Answer Set Programming

Reasoning about Actions

Martin Baláž

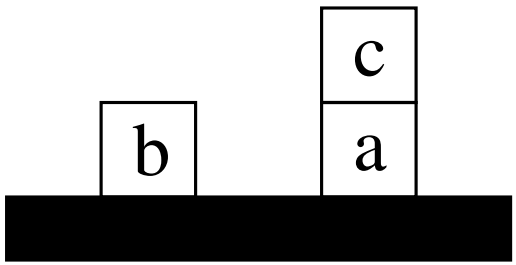
Department of Applied Informatics
Comenius University in Bratislava

2009/2010

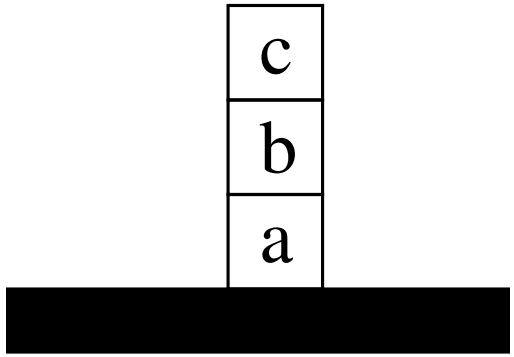
Outline

Blocks World Example

initial:



goal:



Language $\mathcal{K}$

```
fluents:    on(B, L) requires block(B), location(L).
            occupied(B) requires block(B).
actions:    move(B, L) requires block(B), location(L).
initially:  on(a, table). on(b, table). on(c, a).
always:     caused occupied(B) if on(B1, B), block(B).
            executable move(B, L) if B <> L.
            nonexecutable move(B, L) if occupied(B).
            nonexecutable move(B, L) if occupied(L).
            caused on(B, L) after move(B, L).
            caused -on(B, L1) after move(B, L),
                    on(B, L1), L <> L1.
            inertial on(B, L).
            noConcurrency.
goal:       on(c, b), on(b, a), on(a, table)? (3)
```

Background Knowledge

```
block(a).
block(b).
block(c).
location(table).
location(B) :- block(B).

$ dlv -FP blocksworld.plan background.dlv

STATE 0: occupied(a), on(a,table), on(b,table), on(c,a)
ACTIONS: move(c,table)
STATE 1: on(a,table), on(b,table), on(c,table), -on(c,a)
ACTIONS: move(b,a)
...
PLAN: move(c,table); move(b,a); move(c,b)
```

Language

Definition (Alphabet)

- action predicate symbols σ^{act}
- fluent predicate symbols σ^{fl}
- type predicate symbols σ^{typ}
- constant symbols σ^{con}
- variable symbols σ^{var}

Language

Definition (Term)

A *term* t is

- a constant $t \in \sigma^{con}$
- a variable $t \in \sigma^{var}$

Definition (Atom and Literal)

An *action* (resp. *fluent*, *type*) is an atom $p(t_1, \ldots, t_n)$, where $p \in \sigma^{act}$ (resp. $p \in \sigma^{fl}$, $p \in \sigma^{typ}$) is a predicate symbol with the arity $n \geq 0$, and $t_1, \ldots, t_n$ are terms.
An *action* (resp. *fluent*, *type*) *literal* is either an action (resp. fluent , type) or an action (resp. fluent, type) preceded by $\neg$.

Convention

- $a, a_1, \ldots, a_n$ denotes action atoms
- f denotes a fluent literal or *false*
- B denotes b_1 , ... , b_k , not b_{k+1} , ... , not b_l where $b_1, \ldots, b_l$ are fluent or type literals
- C denotes c_1 , ... , c_m , not c_{m+1} , ... , not c_n where $c_1, \ldots, c_n$ are action, fluent or type literals

Action and Fluent Declaration

Definition (Action and Fluent Declaration)

An *action* (resp. *fluent*) *declaration* is an expression of the form

$$p(X_1, \ldots, X_n) \texttt{ requires } t_1\texttt{, } \ldots\texttt{, } t_m\texttt{.}$$

where $p(X_1, \ldots, X_n)$ is an action (resp. fluent), $X_1, \ldots, X_n$ are variables, $t_1, \ldots, t_m$ are types, and and every X_i occurs in $t_1, \ldots, t_m$.

If $m = 0$, the keyword `requires` may be omitted.

Example

```
occupied(B) requires location(B).
move(B, L) requires block(B), location(L).
```

Causation Rule

Definition (Causation Rule)

A *causation rule* is an expression of one of the forms

`caused` *f* `if` *B* `after` *C*`.`

Rules where *C* is empty are *static rules*, all others are *dynamic rules*. When *B* (resp. *C*) is empty, `if` (resp. `after`) is omitted. When *B* and *C* are empty, `caused` is optional.

Example

```
caused on(B, L) after move(B, L).
caused occupied(B) if on(B1, B), block(B).
```

Executability Condition

Definition (Executability Condition)

An *executability condition* is an expression of the form

`executable` *a* `if` *C*`.`

When *C* is empty, `if` is omitted.

Example (Translation)

`executable a.`
`caused` $exec_a$ `after` *C*`.`
`caused false if not` $exec_a$ `after` *a*`.`

Initial State Constraint

Definition (Initial State Constraint)

An *initial state constraint* is an expression of the form

`initially caused` *f* `if` *B*`.`

When *B* is empty, `if` is omitted and `caused` is optional.

Example

```
initially on(a, table).
initially on(b, table).
initially on(c, a).
```

Query

Definition (Query)

A *query* is an expression of the form

$$g_1, \ldots, g_m, \texttt{not}\ g_{m+1}, \ldots, g_n?$$

where $g_1, \ldots, g_n$ are variable free fluent literals, and $0 \leq m \leq n$, $0 < n$.

Example

```
on(c, b), on(b, a), on(a, table)?
```

Planning Problem

Definition (Action Description)

An *action description* is a pair $AD = \langle D, R \rangle$ of a finite set D of action and fluent declarations and a finite set R of causation rules, initial state constraints, and executability conditions which do not contain positive cyclic dependencies among actions.

Definition (Planning Domain)

A *planning domain* is a pair $PD = \langle \Pi, AD \rangle$ of a stratified logic program Π and a an action description AD.

Definition (Planning Problem)

A *planning problem* $\mathcal{P} = \langle PD, q \rangle$ is a pair of a planning domain PD and a query q.

Inertia

Definition (Inertia)

An *inertia rule* is an expression of the form

`inertial` *f* `if` *B* `after` *C*`.`

When *B* (resp. *C*) is empty, `if` (resp. `after`) is omitted.

Example (Translation)

`caused` *f* `if not` $\neg f$`,` *B* `after` *f*`,` *C*`.`

Default

Definition (Default)

A *default rule* is an expression of the form

`default` f.

Example (Translation)

`caused` f `if not` $\neg f$.

Totality

Definition (Totality)

A *totality rule* is an expression of the form

$$\texttt{total } f \texttt{ if } B \texttt{ after } C\texttt{.}$$

When B (resp. C) is empty, `if` (resp. `after`) is omitted.

Example (Translation)

$\texttt{caused } f \texttt{ if not } \neg f\texttt{, } B \texttt{ after } f\texttt{, } C\texttt{.}$
$\texttt{caused } \neg f \texttt{ if not } f\texttt{, } B \texttt{ after } \neg f\texttt{, } C\texttt{.}$

Integrity Constraint

Definition (Integrity Constraint)

An *integrity constraint* is an expression of the form

`forbidden` *B* `after` *C*`.`

When *C* is empty, `after` is omitted.

Example (Translation)

`caused false if` *B* `after` *C*`.`

Non Executability Condition

Definition (Non Executability Condition)

A *non-executability condition* is an expression of the form

`nonexecutable` *a* `if` *B*`.`

If *B* is empty, `if` is omitted.
In case of conflicts, `nonexecutable` overrides `executable`.

Example (Translation)

`caused false after` *a*`,` *B*`.`

Concurrent Actions

By default, actions can be executed in parallel. For execution at most one action per step, we have to use the keyword `noConcurrency`.

Example (Translation)

`caused false after` a_1`,` a_2`.`

where $a_1 \neq a_2$.