

The Art and Science of Action Programming Languages

Michael Thielscher

Chapter 1

Introduction

Introduction

This tutorial is concerned with knowledge representation and reasoning techniques for systems that **act autonomously** in a complex environment.

Introduction

The aim is to endow these systems with **knowledge** of how their world functions, in particular knowledge of their own abilities to act.

Reasoning about this knowledge allows to

- make autonomous decisions
- exhibit goal-oriented behavior

Introduction

Foundations

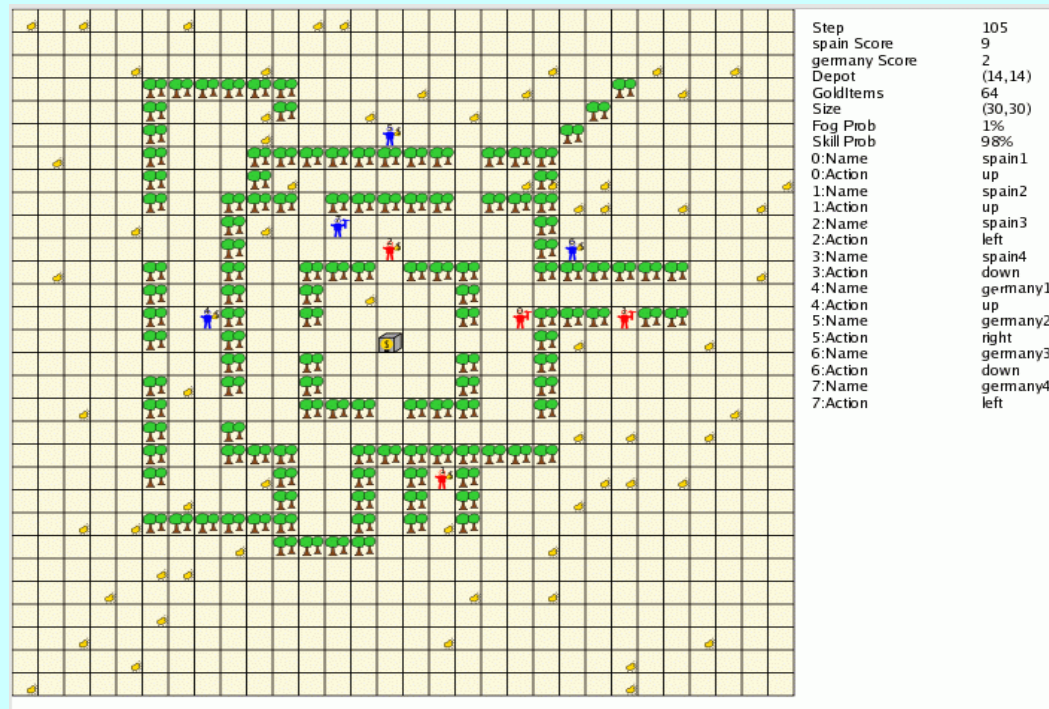
- Symbolic representation
- Logical reasoning

Advantages

- High degree of abstraction
- High-level action programming languages
- Large amount of diverse knowledge
- Uncertainty via disjunction / existential quantification

Application I: Multiagent Systems

Multiagent systems consist of autonomous, intelligent agents



CLIMA-06 Contest: Two competing teams of agents

Reasoning Tasks

- Verifying applicability of actions
Can I go forward now?
- Prediction
Where will the gold be after moving forward?
- Planning / Goal-oriented behavior
Which part of the environment shall I explore?

Application II: Cognitive Robots

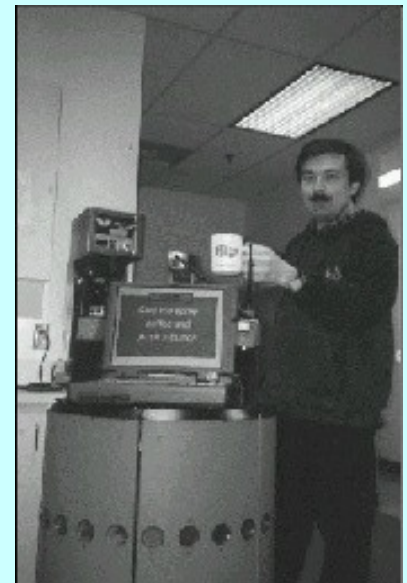
Autonomous robots making high-level decisions



Museum Guide RHINO



Coffee Delivery Robot



Reasoning Tasks

- Verifying applicability of actions
Can I deliver coffee now?
- Prediction
Where will the coffee be after picking it up?
- Planning / Goal-oriented behavior
In which order shall I satisfy the requests?
- Explanation
I can't deliver coffee now, what went wrong?

Application III: General Game Playing

A General Game Player is a system that

- understands formal descriptions of arbitrary games
Solitaire, 4-dimensional Chess,
n-player Monopoly, Texas Hold'Em, ...
- plays these games without human intervention

Reasoning Tasks

- Verifying applicability of actions (aka moves)
Can I move my king now?
- Prediction
Will I still be able to castle afterwards?
- Planning / Goal-oriented behavior
How can I win the game?

The Game Playing Metaphor

Agent in static world

⇒ Single-Player Game

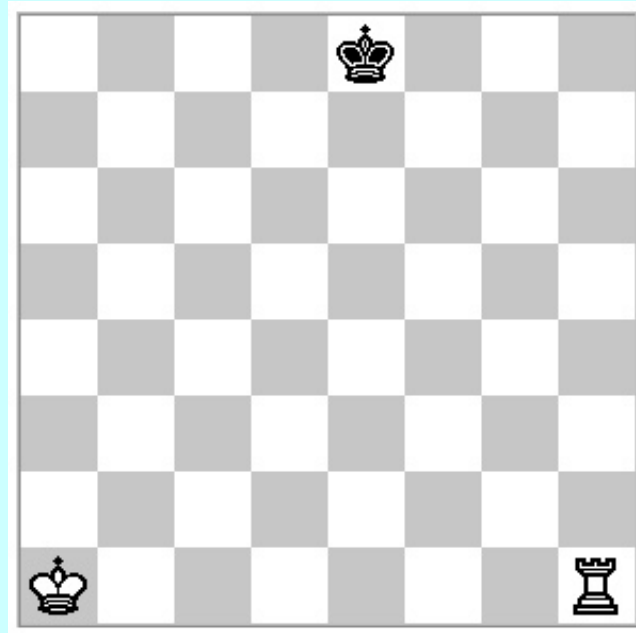
Agent/Robot in dynamic world

⇒ Two-Player Game (world as opponent)

System with multiple agents

⇒ n -Player Game

Example



Three autonomous agents.

Goal of **White-King** and **White-Rook**: checkmate **Black-King**

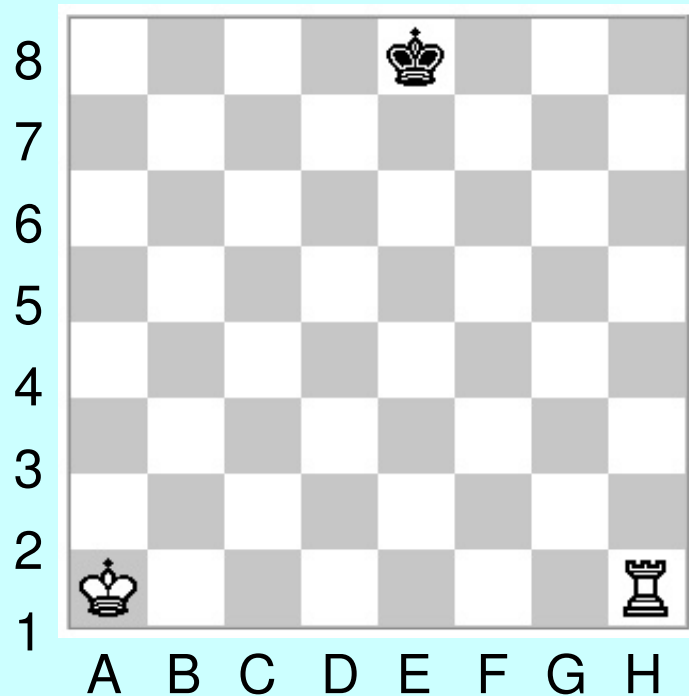
Historical Development

- 1963 Situation Calculus – the oldest KR formalism
- 1969 Frame Problem
- 1972f STRIPS and other planning languages
- 1991f Solving the frame problem:
Situation-, Event-, Fluent Calculus
- 1997f GOLOG, FLUX – action programming languages

Chapter 2

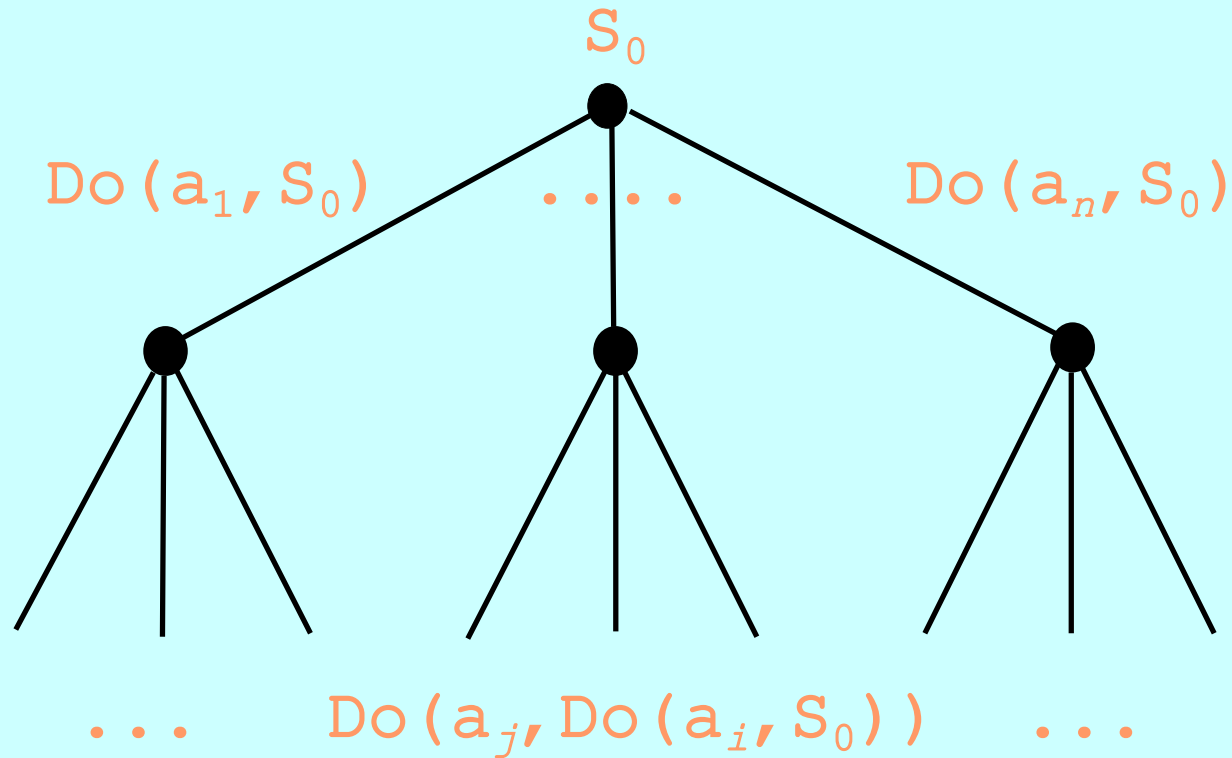
Situation Calculus and the Frame Problem

Representation



- Fluents
`Cell(agent, x, y)`
- Actions
`Move(agent, u, v, x, y)`

Time Structure: Situation Tree



Knowledge I: Abilities

- State Knowledge

$\text{Holds}(\text{Cell}(\text{WhiteKing}, A, 1), S_0) \wedge$

$\text{Holds}(\text{Cell}(\text{WhiteRook}, H, 1), S_0)$

- Precondition Axioms

$\text{Poss}(\text{Move}(\text{WhiteKing}, u, v, x, y), s) \leftrightarrow$

$\text{Holds}(\text{Cell}(\text{WhiteKing}, u, v), s) \wedge$

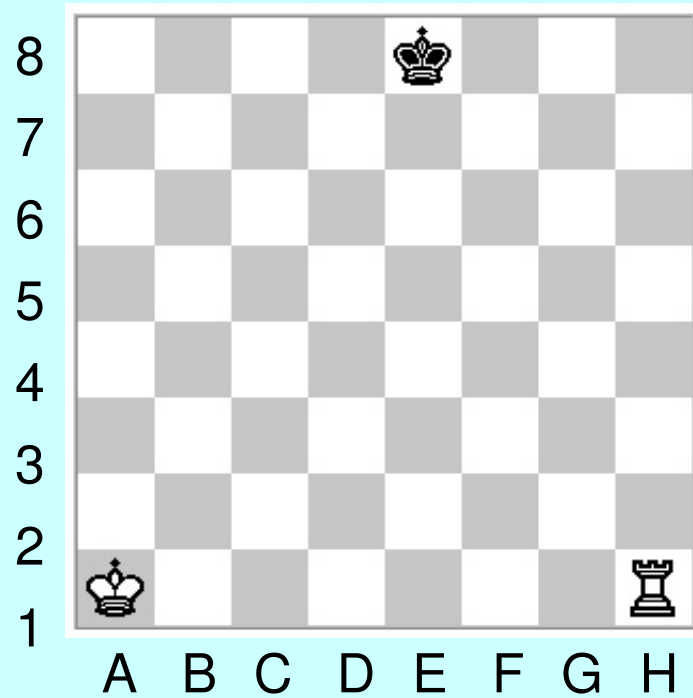
$\text{Legal-King-Move}(u, v, x, y, s)$

$\text{Poss}(\text{Move}(\text{WhiteRook}, u, v, x, y), s) \leftrightarrow$

$\text{Holds}(\text{Cell}(\text{WhiteRook}, u, v), s) \wedge$

$\text{Legal-Rook-Move}(u, v, x, y, s)$

Example



Poss (Move (WhiteKing, A, 1, B, 2), S_0)

Knowledge II: Effects

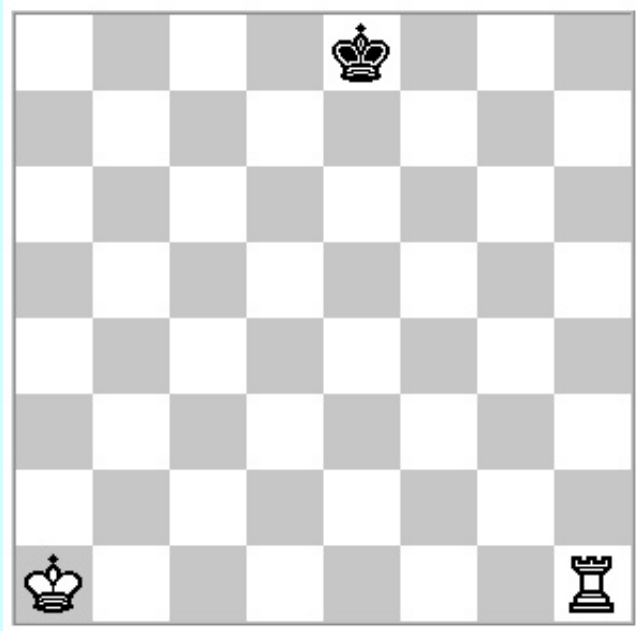
- Effect Axioms

$\text{Holds}(\text{Cell}(\text{agent}, x, y),$
 $\text{Do}(\text{Move}(\text{agent}, u, v, x, y), s))$

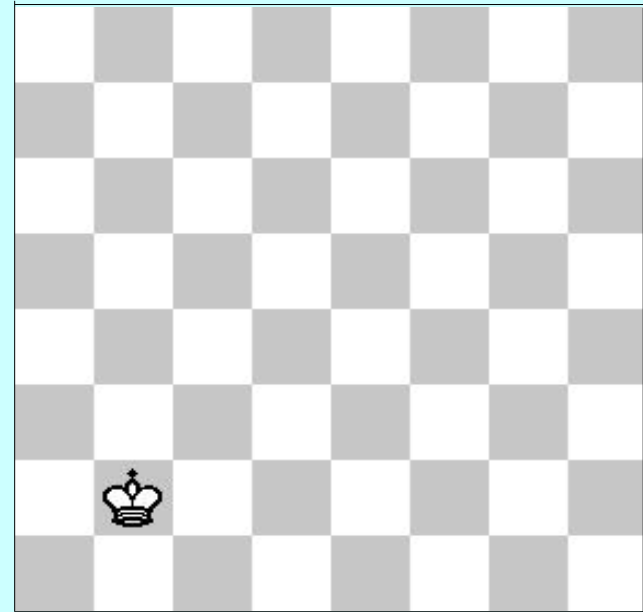
$\neg\text{Holds}(\text{Cell}(\text{agent}, u, v),$
 $\text{Do}(\text{Move}(\text{agent}, u, v, x, y), s))$

$\text{Holds}(\text{Cell}(c, x, y), s) \rightarrow$
 $\neg\text{Holds}(\text{Cell}(c, x, y),$
 $\text{Do}(\text{Move}(\text{agent}, u, v, x, y), s))$

Example



S_0



Do (Move (WhiteKing,
A, 1, B, 2), S_0)

The Frame Problem (1969)

- The effect axioms do **not** allow to conclude

$\text{Holds}(\text{Cell}(\text{WhiteRook}, H, 1),$
 $\text{Do}(\text{Move}(\text{WhiteKing}, A, 1, B, 2), S_0))$

- Additional **frame axioms** are needed

$\text{Holds}(\text{Cell}(c, i, j), \text{Do}(\text{Move}(p, u, v, x, y), s))$
 $\leftarrow \text{Holds}(\text{Cell}(c, i, j), s) \wedge$
 $[i \neq u \vee j \neq v] \wedge [i \neq x \vee j \neq y]$

- Representation not succinct
- Reasoning inefficient

On the Frame Problem

Daniel Dennett:

A new, deep epistemological problem – accessible in principle but unnoticed by generations of philosophers – brought to light by the novel methods of AI and still far from being solved.

On the Frame Problem

Ray Reiter:

If AI can be said to have a classic problem, then the Frame Problem is it.

Like all good open problems it is subtle, challenging, and it has led to significant new technical and conceptual developments in the field.

STRIPS (1972)

```
Move (p, u, v, x, y)
```

```
ADD-LIST: {Cell (p, x, y) }
```

```
DEL-LIST: {Cell (p, u, v) }
```

```
{Cell (WhiteKing, A, 1),  
  Cell (WhiteRook, H, 1),  
  Cell (BlackKing, E, 8) }
```



```
{Cell (WhiteKing, B, 2),  
  Cell (WhiteRook, H, 1),  
  Cell (BlackKing, E, 8) }
```

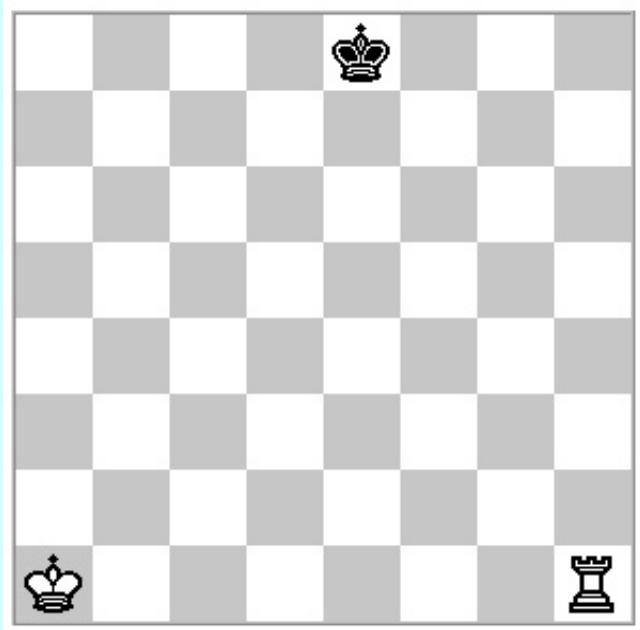
Other Planning Languages: ADL (1989), PDDL (1998)

- Conditional effects
- Fluents that are unknown
- Efficient planning techniques
 - Partial Order Planning
 - Graphplan
 - Planning as satisfiability
- Limited expressiveness:
No disjunctive state knowledge, quantification, ...
 $(\exists x) (\text{Holds}(\text{Dist}(x), S) \wedge 3.8 \leq x \leq 4.7)$

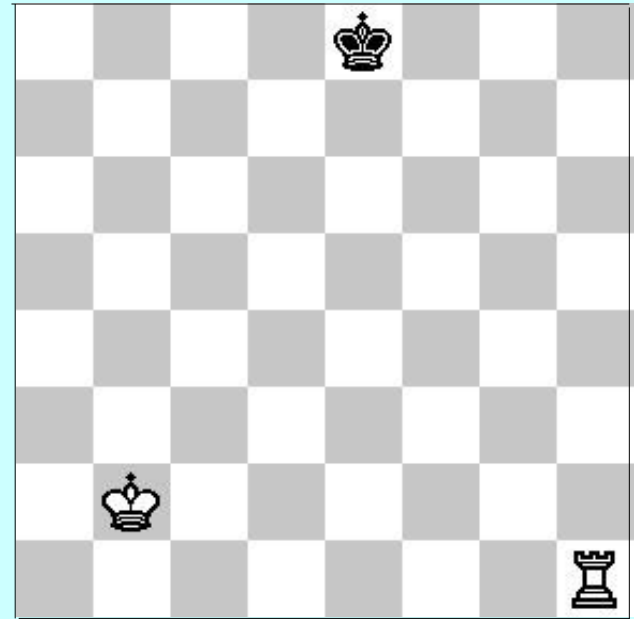
Successor State Axioms (1991)

$$\begin{aligned} \text{Holds}(\textit{Fluent}, \text{Do}(a, s)) &\leftrightarrow \\ &\text{effect}_+(a, s) \\ &\vee [\text{Holds}(\textit{Fluent}, s) \wedge \neg \text{effect}_-(a, s)] \end{aligned}$$
$$\begin{aligned} \text{Holds}(\text{Cell}(c, x, y), \text{Do}(a, s)) &\leftrightarrow \\ &a = \text{Move}(c, u, v, x, y) \\ &\vee [\text{Holds}(\text{Cell}(c, x, y), s) \wedge \\ &\quad (\forall u, v) a \neq \text{Move}(c, x, y, u, v) \wedge \\ &\quad (\forall p, u, v) a \neq \text{Move}(p, u, v, x, y)] \end{aligned}$$

Example



S_0



Do (Move (WhiteKing,
A, 1, B, 2), S_0)

Basic Action Theories

- Precondition axioms for all actions
- Successor state axioms for all fluents
- Initial state axiom

Extensions

- Knowledge and sensing via modality $K(s, s')$
- Nondeterministic actions
- Indirect effects of actions (Ramification Problem)
- Probabilities $P(s) = p$
- Time and continuous processes $Start(s) = t$

Chapter 3

Action Programming in GOLOG

Example Program

```
proc DeliverCoffee
```

```
  while  $(\exists p) \text{wantsCoffee}(p) \wedge \neg \text{hasCoffee}(p)$  do
```

```
     $\pi p$ . goto(coffeeMachine); pickUp(Coffee);  
    goto(p); giveCoffee(p)
```

```
  endWhile
```

```
endProc
```

```
proc goto(loc)
```

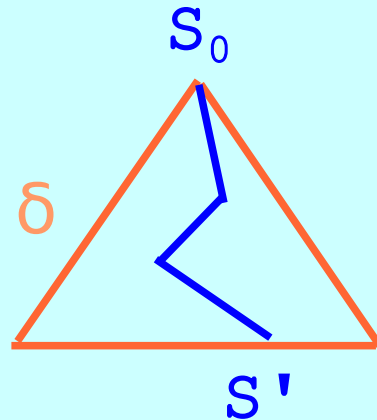
```
  if robotLocation(rloc) then drive(rloc, loc) endif
```

```
endProc
```


Programming Constructs

primitive actions	a
tests	$\phi?$
sequence	$\delta_1; \delta_2$
nondeterministic choice	$\pi \mathbf{x}. \delta(\mathbf{x})$
nondeterministic choice	$\delta_1 \mid \delta_2$
nondeterministic iteration	δ^*
if ϕ then δ_1 else δ_2 endif	$[\phi?; \delta_1] \mid [\neg\phi?; \delta_2]$
while ϕ do δ endwhile	$[\phi?; \delta]^* ; \neg\phi?$

Execution Modes



- **Offline** execution: Find terminating run, then execute
= Planning with search heuristics

```
proc main  
    while  $\neg$ goal do anyAction endWhile  
endProc
```

- Interleaved **Online-/Offline** execution

ConGOLOG

concurrent execution

$\delta_1 \parallel \delta_2$

concurrency w/ priorities

$\delta_1 \gg \delta_2$

concurrent iteration

δ'

interrupt

$\langle \phi \rightarrow \delta \rangle$

Knowledge-Based GOLOG

knowledge tests

$\text{Knows}(\phi)?$

$\text{Kwhether}(\phi)?$

sensing actions

www.cs.toronto.edu/cogrobo/



The Cognitive Robotics Group University of Toronto

[People](#) [Systems](#) [Publications](#)

[Courses](#) [Meetings](#) [Links](#)

[Slides](#) [Where?](#)

The Cognitive Robotics Group is concerned with endowing robotic or software agents with higher level cognitive functions that involve reasoning, for example, about goals, perception, actions, the mental states of other agents, collaborative task execution, etc. To do this, it is necessary to describe, in a language suitable for automated reasoning, enough of the properties of the robot, its abilities, and its environment, to permit it to make high-level decisions about how to act. The group has developed effective methods for representing and reasoning about the prerequisites and effects of actions, perception and other knowledge-producing actions, and natural events and actions by other agents. These methods have been incorporated into a logic programming language for agents called GOLOG (aGOI in LOGic). A prototype implementation of the language has been developed. Experiments have been conducted in using the language to build a high-level robot controller, some software agent applications (e.g. meeting scheduling), and more recently business process modeling tools.

People



Systems



Courses



Meetings



Publications



Links



Slides



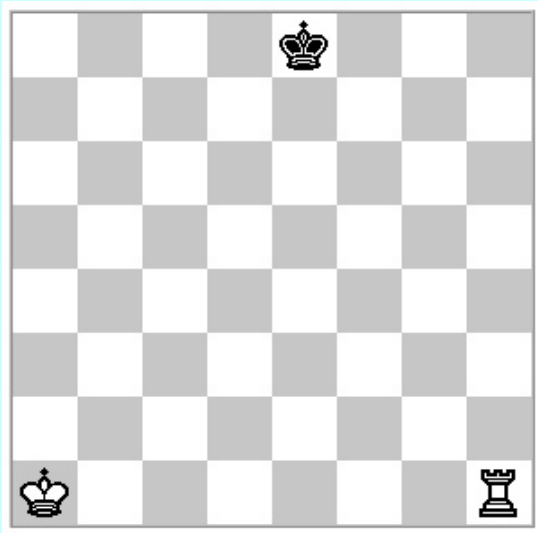
Where are we?



Chapter 4

Fluent Calculus and FLUX

The Frame Problem Revisited



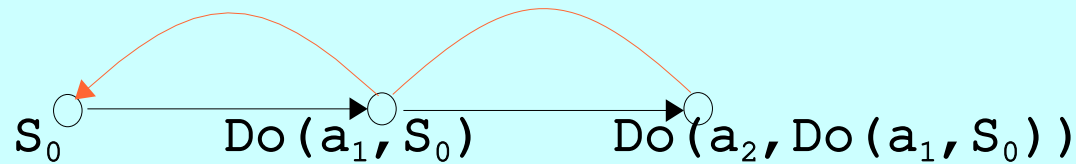
$\text{Holds}(\text{Cell}(c, x, y), \text{Do}(a, s)) \leftrightarrow$
 $a = \text{Move}(c, u, v, x, y)$
 $\vee [\text{Holds}(\text{Cell}(c, x, y), s) \wedge$
 $(\forall) a \neq \text{Move}(c, x, y, u, v) \wedge$
 $(\forall) a \neq \text{Move}(p, u, v, x, y)]$

- 64 instances needed to update the state
- Regression requires to roll back an entire history

Regression vs. Progression

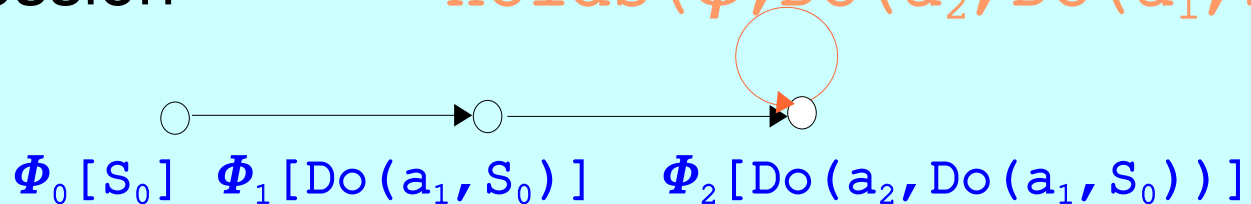
Regression

Holds ($\phi, \text{Do}(a_2, \text{Do}(a_1, S_0))$) ?



Progression

Holds ($\phi, \text{Do}(a_2, \text{Do}(a_1, S_0))$) ?



State Update Axioms (1999)

$$\Delta(s) \rightarrow \text{State}(\text{Do}(\textit{Action}, s)) = \\ \text{State}(s) - \text{effects}_- + \text{effects}_+$$

$$\text{Holds}(\text{Cell}(c, x, y), s) \rightarrow \\ \text{State}(\text{Do}(\text{Move}(p, u, v, x, y), s)) = \\ \text{State}(s) - \text{Cell}(c, x, y) \\ - \text{Cell}(p, u, v) + \text{Cell}(p, x, y)$$

- Axiomatic definition of + and -

Fluent Calculus: Basic Action Theories

- Precondition axioms for all actions
- State update axioms for all actions
- Initial state axiom

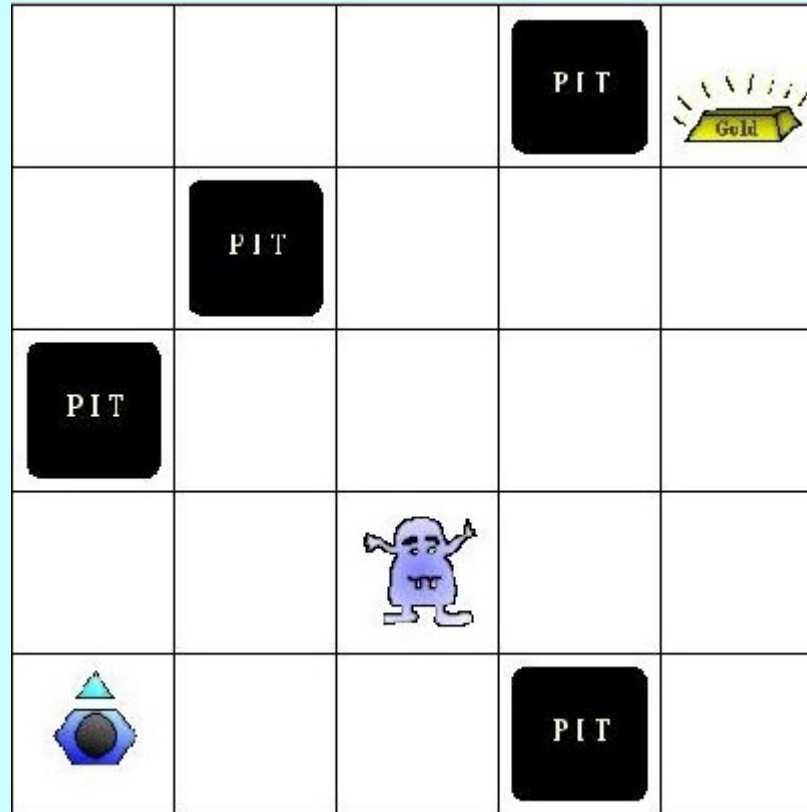
Extensions

- Knowledge and sensing via modality $KState(s, z)$
- Nondeterministic actions
- Indirect effects of actions (Ramification Problem)
- Unexpected action failure (Qualification Problem)
- Probabilities $P(z, s) = p$
- Time and continuous processes

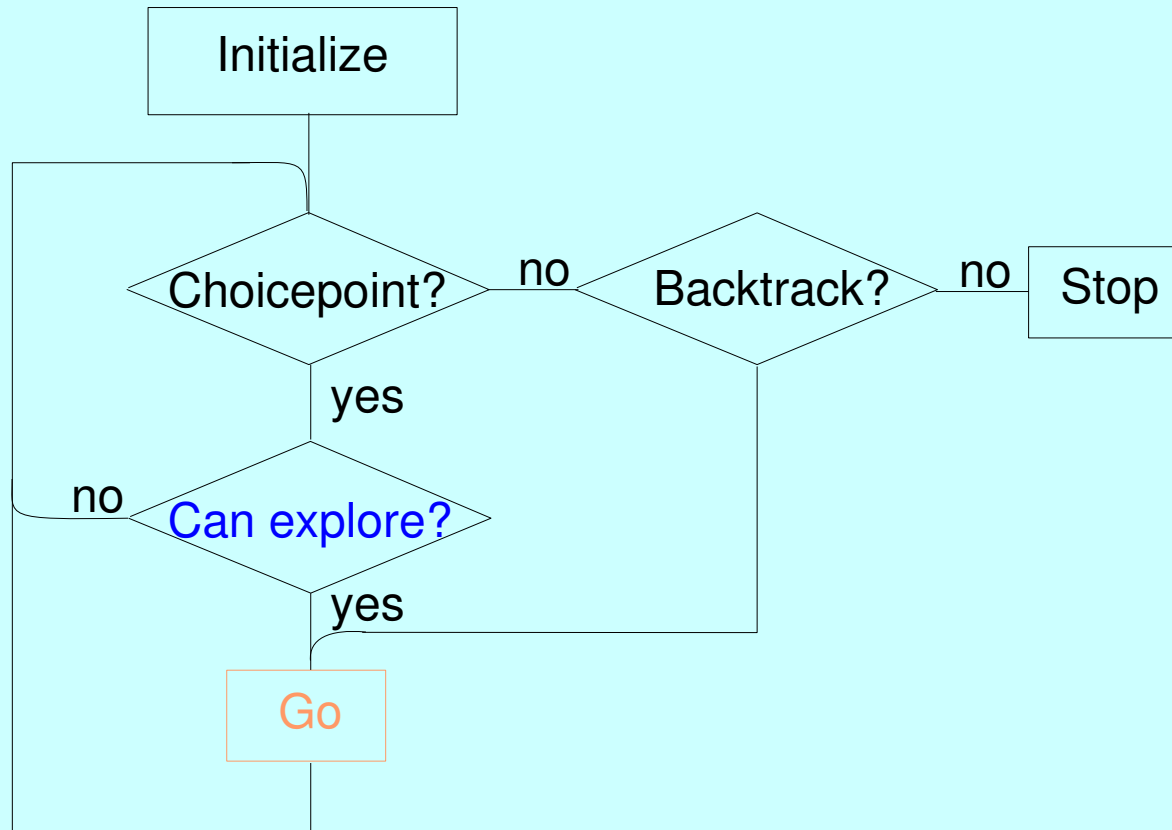
Action Programming with FLUX

- Constraint Logic Programming-based language
- State update as Constraint Rewriting
- Progression
- Interleaved Online-/Offline execution

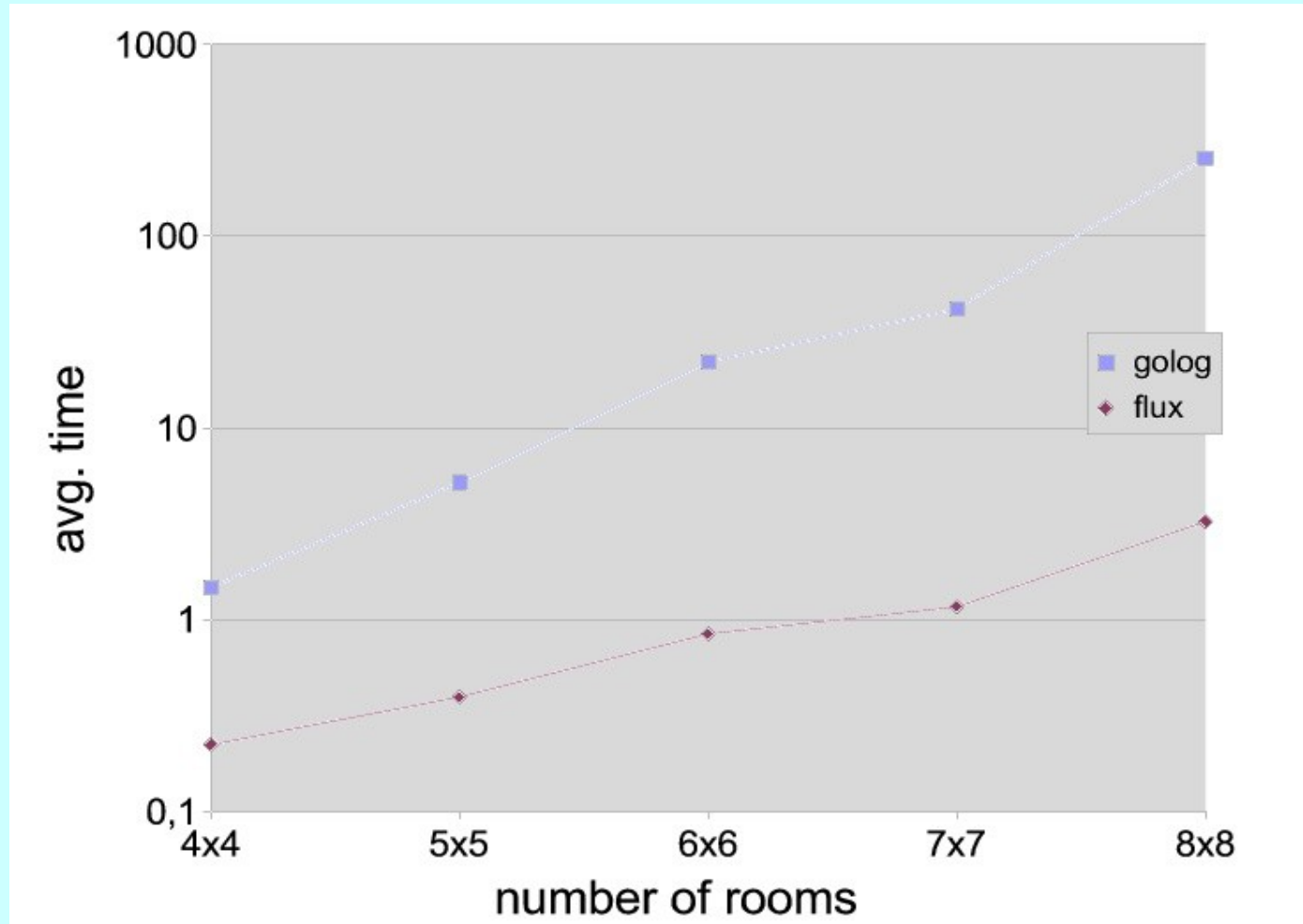
Example: Wumpus World



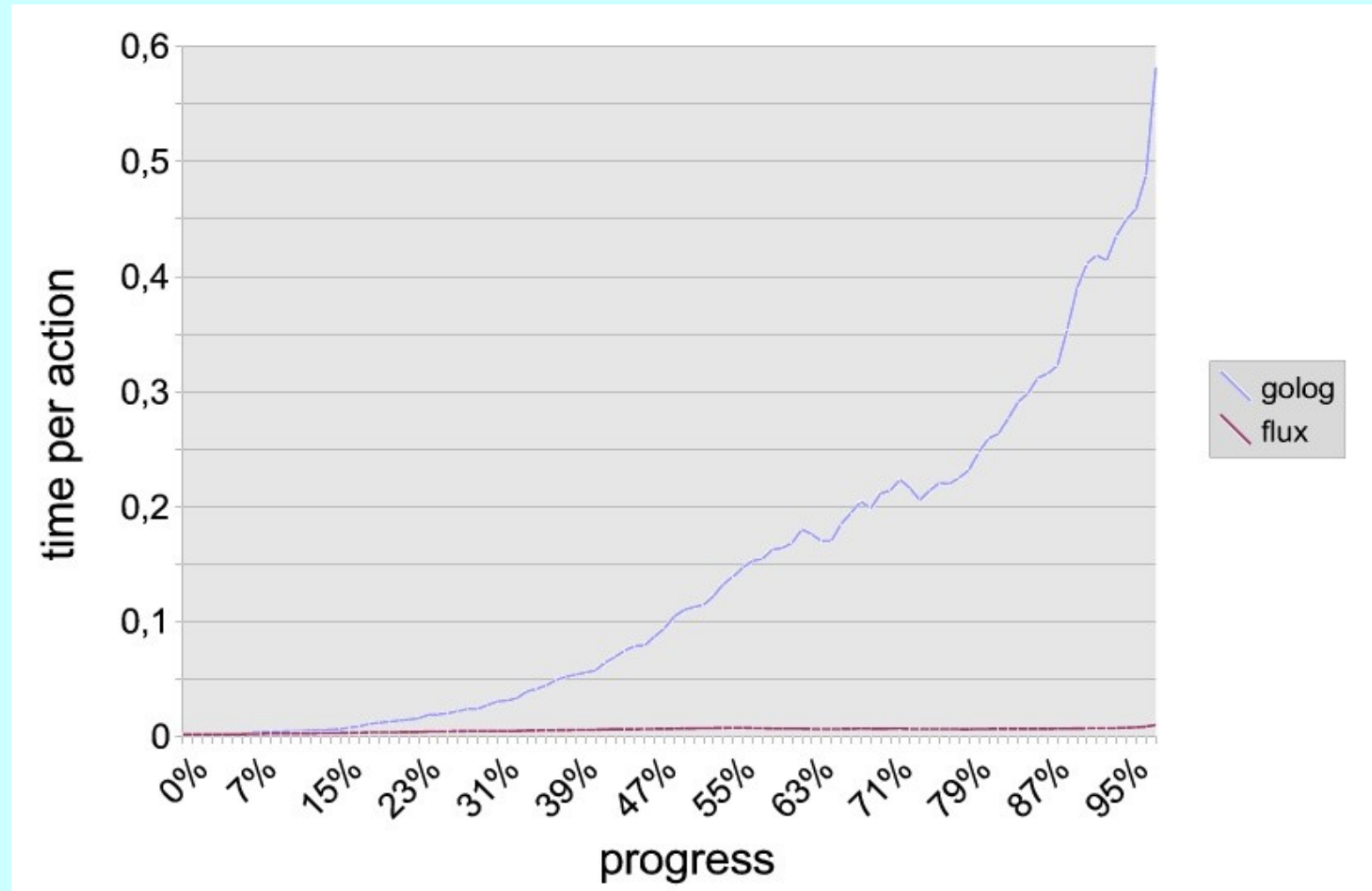
A Systematic Exploration Strategy



Runtime Comparison



Runtime Comparison



www.fluxagent.org



[Home](#)

[System](#)

[Books](#)

[Publications](#)

[Projects](#)

[People](#)

[Demos](#)

[Talks](#)

[FAQs](#)

[Disclaimer](#)

Last updated:

Jul 28 2005

[Web-Admin](#)



FLUX is a high-level programming system for cognitive agents of all kinds, including autonomous robots. Cognitive agents control themselves using an internal model of their environment. The FLUX kernel system endows agents with the general cognitive ability to reason about their actions and sensor data they acquire. FLUX agents are also able to plan ahead their actions in order to achieve specific goals. FLUX allows to implement complex strategies with concise and modular agent programs. An efficient constraint logic program, the FLUX system scales up well to domains which require large states and long action sequences.

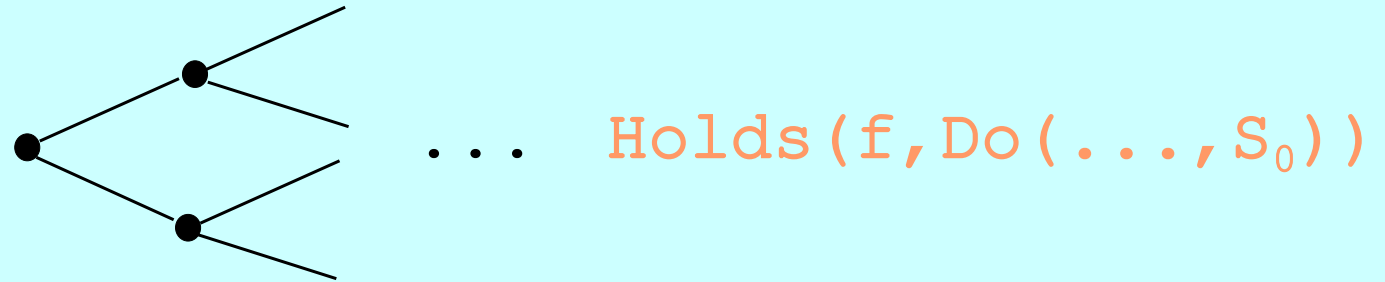
FLUX is an implementation of the Fluent Calculus. A versatile action representation formalism, this calculus provides a basic solution to the classical frame problem using the concept of state update axioms. The Fluent Calculus allows to address a variety of aspects in reasoning about actions, such as

- Ramifications (i.e., indirect effects of actions)
- Qualifications (i.e., unexpected action failure)
- Nondeterministic actions
- Concurrent actions
- Continuous change
- Sensors and effectors with noise

Chapter 5

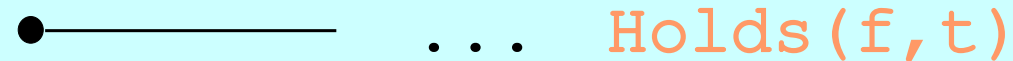
Event Calculus and Other Formalisms

Linear Time Structure



Situation Calculus

Fluent Calculus



Event Calculus

Event Calculus

Holds (f, t)

Happens (e, s, t)

f holds at time t

e happens between times s and t

Initiates (e, f, s, t)

Terminates (e, f, s, t)

e initiates f between s and t

e terminates f between s and t

Event Calculus: Basic Theories

- Narrative (using `Happens`)
- Observations (using `Holds`)
- Effect axioms (using `Initiates` / `Terminates`)
- Frame Problem solved by **Circumscription**

Extensions

- Concurrent events
- Nondeterministic events
- Planning by abduction
- Continuous processes

Other Agent Programming Systems

- A Behavior Language (ABL)
- Practical Reasoning System (PRS)
- SRI Procedural Agent Realization Kit (SPARK)

Systematic Assessment Methods

Meta Action Theories allow to systematically assess the **range of applicability** of specific calculi.

- Features-and-Fluents [Sandewall, 1994]
- Action Description Languages [Lifschitz et al, 1993f]

Chapter 6

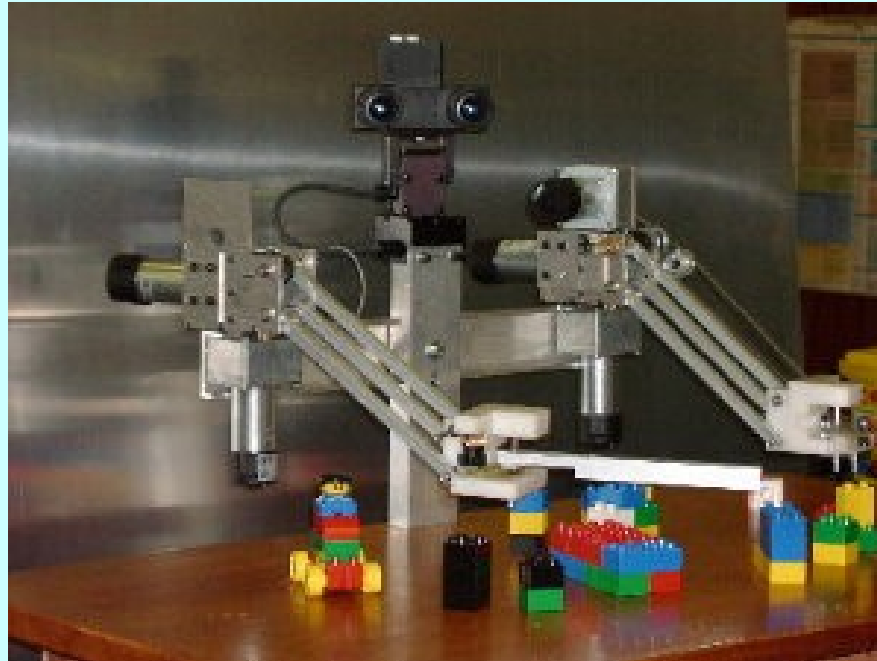
Applications and Future Challenges

Application: Museum Tour Guide Robot



- *Experiences with an interactive museum tour-guide robot* [Wolfram Burgard et al, 1999]
- *GOLEX: Bridging the gap between logic (GOLOG) and a real robot* [Rainer Hähnel et al, 1998]

Application: Upper-Torso Humanoid Robot



Event Calculus for perception and cognition

www.iis.ee.ic.ac.uk/~mpsha/ludwig/

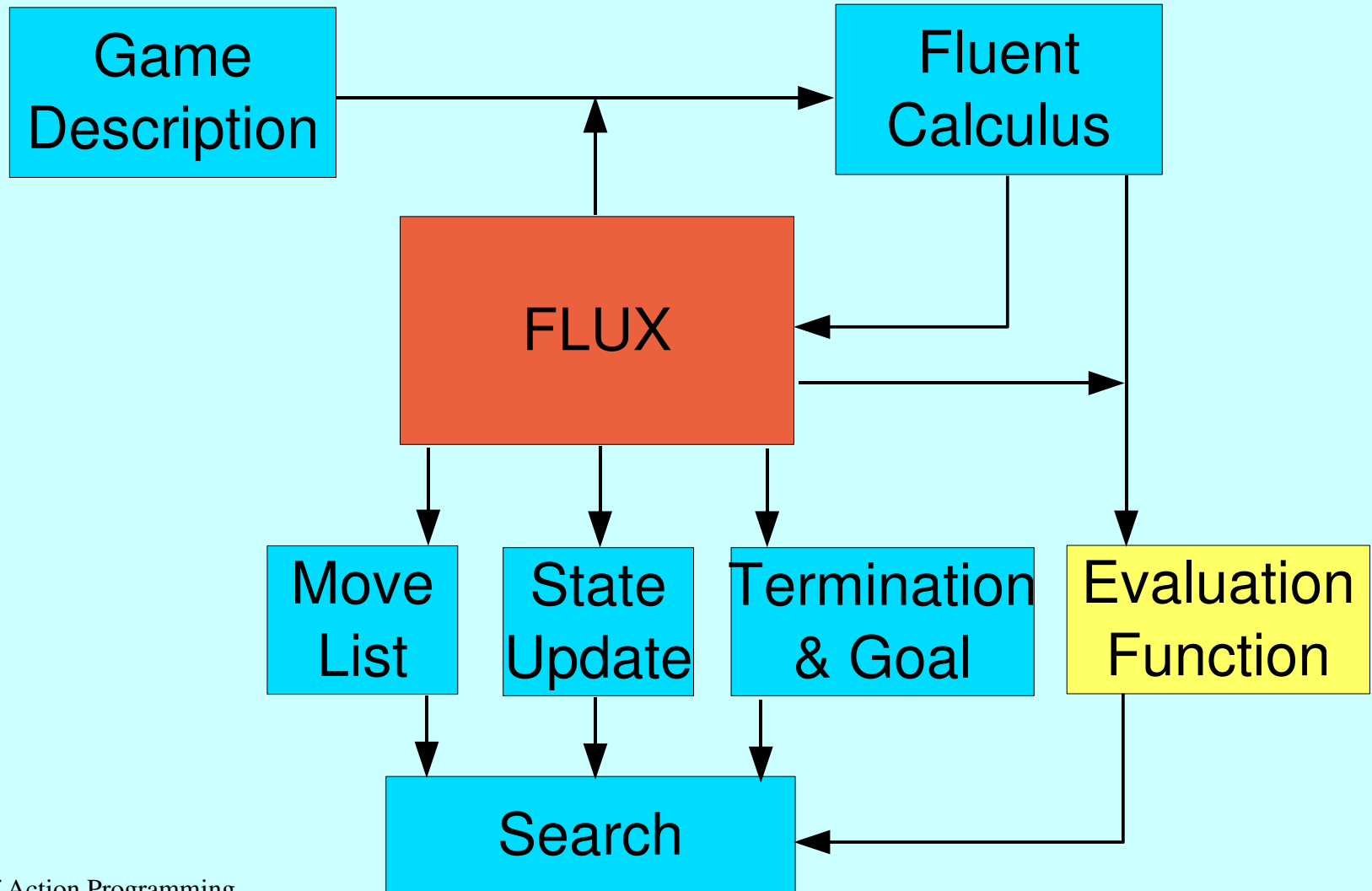
Application: UAVs



A UAV for traffic control monitors the ground and can autonomously track down and follow a car if necessary.

www.ida.liu.se/ext/witas/

Application: FLUXPLAYER



GGP World Championship 2006

AC8	BC8	CC8	DC8	EC8	FC8	GC8	HC8
AC7	●	CC7	DC7	EC7	●	GC7	HC7
AC6	BC6	CC6	DC6	EC6	FC6	GC6	HC6
AC5	BC5	CC5	●	EC5	FC5	GC5	HC5
AC4	BC4	CC4	DC4	EC4	FC4	GC4	HC4
AC3	BC3	CC3	●	EC3	FC3	GC3	HC3
●	BC2	●	DC2	EC2	FC2	●	HC2
AC1	BC1	CC1	DC1	EC1	FC1	GC1	HC1

Piece Count RED: 1 BLACK: 7

Playclock:

Roles:

Red
CLUNEPLAYER
0

Black
FLUXPLAYER
100

Last Moves (step 49):

Red
move(wp,f,c3,d,c5)

Black
noop

Cylindrical Checkers: The Championship Final @AAAI-06

www.fluxagent.org

Future Challenge 1: Uncertainty

- Logic
 - Controlled uncertainty in logic via incomplete state descriptions (disjunction, ...)
 - Symbolic reasoning can deal with large state spaces thanks to abstraction and local inference
 - Probability
 - Robot control in real-world environments requires probabilistic knowledge representation
 - $P(z, s) \triangleq$ probability of z to be actual state in s
- Challenge: A computational model for logic & probability!**

Future Challenge 2: Symbol Grounding

- Symbols (like "**Sandra's-coffee-mug**") need to be grounded in actual perceptions of the real world
- In today's systems, the grounding of symbols is pre-defined
- Cognitive robots should ultimately be able to ground symbols autonomously

Challenge: Solve the Symbol Grounding Problem!

Recommended Literature

- Situation Calculus and GOLOG

Raymond Reiter: *Logic in Action*. MIT Press 2001

- Fluent Calculus and FLUX

Michael Thielscher: *Reasoning Robots*. Springer 2005

- Event Calculus

Murray Shannahan: *Solving the Frame Problem*.

MIT Press 1996

Erik Mueller: *Commonsense Reasoning*.

Morgan Kaufmann 2006