KR-Techniques for General Game Playing

Michael Thielscher

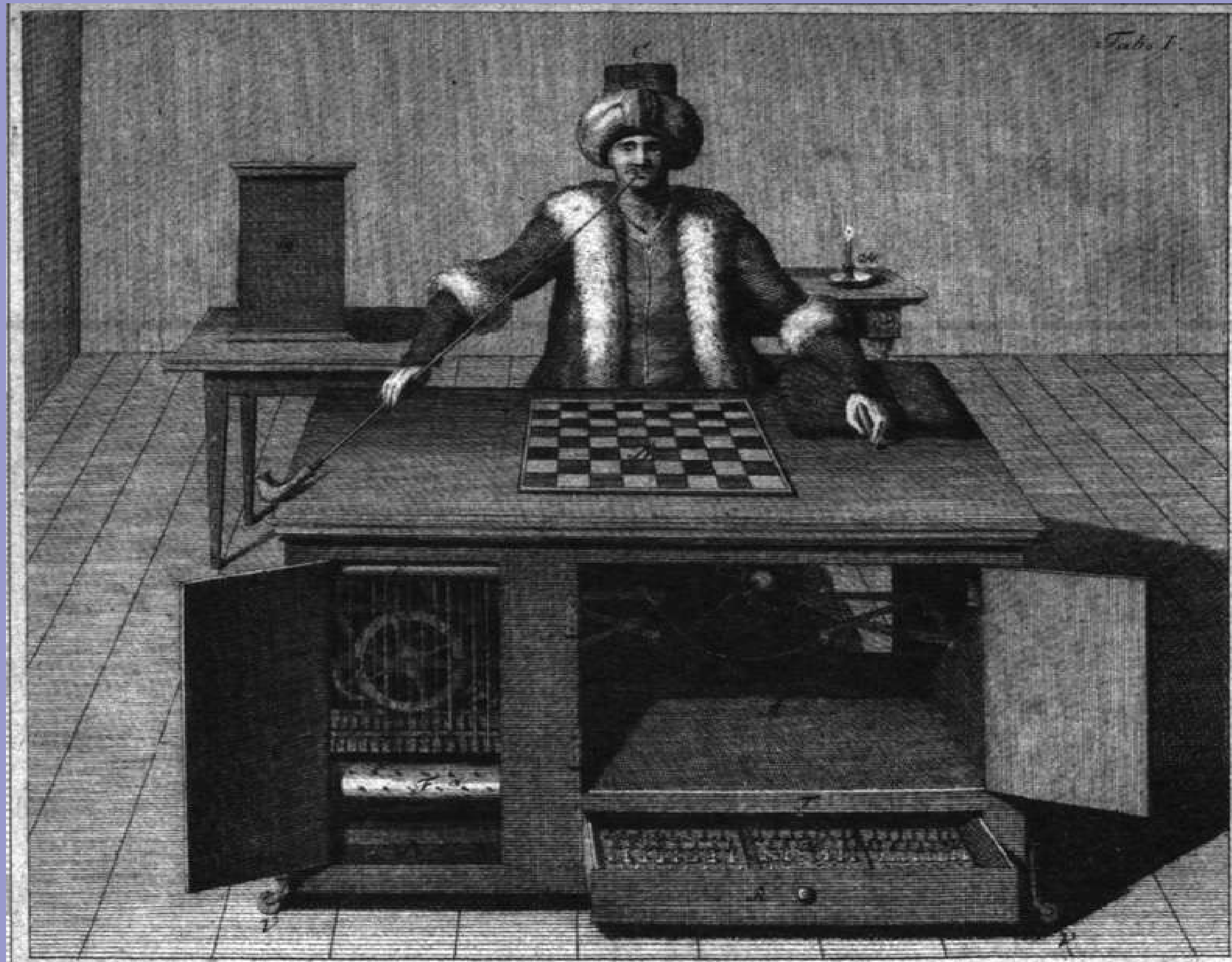
Roadmap

1. General Game Playing – a Grand AI Challenge

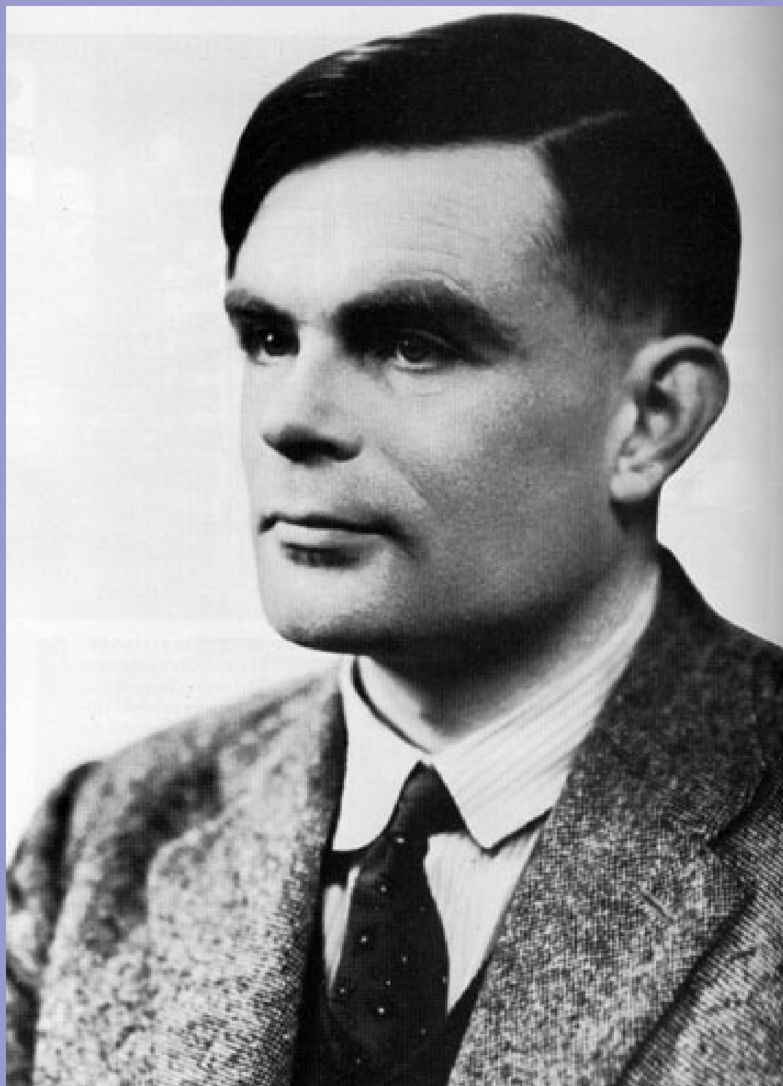
2. KR-Aspects
   - Formalizing game rules:
     Compact representations of state machines
   - Challenge I:
     Mapping game descriptions to efficient representations
   - Extracting useful knowledge from game descriptions
   - Challenge II:
     Proving properties of games

3. Further Aspects: Search + Learning

The Turk (18th Century)



Alan Turing & Claude Shannon (~1950)



Deep-Blue Beats World Champion (1997)

Definition

In the early days, game playing machines were considered a key to Artificial Intelligence (AI).

But chess computers are highly specialized systems. Deep-Blue's intelligence was limited. It couldn't even play a decent game of Tic-Tac-Toe or Rock-Paper-Scissors.

A General Game Player is a system that

- understands formal descriptions of arbitrary strategy games
- learns to play these games well without human intervention

General Game Playing - A Grand AI Challenge

Rather than being concerned with a specialized solution to a narrow problem, General Game Playing encompasses a variety of AI areas.

General Game Playing and AI

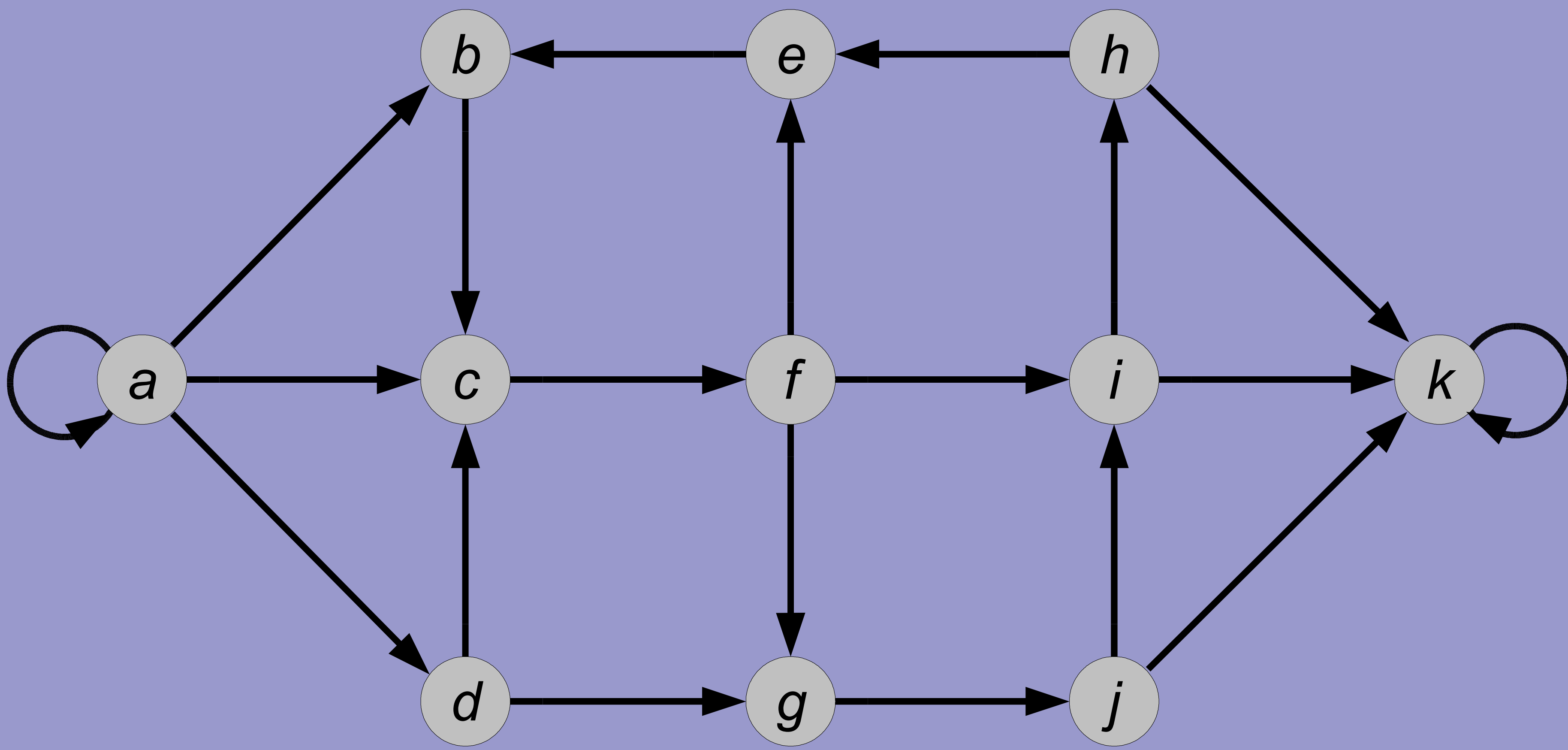
Agents	Games
Competitive environments	Deterministic, complete information
Uncertain environments	Nondeterministic, partially observable
Unknown environment model	Rules partially unknown
Real-world environments	Robotic player

Knowledge Representation for Games

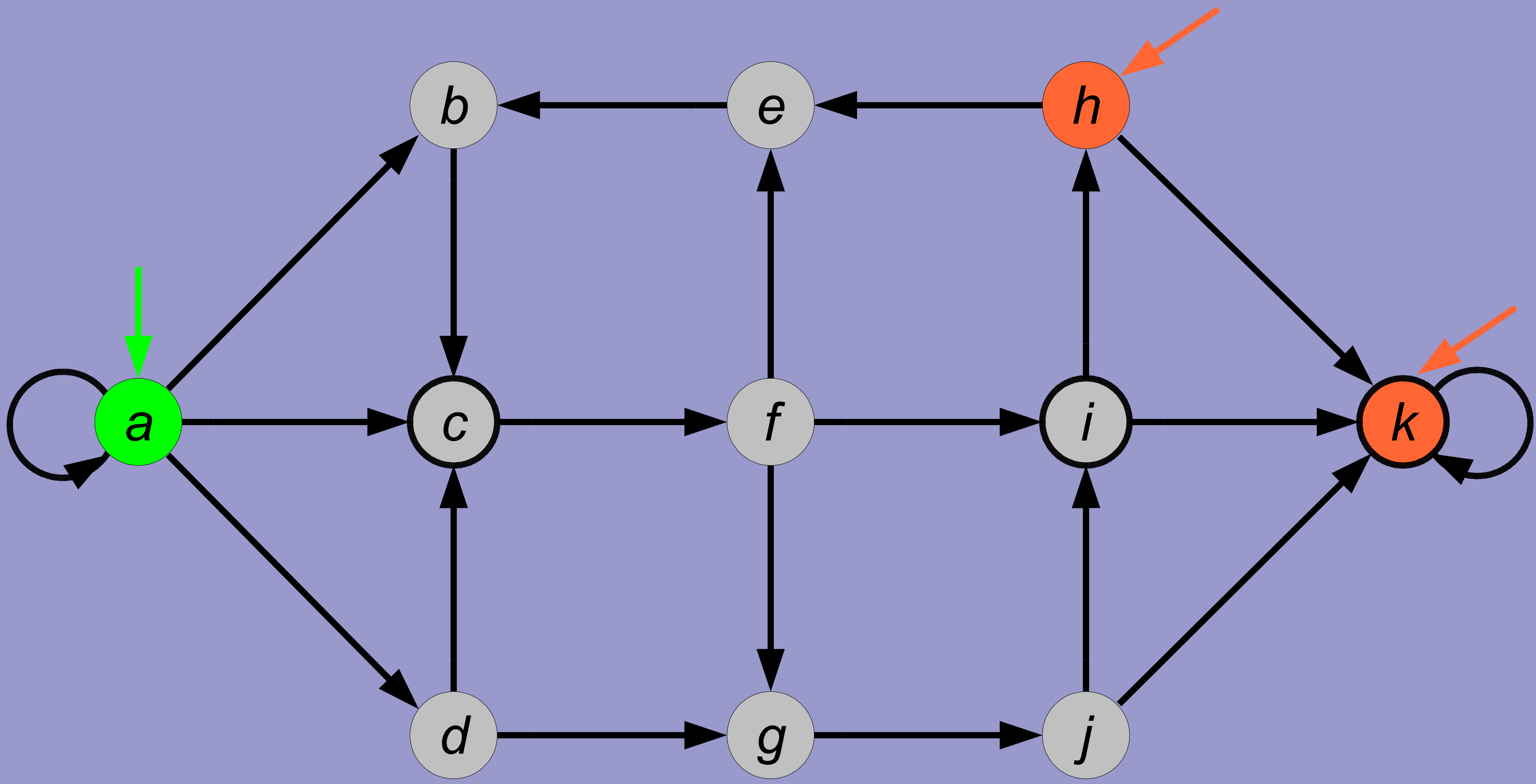
–

The Game Description Language

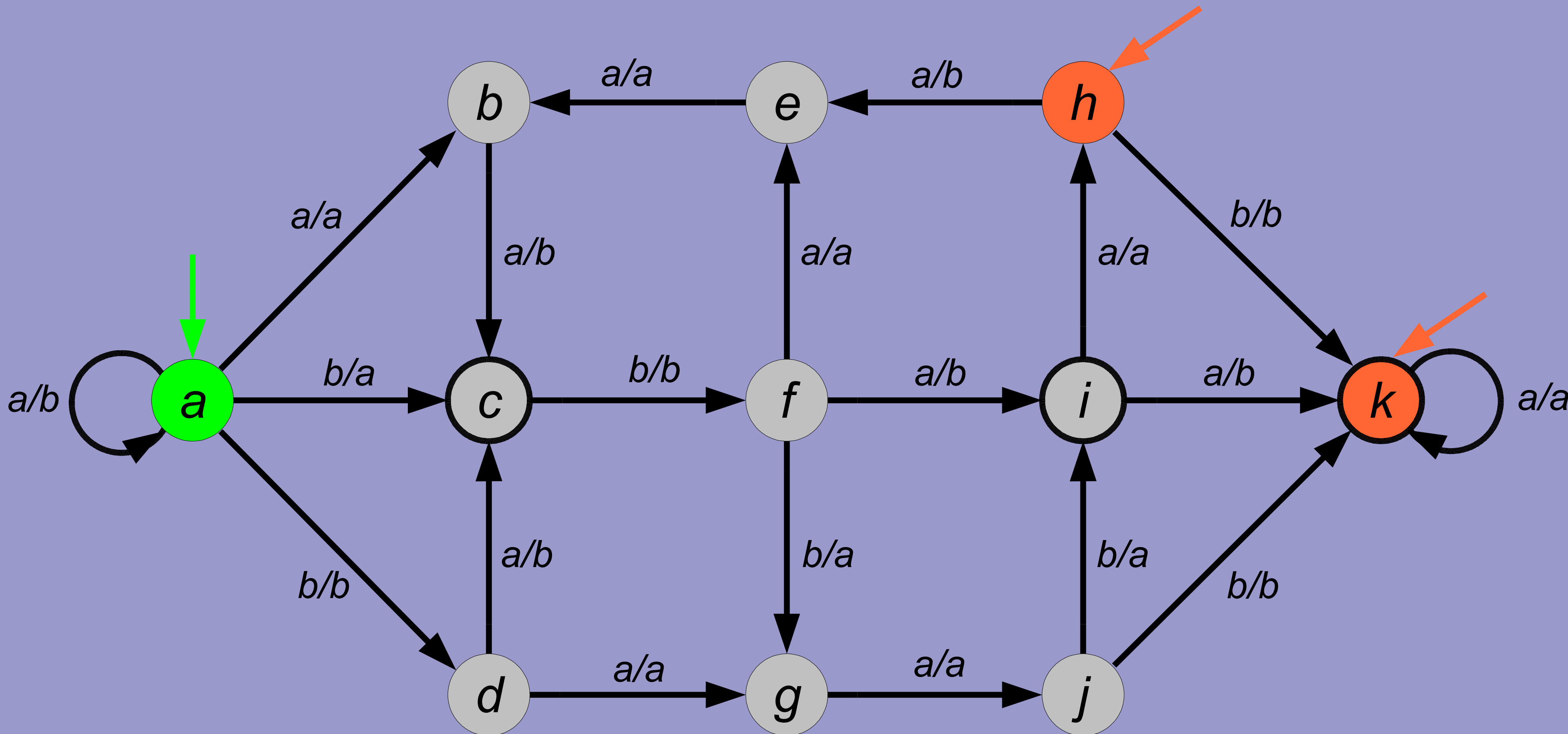
Games as State Machines



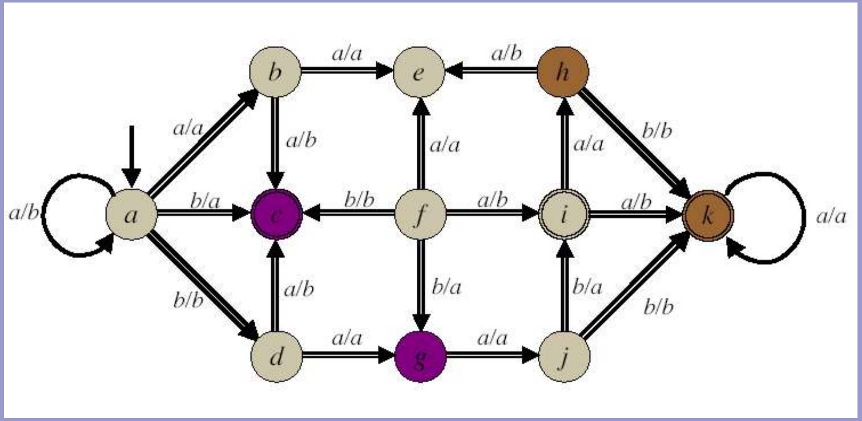
Initial Position and End of Game



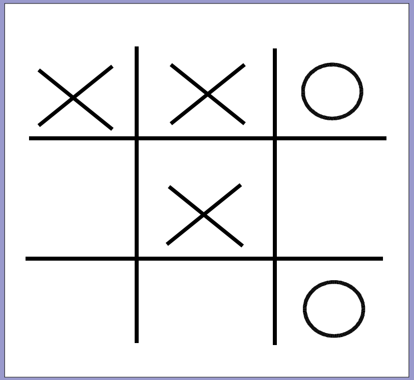
Simultaneous Moves



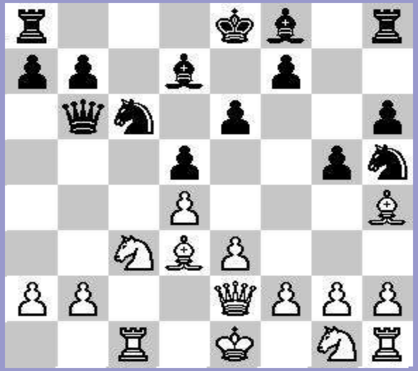
Every finite game can be modeled as a state transition system



But direct encoding impossible in practice

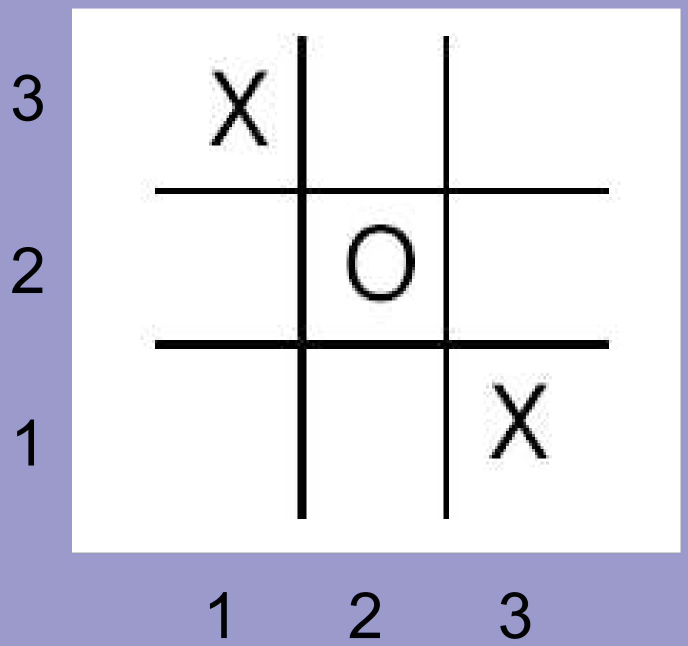


19,683 states



~ 10^{43} legal positions

Modular State Representation: Fluents



`cell(X,Y,M)`

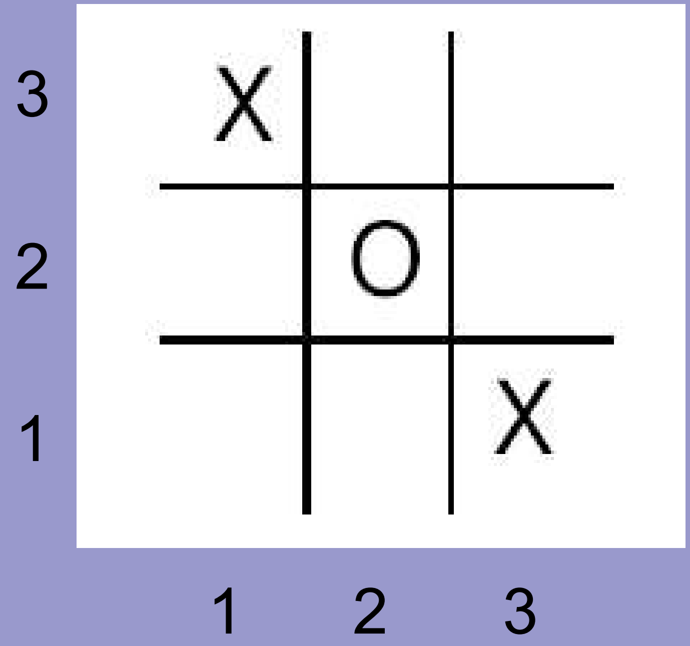
X,Y ∈ {1,2,3}

M ∈ {x,o,b}

`control(P)`

P ∈ {xplayer,oplayer}

Actions



`mark(X,Y)`

X,Y $\in$ **{1,2,3}**

`noop`

Tic-Tac-Toe Game Model

Symbolic expressions: {`xplayer`, `oplayer`, `cell(1,1,b)`, `noop`, ...}

- roles {`xplayer`, `oplayer`}
- initial s_1 = {`cell(1,1,b)`, ..., `cell(3,3,b)`, `control(oplayer)`}
- legal actions {(`xplayer`, `mark(1,1)`, s_1), ..., (`oplayer`, `noop`, s_1), ...}
- update ⟨(⟨`xplayer` ↦ `mark(1,1)`, `oplayer` ↦ `noop`⟩, s_1)
  ↦ {`cell(1,1,x)`, ..., (`cell(3,3,b)`, `control(oplayer)`}⟩,
  ...
- terminals {t_1 = {`cell(1,1,x)`, `cell(1,2,x)`, `cell(1,3,x)`, ...}, ...}
- goal {(`xplayer`, t_1, 100), (`oplayer`, t_1, 0), ...}

Symbolic Game Model

Let Σ be a countable set of ground expressions.

A game is a structure (R, l, u, s_1, t, g)

- $R \in 2^{\Sigma}$ roles
- $l \subseteq R \times \Sigma \times 2^{\Sigma}$ legal actions
- $u: (R \mapsto \Sigma) \times 2^{\Sigma} \mapsto 2^{\Sigma}$ update
- $s_1 \in 2^{\Sigma}$ initial position
- $t \subseteq 2^{\Sigma}$ terminal positions
- $g \subseteq R \times 2^{\Sigma} \times \mathbb{N}$ goal relation

where 2^{Σ} := finite subsets of Σ

Game Description Language GDL

A game description is a stratified, allowed logic program whose signature includes the following game-independent vocabulary:

```
role(player)
init(fluent)
true(fluent)
does(player,move)
next(fluent)
legal(player,move)
goal(player,value)
terminal
```

Describing a Game: Roles

A GDL description P encodes the roles $R = \{\sigma \in \Sigma : P \models \texttt{role}(\sigma)\}$

```
role(xplayer) <=
role(oplayer) <=
```

Describing a Game: Initial Position

A GDL description *P* encodes $s_1 = \{\sigma \in \Sigma : P \models \texttt{init}(\sigma)\}$

```
init(cell(1,1,b)) <=
init(cell(1,2,b)) <=
init(cell(1,3,b)) <=
init(cell(2,1,b)) <=
init(cell(2,2,b)) <=
init(cell(2,3,b)) <=
init(cell(3,1,b)) <=
init(cell(3,2,b)) <=
init(cell(3,3,b)) <=
init(control(xplayer)) <=
```

Preconditions

For $S \subseteq \Sigma$ let $S^{\texttt{true}} := \{\texttt{true}(\sigma) : \sigma \in S\}$

then P encodes $I = \{(r, \sigma, S) : P \cup S^{\texttt{true}} \models \texttt{legal}(r, \sigma)\}$

```
legal(P,mark(X,Y))  <=  true(cell(X,Y,b)) ∧
                        true(control(P))

legal(xplayer,noop) <=  true(cell(X,Y,b)) ∧
                        true(control(oplayer))

legal(oplayer,noop) <=  true(cell(X,Y,b)) ∧
                        true(control(xplayer))
```

Update

For $A : R \mapsto \Sigma$ let $A^{\mathtt{does}} := \{\mathtt{does}(r, A(r)) : r \in R\}$

then P encodes $u(A, S) = \{\sigma : P \cup A^{\mathtt{does}} \cup S^{\mathtt{true}} \models \mathtt{next}(\sigma)\}$

```
next(cell(M,N,x))<= does(xplayer,mark(M,N))

next(cell(M,N,o))<= does(oplayer,mark(M,N))

next(cell(M,N,W))<= true(cell(M,N,W)) ∧ ¬W=b

next(cell(M,N,b))<= true(cell(M,N,b)) ∧
                    does(P,mark(J,K)) ∧ (¬M=J ∨ ¬N=K)

next(control(xplayer)) <= true(control(oplayer))

next(control(oplayer)) <= true(control(xplayer))
```

Termination

P encodes $t = \{S \subseteq \Sigma : P \cup S^{\texttt{true}} \models \texttt{terminal}\}$

```
terminal <= line(x) ∨ line(o)
terminal <= ¬open

line(W) <= row(M,W)
line(W) <= column(N,W)
line(W) <= diagonal(W)

open <= true(cell(M,N,b))
```

Auxiliary Clauses

```
     row(M,W) <= true(cell(M,1,W)) ∧
                 true(cell(M,2,W)) ∧
                 true(cell(M,3,W))

  column(N,W) <= true(cell(1,N,W)) ∧
                 true(cell(2,N,W)) ∧
                 true(cell(3,N,W))

  diagonal(W) <= true(cell(1,1,W)) ∧
                 true(cell(2,2,W)) ∧
                 true(cell(3,3,W))

  diagonal(W) <= true(cell(1,3,W)) ∧
                 true(cell(2,2,W)) ∧
                 true(cell(3,1,W))
```

Goals

P encodes $g = \{(r, S, n): P \cup S^{\texttt{true}} \models \texttt{goal}(r, n)\}$

```
goal(xplayer,100) <= line(x)
goal(xplayer,50)  <= ¬line(x) ∧ ¬line(o) ∧ ¬open
goal(xplayer,0)   <= line(o)

goal(oplayer,100) <= line(o)
goal(oplayer,50)  <= ¬line(x) ∧ ¬line(o) ∧ ¬open
goal(oplayer,0)   <= line(x)
```

Reasoning

Game descriptions are a good example of knowledge representation with formal logic.

Automated reasoning about actions necessary to

- determine legal moves
- update positions
- recognize end of game

Challenge I: Efficient Descriptions

GDL and the Frame Problem

```
next(cell(M,N,x))<= does(xplayer,mark(M,N))
next(cell(M,N,o))<= does(oplayer,mark(M,N))

next(cell(M,N,W))<= true(cell(M,N,W)) ∧ ¬W=b
next(cell(M,N,b))<= true(cell(M,N,b)) ∧
                    does(P,mark(J,K)) ∧ (¬M=J ∨ ¬N=K)

next(control(xplayer)) <= true(control(oplayer))
next(control(oplayer)) <= true(control(xplayer))
```

GDL and the Frame Problem

Effect Axioms

```
next(cell(M,N,x))<= does(xplayer,mark(M,N))
next(cell(M,N,o))<= does(oplayer,mark(M,N))
```

Frame Axioms

```
next(cell(M,N,W))<= true(cell(M,N,W)) ∧ ¬W=b
next(cell(M,N,b))<= true(cell(M,N,b)) ∧
                    does(P,mark(J,K)) ∧ (¬M=J ∨ ¬N=K)
```

Action-Independent Effects

```
next(control(xplayer)) <= true(control(oplayer))
next(control(oplayer)) <= true(control(xplayer))
```

A More Efficient Encoding (PDDL)

```
(:action noop
 :effect (and (when (control xplayer) (control oplayer))
              (when (control oplayer) (control xplayer))) )

(:action mark
 :parameters (?p ?m ?n)
 :effect (and (not cell(?m ?n b))
              (when (= ?p xplayer) (cell(?m ?n x)))
              (when (= ?p oplayer) (cell (?m ?n o)))
              (when (control xplayer) (control oplayer))
              (when (control oplayer) (control xplayer))) )
```

How to Get There?

Using Situation Calculus, the completion of the GDL clauses entails

```
cell(M,N,W,do(mark(xplayer,J,K),S)) <=>
    W=x ∧ M=J ∧ N=K
    ∨ cell(M,N,W,S) ∧ ¬W=b
    ∨ cell(M,N,W,S) ∧ W=b ∧ (¬M=J ∨ ¬N=K)
```

This is equivalent to the (instantiated) Successor State Axiom

```
cell(M,N,W,do(mark(xplayer,J,K),S)) <=>
    W=x ∧ M=J ∧ N=K
    ∨
    cell(M,N,W,S) ∧ ¬(M=J ∧ N=K ∧ W=b)
```

A More Difficult Example

```
succ(0,1)<=
succ(1,2)<=
succ(2,3)<=
init(step(0)) <=
next(step(N)) <= true(step(M)) ∧ succ(M,N)
```

The equivalence

```
step(N,do(P,A,S)) <=> step(M,S) ∧ succ(M,N)
```

does not entail the positive and negative(!) effects

```
(when (and (step ?m) (succ ?m ?n)) (step ?n))
(when (step ?n) (not (step ?n)))
```

Challenge I

Translate GDL effect clauses into an efficient action representation!

- Which formalism?
  Successor state axioms, state update axioms (Fluent Calculus), PDDL, causal laws, ...
- May require to prove state constraints
- Concurrency (for *n*-player games w/ $n \geq 2$)

Challenge II: Proving State Constraints

The Value of Knowledge

Not only are state constraints helpful for better encodings, structural knowledge of a game is crucial for good play.

Examples

- A game is turn-based.
- Each board cell (`X,Y`) has a unique contents `M`.
- Markers `x` and `o` in Tic-Tac-Toe are permanent.
- A game is weakly (strongly) winnable.

Game properties like these can be formalized using ATL; see [W. v. d. Hoek, J. Ruan, M. Wooldridge; 2008]

Induction Proofs

Claim

Fluent `control` has a unique argument in every reachable position.

```
P: init(control(xplayer)) <=
   next(control(xplayer)) <= true(control(oplayer))
   next(control(oplayer)) <= true(control(xplayer))
```

The claim holds if

- uniqueness holds initially, and
- uniqueness holds `next`, provided it is `true` (and every player makes a legal move).

Answer Set Programming

We can use ASP to prove both an induction base and step.

P ∪

```
h0 <= 1{init(control(X)): controldomain1(X)}1
<= h0
```

admits no answer set;

same for

P ∪

```
1{true(control(X)): controldomain1(X)}1 <=
h <= 1{next(control(X)): controldomain1(X)}1
<= h
```

Another Example

Claim

Every board cell has a unique contents.

Let P be the GDL clauses for Tic-Tac-Toe.

P ∪

```
h0(X,Y) <= 1{init(control(X,Y,Z)):
                              celldomain3(Z)}1
h0 <= ¬h0(X,Y)
<= ¬h0
```

admits no answer set.

Another Example (Cont'd)

For the induction step, uniqueness of `control` must be known!

```
P ∪ 1{true(control(X)): controldomain1(X)}1 <=
    1{does(R,A): doesdomain2(A)}1 <=
    <= does(R,A) ∧ ¬legal(R,A)
    1{true(cell(X,Y,Z)): celldomain3(Z)}1 <=
    h(X,Y) <= 1{next(cell(X,Y,Z)): celldomain3(Z)}1
    h <= ¬h(X,Y)
    <= ¬h
```

admits no answer set.

Challenge II

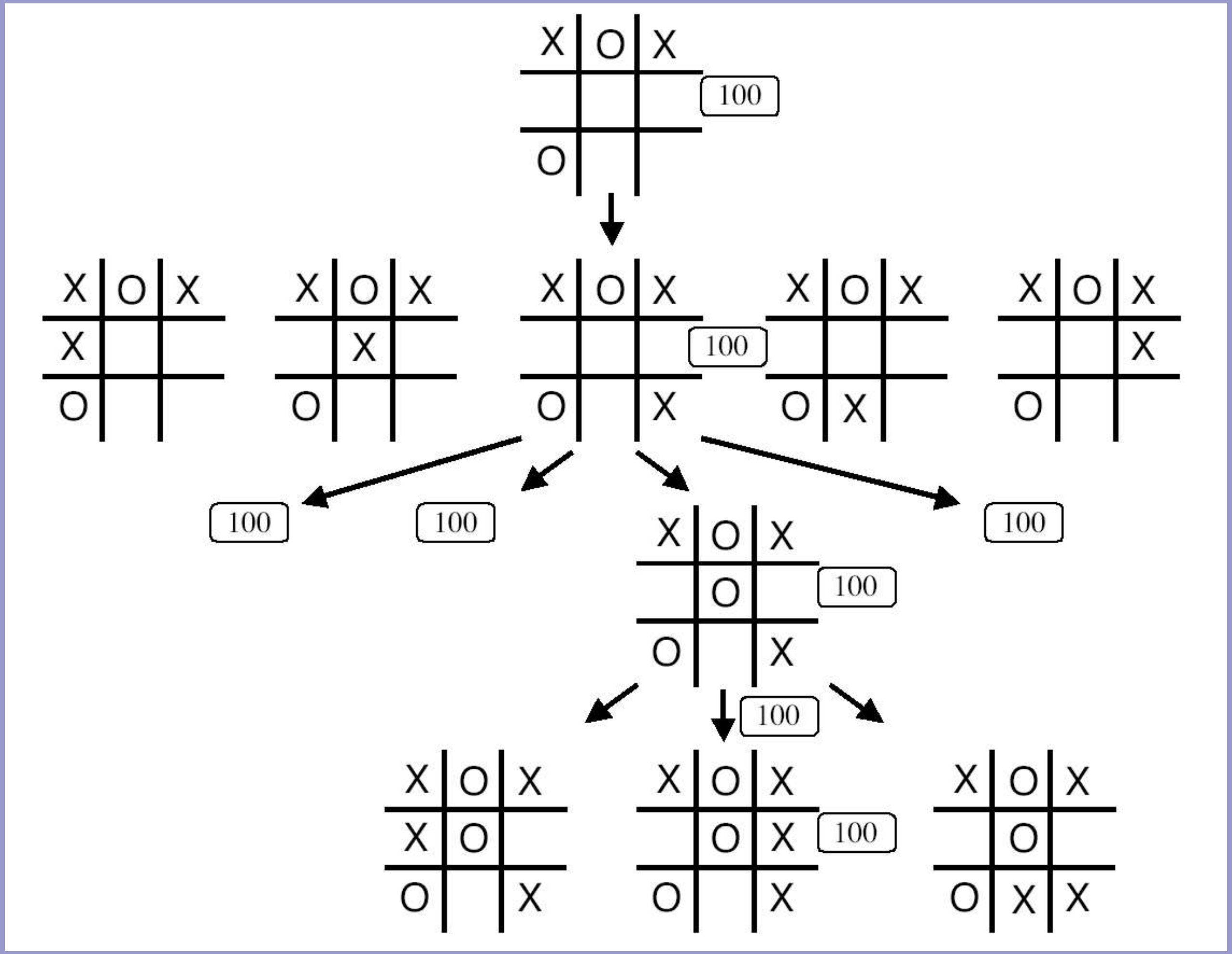
Induction proofs using ASP work fine for reasonably small games.

For complex games, the grounded program becomes too large.

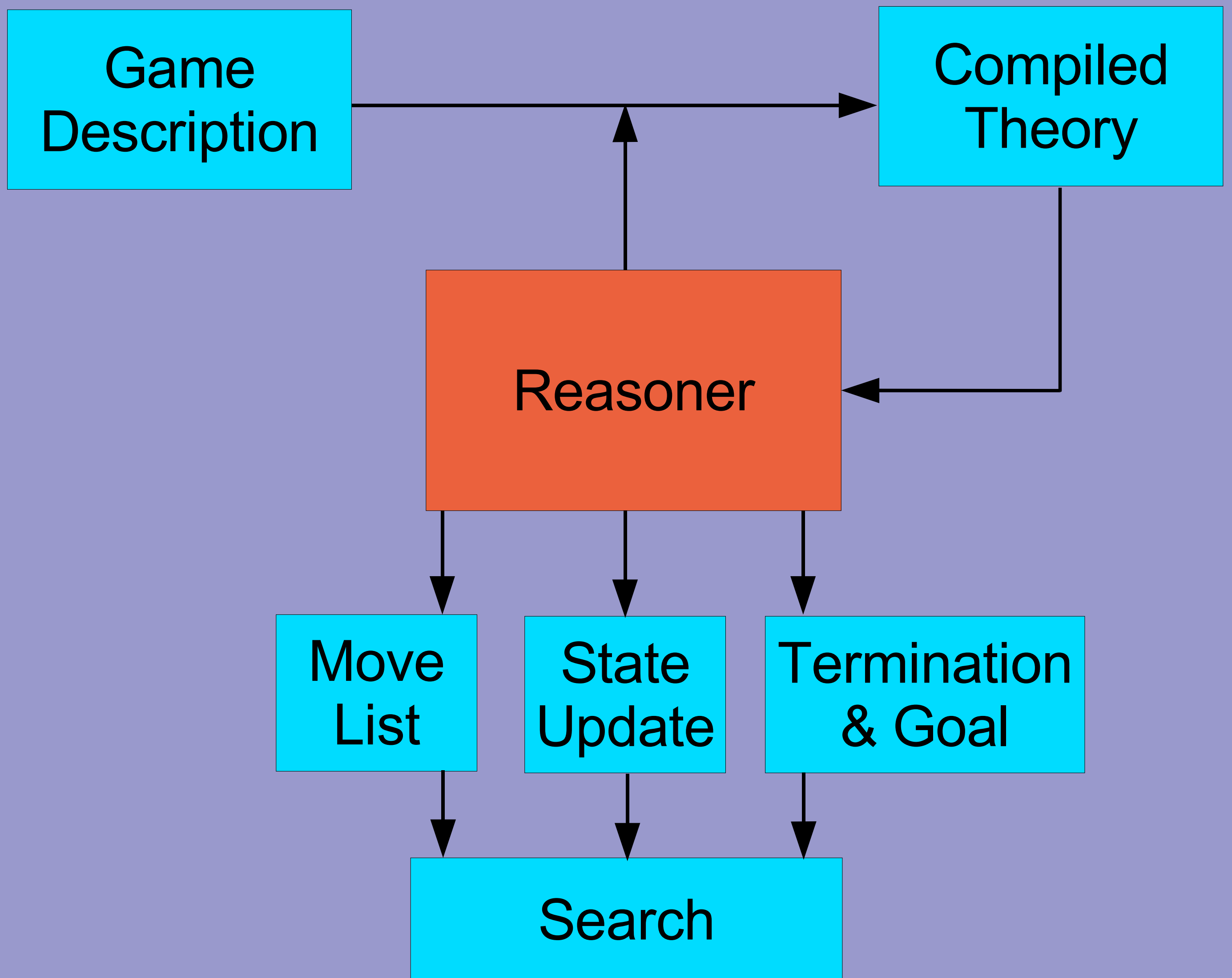
Find a more abstract proof method for GGP!

Planning and Search

Game Tree Search (General Concept)



A General Architecture



Learning

Towards Good Play

Besides efficient inference and search algorithms, the ability to automatically generate a good evaluation function distinguishes good from bad General Game Playing programs.

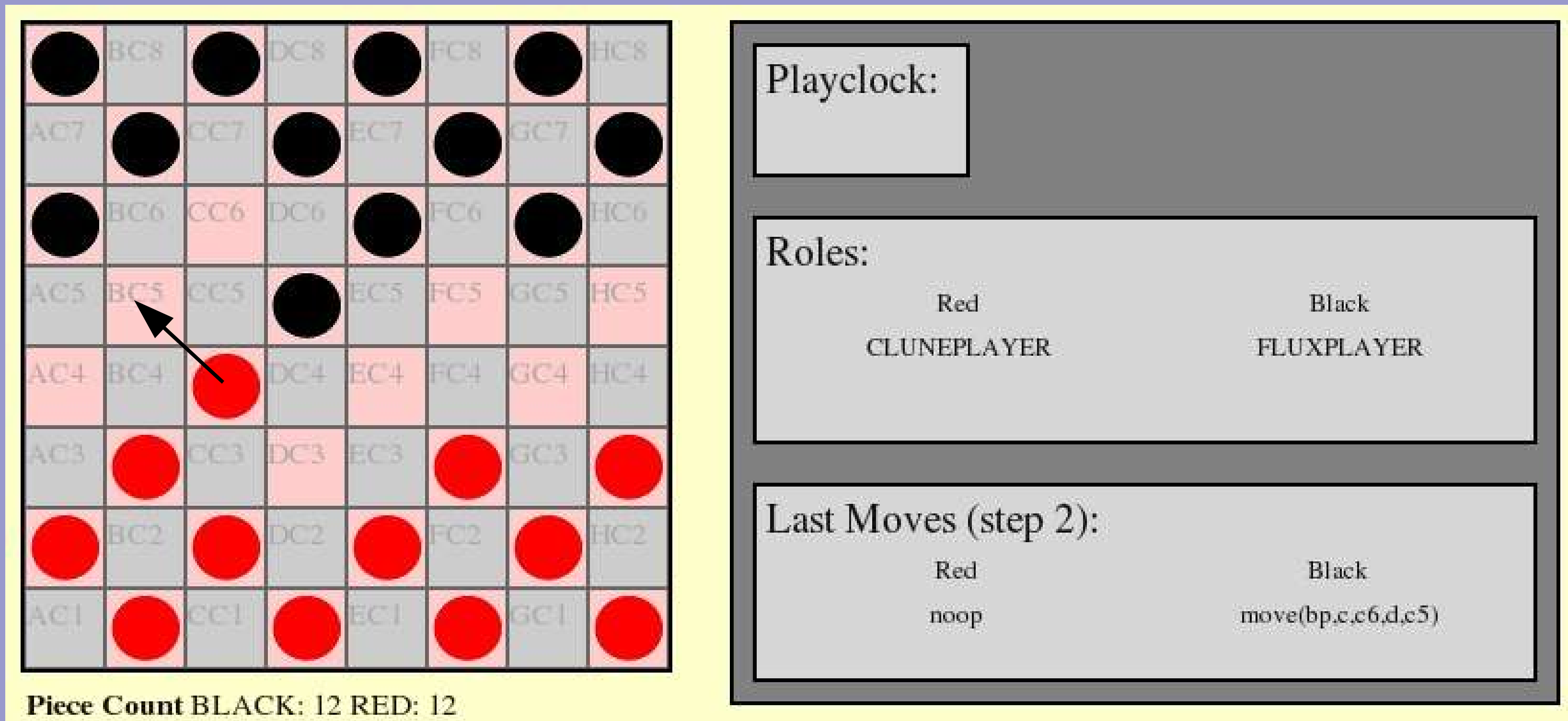
Existing approaches:

- Mobility and Novelty Heuristics
- Structure Detection
- Fuzzy Goal Evaluation
- Monte-Carlo Tree Search

Mobility

- More moves means better state
- Advantage:
  In many games, being cornered or forced into making a move is quite bad
  - In Chess, having fewer moves means having fewer pieces, pieces of lower value, or less control of the board
  - In Chess, when you are in check, you can do relatively few things compared to not being in check
  - In Othello, having few moves means you have little control of the board
- Disadvantage: Mobility is bad for some games

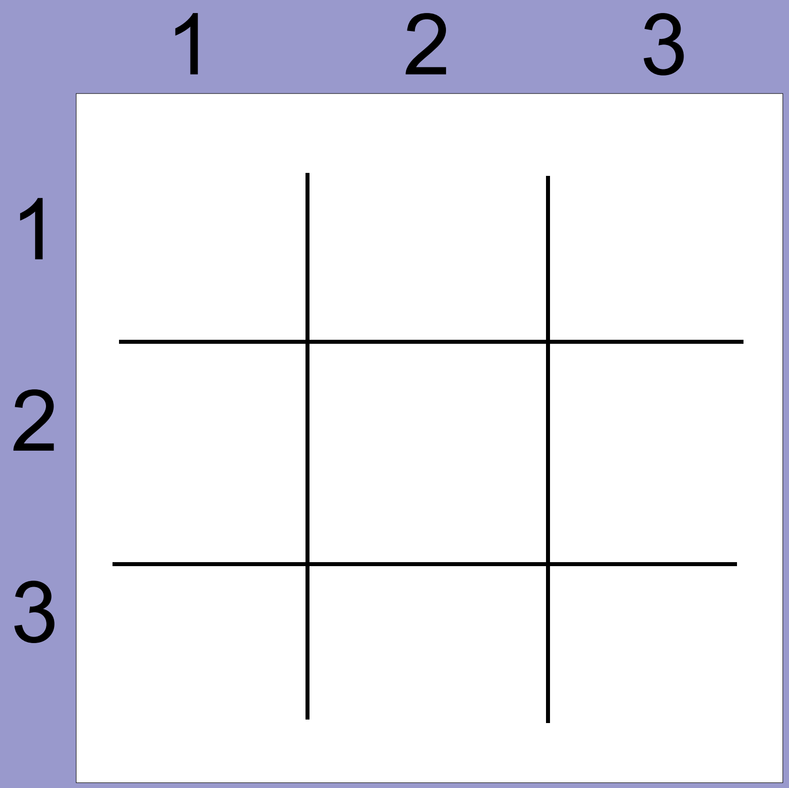
Worldcup 2006: Cluneplayer vs. Fluxplayer



Designing Evaluation Functions

- Typically designed by programmers/humans
- A great deal of thought and empirical testing goes into choosing one or more good functions
- E.g.
  - piece count, piece values in chess
  - holding corners in Othello
- But this requires knowledge of the game's structure, semantics, play order, etc.

Fuzzy Goal Evaluation: Example




```
goal(xplayer,100) <= line(x)
line(P) <= row(P)
             ∨ col(P)
             ∨ diag(P)
```

Value of intermediate state = Degree to which it satisfies the goal

Full Goal Specification

```
goal(xplayer,100) <= line(x)

line(P)     <= row(P) ∨ col(P) ∨ diag(P)

row(P)      <= true(cell(1,Y,P)) ∧ true(cell(2,Y,P)) ∧
               true(cell(3,Y,P))
col(P)      <= true(cell(X,1,P)) ∧ true(cell(X,2,P)) ∧
               true(cell(X,3,P))
diag(P)     <= true(cell(1,1,P)) ∧ true(cell(2,2,P)) ∧
               true(cell(3,3,P))
diag(P)     <= true(cell(3,1,P)) ∧ true(cell(2,2,P)) ∧
               true(cell(1,3,P))
```

After Unfolding

```
goal(x,100) <= true(cell(1,Y,x)) ∧ true(cell(2,Y,x)) ∧
               true(cell(3,Y,x))
               ∨
               true(cell(X,1,x)) ∧ true(cell(X,2,x)) ∧
               true(cell(X,3,x))
               ∨
               true(cell(1,1,x)) ∧ true(cell(2,2,x)) ∧
               true(cell(3,3,x))
               ∨
               true(cell(3,1,x)) ∧ true(cell(2,2,x)) ∧
               true(cell(1,3,x))
```

3 literals are true after **`does(x,mark(1,1))`**

2 literals are true after **`does(x,mark(1,2))`**

4 literals are true after **`does(x,mark(2,2))`**

Evaluating Goal Formula (Cont'd)

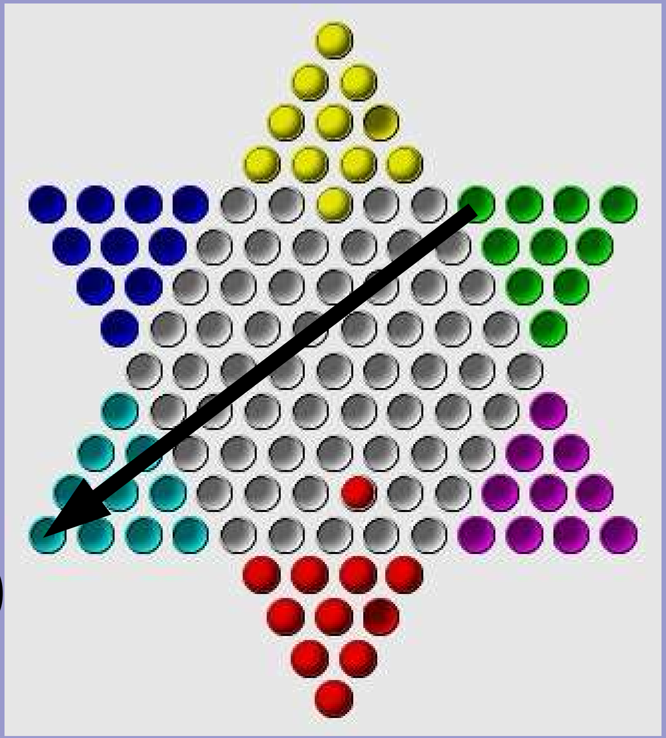
- Our t-norms: Instances of the Yager family (with parameter *q*)

$$T(a,b) = 1 - S(1-a,1-b)$$
$$S(a,b) = (a\hat{}q + b\hat{}q)\ \hat{}\ (1/q)$$

- Evaluation function for formulas

$$eval(f \wedge g) = T'(eval(f),eval(g))$$
$$eval(f \vee g) = S'(eval(f),eval(g))$$
$$eval(\neg f) = 1 - eval(f)$$

Advanced Fuzzy Goal Evaluation: Example



(j,13)

(e,5)

```
init(cell(green,j,13))∧ ...

goal(green,100)
    <= true(cell(green,e,5)
       ∧ ...
```

Truth degree of goal literal = (Distance to current value)$^{-1}$

Identifying Metrics

- Order relations Binary, antisymmetric, functional, injective

```
succ(1,2).  succ(2,3).  succ(3,4).
file(a,b).  file(b,c).  file(c,d).
```

- Order relations define a metric on functional features

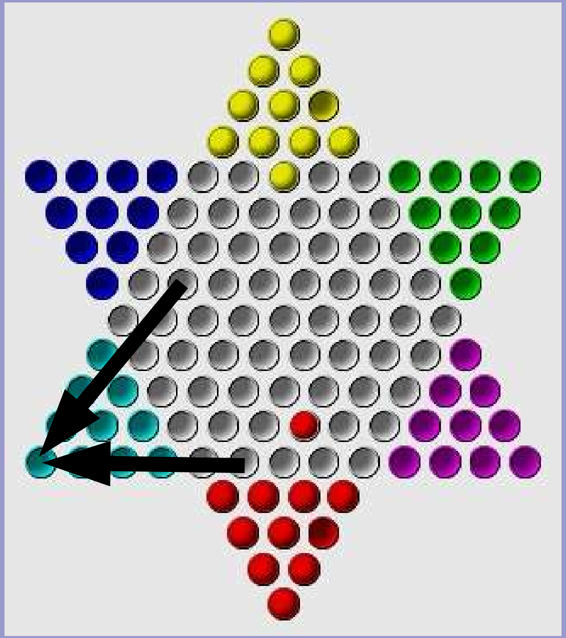
Δ(cell(green,j,13),cell(green,e,5)) = 13

Degree to which *f(x,a)* is true given that *f(x,b)*:

$$(1-p) - (1-p) * \Delta(b,a) / |dom(f(x))|$$

(f,10)

(e,5)



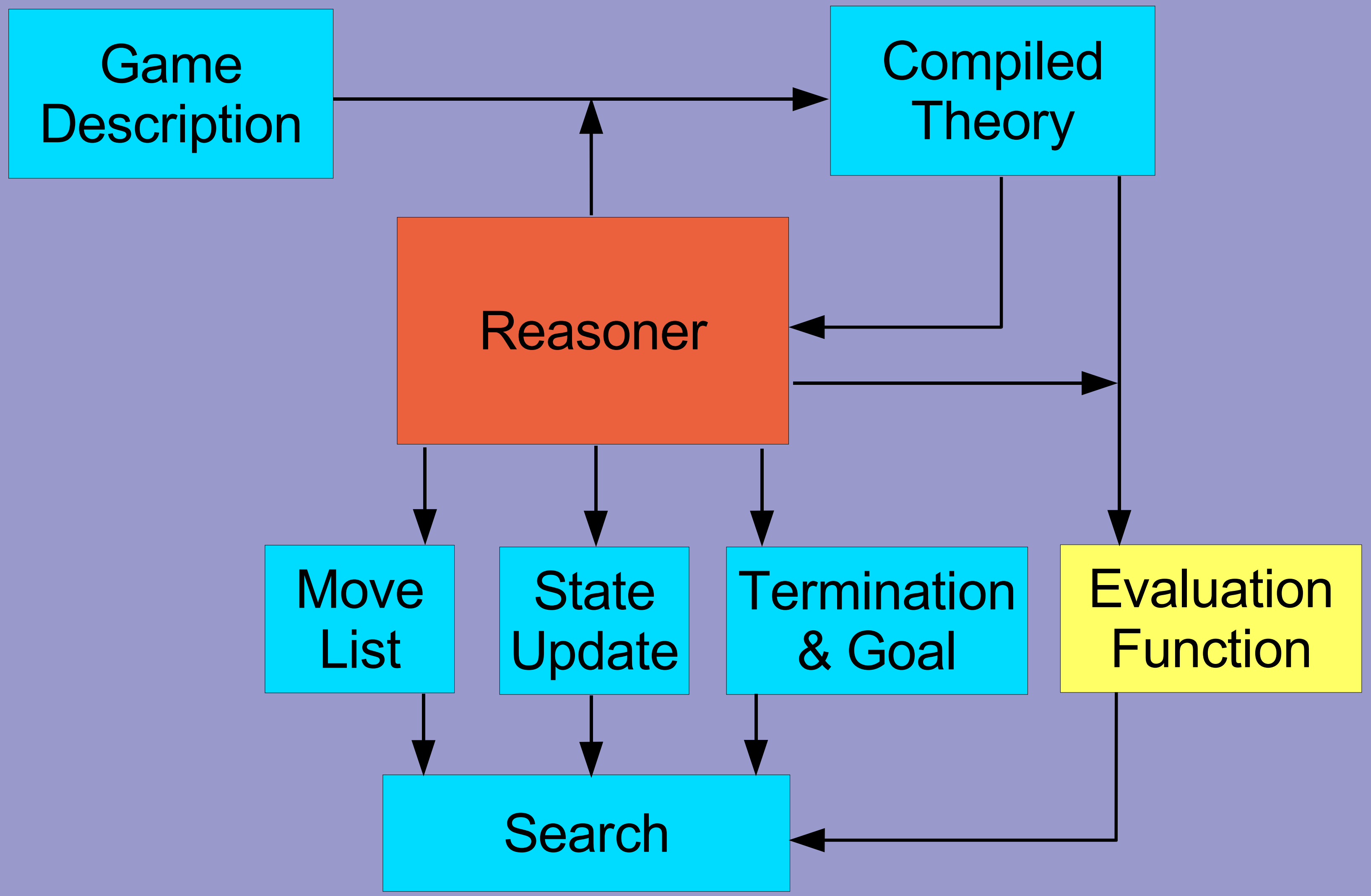
(j,5)

With $p = 0.9$, eval(`cell(green,e,5)`) is

0.082 if **`true(cell(green,f,10))`**

0.085 if **`true(cell(green,j,5))`**

A General Architecture



Assessment

Fuzzy goal evaluation works particularly well for games with

- independent sub-goals
  15-Puzzle
- converge to the goal
  Chinese Checkers
- quantitative goal
  Othello
- partial goals
  Peg Jumping, Chinese Checkers with >2 players

Summary

The GGP Challenge

Much like RoboCup, General Game Playing

- combines a variety of AI areas
- fosters developmental research
- has great public appeal
- has the potential to significantly advance AI

In contrast to RoboCup, GGP has the advantage to

- focus on the high-level knowledge aspect of intelligence
- poses a number of interesting challenges for KRR
- make a great hands-on course for AI+KR students

A Vision for GGP

Uncertainty

- Nondeterministic games with incomplete information

Natural Language Understanding

- Rules of a game given in natural language

Computer Vision

- Vision system sees board, pieces, cards, rule book, ...

Robotics

- Robot playing the actual, physical game

Resources

- Stanford GGP initiative `games.stanford.edu`
  - GDL specification
  - Basic player
- GGP in Germany `general-game-playing.de`
  - Game master
- Palamedes `palamedes-ide.sourceforge.net`
  - GGP/GDL development tool

Recommended Papers

- J. Clune
  Heuristic evaluation functions for general game playing, AAAI 2007
- H. Finnsson, Y. Björnsson
  Simulation-based approach to general game playing, AAAI 2008
- M. Genesereth, N. Love, B. Pell
  General game playing, AI magazine 26(2), 2006
- W. v. d. Hoek, J. Ruan, M. Wooldridge
  Verification of games in the game description language, 2008 (submitted)
- S. Schiffel, M. Thielscher
  Fluxplayer: a successful general game player, AAAI 2007
- S. Schiffel, M. Thielscher
  Specifying multiagent environments in the Game Description Language, 2008 (submitted)