Action Description Languages meet CLP($\mathcal{FD}$) and Linda

Agostino Dovier   Andrea Formisano   Enrico Pontelli

Università di Udine & Università di Perugia & New Mexico State University

CILC 2010 — Cosenza - July 7-9, 2010

Action Description Languages

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

- A seminal work: *Action Languages*, Gelfond & Lifschitz, 1998
- Stable model semantics
- Many proposals, several languages:

$$\mathcal{A},\ \mathcal{B},\ \mathcal{C}+,\ \mathcal{K},\ \mathcal{CARD},\ \mathcal{AL},\ \mathcal{ALAN}, \ldots$$

ADL: an example of $\mathcal{B}$-like encoding

A fragment of action theory for the well-known three-barrels problem:

```
fluent contains(B,L)
```
for each $B \in \{5, 7, 12\}$ and $L \in \{0, ..., 12\}$

...

```
action fill(B1,B2)
```
for $B_1, B_2 \in \{5, 7, 12\}$

...

```
fill(5,12) causes contains(5,0)
        if contains(5,N1), contains(12,N2)
```
for each $N_1, N_2 \in \{0, ..., 12\}, 12 - N_2 \geq N_1$

```
fill(5,12) causes contains(12,N1 + N2)
        if contains(5,N1), contains(12,N2)
```
for each $N_1, N_2 \in \{0, ..., 12\}, 12 - N_2 \geq N_1$

...

ADL: many extensions, several features

- multi-valued fluents and constraints
- backward and forward time references
- durable actions, delayed effects, ...
- costs, preferences, maintenance goals, ...
- ...

An ADL offering these features: $\mathcal{B}^{\text{MV}}$
(CILC'07, CILC'09, TPLP'10,...)

Example in $\mathcal{B}^{MV}$

A fragment of action theory for the three-barrels problem:

`fluent contains(`B`) valued 0..12` for $B \in \{5, 7, 12\}$

...

`action fill(`B_1`,`B_2`)` for $B_1, B_2 \in \{5, 7, 12\}$

...

```
fill(5,12) causes contains(5)=0
          if 12-contains(12) ≥ contains(5)

fill(5,12) causes
          contains(12)=contains(5)^-1+contains(12)^-1
          if 12-contains(12) ≥ contains(5)
```

...

Many agents

Purpose: to extend $\mathcal{B}^{\text{MV}}$ so to enable the declarative specification of

- planning domains with multiple agents (concurrent, collaborative, self-interested,...)
- agents' policies for coordination and interaction
- strategies for concurrent plan-execution (conflict resolution, negotiation, replanning, coordination, ...)

Many agents

Purpose: to extend $\mathcal{B}^{\text{MV}}$ so to enable the declarative specification of

- planning domains with multiple agents (concurrent, collaborative, self-interested,...)
- agents' policies for coordination and interaction
- strategies for concurrent plan-execution (conflict resolution, negotiation, replanning, coordination, ...)

Two approaches:

- a centralized view of MAS (CILC'09, LPNMR'09,...)
- plan execution/integration for concurrent autonomous agents (CILC'10)

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.

So, 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:

- to design an ADL for autonomous agents coordination, to support the specification of strategies and policies for conflict resolution, communication, coordination, ...

$\mathcal{B}^{\text{AAC}}$

- to develop a prototype to execute ADL specifications and enable planning, concurrent plan-execution, and plan revision.

CLP(FD) + Linda

...ensuring extensibility of the ADL and modularity of the prototype!

The language: from $\mathcal{B}^{\text{MV}}$ to $\mathcal{B}^{\text{AAC}}$

Action declaration

$$\texttt{action}\ \ Act$$

Fluents...

$$\texttt{fluent}\ \ f_1, \ldots, f_h\ \ \texttt{valued}\ \ dom$$

expressions...

$$FE \quad ::= \quad n \mid f^t \mid f@r \mid FE_1 \oplus FE_2 \mid rei(C) \mid \ldots$$

...and constraints

A constraint C is a propositional combination of *primitive constraints* of the form $FE_1\ relop\ FE_2$.

The language: from $\mathcal{B}^{\text{MV}}$ to $\mathcal{B}^{\text{AAC}}$

Dynamic causal laws

$$\mathit{Act}\ \texttt{causes}\ C_{\mathit{Eff}}\ \texttt{if}\ C_{\mathit{Prec}}$$

Executability laws

$$\texttt{executable}\ \mathit{Act}\ \texttt{if}\ C$$

Specification of initial...

$$\texttt{initially}\ C$$

...and final states

$$\texttt{goal}\ C$$

The language: from $\mathcal{B}^{MV}$ to $\mathcal{B}^{AAC}$

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, ..., A_k

The language: from $\mathcal{B}^{MV}$ to $\mathcal{B}^{AAC}$

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

$$\texttt{action}\ \mathit{Act}\ \mathit{OPT}$$

$$
\begin{array}{rcl}
\mathit{OPT} & ::= & \texttt{on\_conflict}\ \mathit{OC}\ \mathit{OPT} \\
 & & \mid\ \texttt{on\_failure}\ \mathit{OF}\ \mathit{OPT} \\
\mathit{OC} & ::= & \texttt{retry\_after}\ T\ \texttt{[provided}\ C\texttt{]} \\
 & & \mid\ \texttt{forego [provided}\ C\texttt{]} \\
 & & \mid\ \texttt{arbitration} \\
\mathit{OF} & ::= & \texttt{retry\_after}\ T\ \texttt{[if}\ C\texttt{]} \\
 & & \mid\ \texttt{replan [if}\ C\texttt{] [add\_goal}\ C\texttt{]} \\
 & & \mid\ \texttt{fail [if}\ C\texttt{]}
\end{array}
$$

Johnny&Mary

Action theory for agent Johnny (and similarly for agent Mary)

```
agent johnny

fluent waterTemp, bottleTemp valued 5..100
fluent emptybottle valued 0..1

action open_left on_conflict retry_after 2
action open_right on_conflict retry_after 2
action fill

open_left causes waterTemp>50
open_right causes waterTemp<10
fill causes emptybottle=0 and bottleTemp=waterTemp

initially emptybottle=1 and waterTemp=20
```

Goal for Johnny: `goal emptybottle=0 and bottleTemp<20`
Goal for Mary: `goal emptybottle=0 and bottleTemp>20`

Simple plan-execution scheme

- A supervisor controls the execution of agents' plans, ensuring consistency of the global state
- Each agent sends a message to the supervisor declaring the intention to execute an action

Simple plan-execution scheme

- A supervisor controls the execution of agents' plans, ensuring consistency of the global state
- Each agent sends a message to the supervisor declaring the intention to execute an action
- The supervisor verifies the consistency of the concurrent execution of all "required actions", and
- if the case, determines the (minimal) subsets of conflicting agents/actions (while non-conflicting actions are enabled)

Simple plan-execution scheme

- A supervisor controls the execution of agents' plans, ensuring consistency of the global state
- Each agent sends a message to the supervisor declaring the intention to execute an action
- The supervisor verifies the consistency of the concurrent execution of all "required actions", and
- if the case, determines the (minimal) subsets of conflicting agents/actions (while non-conflicting actions are enabled)
- Conflicts can be resolved by executing various protocols (we implemented just a few basic possibilities: direct interaction among agents, arbitration,...)
- The conflict resolution phase might enable further actions and cause changes in agents' goals and plans
- Agents' actions might involve explicit communication...

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:

$$\texttt{request}\ C_1\ \texttt{if}\ C_2$$

- Point-to-point communication

$$\texttt{request}\ C_1\ \texttt{to\_agent}\ Ag\ \texttt{if}\ C_2$$

A more general scheme:

$$\texttt{request}\ C_1\ [\texttt{to\_agent}\ Ag]\ \texttt{if}\ C_2\ [\texttt{offering}\ C_3]$$

Example: A guitar maker

(fragment of an action theory from the paper)

```
agent guitar_maker
...
action make_guitar
...
make_guitar causes guitars++ and neck-- and
    strings=strings^-1-6 and body-- and
    pickup--    if pickup<2.
...
% interaction with seller:
request strings>5 to_agent seller
     if strings<6
    offering seller_account=seller_account^-1+8
...
```

“Local” and “global” semantics

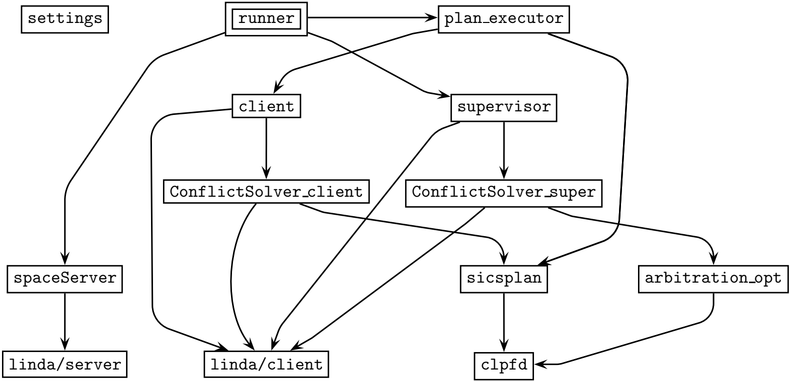
“locally”

Semantics of each single action theory is given in terms of transition systems (analogously to $\mathcal{B}$, $\mathcal{B}^{\text{MV}}$, ...)

“globally”

- Agents’ partial views of the world have to be always “projections” of a consistent global state of the world
- Each agent tries to execute its plan. If conflicts or failures prevent this, then the agent (should) re-plan
- Agents can communicate, ask for help, accept requests (and modify their goals),...

The prototype: system architecture



The future

The ADL and the prototype are open to a number of extensions and much work has to be done

- allow stronger interaction among agents (common plan development, sub-plans, ...)
- further strategies and policies for conflict resolution and coordination can be added
- more expressive communication (requests/answers involving complex conditions/constraints)
- notions of trust and payoff (mediated or not, dynamically evolving w.r.t. agents' behaviour,...)
- ...