# Action Languages and Planning Techniques

## Golog and PDDL

Gabi Röger

Department of Computer Science
University of Freiburg

# Motivation

Dynamic Domains

Planning

Flexible representation

**Planning Techniques**
PDDL
Efficiency
low-level

**Action Languages**
Golog
Expressive Power
high-level

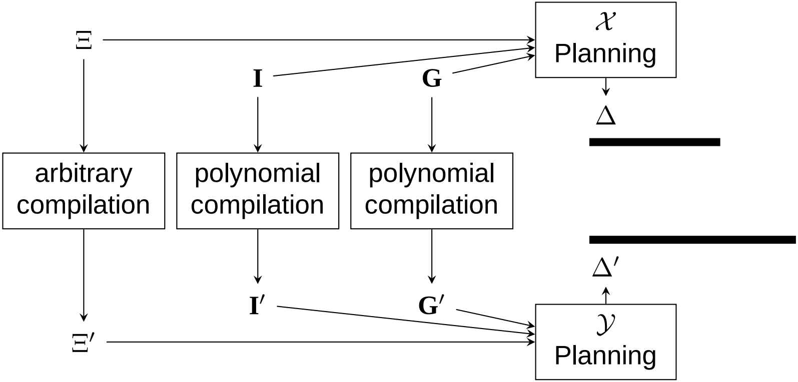
In recent years:
Convergence of Expressiveness

An integration of both fields would provide great advantages:

- Flexible description of a system’s behaviour
- Efficient planning of actual low-level actions

# Compilation Schemes

Framework to compare the expressive power of planning formalisms



# The Action Language Golog

- Logic programming language
- Used for dynamic worlds
- One can constrain a system's (e.g. a robot's) behaviour on a high level, e.g. with
  - Nondeterministic choice of actions
  - Nondeterministic choice of arguments
  - Nondeterministic iteration (execute a command zero or more times)
  - if and while statements
  - Procedures

Advantage: As Golog is based on the situation calculus (using macros), there is a formal theory.

# Situation Calculus

A way to represent dynamically changing worlds with logic

- Changes of the world are the result of actions and each action leads to a new situation.
  - initial situation $s_0$
  - function $do(a, s)$
- Predicate $Poss(a, s)$ states whether it is possible to perform action $a$ in situation $s$.
- Fluents are relations and functions whose values vary from one situation to the next
  - situation term as last argument
  - e.g. $switchedOn(lamp, s)$, $primeMinister(Italy, s)$
- Situation-independent predicates and fluents keep the same value in all situations, e.g. $mathematician(Gauss)$

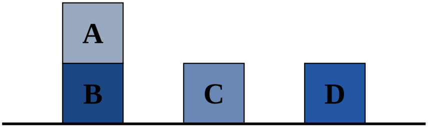
# Example: Blocksworld
## Domain Structure

$$Poss(move(b,f,t),s) \equiv on(b,f,s) \wedge clear(b,s) \wedge clear(t,s)$$
$$Poss(moveToTable(b,f),s) \equiv on(b,f,s) \wedge clear(b,s)$$

$$\begin{aligned} clear(b,do(a,s)) \equiv\ & \exists b',b''(a = move(b',b,b'')) \vee \\ & \exists b'(a = moveToTable(b',b)) \vee \\ & clear(b,s) \wedge \neg(\exists b',b''(a = move(b',b'',b))) \end{aligned}$$

$$\begin{aligned} on(b_1,b_2,do(a,s)) \equiv\ & \exists b(a = moveToTable(b_1,b) \wedge b_2 = \mathbf{table}) \vee \\ & \exists b(a = move(b_1,b,b_2)) \vee on(b_1,b_2,s) \wedge \\ & \neg(\exists b(a = move(b_1,b_2,b)) \vee \\ & a = moveToTable(b_1,b_2)) \end{aligned}$$

Initial Situation:




$$
\begin{aligned}
on(b_1, b_2, s_0) \equiv\ & (b_1 = \mathbf{A} \wedge b_2 = \mathbf{B}) \vee (b_1 = \mathbf{B} \wedge b_2 = \mathbf{table}) \vee \\
& (b_1 = \mathbf{C} \wedge b_2 = \mathbf{table}) \vee (b_1 = \mathbf{D} \wedge b_2 = \mathbf{table}) \\
clear(b, s_0) \equiv\ & b = \mathbf{A} \vee b = \mathbf{C} \vee b = \mathbf{D}
\end{aligned}
$$

Goal:

$$on(\mathbf{C}, \mathbf{B}, s) \wedge on(\mathbf{D}, \mathbf{A}, s)$$

# Integration of planning techniques and action languages

Task

- Create a common semantic basis in the situation calculus
- Analyze expressive power by means of compilation techniques
- Implement a system

Necessary skills for bachelor/master/diploma theses or student projects

- Courses:
  - Logic for computer scientists
  - Theoretical computer science (Informatik III)
- Programming skills
- Interest in complexity issues