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
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

\[ S_0 \]

\[ \text{Do}(a_1, S_0) \quad \text{...} \quad \text{Do}(a_n, S_0) \]

\[ \text{... Do}(a_1, S_0) \quad \text{... Do}(a_j, S_0) \quad \text{...} \]
Knowledge I: Abilities

- **State Knowledge**
  
  \[ \text{Holds(Cell(WhiteKing, A, 1),} S_0) \land \text{Holds(Cell(WhiteRook, H, 1),} S_0) \]

- **Precondition Axioms**

  \[ \text{Poss(Move(WhiteKing,} u,v,x,y),s) \leftrightarrow \text{Holds(Cell(WhiteKing,} u,v),s) \land \text{Legal-King-Move(} u,v,x,y,s) \]

  \[ \text{Poss(Move(WhiteRook,} u,v,x,y),s) \leftrightarrow \text{Holds(Cell(WhiteRook,} u,v),s) \land \text{Legal-Rook-Move(} u,v,x,y,s) \]
Example

\[ \text{Poss}(\text{Move}(\text{WhiteKing},A,1,B,2),S_0) \]
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

\[ S_0 \]

\[ \text{Do(Move(WhiteKing, A,1,B,2),S_0)} \]
The Frame Problem (1969)

• The effect axioms do not allow to conclude
  \[\text{Holds(\text{Cell(WhiteRook, H, 1), Do(Move(WhiteKing, A, 1, B, 2), S_0)})}\]

• Additional frame axioms are needed
  \[\text{Holds(\text{Cell(c, i, j), Do(Move(p, u, v, x, y), s))} \leftarrow \text{Holds(\text{Cell(c, i, j), s}) \land [i \neq u \lor j \neq v] \land [i \neq x \lor j \neq y]}\]

• 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\((\text{WhiteKing},A,1)\),
Cell\((\text{WhiteRook},H,1)\),
Cell\((\text{BlackKing},E,8)\)\}

\[\Rightarrow\]

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

- Conditional effects
- Fluents that are unknown
- Efficient planning techniques
  - Partial Order Planning
  - Graphplan
  - Planning as satisfiability
- Limited expressiveness:
  No disjunctive state knowledge, quantification,...

\[(\exists x) (\text{Holds}(\text{Dist}(x), S) \land 3.8 \leq x \leq 4.7)\]
Successor State Axioms (1991)

\[
\text{Holds}(\text{Fluent}, \text{Do}(a,s)) \leftrightarrow \\
\text{effect}_+(a,s) \\
\lor \left[ \text{Holds}(\text{Fluent}, s) \land \neg \text{effect}_-(a,s) \right]
\]

\[
\text{Holds}(\text{Cell}(c,x,y), \text{Do}(a,s)) \leftrightarrow \\
a = \text{Move}(c,u,v,x,y) \\
\lor \left[ \text{Holds}(\text{Cell}(c,x,y), s) \land \\
(\forall u,v) a \neq \text{Move}(c,x,y,u,v) \land \\
(\forall p,u,v) a \neq \text{Move}(p,u,v,x,y) \right]
\]
Example

$S_0$

$\text{Do(Move(WhiteKing, A,1,B,2),} S_0)$
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 $\text{Start}(s) = t$
Chapter 3

Action Programming in GOLOG
Example Program

proc DeliverCoffee
    while (∃p)wantsCoffee(p) ∧ ¬hasCoffee(p) do
        πp.goto(coffeeMachine); pickUp(Coffee);
        goto(p); giveCoffee(p)
    endwhile
endProc

proc goto(loc)
    if robotLocation(rloc) then drive(rloc,loc) endIf
endProc
# Programming Constructs

<table>
<thead>
<tr>
<th>Construct</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>primitive actions</td>
<td>$a$</td>
</tr>
<tr>
<td>tests</td>
<td>$\phi?$</td>
</tr>
<tr>
<td>sequence</td>
<td>$\delta_1; \delta_2$</td>
</tr>
<tr>
<td>nondeterministic choice</td>
<td>$\pi x. \delta(x)$</td>
</tr>
<tr>
<td>nondeterministic choice</td>
<td>$\delta_1</td>
</tr>
<tr>
<td>nondeterministic iteration</td>
<td>$\delta^*$</td>
</tr>
<tr>
<td>if $\phi$ then $\delta_1$ else $\delta_2$ endIf</td>
<td>$[\phi?; \delta_1]</td>
</tr>
<tr>
<td>while $\phi$ do $\delta$ endWhile</td>
<td>$[\phi?; \delta]^* ; \neg\phi?$</td>
</tr>
</tbody>
</table>
Execution Modes

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

  ```
  proc main
  
  while ¬goal do anyAction endWhile

  endProc
  ```

- **Interleaved Online-/Offline** execution
ConGOLOG

concurrent execution, \( \delta_1 \parallel \delta_2 \)
concurrency w/ priorities, \( \delta_1 >> \delta_2 \)
concurrent iteration, \( \delta' \)
interrupt, \( <\phi \rightarrow \delta> \)
Knowledge-Based GOLOG

knowledge tests

sensing actions

Knows(ϕ)?
Kwhether(ϕ)?
Chapter 4

Fluent Calculus and FLUX
The Frame Problem Revisited

\[ \text{Holds}(\text{Cell}(c,x,y), \text{Do}(a,s)) \leftrightarrow \]
\[ a = \text{Move}(c,u,v,x,y) \]
\[ \lor \left[ \text{Holds}(\text{Cell}(c,x,y),s) \land \right. \]
\[ \left. (\forall) a \neq \text{Move}(c,x,y,u,v) \land \right. \]
\[ (\forall) a \neq \text{Move}(p,u,v,x,y) \right] \]

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

$\Phi_0[S_0]$  $\Phi_1[\text{Do}(a_1, S_0)]$  $\Phi_2[\text{Do}(a_2, \text{Do}(a_1, S_0))]$
State Update Axioms (1999)

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

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

Initialize

Choicepoint? no

Backtrack? no

Stop

yes

Can explore? no

no

Yes

Go
Runtime Comparison

![Graph showing runtime comparison between different number of rooms and different algorithms. The graph plots average time against the number of rooms. Two algorithms are compared: golog and flux. The graph shows a linear increase in average time as the number of rooms increases.](image-url)
Runtime Comparison

![Graph showing time per action vs progress]

- golog
- flux
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

\[ \ldots \quad \text{Holds}(f, \text{Do}(\ldots, S_0)) \]

Situation Calculus

Fluent Calculus

Event Calculus

\[ \ldots \quad \text{Holds}(f, t) \]
# Event Calculus

<table>
<thead>
<tr>
<th>Event Calculus</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holds(f,t)</td>
<td>f holds at time t</td>
</tr>
<tr>
<td>Happens(e,s,t)</td>
<td>e happens between times s and t</td>
</tr>
<tr>
<td>Initiates(e,f,s,t)</td>
<td>e initiates f between s and t</td>
</tr>
<tr>
<td>Terminates(e,f,s,t)</td>
<td>e terminates f between s and t</td>
</tr>
</tbody>
</table>
Event Calculus: Basic Theories

- Narrative (using \texttt{Happens})
- Observations (using \texttt{Holds})
- Effect axioms (using \texttt{Initiates} / \texttt{Terminates})
- Frame Problem solved by \texttt{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

Game Description → Fluent Calculus

FLUX

Move List → State Update

Termination & Goal → Evaluation Function

Search
GGP World Championship 2006

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

• \( P(z, s) \) = 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

• 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