$, the legality relation;
- $u : (R \mapsto A) \times S \rightarrow S$, the transition or update function; and
- $g : R \times S \mapsto \mathbb{N}$, the reward or goal function.

$$u(A, s) = \{ f \in \Sigma : D \cup s^{\text{true}} \cup A^{\text{does}} \models \text{next}(f) \}$$
Question: How do fluents evolve over time?

- Construct a fluent-graph
  - Each fluent of a game is a node
  - A directed edge \((f_i, f)\) is added, if at least one of the predecessor states must hold in the current state, for \(f\) to hold in the next state.

Figure 1: Partial fluent graph for Tic-Tac-Toe.
Fluent graph: definition

Definition 2 (Fluent Graph). Let $\Gamma$ be a game over ground terms $\Sigma$. A graph $G = (V, E)$ is called a fluent graph for $\Gamma$ iff

- $V = \Sigma \cup \{\emptyset\}$ and
- for all fluents $f \in \Sigma$, two valid states $s$ and $s'$

\[(s' \text{ is a successor of } s) \land f' \in s' \Rightarrow (\exists f)(f, f') \in E \land (f \in s \cup \{\emptyset\})\]  

Why do they need a new symbol?
The extra symbol ...

- A fake node to capture fluents in the game that do not have preconditions
  - For instance:
    - next(g):-distinct(a,b)
- In addition, some fluents might not be connected at all
  - For instance:
    - next(g):- distinct(a,a)
Are fluent graphs unique?
Are fluent graphs unique?

• No
• Only some of the necessary preconditions are covered

Figure 2: Alternative partial fluent graph for Tic-Tac-Toe.
Definition 3 (Distance Function). Let $\Delta_G(f, f')$ be the length of the shortest path from node $f$ to node $f'$ in the fluent graph $G$ or $\infty$ if there is no such path. Then

$$\Delta(s, f') = \min_{f \in s \cup \{\emptyset\}} \Delta_G(f, f')$$

Intuitively, each edge $(f, f')$ in the fluent graph corresponds to a state transition of the game from a state in which $f$ holds to a state in which $f'$ holds.

Thus, the length of a path from $f$ to $f'$ in the fluent graph corresponds to the number of steps in the game between a state containing $f$ to a state containing $f'$. 
Admissible heuristic

• The distance function is a lower bound on the actual number of steps
• Thus, it is an admissible heuristic!
• Before we show how the heuristic is applied, we look into the construction of fluent graphs from rules
From Rules to Fluent Graphs

For each ground fluent $f'$ of the game:

1. Construct a ground disjunctive normal form $\phi$ of $\text{next}(f')$, i.e., a formula $\phi$ such that $\text{next}(f') \supset \phi$.

2. For every disjunct $\psi$ in $\phi$:
   - Pick one literal $\text{true}(f)$ from $\psi$ or set $f = \emptyset$ if there is none.
   - Add the edge $(f, f')$ to the fluent graph.
How to pick a literal?

- The distance function is admissible (lower bound)!
- What is the goal ... ?
How to pick a literal?

- The distance function is admissible (lower bound)!
- One wants to increase the shortest paths lengths as much as possible, while still remaining admissible
  - The fluent graph contains as few edges as possible, but as many as strictly necessary
- The other extreme: complete fluent graph
Two open problems

1. How to construct a ground formula in DNF?
2. Which true(f) is selected?
Algorithm 1 Constructing a formula $\phi$ in DNF with $\text{next}(f') \supset \phi$.

Input: game description $D$, ground fluent $f'$

Output: $\phi$, such that $\text{next}(f') \supset \phi$

1: $\phi := \text{next}(f')$
2: $\text{finished} := \text{false}$
3: while $\neg \text{finished}$ do
4: Replace every positive occurrence of $\text{does}(r, a)$ in $\phi$ with $\text{legal}(r, a)$.
5: Select a positive literal $l$ from $\phi$ such that $l \neq \text{true}(t), l \neq \text{distinct}(t_1, t_2)$ and $l$ is not a recursively defined predicate.
6: if there is no such literal then
7: $\text{finished} := \text{true}$
8: else
9: $\tilde{l} := \bigvee_{h: \exists b \in D, l\sigma = h\sigma} b\sigma$
10: $\phi := \phi\{l/\tilde{l}\}$
11: end if
12: end while
13: Transform $\phi$ into disjunctive normal form, i.e., $\phi = \psi_1 \lor \ldots \lor \psi_n$ and each formula $\psi_i$ is a conjunction of literals.
14: for all $\psi_i$ in $\phi$ do
15: Replace $\psi_i$ in $\phi$ by a disjunction of all ground instances of $\psi_i$.
16: end for
Tic-Tac-Toe example

1. \texttt{next(cell(M,N,x)) :- does(xplayer,mark(M,N)).}
2. \texttt{next(cell(M,N,o)) :- does(oplayer,mark(M,N)).}
3. \texttt{next(cell(M,N,C)) :- true(cell(M,N,C)),}
   \texttt{does(P,mark(X,Y)), distinct(X,M).}
4. \texttt{next(cell(M,N,C)) :- true(cell(M,N,C)),}
   \texttt{does(P,mark(X,Y)), distinct(Y,N).}

Formula for \texttt{next(cell(c,1,X))}?
Tic-Tac-Toe example

1. next(cell(M,N,x)) :- does(xplayer,mark(M,N)).
2. next(cell(M,N,o)) :- does(oplayer,mark(M,N)).
3. next(cell(M,N,C)) :- true(cell(M,N,C)), does(P,mark(X,Y)), distinct(X,M).
4. next(cell(M,N,C)) :- true(cell(M,N,C)), does(P,mark(X,Y)), distinct(Y,N).

Formula for next(cell(c,1,X))? 

\[ \phi = \text{does(xplayer,mark(c,1))} \lor \text{true(cell(c,1,x))} \land \text{does(P,mark(X,Y))} \land \text{distinct(X,c)} \lor \text{true(cell(c,1,x))} \land \text{does(P,mark(X,Y))} \land \text{distinct(Y,1)} \]
\[
\phi = \text{does(xplayer,mark(c,1))} \lor \\
\text{true(cell(c,1,x))} \land \text{does(xplayer,mark(a,1))} \land \text{distinct(a,c)} \lor \\
\text{true(cell(c,1,x))} \land \text{does(xplayer,mark(b,1))} \land \text{distinct(b,c)} \lor \\
\ldots \lor \\
\text{true(cell(c,1,x))} \land \text{does(xplayer,mark(c,3))} \land \text{distinct(c,c)} \lor \\
\text{true(cell(c,1,x))} \land \text{does(xplayer,mark(a,1))} \land \text{distinct(1,1)} \lor \\
\ldots \lor \\
\text{true(cell(c,1,x))} \land \text{does(xplayer,mark(c,3))} \land \text{distinct(3,1)} \lor \\
\text{true(cell(c,1,x))} \land \text{does(oplayer,mark(a,1))} \land \text{distinct(a,c)} \lor \\
\ldots \lor \\
\text{true(cell(c,1,x))} \land \text{does(oplayer,mark(c,3))} \land \text{distinct(c,c)} \lor \\
\text{true(cell(c,1,x))} \land \text{does(oplayer,mark(a,1))} \land \text{distinct(1,1)} \lor \\
\ldots \lor \\
\text{true(cell(c,1,x))} \land \text{does(oplayer,mark(c,3))} \land \text{distinct(3,1)}
\]
Tic-Tac-Toe example (legal instead of does!)

1. `next(cell(M,N,x)) :- does(xplayer,mark(M,N)).`
2. `next(cell(M,N,o)) :- does(oplayer,mark(M,N)).`
3. `next(cell(M,N,C)) :- true(cell(M,N,C)),
   does(P,mark(X,Y)), distinct(X,M).`
4. `next(cell(M,N,C)) :- true(cell(M,N,C)),
   does(P,mark(X,Y)), distinct(Y,N).`

\[ \phi = \text{true(control(xplayer))} \land \text{true(cell(c,1,blank))} \]
\[ \lor \]
\[ \text{true(cell(c,1,x))} \land \text{true(control(P))} \land \text{true(cell(X,Y,blank))} \]
\[ \land \text{distinct(X,c)} \]
\[ \lor \]
\[ \text{true(cell(c,1,x))} \land \text{true(control(P))} \land \text{true(cell(X,Y,blank))} \]
\[ \land \text{distinct(Y,1)} \]
Selecting Preconditions for the Fluent Graph

1. Add only edges if necessary
   - Use existing edges, if possible
2. Prefer \text{true}(f) over \text{true}(g), if \(f\) is more similar to \(f'\) than \(g\) is to \(f'\)

We define the similarity \(\text{sim}(t, t')\) recursively over ground terms \(t, t'\):

\[
\text{sim}(t, t') = \begin{cases} 
1: & t, t' \text{ have arity 0 and } t = t' \\
\sum_i \text{sim}(t_i, t'_i): & t = f(t_1, \ldots, t_n) \text{ and } \\
0: & t' = f(t'_1, \ldots, t'_n) \\
& \text{else}
\end{cases}
\]
How to apply distance measures for search?
Distances in Tic-Tac-Toe

Figure 3: Two states of the Tic-Tac-Toe. The first row is still open in state $s_1$ but blocked in state $s_2$.

\begin{verbatim}
1  line(x) :- true(cell(1,1,x)),
2       true(cell(2,1,x)), true(cell(3,1,x)).
\end{verbatim}
Distances in Breakthrough

Figure 4: Initial position in Breakthrough and the move options of a pawn.

```
1 goal(black, 100) :-
2   index(X), true(cellholds(X, 1, black)).
```
Distances in Breakthrough

Figure 5: A partial fluent graph for Breakthrough, role black.
Evaluation

Figure 6: Advantage in Win Rate of flux_distance
Summary

• Search is much more goal-directed
• However, smaller number of evaluated states/second
• Grounding of goal formulae is main problem
  – Out-of-memory or time-out
• Possible solution: non-grounded fluent-graphs
Proving properties of games
Main question

• How can a general game player *deduce* that a fact is true for all finitely reachable states?
  – For instance, the feature control(P) in Tic-Tac-Toe is unique in every single state => Ability to identify turn-taking games
  – Z is output for cell(X,Y,Z); so far no proof!

• Solution?
Main question

• How can a general game player *deduce* that a fact is true for all finitely reachable states?
  – For instance, the feature control(P) in Tic-Tac-Toe is unique in every single state => Ability to identify turn-taking games
  – Z is output for cell(X,Y,Z); so far no proof!

• Solution?

• Induction:
  – Property p holds in the initial state
  – If p holds in the current state, then p holds in all successor states

• Use answer set programming …
Example for induction with answer set progr.

• Proof of initial state is trivial

```prolog
1 init(cell(a,1,blank)). ... init(cell(c,3,blank)).
2 init(control(xplayer)).
3 cdom(xplayer).
4 cdom(oplayer).
5 phi_init :- 1 { init(control(X)) : cdom(X) } 1.
6 :- phi_init.
```
Example for induction

- Inductive step:

```prolog
1 dom_control(xplayer). dom_control(oplayer).
2 dom_cell1(a). dom_cell1(b). dom_cell1(c).
3 dom_cell2(1). dom_cell2(2). dom_cell2(3).
4 dom_cell3(x). dom_cell3(o). dom_cell3(blank).
5
6 dom_fluent(control(X)) :- dom_control(X).
7 dom_fluent(cell(X,Y,C)) :-
8     dom_cell1(X), dom_cell2(Y), dom_cell3(C).
9
10 dom_move(mark(X,Y)) :- dom_cell1(X), dom_cell2(Y).
11 dom_move(nocp).
12
13 0 { true(F) : dom_fluent(F) }.
14
15 1 { does(R,M) : dom_move(M) } 1 :- role(R).
16 :- does(R,M), not legal(R,M).
17
18 phi_true :- 1 { true(control(X)) : dom_control(X) } 1.
19 :- not phi_true.
20
21 phi_next :- 1 { next(control(X)) : dom_control(X) } 1.
22 :- phi_next.
```
Summary FluxPlayer

• Key techniques:
  – Modeling of degree of truth of formulae with fuzzy logic
  – Estimation for similarity of states w.r.t. goals states
  – Extraction of static/dynamic structures from game descriptions
  – Shortest paths in fluent graphs as lower bound for state distances
  – Proving properties of games using induction


Competition history

- 2005: Cluneplayer, by Jim Clune (UCLA)
- 2006: Fluxplayer, by Stephan Schiffel and Michael Thielscher (Dresden University of Technology)
- 2007: Cadiaplayer, by Yngvi Björnsson and Hilmar Finnsson (Reykjavik University)
- 2008: Cadiaplayer, by Yngvi Björnsson, Hilmar Finnsson and Gylfi Þór Guðmundsson (Reykjavik University)
- 2009: Ary, by Jean Méhat (Paris 8 University)
- 2010: Ary, by Jean Méhat (Paris 8 University)
- 2011: TurboTurtle, by Sam Schreiber
- 2012: Cadiaplayer, by Hilmar Finnsson and Yngvi Björnsson (Reykjavik University)
- 2013: TurboTurtle, by Sam Schreiber
Acknowledgements

Main source for the slides was:
Stephan Schiffel: Knowledge-based General Game Playing, dissertation at TU Dresden

In addition:
Michulke et.al.: Distance features for General Game Playing (GIGA 2011)