BAAC: A Prolog System for Action Description and Agents Coordination

Agostino Dovier  Andrea Formisano  Enrico Pontelli

Univ. of Udine & Univ. of Perugia & New Mexico State Univ.

ICLP 2011 — Lexington - July 8, 2011
Action Description Languages

Provide a declarative framework for knowledge representation and reasoning on actions and change

- Many proposals, several languages:
  \[ A, B, C+, K, CARD, AL, ALAN, \ldots \]
Action description languages can be encoded using CLP(FD) implemented in Prolog [ICLP07,MG65].

Fluent values are not forced to be Boolean.

We presented the language $\mathcal{B}^{MV}$ and its encoding [ICLP07,TPLP10].

An extension for multiagent planning $\mathcal{B}^{MAP}$ (with centralized reasoner) was also developed [LPNMR09,FuIn2010].
Concurrent autonomous agents

The basic idea:

- Agents “live” in a common world
- Each agent has a (partial) view of the world and its own goals
- Each agent autonomously develops a plan.
Concurrent autonomous agents

The basic idea:
- Agents “live” in a common world
- Each agent has a (partial) view of the world and its own goals
- Each agent autonomously develops a plan.

Properties of the world (fluents) may be shared by different agents, but
- Agents might not be aware of this, and
- The “local” view of an agent might be affected by other agents’ actions
- The effects of actions of different agents may interfere
- The concurrent execution of agents’ plans might lead to inconsistencies and conflicts among actions’ effects.
Domain specification and plan execution

Two main aims:

- Design an Action Description Language for autonomous agents coordination, to support the specification of strategies and policies for conflict resolution, communication, ...

- Develop a prototype to execute Action Description Language specifications and enable planning, concurrent plan-execution, and plan revision.

...ensuring extensibility of the Action Description Language and modularity of the prototype!
The language \( \mathcal{B}^{\text{AAC}} \)

**Action declaration**

\[
\text{action } \text{Act}
\]

**Fluents...**

\[
\text{fluent } f_1, \ldots, f_h \text{ valued } \text{dom}
\]

**expressions...**

\[
\text{FE} \quad ::= \quad n \mid f^t \mid f@r \mid \text{FE}_1 \oplus \text{FE}_2 \mid \text{rei}(C) \mid \ldots
\]

**...and constraints**

A constraint \( C \) is a propositional combination of *primitive constraints* of the form \( \text{FE}_1 \text{ relop } \text{FE}_2 \).
The language $\mathcal{B}^{\text{AAC}}$

Dynamic causal laws

$$\text{Act} \text{ causes } C_{\text{Eff}} \text{ if } C_{\text{Prec}}$$

Executability laws

executable $\text{Act}$ if $C$

Specification of initial...

initially $C$

...and final states

goal $C$
The language $\mathcal{B}^{AAC}$

Each agent $Ag$ is specified by a different action theory

**Agent identification**

agent $Ag$ [priority $Val$].

**Knowledge about other agents**

known_agents $A_1, A_2, \ldots, A_k$
The language $\mathcal{B}^{\text{AAC}}$

To specify simple reactions to conflicts and failures in plan-execution, we refine the action declarations:

\[
\text{action } \text{Act } \text{OPT}
\]

\[
\text{OPT} ::= \text{on\_conflict } \text{OC } \text{OPT} \\
| \text{on\_failure } \text{OF } \text{OPT}
\]

\[
\text{OC} ::= \text{retry\_after } T \quad [\text{provided } C] \\
| \text{forego} \quad [\text{provided } C]
\]

\[
\text{OF} ::= \text{retry\_after } T \quad [\text{if } C] \\
| \text{replan} \quad [\text{if } C] \quad [\text{add\_goal } C] \\
| \text{fail} \quad [\text{if } C]
\]
Supervisor

- A **supervisor** controls the execution of agents’ plans
  - Ensure consistency of the state of the world
- Each agent sends a message to the supervisor declaring the intention to execute an action
Supervisor

- A **supervisor** controls the execution of agents’ plans
  - Ensure consistency of the state of the world
- Each agent sends a message to the supervisor declaring the intention to execute an action
- The supervisor verifies the consistency of the consequences of all requested actions
- Supervisor determines subsets of conflicting actions
Conflict Resolution

Conflicts can be resolved by executing various protocols

1. Supervisor arbitration (e.g., using priorities, round-robin, etc.)
2. Agent cooperation (e.g., action directives, taking turns, etc.)

Conflict resolution may require agents to modify their plans
- planned actions may be no longer executable
- new goals may have been added
- new state of the world may be incompatible with original plan
- ...

DFP (UniUD-UniPG-NMSU)
Modeling explicit communication

Communication might occur in a conflict-resolution phase, during the execution of a step of the concurrent plans.

Moreover, explicit actions laws can be used to specify

- **Broadcasting communication**:

  \[
  \text{request } C_1 \text{ if } C_2
  \]

- **Point-to-point communication**

  \[
  \text{request } C_1 \text{ to_agent } Ag \text{ if } C_2
  \]

A more general scheme:

\[
\text{request } C_1 \text{ [to_agent } Ag] \text{ if } C_2 \text{ [offering } C_3]\]
A volleyball match in $B^{AAC}$

Several independent Prolog agents (reasoning possibly on different machines) coordinated by Linda. Both the teams wish to score a point.
A volleyball match in \( \mathcal{B}^{\text{AAC}} \)

Several independent Prolog agents (reasoning possibly on different machines) coordinated by Linda. Both the teams wish to score a point.

Questions?