Reasoning about Concurrent Actions and Observations

Renwei Li and Luís Moniz Pereira

Department of Computer Science
Universidade Nova de Lisboa
2825 Monte da Caparica, Portugal
{renwei | lmp}@di.fct.unl.pt

Presented in AGP'96, San Sebastian, Spain
August 15 - 18, 1996
Action description language $\mathcal{A}$
(by Gelfond and Lifschitz)

\[ F \text{ after } A_1; \ldots; A_m \]
\[ \text{initially } F \]
\[ A \text{ causes } F \text{ if } P_1; \ldots; P_n \]
\[ A \text{ causes } F \]

A domain description is a set of value propositions and effect propositions. The semantics is defined by using states and transitions.
Stolen Car Problem $D1$
(by Kautz)

\begin{align*}
\text{initially} & \quad \neg \text{Stolen} \\
\text{Stolen after} & \quad \text{Wait; Wait; Wait} \\
\text{Steal causes} & \quad \text{Stolen}
\end{align*}

It can be shown that the above domain description has no model. But why?

In order to solve the problems such as SCP, we need to extend the action language $A$ with concurrent actions, and to express incomplete knowledge about the past history.
Purposes of this talk:

- extend $\mathcal{A}$ with concurrent actions;
- extend $\mathcal{A}$ with observation propositions;
- Translation of domain descriptions into abductive logic programs;
- Diagnose domains when expected behaviour is in disagreement with observed behaviour.
Concurrent action: A finite nonempty set of primitive actions.

How should we define semantics of concurrent actions?

General observation: the effects of a concurrent action is aggregation of effects of its primitive actions.

Example: Switch a light and open a door.

But ... what’s about primitive actions with conflicting effects? Example: Open and close the same door at the same time.
Lin-Shoham’s solution:

After you open and close the same door at the same time, the door keeps unchanged.

Baral-Gelfond’s solution:

After you open and close the same door at the same time, the resulting situation does not exist, and all successive actions are disabled.

Our solution:

After you open and close the same door at the same time, the status of the door is not known (for the moment).
Action Description Language \( \mathcal{AC} \).

Primitive action alphabet \( \Sigma_A \).

Fluent name alphabet \( \Sigma_F \).

The syntax of \( \mathcal{AC} \) is the same as that of \( \mathcal{A} \) except an action is defined to be a finite nonempty set of primitive actions.

Example: Yale Shooting Problem (Hanks and McDermott):

\[
D_{YSP} = \left\{ \begin{array}{l}
\text{initially } \neg \text{Loaded.} \\
\text{initially } \text{Alive.} \\
\text{Shoot \textbf{causes} } \neg \text{Alive if Loaded.} \\
\text{Shoot \textbf{causes} } \neg \text{Loaded.} \\
\text{Load \textbf{causes} } \text{Loaded}
\end{array} \right\}
\]
Semantics of $\mathcal{AC}$

A state $\sigma$ is a pair of sets of fluent names $\langle \sigma^+, \sigma^- \rangle$ such that $\sigma^+$ and $\sigma^-$ are disjoint, i.e., $\sigma^+ \cap \sigma^- = \emptyset$.

A transition function $\Phi$ is a mapping from the set of pairs $(A, \sigma)$, where $A$ is an action expression and $\sigma$ is a state, into the set of states.

A structure is a pair $(\sigma_0, \Phi)$, where $\sigma_0$ is a state, called the initial state of the structure, and $\Phi$ is a transition function.

The rest of it needs 8 –15 minutes to explain, as it is a little complicated and technical. Read the paper afterwards if you want to know more.
Abductive logic programming

Abductive normal logic programming framework: \((P, A, I)\), where

\(P\) is a set of normal logic programming rules,

\(A\) a set of abducibles, and

\(I\) a set of integrity constraints.

The semantics of an abductive normal logic program is defined to be the union of the integrity constraints and the first-order theory by completing the non-abducible predicates together with Clark’s Equality Theory.

For all the acyclic logic programs, the predicate completion semantics coincide with other semantics such as the stable model semantics and the well-founded model semantics.
Translation into abductive logic programs

1. Auxiliary predicates about subactions: Assume that we have standard rules for set-related predicates $\text{Subseteq}(S_1, S_2)$ and $\text{Member}(A, S)$, by which we can define $\text{subacteq}(S_1, S_2)$ and $\text{subact}(S_1, S_2)$ about subactions.

2. Initialization: Let $s_0$ be a new symbol to denote the initial situation.

   $\text{is\_true}(F, s_0) \leftarrow \text{initially\_true}(F)$.
   $\text{is\_false}(F, s_0) \leftarrow \text{initially\_false}(F)$.

3. Auxiliary Predicates: These auxiliary predicates will be used to define the two main predicates $\text{is\_true}(F, S')$ and $\text{is\_false}(F, S')$, which indicate whether fluent $F$ is true or false in situation $S$. 
\text{initiates}(A, F, S) \\
\quad \leftarrow \text{imm\_initiates}(A, F, S).
\text{initiates}(A, F, S) \\
\quad \leftarrow \text{subacteq}(B, A), \\
\quad \text{imm\_initiates}(B, F, S), \\
\quad \text{not clip\_initiates}(F, B, A, S).
\text{terminates}(A, F, S) \\
\quad \leftarrow \text{imm\_terminates}(A, F, S)
\text{terminates}(A, F, S) \\
\quad \leftarrow \text{subacteq}(B, A), \\
\quad \text{imm\_terminates}(B, F, S), \\
\quad \text{not clip\_terminates}(F, B, A, S).
\text{causes}(F, S, A, B) \\
\quad \leftarrow \text{subacteq}(B, A), \\
\quad \text{imm\_initiates}(B, F, S), \\
\quad \text{not clip\_initiates}(F, B, A, S), \\
\quad \text{not clip\_Cause1}(F, B, A, S).
\text{causes}(F, S, A, B) \\
\quad \leftarrow \text{subacteq}(B, A), \\
\quad \text{imm\_terminates}(B, F, S), \\
\quad \text{not clip\_terminates}(F, B, A, S), \\
\quad \text{not clip\_Cause2}(F, B, A, S).
\[
\text{\textit{delta}}(A, S, F) \\
\quad \leftarrow \text{\textit{initiates}}(A, G, S), \\
\quad \text{\textit{terminates}}(A, G, S), \\
\quad \text{\textit{causes}}(G, S, A, B), \\
\quad \text{\textit{initiates}}(B, F, S).
\]

\[
\text{\textit{delta}}(A, S, F) \\
\quad \leftarrow \text{\textit{initiates}}(A, G, S), \\
\quad \text{\textit{terminates}}(A, G, S), \\
\quad \text{\textit{causes}}(G, S, A, B), \\
\quad \text{\textit{terminates}}(B, F, S).
\]
4. Main Predicates: The following predicates are used to determine whether a fluent is true or false in a situation.

\[
\begin{align*}
\text{is\_true}(F, \text{result}(A, S)) & \leftarrow \text{is\_true}(F, S), \\
& \quad \text{not delta}(A, S, F), \\
& \quad \text{not terminates}(A, F, S).
\end{align*}
\]

\[
\begin{align*}
\text{is\_false}(F, \text{result}(A, S)) & \leftarrow \text{is\_false}(F, S), \\
& \quad \text{not delta}(A, S, F), \\
& \quad \text{not initiates}(A, F, S).
\end{align*}
\]

\[
\begin{align*}
\text{is\_false}(F, \text{result}(A, S)) & \leftarrow \text{terminates}(A, F, S), \\
& \quad \text{not delta}(A, S, F), \\
& \quad \text{not initiates}(A, F, S).
\end{align*}
\]
5. Domain-Specific Predicates

The syntax and semantics of the following predicates depend on domain descriptions. Let $F \in \Sigma_f$ be a fluent name. We write $\text{holds}(F, S)$ and $\text{holds}(\neg F, S)$ to stand for $\text{is\_true}(F, S)$, and $\text{is\_false}(F, S)$, respectively.

- For a causes $f$ if $p_1, \ldots, p_n$ in $D$, where $f$ is positive:
  
  $$\text{imm\_initiates}(a, f, S) \leftarrow \text{holds}(p_1, S), \ldots, \text{holds}(p_n, S).$$

- For a causes $\neg f$ if $p_1, \ldots, p_n$, where $f$ is positive:
  
  $$\text{imm\_terminates}(a, f, S) \leftarrow \text{holds}(p_1, S), \ldots, \text{holds}(p_n, S).$$
Integrity constraints, $IC_D$, is defined as follows: For each value proposition $F$ after $A_1; \ldots; A_m$, we have:

$$\text{holds}(F, \text{result}(A_1; \ldots; A_m, s_0))$$

**Proposition 1** $\pi D$ is an acyclic program with first-order constraints in the sense of Apt.

**Corollary 2** For $\pi D$, $\text{COMP}(\pi D)$ coincides with its generalized stable model semantics and generalized well-founded model semantics.
Theorem 3 (Soundness) Let $D$ be any domain description. For any value proposition $Q$, if $COMP(\pi D) \models \pi Q$, then $D$ entails $Q$.

Theorem 4 (Completeness) Let $D$ be a domain description. For any value proposition $Q$, if $D$ entails $Q$, then $COMP(\pi D) \models \pi Q$.

Thus, proof of entailments in $\mathcal{AC}$ is reduced to queries in logic programming. In our work we have experimented it with our belief revision system REVISE which uses the WFSX semantics. The use of REVISE is not essential, since the major semantics of $\pi D$ coincide.
Observations

**observed** \( F \) after \( A_1; \ldots; A_m \)

An observation tells observed practical behaviour, while a domain description tells the ideal behaviour. Whenever there is a discrepancy between them, there is a diagnosis problem.
Diagnosis Problem

\( D \) — domain description

\( OBS = \{ \text{observed } Q_1, \ldots, \text{observed } Q_m \} \)

Let \( OVP = \{Q_1, \ldots, Q_m\} \). We call \((D, OBS)\), or simply \((D, OVP)\), a diagnostic domain. We also say that there is a diagnostic explanation problem, or simply diagnosis problem, for the domain iff \( D \cup OVP \) is inconsistent.

Expansion

Let \( f \text{ after } a_1; \ldots; a_m \) and \( f \text{ after } b_1; \ldots; b_m \) be two value propositions. The latter is an expansion of the former iff for every \( 1 \leq i \leq m \), \( a_i \subseteq b_i \).

Expansion

Let \( P_1 \) and \( P_2 \) are two sets of value propositions. \( P_2 \) is said to be an expansion of \( P_1 \) iff (i) for every \( Q_1 \) of \( P_1 \) there is a \( Q_2 \) in \( P_2 \) which is an expansion of \( Q_1 \); and (ii) every \( Q'_2 \) of \( P_2 \) is an expansion of a \( Q'_1 \) of \( P_1 \).
Diagnostic explanation
A diagnostic explanation for \((D, OVP)\) is a set of value propositions \(E\) such that (i) \(D \cup E\) is consistent; (ii) \(E\) is an expansion of \(OBS\).

Example: the following three sets are diagnostic explanations for the SCP diagnostic domain \(D_{scp} \cup OBS_{scp}\):

\[
E_1 = \{Stolen \ after \ \{Steal, Wait\}; \ Wait; \ Wait\}
\]
\[
E_2 = \{Stolen \ after \ \{Steal, Wait\}; \ Wait\}
\]
\[
E_3 = \{Stolen \ after \ \{Steal, Wait\}\}
\]

Any of \(E_1, E_2, E_3\) will explain why the car is missing in the parking lot.
Preferred Explanation
Suppose $E_1$ and $E_2$ are two diagnostic explanations for $(D,OVP)$. We say that $E_1$ is preferred to $E_2$ iff $E_2$ is an expansion of $E_1$. A diagnostic explanation $E$ for $(D,OVP)$ is said to be most preferred iff for any diagnostic explanation $E'$ for $(D,OVP)$ if $E'$ is preferred to $E$, then $E = E'$.

Translation into abductive logic programs

$occ(A, S_1, S_2) — A$ occurs in situation $S_1$ and leads to the situation $S_2$ possibly together with some other actions.

Abducible predicate: $happens(A, S_1, S2)$
\(\pi_{OBS}\) consists of a set of logic programming rules \(P_{OBS}\) and a set of integrity constraints \(IC_{OBS}\), defined as follows:

1. For each observed \(f\) after \(a_1; a_2; \ldots; a_m\):

\[
occ(a_1, s_0, s_{a_1}) \leftarrow \\
\ldots \\
occ(a_m, s_{a_1}; a_2; \ldots; a_{m-1}, s_{a_1}; a_2; \ldots; a_m) \leftarrow
\]

And adding the following into \(IC_{OBS}\) as a constraint:

\[
false \leftarrow not\ holds(f, s_{a_1}; a_2; \ldots; a_m)
\]

2. We add the following additional rules for \(occ/3\):

\[
occ(A, S_1, S_2) \leftarrow atomic(A), happens(A, S_1, S_2)
\]

where \(atomic(A)\) denotes that \(A\) is an atomic action, i.e. \(A \in \Sigma_a\).
For a given diagnostic domain \((D, OBS)\), its translation is defined to be \(\pi D \cup \pi OBS\).

\(\pi D \cup \pi OBS\) is still an acyclic program with first-order constraints.

We can use an abductive query procedure to generate abductive answers to queries. In particular, an abductive answer, if any, may be generated for the query \(\leftarrow \neg false\). If such an answer \(\Delta\) exists, the theory \(COMP(P_D \cup P_OBS) \cup IC_D \cup IC_OBS \cup \Delta\) is consistent.
Explanation Soundness
Let \((D, OBS)\) be a diagnostic domain. If there exists \(\Delta_H\) such that \(\pi D \cup \pi OBS \cup \Delta_H\) is consistent, then there is a diagnostic explanation \(E\) for \((D, OBS)\).

Explanation Completeness
Let \((D, OBS)\) be a diagnostic domain. Let \(E\) be a diagnostic explanation. Then, there is a \(\Delta_H\) such that \(\pi D \cup \pi OBS \cup \Delta_H\) is consistent.

If \(D \cup OVP\) is consistent, then there is an abductive answer \(\Delta\) to \(\leftarrow \neg false\) such that no atoms of the form \(happens(a, s_i, s_j)\) appear in \(\Delta\).