Reasoning about Actions in Prioritized Default Theory

by
Tran Cao Son and Enrico Pontelli

JELIA 2002
Tech. Rep. at New Mexico State Univ.

November 8, 2002

Syntax of Action Language $\mathcal{B}$

Domain Signature:

- a set $\mathcal{F}$ of fluents.
- a set $\mathcal{A}$ of actions. $\mathcal{F}$ and $\mathcal{A}$ are disjoint.

Fluent literals: fluents and fluents preceded by symbol $\neg$ (e.g., p, $\neg q$).

Dynamic causal laws are statements of the form:

$$a \textbf{ causes } f \textbf{ if } p_1, \ldots, p_n$$

Informal reading: execution of action a causes fluent literal f to become true at the next moment of time if fluent literals $p_1, \ldots, p_n$ are true when a is executed.

Executability conditions:

$$a \textbf{ executable\_if } p_1, \ldots, p_n$$

Syntax of Action Language $\mathcal{B}$

Static causal laws:

$$f \textbf{ if } p_1, \ldots, p_n$$

Informal reading: at any moment of time, fluent literal f is true if fluent literals $p_1, \ldots, p_n$ are true.

Collections of dynamic laws, static laws, and executability conditions form the *domain description*.

Axioms describing the *initial situation*:

$$\textbf{initially } f$$

Queries are described by statements of the form:

$$\varphi \textbf{ after } \alpha$$

Informal reading: fluent formula φ is true after action sequence α has been executed

Syntax of Action Language $\mathcal{B}$

An *action theory* is a pair $\langle D, \Gamma \rangle$, where D is a domain description and Γ is a collection of axioms describing the initial situation.

Example. A briefcase has two clasps. Actions are available to unfasten each clasp. The briefcase becomes open when the two clasps are unfastened.

Objects: briefcase, clasps (c_1, c_2)
Fluents: open (briefcase is open), fastened(X) (clasp X is fastened)
Actions: unfasten(X) (unfasten clasp X)

% action unfasten(X) causes clasp X to be unfastened.
$unfasten(X)$ **causes** $\neg fastened(X)$

% if both clasps are unfastened, the briefcase pops open
$open$ **if** $\neg fastened(c_1), \neg fastened(c_2)$

Semantics of $\mathcal{B}$

Let D be a domain description in $\mathcal{B}$. D describes a transition diagram, i.e. a directed graph whose nodes correspond to possible states of the world and arcs correspond to transitions of state due to the execution of actions.

An *interpretation*, I, of the fluents in D is a maximal consistent set of fluent literals from $\mathcal{F}$.

A fluent f is true (resp. false) in I if $f \in I$ (resp. $\neg f \in I$).
Truth of fluent formulas is defined as usual.
Formula φ *holds* in I ($I \models \varphi$) if φ is true in I.

Semantics of $\mathcal{B}$

Let F be a consistent set of fluent literals and K be a set of static causal laws. F is *closed under* K if, for every static causal law "f **if** $p_1, \ldots, p_n$" in K, if $\{p_1, \ldots, p_n\} \subseteq F$, then $f \in F$. $CN_K(F)$ denotes the least consistent set of fluent literals from D that contains F and is closed under K (closure of F under K).

An interpretation, σ, of the fluents in D is a *state* (of D) if σ is closed under the static causal laws of D. Action a is executable in a state σ if there exists an executability condition

$$a \ \textbf{executable\_if} \ p_1, \ldots, p_n$$

in D such that $\sigma \models p_1 \wedge \ldots \wedge p_n$.

The *immediate effect of an action* a *in state* σ is:

$$E(a, \sigma) = \{f \mid \text{“}a \ \textbf{causes} \ f \ \textbf{if} \ p_1, \ldots, p_n\text{”} \in D \text{ and } \sigma \models p_1 \wedge \ldots \wedge p_n\}$$

Semantics of $\mathcal{B}$

Successor State

Let D be a domain description, K be the set of static laws of D, σ_0, σ_1 be states and a be an action.

σ_1 is a successor state of σ_0 under the execution of a if:

- a is executable in σ_0, and
- $\sigma_1 = CN_K(E(a, \sigma_0) \cup (\sigma_0 \cap \sigma_1))$.

A sequence $\mathcal{T} = \sigma_0, a_0, \sigma_1, a_1, \sigma_2, \ldots, a_{n-1}, \sigma_n$ is a *trajectory in* D if, for each transition $\sigma_i, a_i, \sigma_{i+1}$ of $\mathcal{T}$, σ_{i+1} is a successor state of σ_i under a_i.

The possible trajectories of an action theory $\langle D, \Gamma \rangle$ are the trajectories in D $\sigma_0, a_0, \sigma_1, \ldots, a_{n-1}, \sigma_n$ where σ_0 is described with Γ.

Semantics of $\mathcal{B}$

Example. Consider the previous briefcase example. Let the initial situation, Γ, be:

$$\begin{array}{l} \textbf{initially } \neg open \\ \textbf{initially } fastened(c_1) \\ \textbf{initially } fastened(c_2) \end{array}$$

The initial state of all possible trajectories is

$$\sigma_0 = \{\neg open, \neg fastened(c_1), \neg fastened(c_2)\}.$$

We can check that, for any trajectory $\sigma_0, unfasten(c_1), \sigma_1$ of $\langle D, \Gamma \rangle$:

$$\sigma_1 \models \neg fastened(c_1).$$

We write that

$$\langle D, \Gamma \rangle \models \quad \neg fastened(c_1) \textbf{ after } unfasten(c_1).$$

Prioritized Default Theories

Default: a rule that can be defeated (i.e., not applied) if its application causes inconsistencies.

Prioritized Default Theories allow for the specification of rules, defaults, and priorities between conflicting defaults.

Example.

1. Normally, cars have 4 seats.
2. Pick-up trucks are cars.
3. Normally, pick-up trucks have 2 seats.
4. My Ranger is a pick-up truck.

Desired conclusion: my Ranger has 2 *seats.*

Syntax of Prioritized Default Theories

The concepts of term, atom, literal are defined as usual in logic languages.

A rule is a statement of the form:

$$rule(r, l_0, [l_1, \ldots, l_m])$$

where r is the name of the rule, $l_0, \ldots, l_m$ are literals and [] is the list operator. $body(r)$ denotes $[l_1, \ldots, l_m]$. $head(r)$ denotes l_0.

A default is a statement of the form:

$$default(d, l_0, [l_1, \ldots, l_m])$$

d is the name of the default rule. $body(d)$ and $head(d)$ are defined as for rules.

A preference statement is:

$$prefer(d_1, d_2)$$

where d_1, d_2 are names of default rules.

A Prioritized Default Theory is a collection of rules, defaults, and preference statements.

Semantics of Prioritized Default Theories

The semantics of Prioritized Default Theories (PDTs) is defined by translation to A-Prolog. Let T be a prioritized default theory. The semantics of T is defined by the answer set semantics of $T \cup Inf \cup Def$, where Inf and Def are:

$$
Inf \left\{
\begin{array}{lll}
holds(L) & \leftarrow & rule(R, L, Body), hold(Body). \\
holds(L) & \leftarrow & default(D, L, Body), hold(Body), \\
 & & \text{not } defeated(D), \text{not } holds(\neg L). \\
hold([]). & & \\
hold([H|T]) & \leftarrow & holds(H), hold(T).
\end{array}
\right.
$$

$$
Def \left\{
\begin{array}{lll}
defeated(D) & \leftarrow & default(D, L, Body), \\
 & & holds(\neg L). \\
defeated(D) & \leftarrow & default(D, L, Body), \\
 & & default(D_1, L_1, Body_1), \\
 & & prefer(D_1, D), \\
 & & hold(Body_1), \\
 & & \text{not } defeated(D_1).
\end{array}
\right.
$$

Notice that these definitions differ from the ones presented in (Gelfond and Son, 1998).

Action Theories as PDTs

Given an action theory $\langle D, \Gamma \rangle$, consider the language containing:

- atoms of the form $f(T)$ *[fluent literal f is true at time T]*
- atoms of the form $possible(a, T)$ *[action a is executable at time T]*
- atoms of the form $occ(a, T)$ *[action a occurs at time T]*
- rule names for dynamic, static laws, and executability conditions
- default names of the form $inertial(f, T)$, where f is a fluent literal and T denotes a time point.

Action Theories as PDTs

Translation. A action theory $\langle D, \Gamma \rangle$, is translated in a prioritized default theory $\Pi^n(D, \Gamma)$ as follows (notice that T ranges from 0 to n):

Dynamic laws "a **causes** f **if** $p_1, \ldots, p_k$" are translated into

$$rule(dynamic(f, a, T), f(T+1), \\ [p_1(T), \ldots, p_k(T), possible(a, T)]) \leftarrow occ(a, T)$$

Executability conditions "a **executable_if** $p_1, \ldots, p_k$" are translated into

$$rule(executable(a, T), possible(a, T), [p_1(T), \ldots, p_k(T)])$$

Static laws "f **if** $p_1, \ldots, p_k$" are translated into

$$rule(causal(f, T), f(T), [p_1(T), \ldots, p_k(T)])$$

The inertia axiom is represented explicitly as

$$default(inertial(f, T), f(T+1), [f(T)])$$

Axioms "**initially** f" are translated into

$$holds(f(0))$$

Action Theories as PDTs

The translation is correct, i.e. the semantics of $\Pi^n(D,\Gamma)$ coincides with the semantics of $\langle D,\Gamma\rangle$. Let M be an answer set of $\Pi^n(D,\Gamma)$ and $s_i(M) = \{f \mid holds(f(i)) \in M\}$.

Theorem 1. Let $\langle D,\Gamma\rangle$ be a complete and consistent action theory.

[*soundness*]
For every sequence of actions $a_0,\dots,a_{n-1}$ such that there exists a trajectory $\sigma_0,a_0,\dots,\sigma_n$, and for every answer set M of
$\Pi^n(D,\Gamma) \cup \{occ(a_i,i) \mid 0 \le i < n\}$,

$s_{i+1}(M)$ is a successor state of $s_i(M)$ under a_i.

[*completeness*]
For every trajectory $\sigma_0,a_0,\dots,\sigma_n$, there exists an answer set M of
$\Pi^n(D,\Gamma) \cup \{occ(a_i,i) \mid 0 \le i < n\}$ such that

$$s_i(M) = \sigma_i \text{ for every } i, 1 \le i \le n$$

Using smodels

Issues to deal with when preparing the encoding for smodels:

Representation of lists of preconditions. The authors give a name to each list and include facts associating names of lists with their elements. For example, list $[p_1(T), \ldots, p_k(T)]$ is represented as:

$$\begin{array}{l} in(p_1(T), list_1). \\ in(p_2(T), list_1). \\ \ldots \\ in(p_k(T), list_1). \end{array}$$

Checking that lists of preconditions are satisfied. The authors introduce a new relation, $holds\_set(List)$, where $List$ is the name of a list. The relation is defined as follows:

$$\begin{array}{lcl} not\_holds\_set(List) & \leftarrow & in(F, List), \text{not } holds(F). \\ holds\_set(List) & \leftarrow & \text{not } not\_holds\_set(List). \end{array}$$

Computing Trajectories

Let $\langle D, \Gamma \rangle$ be a action theory and $\varphi = f_1 \wedge \ldots \wedge f_k$. We can compute the trajectories such that $\sigma_n \models \varphi$ by adding to the smodels encoding the following rules:

$$
\begin{array}{rcl}
goal(T) & \leftarrow & holds(f_1, T), \ldots, holds(f_k, T). \\
 & \leftarrow & \text{not } goal(n). \\
1\{occ(A, T)\}1 & \leftarrow & T < n.
\end{array}
$$

Preferences on Actions

"Riding a bus and a taxi are two alternatives to go to the airport. If someone wants to save money, he will prefer riding the bus."

When action theories are encoded using Prioritized Default Theory, the application of dynamic laws can be controlled by introducing literals of the form

$$block(r, [l_1, \ldots, l_m]).$$

Set Inf from the previous encoding is modified so that the first statement becomes:

$$\begin{array}{rcl} holds(L) & \leftarrow & rule(R, L, Body), hold(Body), \text{not } blocked(R). \\ blocked(R) & \leftarrow & block(R, Body), hold(Body). \end{array}$$

For each preference $prefer_{act}(a, b)$, the new encoding will also include:

$$\begin{array}{l} block(dynamic(F, b, T), \\ \qquad [p_1(T), \ldots, p_k(T), possible(a, T)]) \leftarrow goal(n). \end{array}$$

"If it is possible to execute action a and the goal is achievable, action b should not be executed."

Preferences on Actions

The previous approach does not ensure completeness: if a and b are possible, and a is preferred to b but a does not lead to the goal, then the program may fail to produce a trajectory.

Alternative: use the **maximize** construct of smodels. For each statement $prefer_{act}(a,b)$ and each time point t, include

$$\mathbf{maximize}[occ(a,t) = 1, occ(b,t) = 0].$$

Preferences on actions are static.

Preferences on Literals and smodels

The **maximize** statement can also be used to encode preferences between literals. For example, we may prefer trajectories where the final state contains $health(good)$ instead of $health(bad)$.

Preference between fluent literals is represented by relation $\prec$. $f_2 \prec f_1$ says that models where f_2 is true are preferred to models containing f_1.

If $\prec$ is an irreflexive partial order, and the set of preferences over literals is finite, there exists a finite number of maximal length sequences of literals $f_1, \ldots, f_k$ such that $f_k \prec f_{k-1} \prec \ldots \prec f_1$. For each sequence, we can include in the program a statement:

$$\mathbf{maximize}[f_1 = 0, \ldots, f_k = k - 1].$$

Preferences on literals are static.

Limitation (valid also for preferences on actions): smodels currently does not handle properly multiple **maximize** constructs.

Conclusions

The main points discussed in the paper are:

- Encoding of Action Theories using Prioritized Default Theories;
- Encoding of Prioritized Default Theories in the language of smodels;
- Use of smodels to compute the trajectories for a goal;
- Use of various techniques to select preferred trajectories.

Final Observations

[*Encoding of lists for smodels*].
The encoding of lists for smodels is incorrect. Consider static law

$$p \textbf{ if } p.$$

Applying the translation described in the paper, and simplifying a little the code, we obtain the following encoding of the causal law:

$$\begin{array}{rcl} holds(p(T)) & \leftarrow & holds\_set(l_1(T)). \\ holds\_set(l_1(T)) & \leftarrow & \text{not } not\_holds\_set(l_1(T)). \\ not\_holds\_set(l_1(T)) & \leftarrow & \text{not } holds(p(T)). \end{array}$$

This encoding can be shown to be equivalent to:

$$\begin{array}{rcl} holds(p(T)) & \leftarrow & \text{not } not\_holds\_set(l_1(T)). \\ not\_holds\_set(l_1(T)) & \leftarrow & \text{not } holds(p(T)). \end{array}$$

Consider now initial state $\sigma_0 = \{\neg p\}$ and action a (with no direct effect). The only successor state of σ_0 under a is $\sigma_1 = \{\neg p\}$.

However, the smodels encoding will return two models, one where the successor state is σ_1, the other where the successor state is $\sigma_1' = \{p\}$.

Final Observations

[*Use of* **maximize**].
The use of maximize may lead to unintuitive results, in particular for preferences on literals. Consider the following program (for sake of simplicity, it is not the encoding of a action theory)

$$
\begin{array}{lcl}
a & \leftarrow & \text{not } b. \\
b & \leftarrow & \text{not } a. \\
 & & \\
c & \leftarrow & \text{not } d. \\
d & \leftarrow & \text{not } c. \\
 & & \\
 & \leftarrow & d, b.
\end{array}
$$

Suppose that our preference is $d \prec c \prec b \prec a \prec z$. $\mathbf{maximize}[d = 4, c = 3, b = 2, a = 1, z = 0]$ gives model $\{c, b\}$. Notice that $\{d, a\}$ is also a model. This is *almost* intuitively acceptable (what about $\{d\}$??).

Now, let us introduce a new atom k, such that

$$d \prec k \prec c \prec b \prec a \prec z.$$

This is achieved by replacing the previous **maximize** by:

$$\mathbf{maximize}[d = 5, k = 4, c = 3, b = 2, a = 1, z = 0].$$

Since k is not defined by the program, nothing should change in the relationship between d, c, b, a, and z. However, $\{c, b\}$ is **no more** a model of the program. (smodels returns $\{d, a\}$.)