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
Framework to compare the expressive power of planning formalisms

Compilation Schemes

- arbitrary compilation
- polynomial compilation
- polynomial compilation

Planning

- $\Xi$
- $I$
- $G$
- $\Delta$
- $\Xi'$
- $I'$
- $G'$
- $\Delta'$
- $\gamma$
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.
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. \( \text{switchedOn}(\text{lamp}, s) \), \( \text{primeMinister}(\text{Italy}, s) \)

- Situation-independent predicates and fluents keep the same value in all situations, e.g. \( \text{mathematician}(\text{Gauss}) \)
Example: Blocksworld

Domain Structure

\[ \text{Poss}(\text{move}(b,f,t),s) \equiv \text{on}(b,f,s) \land \text{clear}(b,s) \land \text{clear}(t,s) \]
\[ \text{Poss}(\text{moveToTable}(b,f),s) \equiv \text{on}(b,f,s) \land \text{clear}(b,s) \]

\[ \text{clear}(b, \text{do}(a,s)) \equiv \exists b', b'' (a = \text{move}(b', b, b'')) \lor \exists b' (a = \text{moveToTable}(b', b)) \lor \text{clear}(b,s) \land \neg (\exists b', b'' (a = \text{move}(b', b'', b))) \]

\[ \text{on}(b_1, b_2, \text{do}(a,s)) \equiv \exists b (a = \text{moveToTable}(b_1, b) \land b_2 = \text{table}) \lor \exists b (a = \text{move}(b_1, b, b_2)) \lor \text{on}(b_1, b_2, s) \land \neg (\exists b (a = \text{move}(b_1, b_2, b)) \lor a = \text{moveToTable}(b_1, b_2)) \]
Initial Situation:

\[ on(b_1, b_2, s_0) \equiv (b_1 = A \land b_2 = B) \lor (b_1 = B \land b_2 = \text{table}) \lor (b_1 = C \land b_2 = \text{table}) \lor (b_1 = D \land b_2 = \text{table}) \]

\[ clear(b, s_0) \equiv b = A \lor b = C \lor b = D \]

Goal:

\[ on(C, B, s) \land on(D, A, s) \]
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