Foundations of
Cognitive Robotics

Steffen Hölldobler and Michael Thielscher
Artificial Intelligence Institute
Dresden University of Technology
{sh,mit}@inf.tu-dresden.de

Contents

1. Situation Calculus
2. Fluent Calculus
3. Event Calculus
4. Nondeterministic Actions
5. Ramification Problem
6. Specificity
7. Concurrent Actions
8. Continuous Change
9. Agent Programming Languages
Cognitive Robotics

“We believe that human intelligence depends essentially on the fact that we can represent in language facts about our situation, our goals, and the effects of the various actions we can perform. Moreover, we can draw conclusions from the facts to the effects that certain sequences of actions are likely to achieve our goals.” (John McCarthy 1963)
Situation Calculus

- State, situations, actions and causality
- Fluents
- A first order formalization: the situation calculus
- Frame, ramification, qualification and predication problem
- Frame, effect and successor state axioms
- Plan synthesis and regression
- GOLOG
What’s the Goal?

—we want agents to decide what to do in order to achieve their goals
  ➔ causality, ability, knowledge and believe
  ➔ causes, can, knows, believes
  ➔ knowledge representation and reasoning, strategy, search and control

—we applications:
  ➔ high level control of robots and industrial processes
  ➔ intelligent software agents
  ➔ discrete event simulations
  ➔ etc.

The Framework due to McCarthy (1963)

- “General properties of causality, and certain obvious but until now unformalized facts about the possibility and results of actions, are given as axioms.”

- “It is a logical consequence of the facts of a situation and the general axioms that certain persons can achieve certain goals by taking certain actions.”

- “The formal descriptions of situations should correspond as closely as possible to what people may reasonably be presumed to know about them when deciding what to do.”

States, Actions and Causality

- A **state** is the complete state of affairs of the universe at an instant of time.
- Given a state, **laws of motion** (or **actions**) determine all future states.
- Example: fork lift trucks.
- Neither states nor actions can be completely described.
  -↩ inherent partiality
  -↩ only facts about situations and actions can be specified
  -↩ fluents
- Language: first order logic plus some extensions.
Situations and Fluents

▷ A situation is a term denoting a state. It records the history of how a state has evolved.
  - $s_0$ is called the initial situation and denotes the initial state.
  - $do(move(X, Y), S)$ denotes the situation obtained by executing the action $move(X, Y)$ in situation $S$.

▷ A fluent is a term denoting a fact about a situation that may change when actions are executed.
  - $at(P, X)$ denotes the fact that agent $P$ is at location $X$.
  - $raining(X)$ denotes the fact that it is raining at location $X$.

▷ The binary predicate holds is used to denote that a certain fluent holds in a particular situation.
  - $holds(at(P, X), s_0)$
    denotes that agent $P$ is at location $X$ in the initial situation $s_0$.

Example: Fork Lift Truck (0)

The goal is to have container $a$ on $b$ at $port_2$. 
Example: Fork Lift Truck (1)

- $\text{on}(X, Y)$ denotes that container $X$ is on location or container $Y$.
- $\text{clear}(Y)$ denotes that container or location $Y$ is clear.
- $\text{move}(X, Y)$ denotes that container $X$ is moved onto container or location $Y$.

\[ I \leftrightarrow \text{holds}(\text{on}(a, \text{track1}), s_0) \]
\[ \wedge \text{holds}(\text{on}(b, \text{track2}), s_0) \]
\[ \wedge \text{holds}(\text{clear}(\text{port1}), s_0) \]
\[ \wedge \text{holds}(\text{clear}(\text{port2}), s_0) \]
\[ \wedge \text{holds}(\text{clear}(\text{temp_store}), s_0) \]
\[ \wedge \text{holds}(\text{clear}(a), s_0) \]
\[ \wedge \text{holds}(\text{clear}(b), s_0) \]

\[ G \leftrightarrow \text{holds}(\text{on}(a, b), S') \]
\[ \wedge \text{holds}(\text{on}(b, \text{port2}), S') \]
\[ \wedge \text{holds}(\text{clear}(a), S') \]
\[ \wedge \text{holds}(\text{clear}(\text{track1}), S') \]
\[ \wedge \text{holds}(\text{clear}(\text{track2}), S') \]
\[ \wedge \text{holds}(\text{clear}(\text{temp_store}), S') \]
\[ \wedge \text{holds}(\text{clear}(\text{port1}), S') \]

Example: Fork Lift Truck (2)

▷ How must $A$ be defined such that $\models I \land A \rightarrow G$?

▷ $move(X, Z)$:

$$holds(on(X, Y), S') \land holds(clear(X), S') \land holds(clear(Z), S') \land Z \neq X$$

$$\rightarrow holds(on(X, Z), S') \land holds(clear(Y), S')$$

where $S' = \text{do}(move(X, Z), S)$

▷ Is the stack action sufficient to show $\models I \land A \rightarrow G$?
Frame Problem

▷ Common Assumption: Unless an action explicitly causes a fact to hold or not to hold the facts are preserved by the action.

∼ philosophical and technical aspects!

▷ Frame Axioms $F$:

\[
\begin{align*}
\text{holds}(\text{on}(Y, X), S) \land Y \neq Z & \rightarrow \text{holds}(\text{on}(Y, X), \text{do}(\text{move}(Z, Z'), S)) \\
\text{holds}(\text{clear}(Y), S) \land Y \neq Z' & \rightarrow \text{holds}(\text{clear}(Y), \text{do}(\text{move}(Z, Z'), S))
\end{align*}
\]

▷ $\models I \land \text{move}(X, Y) \land F \land \text{UNA} \rightarrow G$, where $\text{UNA}$ denotes the unique name axioms.

▷ $n \times m$ frame axioms, where $n$ is the number of actions and $m$ is the number of fluents!

∼ Can we do better?

Qualification, Ramification and Prediction Problem

▷ Qualification Problem: What preconditions must be realistically satisfied such that an action can be executed?

~~~ unstack(X) could also have precondition not_glued_on(X).

▷ Ramification Problem: What are the effects of an action?

~~~ If an object is slowly moved, then everything that is on this objects goes with it.

▷ Prediction Problem: How long does a fluent hold?

~~~ How long will an expensive bicycle be standing in front of the office building if it is parked there in the morning?

[12] Cognitive Robotics: Situation Calculus
Positive Effect Axioms

Positive effect axioms specify the emergence of fluents.

Two positive effect axioms for the fluent *broken*:

\[
\begin{align*}
holds(holding(R, X), S) \land Y = X \land fragile(Y) & \rightarrow holds(broken(Y), do(drop(R, X), S)) \\
holds(next_to(B, Y), S) \land bomb(B) & \rightarrow holds(broken(Y), do(explode(B), S))
\end{align*}
\]

Equivalent logical form:

\[
\begin{align*}
poss(A, S) \land [ & (\exists R, X) A = drop(R, X) \land Y = X \land fragile(Y) \\
& \lor (\exists B) A = explode(B) \land holds(next_to(B, Y), S)] \\
& \rightarrow holds(broken(Y), do(A, S))
\end{align*}
\]

\[
\begin{align*}
holds(holding(R, X), S) & \rightarrow poss(drop(R, X), S) \\
bomb(B) & \rightarrow poss(explode(B), S)
\end{align*}
\]

Completeness assumption: The positive effect axioms characterize all the conditions under which an action can lead to *Y* being broken.
Negative Effect Axioms

▷ Negative effect axioms specify the disappearance of fluents.

▷ A negative effect axiom for the fluent $\textit{broken}$:

\[
\text{holds}(\textit{has\_glue}(R), S) \land \text{holds}(\textit{broken}(X), S) \land Y = X \\
\rightarrow \neg\text{holds}(\textit{broken}(Y), \textit{do}(\textit{repair}(R, X), S))
\]

▷ Equivalent logical form:

\[
\text{poss}(A, S) \land (\exists R, X) A = \textit{repair}(R, X) \land Y = X \rightarrow \neg\text{holds}(\textit{broken}(Y), \textit{do}(A, S))
\]

\[
\text{holds}(\textit{has\_glue}(R), S) \land \text{holds}(\textit{broken}(X), S) \rightarrow \text{poss}(\textit{repair}(R, X), S)
\]

▷ Completeness assumption: The negative effect axioms characterize all the conditions under which an action can lead to $Y$ not being broken.
Explanation Closure Axioms

Explanation closure axioms explain the changes of fluents.

Explanation Closure Axiom 1:
\[
poss(A, S) \land \neg holds(broken(Y), S) \land holds(broken(Y), do(A, S)) \\
\rightarrow (\exists R, X) A = drop(R, X) \land X = Y \land fragile(Y) \\
\lor (\exists B) A = explode(B) \land holds(next_to(B, Y), S)
\]

Explanation Closure Axiom 2:
\[
poss(A, S) \land holds(broken(Y), S) \land \neg holds(broken(Y), do(A, S)) \\
\rightarrow (\exists R) A = repair(R, Y)
\]
Situation Calculus: The General Approach (1)

▷ General positive effect axioms for each fluent $f$:

$$\text{poss}(A, S) \land \gamma_{f}^{+}(A, S) \rightarrow \text{holds}(f, \text{do}(A, S))$$

▷ General negative effect axioms for fluent $f$:

$$\text{poss}(A, S) \land \gamma_{f}^{-}(A, S) \rightarrow \neg \text{holds}(f, \text{do}(A, S))$$

▷ Precondition axioms for each action $a$:

$$\pi_{a}(S) \rightarrow \text{poss}(a, S)$$

▷ Completeness Assumption: The general positive and negative effect axioms characterize all conditions under which some action $A$ can lead to $f$ becoming true and false respectively.

▷ This assumption is translated into the explanation closure axioms:

$$\text{poss}(A, S) \land \text{holds}(f, S) \land \neg \text{holds}(f, \text{do}(A, S)) \rightarrow \gamma_{f}^{-}(A, S')$$

$$\text{poss}(A, S) \land \neg \text{holds}(f, S') \land \text{holds}(f, \text{do}(A, S')) \rightarrow \gamma_{f}^{+}(A, S')$$

[16] Cognitive Robotics: Situation Calculus
Unique names axioms for actions:

\[
a(X_1, \ldots, X_n) \neq a'(Y_1, \ldots, Y_m), \text{ where } a \text{ and } a' \text{ are different action names}
\]

\[
a(X_1, \ldots, X_n) = a(Y_1, \ldots, Y_n) \rightarrow X_1 = Y_1 \land \ldots \land X_n = Y_n
\]

Unique names axioms for situations \( \mathcal{F}_{uns} \):

\[
s_0 \neq do(A, S)
\]

\[
do(A, S) = do(A', S') \rightarrow A = A' \land S = S'
\]
Successor State Axioms

- **Theorem:** Let $T$ be a first order theory that entails $\neg\exists (\text{poss}(A, S) \land \gamma_f^+(A, S) \land \gamma_f^-(A, S))$. Then $T$ entails that the general effect axioms together with the explanation closure axioms are logically equivalent to

$$\text{poss}(A, S) \rightarrow \left[ \text{holds}(f, \text{do}(A, S)) \iff \gamma_f^+(A, S) \lor \text{holds}(f, S) \land \neg \gamma_f^-(A, S) \right].$$

The latter formula is called successor state axiom for the fluent $f$.

- The successor state axiom for $\text{broken}$:

$$\text{poss}(A, S) \rightarrow \left[ \text{holds}(\text{broken}(Y), \text{do}(A, S)) \iff \right.$$

$$(\exists R, X) \ A = \text{drop}(R, Y) \land X = Y \land \text{fragile}(Y)$$

$$\lor (\exists B) \ A = \text{explode}(B) \land \text{holds}(\text{next_to}(B, Y), S)$$

$$\lor \text{holds}(\text{broken}(Y), S) \land \neg (\exists R) \ A = \text{repair}(R, Y) \left. \right].$$

- $n + m$ axioms, where $n$ is the number of actions and $m$ is the number of fluents!
Plan Synthesis

▷ Idea: Obtain a plan as answer substitution from the proof of the goal \((\exists S) g(S)\) wrt the axiomatization and the initial situation.

▷ Plans must be executable:

\[
\mathcal{F}_{ex} = \{ ex(S) \iff S = s_0 \lor (\exists A, S') S = do(A, S') \land poss(A, S') \land ex(S') \}
\]

\[
\models F \models (\exists S) g(S) \land ex(S), \text{ where } F \text{ is a suitable axiomatization of the world.}
\]
Simple Formulas

▷ A formula $\mathcal{F}$ is said to be simple if it satisfies the following conditions:

- $\mathcal{F}$ does not contain an occurrence of $\text{poss}$ or $\text{ex}$.
- There does not exist a $\text{holds}$–predicate in $\mathcal{F}$ which contains an occurrence of $\text{do}$.
- There is no quantification over situation variables in $\mathcal{F}$.
- There is at most one free variable $S$ representing a situation in $\mathcal{F}$.

▷ In the following we assume that $\pi_a(S)$, $\gamma^+_f(A, S)$ and $\gamma^-_f(A, S)$ are simple formulas.

▷ $\mathcal{F}_{ss}$: successor state axioms.

▷ $\mathcal{F}_{ap}$: action precondition axioms.
Action Precondition and Successor State Axioms

▷ Action Precondition Axioms:

$$(\forall X, S) \ [\pi_{a_1} \rightarrow poss(a_1(X, S))]$$
$$\vdots$$
$$(\forall Z, S) \ [\pi_{a_n} \rightarrow poss(a_n(Z, S))]$$

▷ Let $\mathcal{D}_{\Psi}(A, S)$ denote the formula

$$(\exists X) \ A = a_1(X) \land \pi_{a_1} \lor \ldots \lor (\exists Z) \ A = a_n(Z) \land \pi_{a_n}.$$  

▷ Successor State Axioms:

$$(\forall A, S, X) \ poss(A, X) \rightarrow [holds(f(X), do(A, S)) \leftrightarrow \Phi_f]$$

[21] Cognitive Robotics: Situation Calculus
A Regression Operator

▷ A regression operator \( R \):

1. Whenever \( W \) is an atom but not of the form \( \text{holds}(f, \text{do}(X, Y)) \) then

\[
R[W] = W.
\]

2. Whenever \( f \) is a fluent whose successor state axiom is of the form

\[
(\forall A, S, \overline{X}) \text{poss}(A, X) \to [\text{holds}(f(\overline{X}), \text{do}(A, S)) \leftrightarrow \Phi_f]
\]

then

\[
R[\text{holds}(f(\overline{t}), \text{do}(a, s))] = \Phi_f{\overline{X}/\overline{t}, A/a, X/s}. \]

3. Whenever \( W, W_1, W_2 \) are formulas then

\[
R[\neg W] = \neg R[W] \\
R[W_1 \land W_2] = R[W_1] \land R[W_2]
\]

and likewise for \( \exists, \lor, \to \) and \( \leftrightarrow \).

[22] Cognitive Robotics: Situation Calculus
Regression

Let $g(S)$ has $S$ as its only free variable

$$\Gamma_0(S) = g(S)$$
$$\Gamma_i(S) = (\exists a_i) \ R[\Gamma_{i-1}(do(a_i, S))] \land \mathcal{D}(a_i, S) \quad i = 1, 2, \ldots$$

A sentence is s-admissible iff it mentions no situation variable at all, or it is of the form $(\forall S) \ W(S)$, where $S$ is a situation variable and $W(S)$ is simple wrt $S$.

Theorem: Suppose

$$\mathcal{F} = \mathcal{F}_{ex} \cup \mathcal{F}_{ss} \cup \mathcal{F}_{ap} \cup \mathcal{F}_{uns} \cup \mathcal{F}_{\forall}$$

where $\mathcal{F}_{\forall}$ is the set of s-admissible sentences that is closed under regression wrt $\mathcal{F}$. Suppose $g(S)$ has $S$ as its only free variable. Then

$$\mathcal{F} \models (\exists S) \ g(S) \land ex(S)$$

iff for some $n$

$$\mathcal{F}_{uns} \cup \mathcal{F}_{\forall} \models \Gamma_0(s_0) \lor \ldots \lor \Gamma_n(s_0).$$

[23] Cognitive Robotics: Situation Calculus
Agent programming language.

Maintains an explicit representation of the world.

User provides axioms about
- the initial situation ($\mathcal{F}_{\forall s}$),
- the preconditions and effects of actions ($\mathcal{F}_{ap} \cup \mathcal{F}_{ss}$)
- and general properties ($\mathcal{F}_{ex} \cup \mathcal{F}_{uns}$).

Let $\mathcal{F} = \mathcal{F}_{ex} \cup \mathcal{F}_{ss} \cup \mathcal{F}_{ap} \cup \mathcal{F}_{uns} \cup \mathcal{F}_{\forall s}$.

Plan is given!

Reason about the situations of the world and consider the effects of various possible plans.
Complex Actions

- \( \text{do}(\delta, S, S') \) holds, whenever \( S' \) is a terminating situation of an execution of a complex action \( \delta \) starting in situation \( S \).

- **Primitive actions:**
  \[
  \text{do}(A, S, S') \overset{\text{def}}{=} \text{poss}(A, S) \land S' = \text{do}(A, S).
  \]

- **Test actions:**
  \[
  \text{do}(\Phi?, S, S') \overset{\text{def}}{=} \text{holds}(\Phi, S) \land S = S'.
  \]

- **Sequence:**
  \[
  \text{do}([\delta_1; \delta_2], S, S') \overset{\text{def}}{=} (\exists S^*) \ (\text{do}(\delta_1, S, S^*) \land \text{do}(\delta_2, S^*, S')).
  \]

- We will introduce more complex actions later in the show.

---

Correctness and Termination

▷ Correctness:
\[ F \models (\forall S) \ do(\delta, s_0, S) \rightarrow p(S) \]

or, even stronger
\[ F \models (\forall S_0, S) \ do(\delta, S_0, S) \rightarrow p(S). \]

▷ Termination:
\[ F \models (\exists S) \ do(\delta, s_0, S) \]

or, even stronger
\[ F \models (\forall S_0) (\exists S) \ do(\delta, S_0, S). \]
An Example: The Tile Crawler

▷ Primitive action: \textit{crawl}(N): crawls to tile \(N\).

▷ Fluents:
  
  – \textit{on}(N): the crawler is on tile \(N\),
  
  – \textit{clean}(N): tile \(N\) is clean.

▷ Action Precondition Axioms:

\[
\text{poss}(\text{crawl}(N), S) \leftrightarrow \text{holds}(\text{on}(N - 1), S).
\]

▷ Successor State Axiom:

\[
\text{poss}(A, S) \rightarrow [\text{holds}(\text{on}(N + 1), \text{do}(A, S)) \leftrightarrow [A = \text{crawl}(N + 1) \land \text{holds}(\text{on}(N), S)] \lor [A \neq \text{crawl}(N) \land \text{holds}(\text{on}(N + 1), S)].
\]

\[
\text{poss}(A, S) \rightarrow [\text{holds}(\text{clean}(N), \text{do}(A, S)) \leftrightarrow \text{holds}(\text{clean}(N), S)].
\]

▷ \(\text{do}([\text{crawl}(1); \text{crawl}(2); \text{clean}(2)?; \text{crawl}(3)], S, S')\)
A GOLOG Interpreter in PROLOG

do(E,S,do(E,S)) :- primitive_action(E), poss(E,S).
do(?(P),S,S) :- holds(P,S).
do([],S,S).
do([E|L],S,S1) :- do(E,S,S2), do(L,S2,S1).

holds(and(P1,P2),S) :- holds(P1,S), holds(P2,S).
holds(or(P1,P2),S) :- holds(P1,S); holds(P2,S).
holds(neg(P),S) :- not holds(P,S).
holds(some(V,P),S) :- sub(V,_,P,P1), holds(P1,S).

/* sub(Name,New,Term1,Term2): Term2 is Term1 with Name replaced by New. */
sub(X1,X2,T1,T2) :- var(T1), T2=T1.
sub(X1,X2,T1,T2) :- not var(T1), T1 = X1, T2 = X2.
sub(X1,X2,T1,T2) :- not T1 = X1, T1=..[F|L1], sublist(X1,X2,L1,L2), T2 =..[F|L2].
sublist(X1,X2,[],[]) 
sublist(X1,X2,[T1|L1],[T2|L2]) :- sub(X1,X2,T1,T2), sublist(X1,X2,L1,L2).
The Tile Crawler in PROLOG

/* primitive actions */
primitive_action(crawl(N)).

/* preconditions for primitive actions */
poss(crawl_s(N),S) :- holds(on(N),S).

/* successor state axioms */
holds(on(s(N)),do(E,S)) :- E = crawl(N), holds(on(N),S);
not E = crawl(N), holds(on(s(N),S)).
holds(clean(N),do(E,S)) :- holds(clean(N),S).

/* initial situation */
holds(on(0),s0).
holds(clean(0),s0).
holds(clean(s(0)),s0).
holds(clean(s(s(0))),s0).
holds(clean(s(s(s(0)))),s0).
Comments

▷ Closed world assumption.

〜 Initial situation must be completely specified.

▷ Precise correspondence between \( do(\delta, S, S') \) and \( do(E, S, S') \) depends on a number of factors.

▷ Plans cannot be computed nor synthesized.

▷ Successor state axioms solve the frame problem in a representationally adequate way, but \( n \) applications of these axioms are needed to compute the successor states, if states are characterized by \( n \) fluents.

〜 Inferential frame problem remains!
Literature


Fluent Calculus

- An Example: A Murder Mystery
- States in the Situation Calculus vs. States in the Fluent Calculus
- The Calculus
- State Update Axioms
- Another Example: Yale Shooting Problem
- Literature
A Murder Mystery

“A reliable witness reported that the murderer poured some milk into a cup of tea before offering it to his aunt. The old lady took a drink or two and then she suddenly fell into the armchair and died an instant later, by poioning as has been diagnosed afterwards. According to the witness, the nephew had no opportunity to poison the tea beforehand. This proves that it was the milk which was poisoned and by which the victim was murdered.”

▷ Fluents:
  - \textit{poisoned}(X): X is poisoned,
  - \textit{alive}(X): X is alive.

▷ Actions:
  - \textit{mix}(P, X, Y): agent P mixes X into Y,
  - \textit{drink}(P, X): agent P drinks X.
Formalizing the Murder Mystery

▶ Action precondition axioms: exactly as in the situation calculus

\[
\text{alive}(P) \rightarrow \text{poss}(\text{mix}(P, X, Y), S)
\]
\[
\text{alive}(P) \rightarrow \text{poss}(\text{drink}(P, X), S)
\]

▶ Effect axioms for each action:

\[
\text{poss}(\text{mix}(P, X, Y), S) \land \text{holds}(\text{poisoned}(X), S) \\
\rightarrow \text{holds}(\text{poisoned}(Y), \text{do}(\text{mix}(P, X, Y), S))
\]
\[
\text{poss}(\text{drink}(P, X), S) \land \text{holds}(\text{alive}(P), S) \land \text{holds}(\text{poisoned}(X), S) \\
\rightarrow \neg \text{holds}(\text{alive}(P), \text{do}(\text{drink}(P, X), S))
\]

▶ Initial situation in the situation calculus:

\[
\neg \text{holds}(\text{poisoned(tea)}, s_0) \land \text{holds}(\text{alive(nephew)}, s_0) \land \text{holds}(\text{alive(aunt)}, s_0)
\]
States in the Situation Calculus

A state in the situation calculus is the union of all relevant fluents, the do or do not hold in a situation.

- The holds-predicate and its negation are used to represent that a fluent holds or does not hold in a situation.
- The union is represented with the help of $\land$ and $\top$;
  $\land$ is associative, commutative, idempotent and $\top$ is its unit element.
- $\leadsto$ Each fact is stated only once ($holds(car, s_0) \land holds(car, s_0) \leftrightarrow holds(car, s_0)$).
- Situational fluents are reified ($holds(F, S)$).
- Initial situations completely specify the initial state, eg.: 
  \[ holds(alive(aunt), s_0) \land holds(alive(nephew), s_0) \land \neg holds(poisoned(tea), s_0). \]
- Negative facts are explicitly stated.

[35] Cognitive Robotics: Fluent Calculus
States in the Fluent Calculus

A state in the fluent calculus is the multiset union of all relevant fluents that hold in a situation.

▷ Fluents are represented by fluent terms.

▷ The multiset union is represented with the help of \( \circ \) and \( \emptyset \);
  \( \circ \) is associative, commutative and \( \emptyset \) is its unit element.

\( \sim \) Fluents may be stated more than once (\( car \circ car \neq car \)).

▷ Multisets of fluents are reified.

▷ Initial situations may be partially specified, eg.:

\[
(\exists Z) \ [state(s_0) = alive(nephew) \circ alive(aunt) \circ Z \land (\forall Z') \ Z \neq poisoned(tea) \circ Z'].
\]

▷ Negative facts are either explicitly stated or are derived using completion.
The Language of the Fluent Calculus

▷ Order sorted language with
  - sorts ACTION, SITUATION, FLUENT, STATE and OBJECT and
  - ordering constraint FLUENT < STATE ( (∀X) (FLUENT(X) → STATE(X)) )

▷ Function symbols $Σ_F = Σ_A \cup Σ_{Sit} \cup Σ_{Fl} \cup Σ_{St} \cup Σ_O$, where
  - $Σ_A$ is a set of function symbols denoting action names,
  - $Σ_{Sit} = \{s_0, do\}$,
  - $Σ_{Fl}$ is a set of function symbols denoting fluent names,
  - $Σ_{St} = \{\emptyset, o, state\}$ and
  - $Σ_O$ is a set of function symbols denoting objects.

▷ Variables $Σ_V = Σ_{V,A} \cup Σ_{V,Sit} \cup Σ_{V,Fl} \cup Σ_{V,St} \cup Σ_{V,O}$ (for each sort)

▷ All sets are mutually disjoint.
Function Symbols

▷ Special function symbols:

\[ s_0 : \rightarrow \text{SITUATION} \]
\[ do : \text{ACTION} \times \text{SITUATION} \rightarrow \text{SITUATION} \]
\[ \emptyset : \rightarrow \text{STATE} \]
\[ \circ : \text{STATE} \times \text{STATE} \rightarrow \text{STATE} \]
\[ state : \text{SITUATION} \rightarrow \text{STATE} \]

▷ Remaining function symbols:

\[ \text{nephew} : \rightarrow \text{OBJECT} \]
\[ \text{aunt} : \rightarrow \text{OBJECT} \]
\[ \text{tea} : \rightarrow \text{OBJECT} \]
\[ \text{milk} : \rightarrow \text{OBJECT} \]
\[ \text{mix} : \text{OBJECT} \times \text{OBJECT} \times \text{OBJECT} \rightarrow \text{ACTION} \]
\[ \text{drink} : \text{OBJECT} \times \text{OBJECT} \rightarrow \text{ACTION} \]
\[ \text{alive} : \text{OBJECT} \rightarrow \text{FLUENT} \]
\[ \text{poisoned} : \text{OBJECT} \rightarrow \text{FLUENT} \]
Action, Situation, Fluent, State and Object Terms

- Object terms: tea, nephew.
- Action terms: mix(nephew, milk, tea), drink(X, Y).
- Situation terms: s₀, do(mix(nephew, milk, tea), s₀).
- Fluent terms: poisoned(milk), alive(X).
- State terms: ∅, Z₁ ◦ Z₂, state(s₀), alive(X).
  - A state term is said to be simple if it contains at most one occurrence of a variable of sort state.
    \[ \sim alive(nephew) ◦ alive(aunt) ◦ Z, state(do(A, S)) ◦ alive(X) \]
  - A state term is said to be a constructor state term if it is built from fluent terms, the constant ∅, the function symbol ◦ and variables of sort STATE only.
    \[ \sim alive(nephew) ◦ alive(aunt) ◦ Z, alive(nephew) ◦ alive(aunt) \]
Formalizing States in the Fluent Calculus

▷ Some properties of \( \circ : \mathcal{F}_{AC1} \)

- **A**ssociative: \( \forall Z_1, Z_2, Z_3 : \text{STATE} \) \( (Z_1 \circ Z_2) \circ Z_3 = Z_1 \circ (Z_2 \circ Z_3) \),

- **C**ommutative: \( \forall Z_1, Z_2 : \text{STATE} \) \( Z_1 \circ Z_2 = Z_2 \circ Z_1 \),

- **1** unit element: \( \forall Z : \text{STATE} \) \( Z \circ \emptyset = Z \).

▷ \( \circ \) is not idempotent!

▷ If each fact shall be stated only once in a state, then add

\[ \forall S : \text{SITUATION}, F : \text{FLUENT}, Z : \text{STATE} \ [\text{state}(S) \neq F \circ F \circ Z]. \]

▷ \( \text{holds}(F, S) \overset{\text{def}}{=} (\exists Z : \text{STATE} \ [\text{state}(S) = F \circ Z]. \)
Extended Unique Names Assumptions

$\triangleright$ Extended unique names assumptions $\mathcal{F}_{euna} = \mathcal{F}_E \cup \mathcal{F}_{AC1} \cup \mathcal{F}_{uc}$, where

- $\mathcal{F}_E$ are the axioms of equality,
- $\mathcal{F}_{AC1}$ are the AC1–axioms for $\circ$ and $\emptyset$ and
- $\mathcal{F}_{uc}$ specifies unification completeness: for any terms $t_1$ and $t_2$ with variables $\overline{X}$, which are either not of sort STATE or are constructor state terms.

1. if $t_1$ and $t_2$ are not AC1–unifiable, then

    $\neg(\exists \overline{X}) \ t_1 = t_2,$

2. if $t_1$ and $t_2$ are AC1–unifiable with the complete set of unifiers $cU_{AC1}(t_1, t_2)$, then

    $\forall \overline{X} \ [t_1 = t_2 \rightarrow \bigvee_{\theta \in cU_{AC1}(t_1, t_2)} (\exists \overline{Y}) \ \bigwedge_{X \neq X\theta} X = X\theta],$

    where $\overline{Y}$ denotes the variables occurring in $\bigwedge_{X \neq X\theta} X = X\theta$ and not in $\overline{X}$.

$\leadsto \mathcal{F}_{uc}$ implies $\mathcal{F}_{una} \cup \mathcal{F}_{uns}.$

[41] Cognitive Robotics: Fluent Calculus
State Update Axioms

\[ \Delta(S) \rightarrow \Gamma[\textit{state}(do(a, S)), \textit{state}(S)] \]

Examples:

\[ \text{poss}(\textit{do(mix}(P, X, Y), S) \land \textit{holds}(\textit{poisoned}(X), S) \land \neg \textit{holds}(\textit{poisoned}(Y), S) \rightarrow \textit{state}(\textit{do(mix}(P, X, Y), S)) = \textit{state}(S) \circ \textit{poisoned}(Y) \]  

\[ \text{poss}(\textit{do(mix}(P, X, Y), S) \land \neg \textit{holds}(\textit{poisoned}(X), S) \lor \textit{holds}(\textit{poisoned}(Y), S) \rightarrow \textit{state}(\textit{do(mix}(P, X, Y), S)) = \textit{state}(X) \]  

\[ \text{poss}(\textit{do(drink}(P, X), S)) \land \textit{holds}(\textit{alive}(P), S) \land \textit{holds}(\textit{poisoned}(X), S) \rightarrow \textit{state}(\textit{do(drink}(P, X), S)) \circ \textit{alive}(P) = \textit{state}(S) \]  

\[ \text{poss}(\textit{do(drink}(P, X), S)) \land \neg \textit{holds}(\textit{alive}(P), S) \lor \neg \textit{holds}(\textit{poisoned}(X), S) \rightarrow \textit{state}(\textit{do(drink}(P, X), S)) = \textit{state}(S) \]
Fluent Calculus: The General Approach (1)

▷ Positive effect axioms:
\[
\text{poss}(a(\overline{X}), S) \land \epsilon^+_{a,f}(\overline{X}, S) \rightarrow \text{holds}(f(\overline{Y}), \text{do}(a(\overline{X}), S)),
\]
where each variable occurring in \(\overline{Y}\) occurs also in \(\overline{X}\).

▷ Negative effect axioms:
\[
\text{poss}(a(\overline{X}), S') \land \epsilon^-_{a,f}(\overline{X}, S') \rightarrow \neg\text{holds}(f(\overline{Y}), \text{do}(a(\overline{X}), S')).
\]
where each variable occurring in \(\overline{Y}\) occurs also in \(\overline{X}\).

▷ Precondition axioms for each action \(a\):
\[
\pi_a(\overline{X}, S) \rightarrow \text{poss}(a(\overline{X}, S)).
\]

▷ Extended Unique Names Assumptions
Fluent Calculus: The General Approach (2)

▷ **Completeness assumption:** A given set of effect axioms is complete in the sense that all relevant effects of all involved actions are specified.

▷ **Consistency assumption:** For all \( a \) and \( f \) we find

\[
-(\exists X, S) \ [\text{poss}(a(X), S) \land \epsilon_{a,f}^+(X, S) \land \epsilon_{a,f}^-(X, S)].
\]

▷ **Theorem:** The consistency and completeness assumptions allow to compile the effect axioms into successor state axioms of the form

\[
\Delta(S) \rightarrow \text{state}(\text{do}(a, S)) \circ \vartheta^- = \text{state}(S) \circ \vartheta^+,
\]

where \( \vartheta^- \) and \( \vartheta^+ \) are the negative and positive effects of action \( a \) under condition \( \Delta(S) \) respectively.
Another Example: Yale Shooting Problem (1)

▷ Fluents:
  - $\text{loaded}(X)$: gun $X$ is loaded
  - $\text{dead}(Y)$: individual $Y$ is dead

▷ Actions:
  - $\text{shoot}(X, Y)$: gun $X$ is aimed at $Y$ and the trigger is pulled.

▷ For simplicity, actions are always possible.

▷ Effect axioms:

\[
\begin{align*}
\text{loaded}(X, S) & \rightarrow \text{holds}(\text{dead}(Y), \text{do}(\text{shoot}(X, Y), S)) \\
T & \rightarrow \neg\text{holds}(\text{loaded}(X), \text{do}(\text{shoot}(X, Y), S))
\end{align*}
\]
Yale Shooting Problem (2)

▷ State update axioms:

\[ \neg \text{holds}(\text{loaded}(X), S) \]
\[ \rightarrow \text{state}(\text{do}(\text{shoot}(X, Y), S)) = \text{state}(S) \]

\[ \text{holds}(\text{dead}(Y), S) \land \text{holds}(\text{loaded}(X), S) \]
\[ \rightarrow \text{state}(\text{do}(\text{shoot}(X, Y), S)) \circ \text{loaded}(X) = \text{state}(S) \]

\[ \text{holds}(\text{loaded}(X), S) \land \neg \text{holds}(\text{dead}(Y), S) \]
\[ \rightarrow \text{state}(\text{do}(\text{shoot}(X, Y), S)) \circ \text{loaded}(X) = \text{state}(S) \circ \text{dead}(Y) \]
Comments

▷ State update axioms involve only simple state terms.
▷ State axioms should be designed such that they do not violate the axiom

\[(\forall S : \textit{situation}, X : \textit{fluent}, Z : \textit{state}) \ [\text{state}(S) = X \circ X \circ Z \rightarrow X = \emptyset].\]

It is still crucial, however, in cases of incompletely specified situations.
▷ AC1-unification is decidable; complete unification algorithms exist.
Literature


Event Calculus

- Branching vs. linear time structure
- Axioms of the event calculus
- Circumscription
- Event Calculus and the fork lift truck in PROLOG
Branching Time Structure

\[ \text{do}(\text{mix}(\text{nephew, milk, tea}), s_0) \]

\[ \rightarrow \]

\[ \text{do}(\text{drink(aunt, tea)}, \text{do}(\text{mix(nephew, milk, tea)}, s_0)) \]

\[ \downarrow \]

\[ \text{do}(\text{drink(aunt, milk)}, \text{do}(\text{mix(nephew, milk, tea)}, s_0)) \]

\[ \rightarrow \]

\[ \text{do}(\text{drink(aunt, milk)}, s_0) \]
Reasoning About Counterfactual Action Sequences

The observation

\[\text{holds}(\text{alive}(\text{aunt}), s_0) \land \text{holds}(\text{alive}(\text{nephew}), s_0) \land \neg \text{holds}(\text{poisoned}(\text{tea}), s_0) \land \neg \text{holds}(\text{alive}(\text{aunt}), \text{do}(\text{drink}(\text{aunt}, \text{tea}), \text{do}(\text{mix}(\text{nephew}, \text{milk}, \text{tea}), s_0)))\]

entails this statement about a hypothetical course of events:

\[\neg \text{holds}(\text{alive}(\text{aunt}), \text{do}(\text{drink}(\text{aunt}, \text{milk}), s_0))\]
Linear Time Structure

\[ \text{happens}(E, T) \iff \text{event } E \text{ happens at time } T \]

\[ \text{happens}(\text{mix(nephew, milk, tea)}, 1.5) \]

\[ t = 0 \quad \text{happens}(\text{drink(aunt, tea)}, t) \land t \geq 3.9 \land t \leq 4.1 \]
A Task for the Fork Lift Truck

The goal is to have container $a$ on $b$ at $port1$, container $c$ at $port2$, and container $d$ at $track2$.
Fluents, Events, Time Points

Fluents: \( \text{on}(X, Y) \)
- \( X \) container
- \( Y \) container or location

\( \text{clear}(X) \)
- \( X \) container or location

Events: \( \text{move}(X, Y) \)
- \( X \) container
- \( Y \) container or location

Time points: positive real numbers (incl. 0)
Effect Axioms

\(\text{holds\_at}(F, T)\) \: \Leftrightarrow \: \text{fluent } F \text{ holds at time } T

\(\text{initiates}(E, F, T)\) \: \Leftrightarrow \: \text{event } E \text{ makes fluent } F \text{ true at time } T

\(\text{terminates}(E, F, T)\) \: \Leftrightarrow \: \text{event } E \text{ makes fluent } F \text{ false at time } T

\(\text{initiates}(\text{move}(X, Y), \text{on}(X, Y), T) \leftarrow \)
\(\ \ \ \ \text{holds\_at}(\text{clear}(X), T) \land \text{holds\_at}(\text{clear}(Y), T) \land X \neq Y\)

\(\text{initiates}(\text{move}(X, Y), \text{clear}(Z), T) \leftarrow \)
\(\ \ \ \ \text{holds\_at}(\text{clear}(X), T) \land \text{holds\_at}(\text{clear}(Y), T) \land \text{holds\_at}(\text{on}(X, Z), T) \land X \neq Y \land Y \neq Z\)

\(\text{terminates}(\text{move}(X, Y), \text{on}(X, Z), T) \leftarrow \)
\(\ \ \ \ \text{holds\_at}(\text{clear}(X), T) \land \text{holds\_at}(\text{clear}(Y), T) \land \text{holds\_at}(\text{on}(X, Z), T) \land X \neq Y \land Y \neq Z\)

\(\text{terminates}(\text{move}(X, Y), \text{clear}(Y), T) \leftarrow \)
\(\ \ \ \ \text{holds\_at}(\text{clear}(X), T) \land \text{holds\_at}(\text{clear}(Y), T) \land X \neq Y\)

[55] Cognitive Robotics: Event Calculus
Initial Situation, Course of Events

\[\text{initially}(F) \iff \text{fluent } F \text{ is initiated at time } 0\]
\[\text{n\_initially}(F) \iff \text{fluent } F \text{ is terminated at time } 0\]

\[
\begin{align*}
\text{initially(on}(a, \text{track}1)) \land \text{initially(clear}(a)) \land \\
\text{initially(on}(b, \text{track}2)) \land \text{initially(on}(c, b) \land \text{initially(clear}(c)) \\
\text{initially(clear(temp\_store))} \land \text{initially(clear(port1))} \land \\
\text{initially(on}(d, \text{port}2)) \land \text{initially(clear}(d)) \\
[\text{initially(on}(X, Y)) \rightarrow \text{n\_initially(clear}(Y)))] \land \\
[\text{initially(clear}(X)) \rightarrow \text{n\_initially(on}(Y, X)))] \\
[\text{initially(on}(X, Y)) \land Y \neq Z \rightarrow \text{n\_initially(on}(X, Z))] \land \\
[\text{initially(on}(X, Z)) \land X \neq Y \rightarrow \text{n\_initially(on}(Y, Z))] \land
\end{align*}
\]

\[
\begin{align*}
\text{happens(move}(c, \text{temp\_store}), 3) \land \text{happens(move}(b, \text{port}1), 5) \land \\
\text{happens(move}(a, b), 8) \land \text{happens(move}(d, \text{track}2), 11) \land \text{happens(move}(c, \text{port}2), 13)
\end{align*}
\]
Foundational Axioms of the Event Calculus

\[
\text{clipped}(T_1, F, T_2) \iff \text{fluent } F \text{ becomes false between time } T_1 \text{ and time } T_2
\]

\[
\text{declipped}(T_1, F, T_2) \iff \text{fluent } F \text{ becomes true between time } T_1 \text{ and time } T_2
\]

\[
\text{holds\textunderscore at}(F, T) \leftarrow \text{initially}(F) \land \neg\text{clipped}(0, F, T)
\]

\[
\text{holds\textunderscore at}(F, T_2) \leftarrow (\exists E) \; \text{happens}(E, T_1) \land \text{initiates}(E, F, T_1) \land T_1 < T_2 \land \neg\text{clipped}(T_1, F, T_2)
\]

\[
\neg\text{holds\textunderscore at}(F, T) \leftarrow \text{n\textunderscore initially}(F) \land \neg\text{declipped}(0, F, T)
\]

\[
\neg\text{holds\textunderscore at}(F, T_2) \leftarrow (\exists E) \; \text{happens}(E, T_1) \land \text{terminates}(E, F, T_1) \land T_1 < T_2 \land \neg\text{declipped}(T_1, F, T_2)
\]

\[
\text{clipped}(T_1, F, T_2) \iff (\exists E, T) \; \text{happens}(E, T) \land \text{terminates}(E, F, T) \land T_1 < T \land T < T_2
\]

\[
\text{declipped}(T_1, F, T_2) \iff (\exists E, T) \; \text{happens}(E, T) \land \text{initiates}(E, F, T) \land T_1 < T \land T < T_2
\]
Summary: Event Calculus Signatures

- **Sorts**: FLUENT, EVENT, TIMEPOINT

- **Predicates**:
  - `happens`: EVENT × TIMEPOINT
  - `holds_at`: FLUENT × TIMEPOINT
  - `initially`: FLUENT
  - `n_initially`: FLUENT
  - `initiates`: EVENT × FLUENT × TIMEPOINT
  - `terminates`: EVENT × FLUENT × TIMEPOINT
  - `clipped`: TIMEPOINT × FLUENT × TIMEPOINT
  - `declipped`: TIMEPOINT × FLUENT × TIMEPOINT
  - `≤`: TIMEPOINT × TIMEPOINT
So far the axiomatization does not entail many useful conclusions.

For example, the first event is \( \text{happens}(\text{move}(c, \text{temp}.\text{store}), 3) \).

It would be useful to prove that \( \text{holds}(@\text{clear}(c), 3) \).

But from

\[
\text{holds}(@\text{clear}(c), 3) \leftarrow \text{initially}(@\text{clear}(c)) \land \neg \text{clipped}(0, \text{clear}(c), 3)
\]

\[
\text{clipped}(0, \text{clear}(c), 3) \leftrightarrow (\exists E, T) \text{happens}(E, T) \land \text{terminates}(E, \text{clear}(c), T) \land 0 < T < 3
\]

this does not follow.
Circumscription

“Circumscription allows us to conjecture that no relevant objects exist in certain categories except those whose existence follows from the statement of the problem.” [McCarthy, 1980]

Let $A$ be a sentence of first order formulas containing a predicate symbol $p(\overline{X})$.

$A(p/\Phi) \iff$ replace in $A$ all occurrences of $p$ by $\Phi$

$\text{CIRC}[A; p] \iff A \land (\forall \Phi) \{ A(p/\Phi) \land [(\forall \overline{X}) \Phi(\overline{X}) \rightarrow p(\overline{X})] \rightarrow [(\forall \overline{X}) p(\overline{X}) \rightarrow \Phi(\overline{X})] \}$
Example

Let $N$ be,

$$\text{happens}(\text{move}(c, \text{temp}\_\text{store}), 3) \land \text{happens}(\text{move}(b, \text{port1}), 5)$$
$$\land \text{happens}(\text{move}(a, b), 8) \land \text{happens}(\text{move}(d, \text{track2}), 11)$$
$$\land \text{happens}(\text{move}(c, \text{port2}), 13)$$

Then $\text{CIRC}[N; \text{happens}]$ entails,

$$\text{happens}(E, T)$$
$$\iff$$
$$E = \text{move}(c, \text{temp}\_\text{store}) \land T = 3 \lor$$
$$E = \text{move}(b, \text{port1}) \land T = 5 \lor$$
$$E = \text{move}(a, b) \land T = 8 \lor$$
$$E = \text{move}(d, \text{track2}) \land T = 11 \lor$$
$$E = \text{move}(c, \text{port2}) \land T = 13$$
Unique Names Assumptions

\[ \text{UNA}[c_1, \ldots, c_n] \iff \bigwedge_{i=1\ldots n; j=1\ldots n; i \neq j} c_i \neq c_j \]

UNA[move]

UNA[clear, on]

UNA[a, b, c, d, track1, track2, port1, port2, temp_store]
Joint Circumscription

Let $A$ be a sentence of first order formulas containing predicate symbols $p(\overline{X})$ and $q(\overline{Y})$.

$$\text{CIRC}[A; p, q]$$

$$: \iff$$

$$A \land \{(\forall \Phi, \Psi) \{ A(p/\Phi)(q/\Psi) \land [(\forall X) \Phi(\overline{X}) \rightarrow p(\overline{X})] \land [(\forall Y) \Psi(\overline{Y}) \rightarrow q(\overline{Y})] \rightarrow [(\forall X) p(\overline{X}) \rightarrow \Phi(\overline{X})] \land [(\forall Y) q(\overline{Y}) \rightarrow \Psi(\overline{Y})] \} \}$$

[63] Cognitive Robotics: Event Calculus
Event Calculus: The General Approach

Given are

- conjunction of *initiates* and *terminates* formulas $E$
- conjunction of *initially* formulas $I$
- conjunction of *happens* formulas $N$
- unique names assumptions $U$
- foundational axioms $EC$

The intended meaning is given by the formula

$$\text{CIRC}[N \land I; \textit{happens}] \land \text{CIRC}[E; \textit{initiates, terminates}] \land U \land EC$$
The Event Calculus in PROLOG

\[
\begin{align*}
\text{initiates}(\text{move}(X,Y), \text{on}(X,Y), T) & : - \\
& \text{holds}_\text{at}(\text{clear}(X), T), \text{holds}_\text{at}(\text{clear}(Y), T), \text{not } X=Y. \\

\text{initiates}(\text{move}(X,Y), \text{clear}(Z), T) & : - \\
& \text{holds}_\text{at}(\text{clear}(X), T), \text{holds}_\text{at}(\text{clear}(Y), T), \\
& \text{holds}_\text{at}(\text{on}(X,Z), T), \text{not } X=Y, \text{not } Y=Z. \\

\text{terminates}(\text{move}(X,Y), \text{on}(X,Z), T) & : - \\
& \text{holds}_\text{at}(\text{clear}(X), T), \text{holds}_\text{at}(\text{clear}(Y), T), \\
& \text{holds}_\text{at}(\text{on}(X,Z), T), \text{not } X=Y, \text{not } Y=Z. \\

\text{terminates}(\text{move}(X,Y), \text{clear}(Y), T) & : - \\
& \text{holds}_\text{at}(\text{clear}(X), T), \text{holds}_\text{at}(\text{clear}(Y), T), \text{not } X=Y. \\

\text{holds}_\text{at}(F, T) & : - \text{initially}(F), \text{not clipped}(0, F, T). \\

\text{holds}_\text{at}(F, T2) & : - \text{happens}(E, T1), T1<T2, \text{initiates}(E, F, T1), \\
& \text{not clipped}(T1, F, T2). \\

\text{clipped}(T1, F, T2) & : - \text{happens}(E, T), T1<T, T<T2, \text{terminates}(E, F, T).
\end{align*}
\]

Cognitive Robotics: Event Calculus
The Fork Lift Truck Scenario in PROLOG

initially(on(a,track1)).
initially(on(b,track2)).
initially(clear(c)).
initially(clear(port1)).
initially(clear(d)).

happens(move(c,temp_store),3).
happens(move(a,b),8).
happens(move(c,port2),13).

| ?- holds_at(F,15).

F = clear(a)
F = clear(d)
F = on(a,b)
F = on(d,track2)
F = clear(temp_store)

F = clear(c)
F = on(b,port1)
F = clear(track1)
F = on(c,port2)
Comments

▷ Circumscription is implemented via negation-as-failure.

▷ Closed world assumption
  ▷ Initial situation must be completely specified.

▷ Synthesizing plans requires abduction.

▷ The foundational axioms on persistence solve the frame problem in a representationally adequate way, but $n$ applications of these axioms are needed to compute what holds after an event, if states are characterized by $n$ fluents.

  ▷ Same inferential frame problem as with the Situation Calculus!

[67] Cognitive Robotics: Event Calculus
Planning by Abduction

Given are

- conjunction of initiates and terminates formulas $E$
- conjunction of initially formulas $I$
- conjunction of holds-at formulas $G$
- unique names assumptions $U$
- foundational axioms $EC$

A plan is a conjunction of happens formulas $N$ such that

- $\text{CIRC}[N \land I; \text{happens}] \land \text{CIRC}[E; \text{initiates, terminates}] \land U \land EC$ is consistent
- $\text{CIRC}[N \land I; \text{happens}] \land \text{CIRC}[E; \text{initiates, terminates}] \land U \land EC \models G$

[68] Cognitive Robotics: Event Calculus
Literature


Nondeterministic Actions

- What happens when calling an elevator
- Modeling nondeterministic actions in the Situation Calculus
- Modeling nondeterministic actions in the Fluent Calculus
- Modeling nondeterministic actions in the Event Calculus

Cognitive Robotics: Nondeterministic Actions
Two Elevators

\[\text{holds}(\text{at\_floor}(X, Y), S) \iff \text{elevator } X \text{ is at floor } Y \text{ in situation } S\]

\[\text{holds}(\text{at\_floor}(X, Y), S) \rightarrow X = \text{e\_left} \lor X = \text{e\_right}\]

\[\text{holds}(\text{at\_floor}(X, Y), S) \land \text{holds}(\text{at\_floor}(X, Z), S') \rightarrow Y = Z\]

\[\text{poss}(\text{call\_floor}(Y), S) \iff \neg(\exists X) \text{ holds}(\text{at\_floor}(X, Y), S)\]

\[\left(\exists Y, Z\right) \text{ holds}(\text{at\_floor}(\text{e\_left}, Y), s_0) \land \text{holds}(\text{at\_floor}(\text{e\_right}, Z), s_0) \land Y \neq 1 \land Z \neq 1\]

\[\text{poss}(\text{call\_floor}(Y), S) \rightarrow\]

\[\text{caused}(\text{at\_floor}(\text{e\_left}, Y), \text{true}, \text{do}(\text{call\_floor}(Y), S)) \oplus\]

\[\text{caused}(\text{at\_floor}(\text{e\_right}, Y), \text{true}, \text{do}(\text{call\_floor}(Y), S))\]

(\text{where } F \oplus G \doteq F \land \neg G \lor \neg F \land G)
Case Distinction and Causal Rules

> Idea: In the situation calculus, an auxiliary predicate $case$ is introduced to distinguish the possible outcomes of a nondeterministic action—based on a systematic ordering.

$$case(N, A, S) :\Leftrightarrow \text{in situation } S, \text{ performing the nondeterministic action } A$$

$$\text{results in outcome no. } N$$

$$poss(call\_floor(Y), S) \land case(1, call\_floor(Y), S) \rightarrow caused(at\_floor(e\_left, Y), true, do(call\_floor(Y), S))$$

$$poss(call\_floor(Y), S) \land case(2, call\_floor(Y), S) \rightarrow caused(at\_floor(e\_right, Y), true, do(call\_floor(Y), S))$$

In addition, we need this causal rule to solve the Ramification Problem:

$$caused(at\_floor(X, Y), true, S) \land Y \neq Z \rightarrow caused(at\_floor(X, Z), false, S)$$

[72] Cognitive Robotics: Nondeterministic Actions
On the Predicate ‘case’

▷ There is no successor state axiom for case, because its truth value may arbitrarily vary from one situation to the other.

▷ Let \( n_a \geq 2 \) be the total number of cases for nondeterministic action \( a \), then

\[
\text{case}(1, a(X), S') \oplus \ldots \oplus \text{case}(n, a(X), S')
\]
The Resulting Successor State Axiom

Let $\Psi(X, Y, A, S)$ be an abbreviation of the formula

\[
\begin{align*}
[A = \text{call\_floor}(Y) \land X = \text{e\_left} \land \text{case}(1, A, S)] \lor \\
[A = \text{call\_floor}(Y) \land X = \text{e\_right} \land \text{case}(2, A, S)]
\end{align*}
\]

\[
\begin{align*}
\text{poss}(A, S) \rightarrow \\
\left[ \text{holds}(\text{at\_floor}(X, Y), \text{do}(A, S)) \leftrightarrow \\
\Psi(X, Y, A, S) \lor \\
\text{holds}(\text{at\_floor}(X, Y), S) \land \neg(\exists Z)[\Psi(X, Z, A, S) \land Y \neq Z] \right]
\end{align*}
\]
Disjunctive State Update Axioms

▷ Idea: In the Fluent Calculus, nondeterministic actions are modeled by disjunctions in the consequent of state update axioms.

\[
\begin{align*}
\text{holds}(\text{at\_floor}(X, Y), S) & \rightarrow X = \text{e\_left} \vee X = \text{e\_right} \\
\text{holds}(\text{at\_floor}(X, Y), S) \land \text{holds}(\text{at\_floor}(X, Z), S) & \rightarrow Y = Z \\
\text{poss}(\text{call\_floor}(Y), S) & \leftrightarrow \neg(\exists X) \text{holds}(\text{at\_floor}(X, Y), S)
\end{align*}
\]

\[
\begin{align*}
\text{poss}(\text{call\_floor}(Y), S) \land \text{holds}(\text{at\_floor}(\text{e\_left}, Z_1), S) \land \text{holds}(\text{at\_floor}(\text{e\_right}, Z_2), S) & \rightarrow \\
& \left[ \text{state}(\text{do}(\text{call\_floor}(Y), S)) \circ \text{at\_floor}(\text{e\_left}, Z_1) = \text{state}(S) \circ \text{at\_floor}(\text{e\_left}, Y) \right] \lor \\
& \left[ \text{state}(\text{do}(\text{call\_floor}(Y), S)) \circ \text{at\_floor}(\text{e\_right}, Z_2) = \text{state}(S) \circ \text{at\_floor}(\text{e\_right}, Y) \right]
\end{align*}
\]

\[
(\exists Y_1, Y_2, Z) \text{ state}(s_0) = \text{at\_floor}(\text{e\_left}, Y_1) \circ \text{at\_floor}(\text{e\_right}, Y_2) \circ Z \land Y_1 \neq 1 \land Y_2 \neq 1
\]

[Cognitive Robotics: Nondeterministic Actions](#)
Nondeterministic Actions plus Ramifications

\[
\text{causes}(Z_1, E_1, Z_2, E_2) \leftrightarrow \\
\text{holds\_in}(\text{at\_floor}(X, Y_1), E_1) \land \text{holds\_in}(\text{at\_floor}(X, Y_2), Z_1) \land Y_1 \neq Y_2 \land \\
Z_2 \circ \text{at\_floor}(X, Y_2) = Z_1 \land E_2 = E_1 \circ \neg \text{at\_floor}(X, Y_2)
\]

\[
\text{poss}(\text{call\_floor}(Y), S) \rightarrow \\
\left[ (\exists Z) \; Z = \text{state}(S) \circ \text{at\_floor}(\text{e\_left}, Y) \land \\
\text{ramify}(Z, \text{at\_floor}(\text{e\_left}, Y), \text{state}(\text{do}(\text{call\_floor}(Y), S))) \right] \lor \\
\left[ (\exists Z) \; Z = \text{state}(S) \circ \text{at\_floor}(\text{e\_right}, Y) \land \\
\text{ramify}(Z, \text{at\_floor}(\text{e\_right}, Y), \text{state}(\text{do}(\text{call\_floor}(Y), S))) \right]
\]
Conditional Plans

▷ action $if(F, A_1, A_2) \iff$ if fluent $F$ holds then action $A_1$ else action $A_2$

▷ Foundational axioms $F_{if}$:

$poss(if(F, A_1, A_2), S) \iff$

$[holds(F, S) \rightarrow poss(A_1, S)] \land [\neg holds(F, S) \rightarrow poss(A_2, S)]$

$poss(if(F, A_1, A_2), S) \rightarrow$

$[holds(F, S) \rightarrow state(do(if(F, A_1, A_2), S)) = state(do(A_1, S))] \land$

$[\neg holds(F, S) \rightarrow state(do(if(F, A_1, A_2), S)) = state(do(A_2, S))]$
An Example Conditional Plan

\[\text{holds}(\text{at}(Y), S) \iff \text{the robot is at floor } Y \text{ in situation } S\]
\[\text{holds}(\text{inside}(X), S) \iff \text{the robot is inside of elevator } X \text{ in situation } S\]
\[\text{action } \text{enter}(X) \iff \text{the robot enters elevator } X\]

\[\text{poss}(\text{enter}(X), S) \iff \neg \text{holds}(\text{inside}(X), S) \land (\exists Y) \left[ \text{holds}(\text{at}(Y), S) \land \text{holds}(\text{at}_\text{floor}(X, Y), S) \right]\]

\[\text{poss}(\text{enter}(X), S) \rightarrow \text{state}(\text{do}(\text{enter}(X), S)) = \text{state}(S) \circ \text{inside}(X)\]

Then, from
\[\left(\exists Y_1, Y_2, Z\right) \text{state}(s_0) = \text{at}(1) \circ \text{at}_\text{floor}(\text{e}_\text{left}, Y_1) \circ \text{at}_\text{floor}(\text{e}_\text{right}, Y_2) \circ Z \land\]
\[Y_1 \neq 1 \land Y_2 \neq 1 \land (\forall X, Z') Z \neq \text{inside}(X) \circ Z'\]

it follows that
\[\text{poss}(\text{if}(\text{at}_\text{floor}(\text{e}_\text{left}, 1), \text{enter}(\text{e}_\text{left}), \text{enter}(\text{e}_\text{right})), \text{do}(\text{call}_\text{floor}(1), s_0))\]

[78] Cognitive Robotics: Nondeterministic Actions
Disjunctive Events

▶ Idea: In the Event Calculus, nondeterministic actions are modeled by disjunctive events, each of which has a deterministic outcome.

\[
\text{happens}(\text{arrives\_left}(Y), T) \iff \text{at time } T, \text{ the left elevator arrives when calling at floor } Y
\]

\[
\text{happens}(\text{arrives\_right}(Y), T) \iff \text{at time } T, \text{ the right elevator arrives when calling at floor } Y
\]

\[
\text{happens}(\text{call\_floor}(Y), T) \land (\neg \exists X \text{ holds\_at}(\text{at\_floor}(X, Y), T) \rightarrow \text{happens}(\text{arrives\_left}(Y), T) \lor \text{happens}(\text{arrives\_right}(Y), T))
\]
Modeling the Elevator Domain in the Event Calculus

Let $E$ be,

\[ \text{initiates}(\text{at\_floor}(e\_left, Y), \text{arrives\_left}(Y), T) \]
\[ \text{terminates}(\text{at\_floor}(e\_left, Y), \text{arrives\_left}(Z), T) \leftarrow Y \neq Z \]
\[ \text{initiates}(\text{at\_floor}(e\_right, Y), \text{arrives\_right}(Y), T) \]
\[ \text{terminates}(\text{at\_floor}(e\_right, Y), \text{arrives\_right}(Z), T) \leftarrow Y \neq Z \]

Let $I$ be,

\[ (\exists Y, Z) \quad \text{initially}(\text{at\_floor}(e\_left, Y)) \land \text{initially}(\text{at\_floor}(e\_right, Z)) \land Y \neq 1 \land Z \neq 1 \]

Let $N$ be,

\[ \text{happens}(\text{call\_floor}(1), 5) \]
\[ \text{happens}(\text{call\_floor}(Y), T) \land (\neg \exists X \quad \text{holds\_at}(\text{at\_floor}(X, Y), T) \rightarrow \text{happens}(\text{arrives\_left}(Y), T) \lor \text{happens}(\text{arrives\_right}(Y), T) \]

Then $\text{CIRC}[N \land I; \text{happens}] \land \text{CIRC}[E; \text{initiates, terminates}] \land \text{UNA}[e\_left, e\_right] \land EC \models$

\[ T > 5 \rightarrow \text{holds\_at}(\text{at\_floor}(e\_left, 1), T) \lor \text{holds\_at}(\text{at\_floor}(e\_right, 1), T) \]

[80] Cognitive Robotics: Nondeterministic Actions
Literature


[81] Cognitive Robotics: Nondeterministic Actions
Ramification Problem

- What are indirect effects (i.e., ramifications)?
- The frame/non-frame distinction
- Causality in the Situation Calculus
- Causal relationships in the Fluent Calculus
What are Indirect Effects?

- Action specifications may not describe all effects.
  - Indirect effects follow by general dependencies among fluents.

![Diagram of a circuit with switch and light bulb]

- Direct effect of \( \text{toggle}(sw1) \): \( \text{holds}(closed(sw1), do(\text{toggle}(sw1), s_0)) \)

- Indirect effect of \( \text{toggle}(sw1) \): \( \text{holds}(on(light), do(\text{toggle}(sw1), s_0)) \)

  State constraint: \( (\forall S) \text{holds}(on(light), S) \leftrightarrow \text{holds}(closed(sw1), S) \)

---

[83] Cognitive Robotics: Ramification Problem
Another Example: Moving Two Containers Simultaneously

\[ \text{holds}(at(X, Y), S) \iff \text{container } X \text{ is at location } Y \text{ in situation } S \]

\[
\begin{array}{cccc}
\text{b} & \text{a} \\
\text{track1} & \text{track2} & \text{temp\_store} & \text{port1} & \text{port2} \\
\text{railway} & \text{temporary store} & \text{harbor} \\
\end{array}
\]

\[ \text{holds(on(b, a), } s_0) \land \text{holds(at(a, track1), } s_0) \land \text{holds(at(b, track1), } s_0) \]

\[
\text{direct effect} : \text{holds(at(a, port2), do(move(a, port2), } s_0)) \\
\text{indirect effect} : \text{holds(at(b, port2), do(move(a, port2), } s_0)) \\
\text{state constraint} : (\forall X, Y, L, S) \text{ holds(at(X, L), } S) \land \text{holds(on(Y, X), } S) \rightarrow \text{holds(at(Y, L), } S)
\]

[84] Cognitive Robotics: Ramification Problem
A First Approach: Compiling Away the Problem

Encode direct and indirect effects in successor state (or: state update, or: effect) axioms.

\[
\text{poss}(A, S) \rightarrow \\
\quad \left[ \text{holds}(\text{at}(X, L), \text{do}(A, S)) \iff \\
\quad \quad a = \text{move}(X, L) \lor \\
\quad \quad (\exists Y) a = \text{move}(Y, L) \land \text{holds}(\text{on}(X, Y), S) \lor \\
\quad \quad \text{holds}(\text{at}(X, L), S) \land \neg(\exists L') A = \text{move}(X, L') \\
\quad \quad \quad \land \neg(\exists Y, L') A = \text{move}(Y, L') \land \text{holds}(\text{on}(X, Y), S) \right]
\]

\[
\text{poss}(A, S) \rightarrow \\
\quad \left[ \text{holds}(\text{light}(X), \text{do}(A, S)) \iff \\
\quad \quad A = \text{toggle}(\text{sw}1) \land \neg\text{holds}(\text{closed}(\text{sw}1), S) \land X = \text{bulb} \lor \\
\quad \quad \text{holds}(\text{light}(X), S) \land \neg( A = \text{toggle}(\text{sw}1) \land \text{holds}(\text{closed}(\text{sw}1), S) \land X = \text{bulb} ) \right]
\]
A Problem with this Solution

- Encoding all indirect effects in successor state (or: ...) axioms leads to less concise and succinct axiomatizations, which are more difficult to elaborate upon.

\[ sw1 \]

\[ \sim \] With ordinary state update axioms, this example requires \(2^{n+1}\) update axioms for \(\text{toggle}(sw1)\).

\[ \sim \] Adding just another switch-bulb pair would amount to rewriting the successor state axiom for \(\text{light}(X)\) rather than just adding the corresponding new state constraint.

[86] Cognitive Robotics: Ramification Problem
An Even More Serious Problem

\[
\text{holds}(\text{on}(b, a), s_0) \land \text{holds}(\text{at}(a, \text{track1}), s_0) \land \text{holds}(\text{at}(b, \text{track1}), s_0) \land \text{holds}(\text{at}(b, \text{track1}), s_0)
\]

dir. eff.: \( \text{holds}(\text{at}(a, \text{port2}), \text{do}(\text{move}(a, \text{port2}), s_0)) \)

indir. eff.: \( \text{holds}(\text{at}(b, \text{port2}), \text{do}(\text{move}(a, \text{port2}), s_0)); \text{holds}(\text{at}(c, \text{port2}), \text{do}(\text{move}(a, \text{port2}), s_0)) \)

constr.: \( (\forall X, Y, L, S) \text{ holds}(\text{at}(X, L), S) \land \text{holds}(\text{on}(Y, X), S) \rightarrow \text{holds}(\text{at}(Y, L), S) \)

A successor state axiom for \( \text{at}(X, L) \) along the line of slide [85] sets an upper bound for the number of containers indirectly moved.

[87] Cognitive Robotics: Ramification Problem
A First Solution: The Frame-/Non-Frame Distinction

▷ Idea: Only frame fluents have successor state axioms.

\[
\begin{align*}
\text{f frame fluent} & \iff \text{f directly manipulated by actions, never an indirect effect} \\
\text{f non-frame fluent} & \iff \text{f never directly manipulated by actions,} \\
& \quad \text{f completely determined by all frame fluents}
\end{align*}
\]

\[
\begin{align*}
\text{poss}(A, S) & \rightarrow \\
& \quad \left[ \text{holds(closed}(X), \text{do}(A, S)) \leftrightarrow \\
& \quad A = \text{toggle}(X) \land \neg \text{holds(closed}(X), S) \lor \\
& \quad \text{holds(closed}(X), S) \land A \neq \text{toggle}(X) \right] \\
\text{holds(on(light), S)} & \leftrightarrow \text{holds(closed(sw1), S)}
\end{align*}
\]
Why this Solution is Restricted

▷ Not always is it possible to separate frame and non-frame fluents.

\[ \neg \text{holds}(\text{closed}(\text{sw1}), s_0) \]
\[ \text{holds}(\text{closed}(\text{sw2}), s_0) \]
\[ \text{holds}(\text{closed}(\text{sw3}), s_0) \]
\[ \neg \text{holds}(\text{active}(\text{relay}), s_0) \]
\[ \neg \text{holds}(\text{light}(\text{bulb}), s_0) \]

\[ \text{holds}(\text{light}(\text{bulb}), S) \leftrightarrow \text{holds}(\text{closed}(\text{sw1}), S) \land \text{holds}(\text{closed}(\text{sw2}), S) \]
\[ \text{holds}(\text{active}(\text{relay}), S) \leftrightarrow \text{holds}(\text{closed}(\text{sw1}), S) \land \text{holds}(\text{closed}(\text{sw3}), S) \]
\[ \text{holds}(\text{active}(\text{relay}), S) \rightarrow \neg \text{holds}(\text{closed}(\text{sw2}), S) \]

Frame fluents are: \text{closed}(\text{sw1}), \text{closed}(\text{sw3})
Non-frame fluents are: \text{active}(\text{relay}), \text{light}(\text{bulb})

But what about \text{closed}(\text{sw2})?
Causality in the Situation Calculus: Example

▷ Idea: Formulate both direct and indirect effects succinctly by ‘local’ causal relations.

\[
\text{caused}(F, V, S) \Leftrightarrow \text{fluent } F \text{ is caused to have truth value } V \text{ in situation } S
\]

\[
\text{poss}(\text{toggle}(X), S) \rightarrow \\
\neg \text{holds}(\text{closed}(X), S) \rightarrow \text{caused}(\text{closed}(X), \text{true}, \text{do(\text{toggle}(X), S)})
\]

\[
\text{poss}(\text{toggle}(X), S) \rightarrow \\
\text{holds}(\text{closed}(X), S) \rightarrow \text{caused}(\text{closed}(X), \text{false}, \text{do(\text{toggle}(X), S)})
\]

\[
\text{holds}(\text{closed}(\text{sw1}), S) \land \text{holds}(\text{closed}(\text{sw2}), S) \rightarrow \text{caused}(\text{light(bulb)}, \text{true}, S)
\]

\[
\neg \text{holds}(\text{closed}(\text{sw1}), S) \lor \neg \text{holds}(\text{closed}(\text{sw2}), S) \rightarrow \text{caused}(\text{light(bulb)}, \text{false}, S)
\]

\[
\text{holds}(\text{closed}(\text{sw1}), S) \land \text{holds}(\text{closed}(\text{sw3}), S) \rightarrow \text{caused}(\text{active(relay)}, \text{true}, S)
\]

\[
\neg \text{holds}(\text{closed}(\text{sw1}), S) \lor \neg \text{holds}(\text{closed}(\text{sw3}), S) \rightarrow \text{caused}(\text{active(relay)}, \text{false}, S)
\]

\[
\text{holds}(\text{active(relay)}, S) \rightarrow \text{caused}(\text{closed(\text{sw2})}, \text{false}, S)
\]
The General Approach (1)

( Formulas $\Phi(S)$ and $\pi_a(S)$ are simple formulas; c.f. slide [12] of SitCalc. )

▷ Direct effect axioms $\mathcal{F}_{ef}$ for each action $a$:

\[
\text{poss}(a(\overline{X}), S) \rightarrow \Phi(S) \rightarrow \text{caused}(f(\overline{Y}), v, \text{do}(a(\overline{X}), S))
\]

▷ Causal rules $\mathcal{F}_{cr}$:

\[
\Phi(S) \land \text{caused}(f_1, v_1, S) \land \ldots \land \text{caused}(f_n, v_n, S) \rightarrow \text{caused}(f(\overline{X}), v, S)
\]

▷ CIRC[$\mathcal{F}_{ef} \land \mathcal{F}_{cr}; \text{caused}$]

▷ Precondition axioms $\mathcal{F}_{ap}$ for each action $a$:

\[
\pi_a(S) \rightarrow \text{poss}(a, S)
\]
The General Approach (2)

▷ Foundational axioms $\mathcal{F}_{\text{fnd}}$:

\[
\text{caused}(F, \text{true}, S) \rightarrow \text{holds}(F, S) \\
\text{caused}(F, \text{false}, S) \rightarrow \neg\text{holds}(F, S) \\
\text{true} \neq \text{false} \land (\forall V: \text{truth_value}) \ V = \text{true} \lor V = \text{false}
\]

▷ Unique names assumptions $\mathcal{F}_{\text{una}}$ and $\mathcal{F}_{\text{uns}}$
Generating Successor State Axioms

\( \mathcal{F}_{cr} \) is stratified : \( \iff \) no chain of fluents \( f_0, f_1, \ldots, f_n \) exists such that \( f_0 \leadsto f_1 \leadsto \ldots \leadsto f_n \leadsto f_0 \)

where \( f \leadsto f' : \iff \mathcal{F}_{cr} \) contains a causal rule with \( f \) occurring to the left of the implication and \( f' \) to the right

Proposition:
If \( \mathcal{F}_{cr} \) is stratified, then there is a simple rewriting procedure by which we can obtain a successor state axiom for each fluent \( f(\overline{X}) \), namely, by using formulas \( \Psi_{f,v} \) such that

\[
\text{CIRC}[\mathcal{F}_{ef} \land \mathcal{F}_{cr}; \text{caused}] \models \text{caused}(f(\overline{X}), v, S) \leftrightarrow \Psi_{f,v}
\]

and the schema
\[
\begin{align*}
\text{poss}(A, S) & \rightarrow \\
& \left[ \text{holds}(f(\overline{X}), \text{do}(A, S)) \leftrightarrow \text{caused}(f(\overline{X}), \text{true, do}(A, S)) \lor \\
& \quad \text{holds}(f(\overline{X}), S) \land \neg\text{caused}(f(\overline{X}), \text{false, do}(A, S)) \right]
\end{align*}
\]
The Circuit in PROLOG

holds(light(X), do(A,S)) :-
  X=bulb, holds(closed(sw1),do(A,S)), holds(closed(sw2),do(A,S)) ;
  holds(light(X),S), (not X=bulb ; holds(closed(sw1),do(A,S))),
  holds(closed(sw2),do(A,S))
).

holds(active(X), do(A,S)) :-
  X=relay, holds(closed(sw1),do(A,S)), holds(closed(sw3),do(A,S)) ;
  holds(active(X),S), (not X=relay ; holds(closed(sw1),do(A,S))),
  holds(closed(sw3),do(A,S))
).

holds(closed(X), do(A,S)) :-
  A=toggle(X), not holds(closed(X),S) ;
  holds(closed(X),S), not A=toggle(X), ( not X=sw2 ;
    not holds(active(relay),do(A,S))
  )
).

holds(closed(sw2), s0). holds(closed(sw3), s0).

| ?- holds(F, do(toggle(sw1),s0)).
F = active(relay);
F = closed(sw1);
F = closed(sw3)
Restrictions of this Approach (1)

▷ The proposition on slide [93] does not apply if $\mathcal{F}_{cr}$ is not stratified, as in

\[
\text{holds}(\text{joined}(X, Y), S) \land \text{holds}(\text{closed}(X), S) \rightarrow \text{caused}(\text{closed}(Y), \text{true}, S)
\]

\[
\text{holds}(\text{joined}(X, Y), S) \land \neg \text{holds}(\text{closed}(X), S) \rightarrow \text{caused}(\text{closed}(Y), \text{false}, S)
\]

\[
\text{holds}(\text{joined}(X,Y), \text{do}(A,S)) :\text{ holds}(\text{joined}(X,Y), S).
\]

\[
\text{holds}(\text{closed}(X), \text{do}(A,S)) :\text{-}
\]

\[
A=\text{toggle}(X), \text{not} \text{ holds}(\text{closed}(X), S);
\]

\[
\text{holds}(\text{joined}(Y,X), \text{do}(A,S)), \text{ holds}(\text{closed}(Y), \text{do}(A,S));
\]

\[
\text{holds}(\text{closed}(X), S),
\]

\[
\text{not} A=\text{toggle}(X), \text{not} (\text{ holds}(\text{joined}(Y,X), \text{do}(A,S)),
\]

\[
\text{not} \text{ holds}(\text{closed}(Y), \text{do}(A,S))).
\]

\[
\text{holds}(\text{joined}(\text{sw1}, \text{sw2}), s0). \quad \text{holds}(\text{joined}(\text{sw2}, \text{sw1}), s0).
\]

\[
\text{holds}(\text{closed}(\text{sw1}), s0). \quad \text{holds}(\text{closed}(\text{sw2}), s0).
\]

\[
\mid \text{ ?- holds}(\text{closed}(X), \text{do}(\text{idle},s0)).
\]

The query diverges!

[95] Cognitive Robotics: Ramification Problem
Restrictions of this Approach (2)

\[
\begin{align*}
\text{holds}(\text{closed}(sw1), S) \land \text{holds}(\text{closed}(sw2), S) & \rightarrow \text{caused}(\text{active}(re2), \text{true}, S) \\
\neg\text{holds}(\text{closed}(sw1), S) \lor \neg\text{holds}(\text{closed}(sw2), S) & \rightarrow \text{caused}(\text{active}(re2), \text{false}, S) \\
\text{holds}(\text{closed}(sw1), S) \land \text{holds}(\text{closed}(sw3), S) & \rightarrow \text{caused}(\text{active}(re1), \text{true}, S) \\
\neg\text{holds}(\text{closed}(sw1), S) \lor \neg\text{holds}(\text{closed}(sw3), S) & \rightarrow \text{caused}(\text{active}(re1), \text{false}, S) \\
\text{holds}(\text{active}(re1), S) & \rightarrow \text{caused}(\text{closed}(sw2), \text{false}, S) \\
\text{holds}(\text{active}(re2), S) & \rightarrow \text{caused}(\text{closed}(sw4), \text{false}, S)
\end{align*}
\]

\[\rightarrow\] The possible indirect effect \(\neg\text{holds}(\text{closed}(sw4), \text{do}(\text{toggle}(sw1), s_0))\) cannot be obtained. (The reason being that non-minimal effects are ruled out by circumscribing \(\mathcal{F}_{cr}\).)
Causal Relationships in the Fluent Calculus

▷ Idea: Indirect effects are computed via causal propagation.

\[ \text{causes}(Z_1, E_1, Z_2, E_2) :\Leftrightarrow \text{in state } Z_1 \text{ effects } E_1 \text{ trigger an update to state } Z_2 \]
(and hence from \( E_1 \) to \( E_2 \))

\[ \text{holds}_\text{in}(E_1, E) \overset{\text{def}}{=} \exists E' \ E = E_1 \circ E' \]

Two examples of causal relationships:

\[ \text{causes}(Z \circ \text{closed}(\text{sw2}), E \circ \text{active}(\text{relay}), Z, E \circ \text{active}(\text{relay}) \circ \lnot \text{closed}(\text{sw2})) \]

\[ \text{holds}_\text{in}((\text{on}(X, Y), Z)) \rightarrow \]
\[ \text{causes}(Z \circ \text{at}(X, L), E \circ \text{at}(Y, L'), Z \circ \text{at}(X, L'), E \circ \text{at}(Y, L') \circ \lnot \text{at}(X, L) \circ \text{at}(X, L')) \]

[97] Cognitive Robotics: Ramification Problem
The General Approach (1)

▷ Sort \textbf{EFFECTS} such that

\begin{itemize}
  \item STATE < EFFECTS
  \item \(\Sigma_{E ff} = \{-\}\) where \(- : \text{FLUENT} \mapsto \text{EFFECTS}\)
\end{itemize}

▷ State constraints \(\mathcal{F}_{sc}\)

▷ Causal relationships \(\mathcal{F}_{cr}\):

\[\Phi \rightarrow causes(Z_1, E_1, Z_2, E_2)\]

where \(\Phi\) is an equational formula

▷ \(\text{COMP}[\mathcal{F}_{cr}]\)
The General Approach (2)

\[ \text{ramify}(Z, E, Z') \iff \text{state } Z' \text{ is reachable from state } Z \text{ and effects } E \]

\[ \vdash \text{Second-order foundational axiom } \mathcal{F}_{ra} : \]

\[ \text{ramify}(Z, E, Z') \leftrightarrow \forall \Pi \begin{align*}
& (\forall Z_1, E_1) \Pi(Z_1, E_1, Z_1, E_1) \\
& \wedge \\
& (\forall Z_1, E_1, Z_2, E_2, Z_3, E_3) \\
& \Pi(Z_1, E_1, Z_2, E_2) \wedge \text{causes}(Z_2, E_2, Z_3, E_3) \\
& \rightarrow \Pi(Z_1, E_1, Z_3, E_3) \\
& \rightarrow \\
& (\exists E') \Pi(Z, E, Z', E')
\end{align*} \]

\[ \vdash \text{State update axioms } \mathcal{F}_{sua} : \]

\[ \Delta(S) \rightarrow Z \circ \vartheta^- = \text{state}(S) \circ \vartheta^+ \rightarrow \text{ramify}(Z, \vartheta^+ \circ -\vartheta^-, \text{state}(\text{do}(a, S))) \]

where \( -(F_1 \circ \ldots \circ F_n) \overset{\text{def}}{=} -F_1 \circ \ldots \circ -F_n \)

[99]  Cognitive Robotics: Ramification Problem
The Jointed Switches in the Fluent Calculus

\[ \mathcal{F}_{sc} \iff \text{holds}(\text{joined}(X, Y), S) \rightarrow [\text{holds}(\text{closed}(Y), S) \iff \text{holds}(\text{closed}(X), S)] \]

\[ \mathcal{F}_{ap} \iff \text{poss}(\text{toggle}(X), S) \]

\[ \mathcal{F}_{sua} \iff \text{poss}(\text{toggle}(X), S) \land \neg \text{holds}(\text{closed}(X), S) \rightarrow \]
\[ Z = \text{state}(S) \circ \text{closed}(X) \rightarrow \]
\[ \text{ramify}(Z, \text{closed}(X), \text{state}(\text{do}(\text{toggle}(X), S))) \]

\[ \text{poss}(\text{toggle}(X), S) \land \text{holds}(\text{closed}(X), S) \rightarrow \]
\[ Z \circ \text{closed}(X) = \text{state}(S) \rightarrow \]
\[ \text{ramify}(Z, \neg \text{closed}(X), \text{state}(\text{do}(\text{toggle}(X), S))) \]

\[ \text{COMP}[\mathcal{F}_{cr}] \iff \]
\[ \text{causes}(Z_1, E_1, Z_2, E_2) \iff \]
\[ \text{holds\_in}(\text{closed}(X), E_1) \land \text{holds\_in}(\text{joined}(X, Y), Z_1) \land \neg \text{holds\_in}(\text{closed}(Y), Z_1) \land \]
\[ Z_2 = Z_1 \circ \text{closed}(Y) \land E_2 = E_1 \circ \text{closed}(Y) \lor \]
\[ \text{holds\_in}(\neg \text{closed}(X), E_1) \land \text{holds\_in}(\text{joined}(X, Y), Z_1) \land \text{holds\_in}(\text{closed}(Y), Z_1) \land \]
\[ Z_2 \circ \text{closed}(Y) = Z_1 \land E_2 = E_1 \circ \neg \text{closed}(Y) \]
The Double Relay Circuit in the Fluent Calculus (1)

\[ F_{sc} :\iff \text{holds}(\text{active}(re1), S) \iff \text{holds}(\text{closed}(sw1), S) \land \text{holds}(\text{closed}(sw3), S) \]

\[ \text{holds}(\text{active}(re2), S) \iff \text{holds}(\text{closed}(sw1), S) \land \text{holds}(\text{closed}(sw2), S) \]

\[ \text{holds}(\text{active}(re1), S) \rightarrow \neg \text{holds}(\text{closed}(sw2), S) \]

\[ \text{holds}(\text{active}(re2), S) \rightarrow \neg \text{holds}(\text{closed}(sw4), S) \]

\[ F_{ap} :\iff \text{poss}(\text{toggle}(X), S) \]

\[ F_{sua} :\iff \text{poss}(\text{toggle}(X), S) \land \neg \text{holds}(\text{closed}(X), S) \rightarrow \]

\[ Z = \text{state}(S) \circ \text{closed}(X) \rightarrow \]

\[ \text{ramify}(Z, \text{closed}(X), \text{state}(\text{do}(\text{toggle}(X), S))) \]

\[ \text{poss}(\text{toggle}(X), S) \land \text{holds}(\text{closed}(X), S) \rightarrow \]

\[ Z \circ \text{closed}(X) = \text{state}(S) \rightarrow \]

\[ \text{ramify}(Z, \neg \text{closed}(X), \text{state}(\text{do}(\text{toggle}(X), S))) \]
The Double Relay Circuit in the Fluent Calculus (2)

\[
\text{COMP}[\mathcal{F}_{cr}] :\iff
\]

\[
\begin{align*}
\text{causes}(Z_1, E_1, Z_2, E_2) & \iff \\
\ & \text{holds}_\text{in}(\text{closed(sw1)}, E_1) \land \text{holds}_\text{in}(\text{closed(sw3)}, Z_1) \land \neg \text{holds}_\text{in}(\text{active(re1)}, Z_1) \land \\
\ & \quad Z_2 = Z_1 \circ \text{active(re1)} \land E_2 = E_1 \circ \text{active(re1)} \lor \\
\ & \text{holds}_\text{in}(\text{closed(sw3)}, E_1) \land \text{holds}_\text{in}(\text{closed(sw1)}, Z_1) \land \neg \text{holds}_\text{in}(\text{active(re1)}, Z_1) \land \\
\ & \quad Z_2 = Z_1 \circ \text{active(re1)} \land E_2 = E_1 \circ \text{active(re1)} \lor \\
\ & \text{holds}_\text{in}(\neg \text{closed(sw1)}, E_1) \land \text{holds}_\text{in}(\text{active(re1)}, Z_1) \land \\
\ & \quad Z_2 \circ \text{active(re1)} = Z_1 \land E_2 = E_1 \circ \neg \text{active(re1)} \lor \\
\ & \text{holds}_\text{in}(\neg \text{closed(sw3)}, E_1) \land \text{holds}_\text{in}(\text{active(re1)}, Z_1) \land \\
\ & \quad Z_2 \circ \text{active(re1)} = Z_1 \land E_2 = E_1 \circ \neg \text{active(re1)} \lor
\end{align*}
\]

\ldots
The Double Relay Circuit in the Fluent Calculus (3)

\[
\begin{align*}
\text{holds\_in}(\text{closed}(\text{sw1}), E_1) \land \text{holds\_in}(\text{closed}(\text{sw2}), Z_1) \land \neg \text{holds\_in}(\text{active}(\text{re2}), Z_1) \land \\
Z_2 &= Z_1 \circ \text{active}(\text{re2}) \land E_2 = E_1 \circ \text{active}(\text{re2}) \\
\text{holds\_in}(\text{closed}(\text{sw2}), E_1) \land \text{holds\_in}(\text{closed}(\text{sw1}), Z_1) \land \neg \text{holds\_in}(\text{active}(\text{re2}), Z_1) \land \\
Z_2 &= Z_1 \circ \text{active}(\text{re2}) \land E_2 = E_1 \circ \text{active}(\text{re2}) \\
\text{holds\_in}(\neg\text{closed}(\text{sw1}), E_1) \land \text{holds\_in}(\text{active}(\text{re2}), Z_1) \land \\
Z_2 \circ \text{active}(\text{re2}) &= Z_1 \land E_2 = E_1 \circ \neg\text{active}(\text{re2}) \\
\text{holds\_in}(\neg\text{closed}(\text{sw2}), E_1) \land \text{holds\_in}(\text{active}(\text{re2}), Z_1) \land \\
Z_2 \circ \text{active}(\text{re2}) &= Z_1 \land E_2 = E_1 \circ \neg\text{active}(\text{re2}) \\
\text{holds\_in}(\text{active}(\text{re1}), E_1) \land \text{holds\_in}(\text{closed}(\text{sw2}), Z_1) \land \\
Z_2 \circ \text{closed}(\text{sw2}) &= Z_1 \land E_2 = E_1 \circ \neg\text{closed}(\text{sw2}) \\
\text{holds\_in}(\text{active}(\text{re2}), E_1) \land \text{holds\_in}(\text{closed}(\text{sw4}), Z_1) \land \\
Z_2 \circ \text{closed}(\text{sw4}) &= Z_1 \land E_2 = E_1 \circ \neg\text{closed}(\text{sw4})
\end{align*}
\]
Causal relationships can be automatically generated from state constraints and a binary ‘influence relation,’ which conveys additional domain knowledge of which fluents may directly influence which fluents.

It can be necessary, depending on the application, to distinguish ‘stabilizing’ causal relationships (which describe indirect effects with small temporal lag) from ‘steady’ causal relationships (which describe truly instantaneous indirect effects).
Literature


Specificity

- Objects, Classes and Methods
- Fluent Calculus Formalization
- Specificity
- The General Approach
- Specificity and State Constraints
Objects, Classes and Methods

▷ **Objects** are characterized by an (internal) **state**.
▷ They are grouped into **classes**.
▷ For each class certain **methods** are defined which, if applied to an object of this class, modify the object’s state.
▷ Classes are **ordered** wrt some partial ordering.
▷ Methods of a class \( C \) are **inherited** by its subclasses.
▷ Inherited methods may be **overridden** if more specific methods are defined.
An Example Hierarchy of Classes

The class of movable objects

\[ \text{loc}(X, Y) \]

The class of movable fragile objects

\[ \text{loc}(X, Y), \text{fragile}(X) \]

\[ \text{loc}(X, Y), \text{intact}(X), \text{fragile}(X) \]

\[ \text{loc}(X, Y), \text{broken}(X), \text{fragile}(X) \]

\[ \text{loc}(X, Y) \]: object \( X \) is at location \( Y \)

\[ \text{fragile}(X) \]: object \( X \) is fragile

\[ \text{broken}(X) \]: object \( X \) is broken

\[ \text{intact}(X) \]: object \( X \) is intact
Formalizing Objects and Classes

▷ An object is a ground constructor state term, eg.
\[
\text{loc(nugget, table)} \quad \text{or} \\
\text{loc(vase, table)} \circ \text{fragile(vase)} \quad \text{or} \\
\text{loc(vase, floor)} \circ \text{fragile(vase)} \circ \text{broken(vase)}. 
\]

▷ A class is a constructor state term without occurrences of constants, eg.
\[
\text{loc}(X, Y) \quad \text{or} \\
\text{loc}(X, Y) \circ \text{fragile}(X) \quad \text{or} \\
\text{loc}(X, Y) \circ \text{fragile}(X) \circ \text{broken}(X).
\]

▷ An object \(o\) belongs to a class \(c\) iff \((\exists \sigma) \\ o = c \sigma\), eg.
the object \(\text{loc(nugget, table)}\) belongs to the class \(\text{loc}(X, Y)\).

▷ The initial state is a ground constructor state term and is denoted by the situation \(s_0\), eg.
\[
\text{state}(s_0) = \text{loc(nugget, table)} \quad \text{or} \\
\text{state}(s_0) = \text{loc(nugget, table)} \circ \text{loc(vase, table)} \circ \text{fragile(vase)}. 
\]
Formalizing Methods

- A **method** is an action and is specified by a state update axiom, eg.
  \[ \text{holds}(\text{loc}(X, Y), S) \rightarrow \text{state}(\text{do}(\text{move}(X, Y, Z), S)) \circ \text{loc}(X, Z) = \text{state}(S) \circ \text{loc}(X, Y) \]

- Methods are **applied** to an object in some situation by applying the corresponding state update axiom, eg.
  \[ \text{state}(\text{do}(\text{move}(\text{nugget, table, cupboard}), s_0)) = \text{loc}(\text{nugget, cupboard}) \]

- Inheritance comes for free: Let \( \text{state}(s_0) = \text{loc}(\text{vase, table}) \circ \text{fragile(\text{vase})} \) then
  \[ \text{state}(\text{do}(\text{move}(\text{vase, table, cupboard}), s_0)) = \text{loc}(\text{vase, cupboard}) \circ \text{fragile(\text{vase})} \]
Dropping Objects

▷ Let $\text{drop}(X)$ be an action, in which object $X$ is lifted up and dropped to the floor.

▷ For the class of movable objects we define

$$\text{holds}(\text{loc}(X, Y), S) \implies \text{state}(\text{do(\text{drop}(X), S))) \circ \text{loc}(X, Y) = \text{state}(S) \circ \text{loc}(X, \text{floor})$$

(1)

▷ Let $\text{state}(s_0) = \text{loc}(\text{nugget, table})$ then

$$\text{state}(\text{do(\text{drop(nugget), s_0))}) = \text{loc(\text{nugget, floor})}.$$
How about Overriding?

▷ For the class of fragile objects we define a more specific method:

\[
\text{holds}(\text{loc}(X, Y), S) \land \text{holds}(\text{fragile}(X), S) \\
\Rightarrow \text{state}(\text{do}(\text{drop}(X), S)) \circ \text{loc}(X, Y) = \text{state}(S) \circ \text{loc}(X, \text{floor}) \circ \text{broken}(X) \tag{2}
\]

▷ Let \( \text{state}(s_0) = \text{loc}(\text{vase}, \text{table}) \circ \text{fragile}(\text{vase}) \) then by (2)

\[
\text{state}(\text{do}(\text{drop}(\text{vase}), s_0)) = \text{loc}(\text{vase}, \text{floor}) \circ \text{fragile}(\text{vase}) \circ \text{broken}(\text{vase}).
\]

But (1) is also applicable and, if applied, yields

\[
\text{state}(\text{do}(\text{drop}(\text{vase}), s_0)) = \text{loc}(\text{vase}, \text{floor}) \circ \text{fragile}(\text{vase}).
\]

\[\Rightarrow\] There is a contradiction concerning the fluent \( \text{broken}(\text{vase}) \).

\[\Rightarrow\] We would like to block the application of (1).

[112] Cognitive Robotics: Specificity
Specificity

Let \( \text{cond}(1, \text{drop}(X), X, Y, S) \leftrightarrow \text{holds}(\text{loc}(X, Y), S) \)
\[ \text{cond}(2, \text{drop}(X), X, Y, S) \leftrightarrow \text{holds}(\text{loc}(X, Y), S) \land \text{holds}(\text{fragile}(X), S) \]

\( \sim \) \( \text{cond}(1, \text{drop}(X), X, Y, S) \) subsumes \( \text{cond}(2, \text{drop}(X), X, Y, S) \).

A conditional is an equivalence of the form \( \text{cond}(N, a, \overline{X}, S) \leftrightarrow \Delta(\overline{X}, S) \), where \( N \) is a natural number, \( a \) a term of sort \( \text{ACTION} \), \( \overline{X} \) a list of variables of sort \( \text{OBJECT} \), \( S \) a variable of sort \( \text{SITUATION} \) and \( \Delta(\overline{X}, S) \) the condition of a state update axiom for \( A \).

Let \( \mathcal{F}_C \) be the set of conditionals for a given set of state update axioms such that no natural number occurs more than once as first argument of \( \text{cond} \).

\( \text{cond}(N, a, \overline{X}, S) \) is more specific than \( \text{cond}(M, a, \overline{X}, S) \) iff \( \text{cond}(N, a, \overline{X}, S) \rightarrow \text{cond}(M, a, \overline{X}, S) \) is valid.

\( \text{cond}(N, a, \overline{X}, S) \) is strictly more specific than \( \text{cond}(M, a, \overline{X}, S) \) iff \( [\text{cond}(N, a, \overline{X}, S) \rightarrow \text{cond}(M, a, \overline{X}, S)] \land \neg[\text{cond}(M, a, \overline{X}, S) \rightarrow \text{cond}(N, a, \overline{X}, S)] \) is valid.

\( \sim \) \( \text{cond}(2, \text{drop}(X), X, Y, S) \) is strictly more specific \( \text{cond}(1, \text{drop}(X), X, Y, S) \).

[113] Cognitive Robotics: Specificity
Most Specific Actions

\[ \text{sms}(N, M, a, X, S) \]
\[ \overset{\text{def}}{=} [\text{cond}(N, a, X, S) \rightarrow \text{cond}(M, a, X, S)] \land \neg[\text{cond}(M, a, X, S) \rightarrow \text{cond}(N, a, X, S)] \]
\[ \leadsto \text{sms}(2, 1, \text{drop}(X), X, Y, S) \]

- Idea: Apply only most specific applicable action.

- Transform (1) into
\[ \text{cond}(1, \text{drop}(X), X, Y, S) \land \neg(\exists N) \text{sms}(N, 1, \text{drop}(X), X, Y, S) \]
\[ \rightarrow \text{state}(\text{do}(\text{drop}(X), S)) \circ \text{loc}(X, Y) = \text{state}(S) \circ \text{loc}(X, \text{floor}) \]  \hspace{1cm} (3)

and (2) into
\[ \text{cond}(2, \text{drop}(X), X, Y, S) \land \neg(\exists N) \text{sms}(N, 1, \text{drop}(X), X, Y, S) \]
\[ \rightarrow \text{state}(\text{do}(\text{drop}(X), S)) \circ \text{loc}(X, Y) = \text{state}(S) \circ \text{loc}(X, \text{floor}) \circ \text{broken}(X). \]  \hspace{1cm} (4)

[114] Cognitive Robotics: Specificity
Circumscription

Let 
\[
state(s_0) = \text{loc}(\text{vase, table}) \circ \text{fragile}(\text{vase}).
\]

\(\rightsquigarrow\) (3) is blocked!
\(\rightsquigarrow\) (4) is also blocked!
\(\rightsquigarrow\) We have to minimize the extension of \textit{cond} to ensure that (4) is not blocked.

Let \(\mathcal{F} = \{(3),(4)\} \cup \mathcal{F}_C\) and consider \(\text{CIRC}(\mathcal{F}; \text{cond})\) instead of \(\mathcal{F}\).

\(\rightsquigarrow\) (3) is still blocked!
\(\rightsquigarrow\) (4) is applicable and yields
\[
state(\text{do(drop(vase), } s_0)) = \text{loc}(\text{vase, floor}) \circ \text{fragile(vase)} \circ \text{broken(vase)}.
\]

\(\rightsquigarrow\) The vase is broken.
Specificity: The General Approach

Let $F_{sua}$ be the set of successor state axioms of the form
\[
\Delta(\overline{X}, S) \rightarrow \text{state}(do(a, S)) \circ \vartheta^- = \text{state}(S) \circ \vartheta^+.
\]

Let $F_C$ be the set of conditionals for $F$, i.e. for each element (5) in $F$ the set $F_C$ contains an element of the form
\[
\text{cond}(N, a, \overline{X}, S) \leftrightarrow \Delta(\overline{X}, S).
\]

Let $F_E$ be the set of extended successor state axioms, which is obtained by replacing each element (5) of $F$ by
\[
\text{cond}(N, a, \overline{X}, S) \land \neg(\exists M) \: \text{sms}(M, N, a, \overline{X}, S)
\rightarrow \text{state}(do(a, S)) \circ \vartheta^- = \text{state}(S) \circ \vartheta^+
\]
if $\text{cond}(N, a, \overline{X}, S) \leftrightarrow \Delta(\overline{X}, S) \in F_C$.

Let $F_{euna}$ be the extended unique names assumptions.

Consider $F_{euna} \cup F_C \cup \text{CIRC}(F_E; \text{cond})$.

[116] Cognitive Robotics: Specificity
Specificity and Ramification

▷ In the presence of state constraints it may be necessary to combine specificity with ramifications.

▷ Consider the state constraint

\[-(\exists X, S, Z) \text{intact}(X) \circ \text{broken}(X) \circ Z = \text{state}(S).\]

▷ Let

\[\text{state}(s_0) = \text{loc}(\text{vase, table}) \circ \text{fragile(vase)} \circ \text{intact(vase)}\]

▷ Applying (4) yields

\[\text{state}(\text{do(drop(vase), s_0)}) = \text{loc}(\text{vase, floor}) \circ \text{fragile}(X) \circ \text{intact}(vase) \circ \text{broken}(X)\]

which is inconsistent wrt the state constraint.

~ An additional ramification step yields

\[\text{state}(\text{do(drop(vase), s_0))) = \text{loc}(\text{vase, floor}) \circ \text{fragile}(X) \circ \text{broken}(X).\]
Specificity and the Need to Add Methods

▷ In the presence of state constraints it may be necessary to add methods.

▷ Consider the state constraint

\[(\forall F, S, Z) \ F \circ F \circ Z \neq state(S).\]

▷ Let

\[state(s_0) = loc(vase, table) \circ fragile(vase) \circ broken(vase)\]

▷ Applying (4) yields

\[state(do(drop(vase), s_0)) = loc(vase, floor) \circ fragile(X) \circ fragile(vase) \circ broken(X)\]

which is inconsistent wrt the state constraint.

▷ Adding the state update axioms

\[holds(loc(X, Y), S) \land holds(fragile(X), S) \land holds(broken(X))\]

\[\rightarrow state(do(drop(X), S)) \circ loc(X, Y) = state(S) \circ loc(X, floor)\]

will remedy this problem. This can be done automatically.

[118] Cognitive Robotics: Specificity
Literature


[119] Cognitive Robotics: Specificity
Concurrent Actions

- Modeling a fleet of fork lift trucks
- Concurrent actions in the Event Calculus
- Concurrent actions in the Situation Calculus
- Concurrent actions in the Fluent Calculus
Two Fork Lift Trucks

\[
\text{event } \text{move}(F, X, Y) : \Leftrightarrow \text{ fork lift truck } F \text{ moves } X \text{ to } Y
\]

\[
\text{happens}(\text{move}(\text{fork1}, a, \text{port1}), 1) \land \text{happens}(\text{move}(\text{fork2}, d, \text{temp\_store}), 1)
\]

[Cognitive Robotics: Concurrent Actions]
The New Effect Axioms

\[
\text{initiates}(\text{move}(F, X, Y), \text{on}(X, Y), T) \leftarrow \\
\text{holds}_\text{at}(\text{clear}(X), T) \land \text{holds}_\text{at}(\text{clear}(Y), T) \land X \neq Y
\]

\[
\text{initiates}(\text{move}(F, X, Y), \text{clear}(Z), T) \leftarrow \\
\text{holds}_\text{at}(\text{clear}(X), T) \land \text{holds}_\text{at}(\text{clear}(Y), T) \land \text{holds}_\text{at}(\text{on}(X, Z), T) \land X \neq Y \land Y \neq Z
\]

\[
\text{terminates}(\text{move}(F, X, Y), \text{on}(X, Z), T) \leftarrow \\
\text{holds}_\text{at}(\text{clear}(X), T) \land \text{holds}_\text{at}(\text{clear}(Y), T) \land \text{holds}_\text{at}(\text{on}(X, Z), T) \land X \neq Y \land Y \neq Z
\]

\[
\text{terminates}(\text{move}(F, X, Y), \text{clear}(Y), T) \leftarrow \\
\text{holds}_\text{at}(\text{clear}(X), T) \land \text{holds}_\text{at}(\text{clear}(Y), T) \land X \neq Y
\]
Cancellation

cancels\( (E_1, E_2, T) \) \iff\ event \( E_1 \) cancels the effects of event \( E_2 \) at time \( T \)

cancels\( (\text{move}(F_1, X_1, Y_1), \text{move}(F_2, X_2, Y_2), T) \) \iff
\[
F_1 \neq F_2 \land (X_1 = X_2 \lor Y_1 = Y_2) \lor
F_1 = F_2 \land (X_1 \neq X_2 \lor Y_1 \neq Y_2)
\]

holds\_at\( (F, T) \) \iff\ initially\( (F) \land \neg\text{clipped}(0, F, T) \)

holds\_at\( (F, T_2) \) \iff
\[
(\exists E) \ \text{happens}(E, T_1) \land \text{initiates}(E, F, T_1) \land T_1 < T_2 \land \neg\text{clipped}(T_1, F, T_2) \land
\neg(\exists E') [\ \text{happens}(E', T_1) \land \text{cancels}(E', E, T_1)]
\]

clipped\( (T_1, F, T_2) \) \iff
\[
(\exists E, T) \ \text{happens}(E, T) \land \text{terminates}(E, F, T) \land T_1 < T \land T < T_2 \land
\neg(\exists E') [\ \text{happens}(E', T) \land \text{cancels}(E', E, T)]
\]
Cancellation (continued)

\[ \neg \text{holds\_at}(F, T_2) \leftarrow \]
\[ (\exists E) \ \text{happens}(E, T_1) \land \text{terminates}(E, F, T_1) \land T_1 < T_2 \land \neg \text{declipped}(T_1, F, T_2) \land \]
\[ \neg (\exists E') [ \ \text{happens}(E', T_1) \land \text{cancels}(E', E, T_1) ] \]

\[ \text{declipped}(T_1, F, T_2) \leftrightarrow \]
\[ (\exists E, T) \ \text{happens}(E, T) \land \text{initiates}(E, F, T) \land T_1 < T \land T < T_2 \land \]
\[ \neg (\exists E') [ \ \text{happens}(E', T) \land \text{cancels}(E', E, T) ] \]
Collaboration

The concurrent events,

\[ \text{happens}(\text{move}(\text{fork1}, b, \text{port1}), 1) \land \text{happens}(\text{move}(\text{fork2}, b, \text{port1}), 1) \]

shall now be possible, resulting in this state:
A New Event Type

\[ \text{happens}(E_1 \& E_2, T) \iff \text{events } E_1 \text{ and } E_2 \text{ happen concurrently at time } T. \]

\[ \text{happens}(E_1 \& E_2, T) \leftarrow \text{happens}(E_1, T) \land \text{happens}(E_2, T) \land E_1 \neq E_2 \]

\[ \text{initiates}(\text{move}(F_1, X, Y) \& \text{move}(F_2, X, Y), \text{on}(X, Y), T) \leftarrow \]
\[ (\exists U) \text{holds}_{\text{at}}(\text{on}(U, X), T) \land \text{holds}_{\text{at}}(\text{clear}(U), T) \land \text{holds}_{\text{at}}(\text{clear}(Y), T) \land U \neq Y \]

\[ \text{initiates}(\text{move}(F_1, X, Y) \& \text{move}(F_2, X, Y), \text{clear}(Z), T) \leftarrow \]
\[ (\exists U) \text{holds}_{\text{at}}(\text{on}(U, X), T) \land \text{holds}_{\text{at}}(\text{clear}(U), T) \land \]
\[ \text{holds}_{\text{at}}(\text{clear}(Y), T) \land \text{holds}_{\text{at}}(\text{on}(X, Z), T) \land U \neq Y \land Y \neq Z \]

\[ \text{terminates}(\text{move}(F_1, X, Y) \& \text{move}(F_2, X, Y), \text{on}(X, Z), T) \leftarrow \]
\[ (\exists U) \text{holds}_{\text{at}}(\text{on}(U, X), T) \land \text{holds}_{\text{at}}(\text{clear}(U), T) \land \]
\[ \text{holds}_{\text{at}}(\text{clear}(Y), T) \land \text{holds}_{\text{at}}(\text{on}(X, Z), T) \land U \neq Y \land Y \neq Z \]

\[ \text{terminates}(\text{move}(F_1, X, Y) \& \text{move}(F_2, X, Y), \text{clear}(Y), T) \leftarrow \]
\[ (\exists U) \text{holds}_{\text{at}}(\text{on}(U, X), T) \land \text{holds}_{\text{at}}(\text{clear}(U), T) \land \]
\[ \text{holds}_{\text{at}}(\text{clear}(Y), T) \land U \neq Y \]
The General Approach

Given are

- conjunction of *initiates*, *terminates*, and *cancels* formulas \( E \)
- conjunction of *initially* formulas \( I \)
- conjunction of *happens* formulas \( N \)
- unique names assumptions \( U \)
- foundational axioms \( EC \)

The intended meaning is given by the formula

\[
\text{CIRC}[N \land I; \text{happens}] \land \text{CIRC}[E; \text{initiates, terminates, cancels}] \land U \land EC
\]
The extended Event Calculus in PROLOG (1)

:- op(600, yfx, &).

holds_at(F, T) :- initially(F), not clipped(0, F, T).

holds_at(F, T2) :- happens(E, T1), T1<T2, initiates(E, F, T1),
               not cancelled(E, T1), not clipped(T1, F, T2).

clipped(T1, F, T2) :- happens(E, T), T1<T, T<T2, terminates(E, F, T),
                not cancelled(E, T).

cancelled(E, T) :- happens(E1, T), cancels(E1, E).

cancels(move(F1, X1, Y1), move(F2, X2, Y2)) :-
    not F1=F2, (X1=X2 ; Y1=Y2);
    F1=F2, (not X1=X2 ; not Y1=Y2).
The extended Event Calculus in PROLOG (2)

\[
\text{initiates}(\text{move}(F,X,Y), \text{on}(X,Y), T) :-
\hspace{1em}
\text{holds_at}(\text{clear}(X), T), \text{holds_at}(\text{clear}(Y), T), \text{not } X=\text{Y}.
\]

\[
\text{initiates}(\text{move}(F,X,Y), \text{clear}(Z), T) :-
\hspace{1em}
\text{holds_at}(\text{clear}(X), T), \text{holds_at}(\text{clear}(Y), T),
\hspace{1em}
\text{holds_at}(\text{on}(X,Z), T), \text{not } X=\text{Y}, \text{not } Y=\text{Z}.
\]

\[
\text{terminates}(\text{move}(F,X,Y), \text{on}(X,Z), T) :-
\hspace{1em}
\text{holds_at}(\text{clear}(X), T), \text{holds_at}(\text{clear}(Y), T),
\hspace{1em}
\text{holds_at}(\text{on}(X,Z), T), \text{not } X=\text{Y}, \text{not } Y=\text{Z}.
\]

\[
\text{terminates}(\text{move}(F,X,Y), \text{clear}(Y), T) :-
\hspace{1em}
\text{holds_at}(\text{clear}(X), T), \text{holds_at}(\text{clear}(Y), T), \text{not } X=\text{Y}.
\]
The extended Event Calculus in PROLOG (3)

\[
\text{happens}(E1 & E2, T) :- E1 = \text{move}(_,-,-), E2 = \text{move}(_,-,-), \\
\quad \text{happens}(E1, T), \text{happens}(E2, T), \text{not } E1=E2.
\]

\[
\text{initiates}(\text{move}(F1,X,Y) & \text{move}(F2,X,Y), \text{on}(X,Y), T) :- \\
\quad \text{holds\_at}(\text{on}(U,X), T), \text{holds\_at}(\text{clear}(U), T), \\
\quad \text{holds\_at}(\text{clear}(Y), T), \text{not } U=Y.
\]

\[
\text{initiates}(\text{move}(F1,X,Y) & \text{move}(F2,X,Y), \text{clear}(Z), T) :- \\
\quad \text{holds\_at}(\text{on}(U,X), T), \text{holds\_at}(\text{clear}(U), T), \\
\quad \text{holds\_at}(\text{clear}(Y), T), \text{holds\_at}(\text{on}(X,Z), T), \text{not } U=Y, \text{not } Y=Z.
\]

\[
\text{terminates}(\text{move}(F1,X,Y) & \text{move}(F2,X,Y), \text{on}(X,Z), T) :- \\
\quad \text{holds\_at}(\text{on}(U,X), T), \text{holds\_at}(\text{clear}(U), T), \\
\quad \text{holds\_at}(\text{clear}(Y), T), \text{holds\_at}(\text{on}(X,Z), T), \text{not } U=Y, \text{not } Y=Z.
\]

\[
\text{terminates}(\text{move}(F1,X,Y) & \text{move}(F2,X,Y), \text{clear}(Y), T) :- \\
\quad \text{holds\_at}(\text{on}(U,X), T), \text{holds\_at}(\text{clear}(U), T), \\
\quad \text{holds\_at}(\text{clear}(Y), T), \text{not } U=Y.
\]
The extended Event Calculus in PROLOG (4)

Initially:

\[
\begin{align*}
\text{initially(on}(a, \text{track1})) & \quad \text{initially(clear}(c)) \\
\text{initially(on}(b, \text{track2})) & \quad \text{initially(on}(c,b)) \\
\text{initially(clear}(c)) & \quad \text{initially(clear}(\text{temp}_{-}\text{store})) \\
\text{initially(clear}(\text{port1})) & \quad \text{initially(on}(d, \text{port2})) \\
\text{initially(clear}(d)) & \\
\text{happens(move}(\text{fork1}, a, \text{port1}), 1) & \quad \text{happens(move}(\text{fork2}, d, \text{temp}_{-}\text{store}), 1) \\
\text{happens(move}(\text{fork1}, b, \text{port2}), 3) & \quad \text{happens(move}(\text{fork2}, b, \text{port2}), 3) \\
\text{happens(move}(\text{fork1}, d, \text{track2}), 6) & \quad \text{happens(move}(\text{fork2}, d, \text{temp}_{-}\text{store}), 6)
\end{align*}
\]

\[
\begin{align*}
\text{F = clear}(a) & \quad \text{F = on}(c,b) \\
\text{F = clear}(c) & \quad \text{F = clear}(d) \\
\text{F = on}(a, \text{port1}) & \quad \text{F = clear}(\text{track1}) \\
\text{F = on}(d, \text{track2}) & \quad \text{F = clear}(\text{temp}_{-}\text{store}) \\
\text{F = on}(b, \text{port2}) & \\
\end{align*}
\]
Extending the Situation Calculus

▷ New sort \texttt{SET\_OF\_ACTION} :\iff sets of actions

▷ \texttt{do} : \texttt{SET\_OF\_ACTION} \times \texttt{SIT} \mapsto \texttt{SIT}
   
   \texttt{poss} : \texttt{SET\_OF\_ACTION} \times \texttt{SIT}

▷ Standard set functions and relations \{\}, \{a_1, \ldots, a_n\}, and \in.
Two Fork Lift Trucks: Precondition Axioms

\[\text{poss}(C, S) \leftrightarrow\]
\[\exists F, X, Y \left[ C = \{\text{move}(F, X, Y)\} \land \text{holds}(\text{clear}(X), S) \land \text{holds}(\text{clear}(Y), S)\right]\]
\[\lor\]
\[\exists F_1, F_2, X_1, X_2, Y_1, Y_2\]
\[\left[ C = \{\text{move}(F_1, X_1, Y_1), \text{move}(F_2, X_2, Y_2)\} \land F_1 \neq F_2 \land\right]\]
\[\left( X_1 \neq X_2 \rightarrow Y_1 \neq Y_2 \land \text{holds}(\text{clear}(X_1), S) \land \text{holds}(\text{clear}(Y_1), S) \land \right.\]
\[\text{holds}(\text{clear}(X_2), S) \land \text{holds}(\text{clear}(Y_2), S)\right)\]
\[\left( X_1 = X_2 \rightarrow Y_1 = Y_2 \land \exists Z \text{ holds}(\text{on}(Z, X_1), S) \land \text{holds}(\text{clear}(Z), S) \land \right.\]
\[\text{holds}(\text{clear}(Y_1), S)\right]\]
Two Fork Lift Trucks: Effect Axioms

\[
\text{holds}(on(X, Y), do(C, S)) \leftarrow \\
\text{poss}(C, S) \land (\exists F) \text{move}(F, X, Y) \in C
\]

\[
\neg\text{holds}(on(X, Y), do(C, S)) \leftarrow \\
\text{poss}(C, S) \land (\exists F, Z)(\text{move}(F, X, Z) \in C \land Y \neq Z)
\]

\[
\text{holds}(\text{clear}(X), do(C, S)) \leftarrow \\
\text{poss}(C, S) \land (\exists F, Y, Z)(\text{move}(F, Y, Z) \in C \land \text{holds}(on(Y, X), S) \land X \neq Z)
\]

\[
\neg\text{holds}(\text{clear}(X), do(C, S)) \leftarrow \\
\text{poss}(C, S) \land (\exists F, Y)\text{move}(F, Y, X) \in C
\]
Two Fork Lift Trucks in PROLOG (1)

executable(s0).
executable(do(A,S)) :- executable(S), poss(A,S).

poss(move(_,X,Y),S) :- holds(clear(X),S), holds(clear(Y), S).

poss(conc(C),S) :-
    C = [move(F,X,Y)], poss(move(F,X,Y), S)
    ;
    C = [move(F1,X1,Y1), move(F2,X2,Y2)], not F1=F2,
    ( not X1=X2, not Y1=Y2, poss(move(F1,X1,Y1), S), poss(move(F2,X2,Y2), S) 
    ;
    X1=X2, Y1=Y2,
    holds(on(Z,X1), S), holds(clear(Z), S), holds(clear(Y1), S)
    ).

[135] Cognitive Robotics: Concurrent Actions
Two Fork Lift Trucks in PROLOG (2)

holds(on(X,Y), do(conc(C),S)) :-
  member(move(_, X, Y), C)
;
  holds(on(X,Y), S), not ( member(move(F,X,Z), C), not Y=Z ).

holds(clear(X), do(conc(C),S)) :-
  member(move(_,Y,Z), C), holds(on(Y,X), S), not X=Z
;
  holds(clear(X), S), not member(move(_,_,X), C).
An Example Scenario

holds(clear(track1), s0).
holds(clear(d), s0).
holds(on(a,port1), s0).
holds(on(c,b), s0).
holds(clear(c), s0).

holds(on(d,track2), s0).
holds(clear(temp_store), s0).
holds(clear(a), s0).
holds(on(b,port2), s0).

| ?- S2=do(conc([move(fork1,a,track1),move(fork2,a,track1)]),
do(conc([move(fork1,c,a)]),
s0))),
executable(S2), holds_at(on(X,Y),S2).

X = a, Y = track1
X = c, Y = a
X = d, Y = track2
X = b, Y = port2
Extending the Fluent Calculus

▷ New sort \textbf{CONCURRENT} \supset \text{ACTION}

▷ \( \epsilon : \text{CONCURRENT} \)

▷ \( \circ : \text{CONCURRENT} \times \text{CONCURRENT} \leftrightarrow \text{CONCURRENT} \)

▷ \textbf{EUNA} for \( \circ; \epsilon \) (wrt. \text{CONCURRENT})

▷ Foundational axioms \( \mathcal{F}_{co} : \)

\[
(\forall A : \text{ACTION}, C : \text{CONCURRENT}, S : \text{SIT}) \quad \neg \text{poss}(A \circ A \circ C, S) \\
(\forall S : \text{SIT}) \quad \text{poss}(\epsilon, S) \land \text{state}(\text{do}(\epsilon, S)) = \text{state}(S)
\]

▷ \textbf{cancels}(C, C_1, S) \iff \text{concurrent actions} \ C \ \text{cancels concurrent actions} \ C_1 \ \text{in situation} \ S

▷ State update axioms \( \mathcal{F}_{sua} \) (only deterministic actions, only direct effects):

\[
\text{poss}(C_1 \circ C, S) \land \neg \text{cancels}(C, C_1, S) \land \Delta(S) \\
\rightarrow \text{state}(\text{do}(C_1 \circ C, S)) \circ \vartheta^- = \text{state}(\text{do}(C, S)) \circ \vartheta^+
\]

▷ Macro: \( \text{holds-in}(C_1, C) \overset{\text{def}}{=} (\exists C') \quad C = C_1 \circ C' \)
Two Fork Lift Trucks: Precondition Axiom

\[
\text{poss}(C, S) \leftrightarrow \\
(\forall F, X, Y) \\
[ \text{holds\_in}(\text{move}(F, X, Y), C) \rightarrow \\
\neg(\exists X_1, Y_1) \text{ holds\_in}(\text{move}(F, X_1, Y_1), C) \land (X_1 \neq X \lor Y_1 \neq Y) \\
\land \\
\text{holds}(\text{clear}(Y), S) \\
\land \\
[ \neg(\exists F_1, X_1, Y_1) ( \text{ holds\_in}(\text{move}(F_1, X_1, Y_1), C) \land F_1 \neq F \land (X_1 = X \lor Y_1 = Y) ) \\
\rightarrow \text{holds}(\text{clear}(X), S) ] \\
\land \\
[ (\exists F_1, X_1, Y_1) ( \text{ holds\_in}(\text{move}(F_1, X_1, Y_1), C) \land F_1 \neq F \land (X_1 = X \lor Y_1 = Y) ) \\
\rightarrow X_1 = X \land Y_1 = Y \land (\exists Z) \text{ holds}(\text{on}(Z, X), S) \land \text{holds}(\text{clear}(Z), S) ]
\]

[139] Cognitive Robotics: Concurrent Actions
Two Fork Lift Trucks: State Update Axioms

\[(\exists F_1, X_1, Y_1)\ (\text{holds\_in}(\text{move}(F_1, X_1, Y_1), C) \land F_1 \neq F \land X_1 = X \land Y_1 = Y) \rightarrow \text{cancels}(C, \text{move}(F, X, Y), S)\]

\[\text{poss}(\text{move}(F, X, Y) \circ C, S) \land \neg\text{cancels}(C, \text{move}(F, X, Y), S) \land \text{holds}(\text{on}(X, Z), S) \rightarrow \text{state}(\text{do}(\text{move}(F, X, Y) \circ C, S)) \circ \text{on}(X, Z) \circ \text{clear}(Y) = \text{state}(\text{do}(C, S)) \circ \text{on}(X, Y) \circ \text{clear}(Z)\]

\[\text{poss}(\text{move}(F_1, X, Y) \circ \text{move}(F_2, X, Y) \circ C, S) \land \text{holds}(\text{on}(X, Z), S) \rightarrow \text{state}(\text{do}(\text{move}(F_1, X, Y) \circ \text{move}(F_1, X, Y) \circ C, S)) \circ \text{on}(X, Z) \circ \text{clear}(Y) = \text{state}(\text{do}(C, S)) \circ \text{on}(X, Y) \circ \text{clear}(Z)\]
Literature

▷ M. Shanahan: Solving the Frame Problem: A Mathematical Investigation of the Common

▷ R. Reiter: Natural actions, concurrency and continuous time in the Situation Calculus. In:
L. C. Aiello and J. Doyle and S. Shapiro (ed.’s), Proceedings of the International Conference
on Principles of Knowledge Representation and Reasoning, pp. 2–13. Morgan Kaufmann
1996.

Continuous Change

- Process Fluents
- Continuous change in the Situation Calculus
- Planning with incomplete initial knowledge & Zeno’s paradox
- Continuous change in the Fluent Calculus
- Continuous change in the Event Calculus
Process Fluents

▶ Idea: A fluent, which itself is stable, represents a process of continuous change.

\[
\text{holds}(\text{movement}(X, \vec{P}_0, \vec{V}, T_0), S) \\
\quad \iff \\
\text{In situation } S, \text{ object } X \text{ moves with constant (spatial) velocity } \vec{V}. \\
\text{The object started at time } T_0 \text{ at position } \vec{P}_0.
\]

(Other examples of process fluents are: acceleration, water flow, heating, . . .)

▶ Actions may disturb processes by causing (process) fluents to become true and false, resp., as usual.
A Detail of a Production Line (1)

\[ holds(has(X), S) \iff \text{the robot is in possession of } X \text{ in situation } S \]

\[ put(X, T) \iff \text{the action of putting } X \text{ onto } belt_1 \text{ at time } T \]

\[ vel(belt_1) = (1, 0) \]

\[ holds(has(a), s_0) \land holds(has(b), s_0) \land \neg(\exists) holds(movement(X, P_0, V, T_0), s_0) \]
Precondition and Effect Axioms

The precondition of \( \text{put} \) is to have the object ready and that no other object happens to be at location \((0,0)\):

\[
\text{poss} (\text{put}(X, T), S) \iff \\
\text{holds}(\text{has}(X), S) \land \\
\neg (\exists Y, P_0, V, T_0) [\text{holds}(\text{movement}(Y, P_0, V, T_0), S) \land P_0 + V \cdot (T - T_0) = (0,0)]
\]

The effect of \( \text{put} \) is to lose possession of the object and to initiate a certain movement:

\[
\text{poss}(\text{put}(X, T), S) \rightarrow \neg\text{holds}(\text{has}(X), \text{do}(\text{put}(X, T), S)) \\
\text{poss}(\text{put}(X, T), S) \rightarrow \text{holds}(\text{movement}(X, (0,0), \text{vel}(\text{belt}_1), T), \text{do}(\text{put}(X, T), S))
\]
A Detail of a Production Line (2)

\[ \text{vel}(belt_1) = (1, 0) \land \text{vel}(belt_2) = (0, -0.5) \]

\[ \text{holds}(\text{has}(a), s_0) \land \text{holds}(\text{has}(b), s_0) \land \]

\[ (\forall X, P_0, V, T_0) \left[ \text{holds}(\text{movement}(X, P_0, V, T_0), s_0) \leftrightarrow \right. \]

\[ X = o_1 \land P_0 = (7, 6) \land V = \text{vel}(belt_2) \land T_0 = 0 \lor \]

\[ X = o_2 \land P_0 = (7, 4) \land V = \text{vel}(belt_2) \land T_0 = 0 \]
A Natural Action and its Specification

\[ \text{falls}(X, T) \iff \text{the natural action of } X \text{ falling down onto } belt_1 \]

The precondition of \( \text{falls} \) is that the object reaches the end of the belt:

\[
\text{poss}(\text{falls}(X, T), S) \iff (\exists P_y, T_0) \left[ \text{holds}(\text{movement}(X, (7, P_y), \text{vel}(belt_2), T_0), S) \land (7, P_y) + \text{vel}(belt_2) \cdot (T - T_0) = (7, 0) \right]
\]

The effect of \( \text{falls} \) is to terminate the current movement and to initiate a new one:

\[
\text{poss}(\text{falls}(X, T), S) \rightarrow \neg \text{holds}(\text{movement}(X, P_0, \text{vel}(belt_2), T_0), \text{do}(\text{falls}(X, T), S))
\]
\[
\text{poss}(\text{falls}(X, T), S) \rightarrow \text{holds}(\text{movement}(X, (7, 0), \text{vel}(belt_1), T), \text{do}(\text{falls}(X, T), S))
\]
The Successor State Axioms

\[\text{poss}(A, S) \rightarrow\]
\[\text{holds}(\text{has}(X), \text{do}(A, S)) \leftrightarrow\]
\[\text{holds}(\text{has}(X), S) \land \neg(\exists T) A = \text{put}(X, T)\]

\[\text{poss}(A, S) \rightarrow\]
\[\text{holds}(\text{movement}(X, P_0, V, T_0), \text{do}(A, S)) \leftrightarrow\]
\[A = \text{put}(X, T_0) \land P_0 = (0, 0) \land V = \text{vel}(belt_1)\]
\[\lor\]
\[A = \text{falls}(X, T_0) \land P_0 = (7, 0) \land V = \text{vel}(belt_1)\]
\[\lor\]
\[\text{holds}(\text{movement}(X, P_0, V, T_0), S) \land \neg(\exists T) [A = \text{falls}(X, T) \land V = \text{vel}(belt_2)]\]
Why Natural Actions Require Special Treatment

\[
\begin{align*}
\text{holds}(\text{location}(\text{achilles}, (7, 2)), s_0) & \land \\
\text{holds}(\text{location}(\text{ajax}, (7, -3)), s_0) & \land \\
\text{holds}(\text{movement}(\text{o}_1, (7, 6), (0, -0.5), 0), s_0)
\end{align*}
\]

\[
\begin{align*}
\text{poss}(\text{grab}(R, X, T), S) & \iff \\
\text{holds}(\text{location}(R, P), S) & \land \text{holds}(\text{movement}(X, P_0, V, T_0), S) & \land P = P_0 + V \cdot (T - T_0)
\end{align*}
\]

\[
\models \text{poss}(\text{grab}(\text{achilles}, \text{o}_1, 8), s_0) & \land \text{poss}(\text{falls}(\text{o}_1, 12), s_0) & \land \text{poss}(\text{grab}(\text{ajax}, \text{o}_1, 18), s_0)
\]

[149] Cognitive Robotics: Continuous Change
The General Approach (1)

predicate $\text{natural}(A) : \iff A$ is a natural action

predicate $\text{legal}(S) : \iff \text{situation } S \text{ respects the property of natural actions}$

that they must occur at their predicted times,

provided no earlier actions prevent them from occurring

function $\text{start}(S) : \iff \text{start time of situation } S$

function $\text{time}(A) : \iff \text{time of action } A$

The sort $\text{TIMEPOINT}$ ranges over the real numbers.

We assume a standard interpretation of the reals and their usual operations.

▽ Execution time for each action $A(X, T)$:

\[ \text{time}(A(X, T)) = T \]

▽ Start time of a situation (axiom $F_{\text{start}}$):

\[ \text{start}(s_0) = 0 \land \text{start}(\text{do}(A, S)) = \text{time}(A) \]
The General Approach (2)

- Natural action condition:
  \[
  \text{natural}(A) \leftrightarrow \Psi(A)
  \]
  where \(\Psi\) is a formula with free variable \(A\).

Example: \(\text{natural}(A) \leftrightarrow (\exists X, T) A = \text{falls}(X, T)\)

- Foundational axioms \(\mathcal{F}_{\text{legal}}\) for legal situations:
  \[
  \begin{align*}
  \text{legal}(s_0) \land \\
  \text{legal}(do(A, S')) & \leftrightarrow \text{legal}(S) \land \text{poss}(A, S) \land \text{start}(S) \leq \text{time}(A) \land \\
  (\forall A') [ \text{natural}(A') \land \text{poss}(A', S) & \rightarrow A = A' \lor \text{time}(A) < \text{time}(A')] 
  \end{align*}
  \]

- Plan synthesis: If \(\mathcal{F}\) is the axiomatization of an application domain along with a specification of an initial situation, then \(s\) is a solution to the planning problem of achieving \(g\) iff
  \[
  \mathcal{F} \models g(s) \land \text{legal}(s)
  \]

[151] Cognitive Robotics: Continuous Change
The Conveyor Belt Robot in ECLiPSE (1)

:- lib(clpr).

start(s0, 0).
start(do(A, _), T) :- time(A, T).

legal(s0).
legal(do(A, S)) :-
    legal(S), poss(A, S), time(A, Ta), start(S, Ts), \{ Ts <= Ta \},
    not ( natural(A1), poss(A1, S), not A=A1, time(A1, Ta1), \{ Ta1 <= Ta \} ).

time(put(_, T), T).
time(falls(_, T), T).

natural(falls(_, _)).

vel(belt1, [1.0, 0.0]).
vel(belt2, [0.0, -0.5]).
The Conveyor Belt Robot in ECLiPSE (2)

\[
\text{poss(put}(X,T), S) :\neg \text{ holds(has}(X), S), \not\text{ occupied}([0,0], T, S).
\]

\[
\text{occupied}([Px,Py], T, S) :\neg
\begin{align*}
\text{holds(movement}_\text{,} &[Pxo,Pyo],[Vx,Vy],T0), S), \\
\&\{ Pxo + Vx*(T-T0) = Px, Pyo + Vy*(T-T0) = Py \}.
\end{align*}
\]

\[
\text{poss(falls}(X,T), S) :\neg
\begin{align*}
\text{holds(movement}(X,[7.0,Py],[Vx,Vy],T0), S), \\
\&\text{vel(belt2, [Vx,Vy])}, \\
\&\{ Py + Vy*(T-T0) = 0.0 \}.
\end{align*}
\]

\[
\text{holds(has}(X), \text{do}(A,S)) :\neg
\begin{align*}
\text{holds(has}(X), S), \not\text{ A=put}(X,\_).
\end{align*}
\]

\[
\text{holds(movement}(X,P0,V,T0), \text{do}(A,S)) :\neg
\begin{align*}
\text{A=put}(X,T0), P0=[0.0,0.0], \text{vel}(belt1, V); \\
\text{A=falls}(X,T0), P0=[7.0,0.0], \text{vel}(belt1, V); \\
\text{holds(movement}(X,P0,V,T0), S), \not( A=falls}(X,\_), \text{vel}(belt2, V) ).
\end{align*}
\]
The Conveyor Belt Robot in ECLiPSE (3)

```
holds(has(a), s0).
holds(has(b), s0).
holds(movement(o1,[7.0,6.0],V,0.0), s0) :- vel(belt2, V).
holds(movement(o2,[7.0,4.0],V,0.0), s0) :- vel(belt2, V).
```

```
[eclipse 2]: poss(A, s0).
A = put(a, _);
A = put(b, _);
A = falls(o1, 12.0);
A = falls(o2, 8.0)
```

```
[eclipse 3]: S = do(falls(o1,12.0), do(falls(o2,8.0), do(put(a,5.0),
    do(put(b,1.0), s0)))))}, legal(S), holds(F, S).
F = movement(o1, [7.0, 0.0], [1.0, 0.0], 12.0);
F = movement(o2, [7.0, 0.0], [1.0, 0.0], 8.0);
F = movement(a, [0.0, 0.0], [1.0, 0.0], 5.0);
F = movement(b, [0.0, 0.0], [1.0, 0.0], 1.0)
```
Planning with Incomplete Initial Knowledge

\[
\text{holds}(\text{has}(a), s_0) \land \text{holds}(\text{movement}(o_1, (7, 6), \text{vel}(belt_2), 0), s_0) \land \\
(\exists U)(\forall X, P_0, V, T_0)[ \text{holds}(\text{movement}(X, P_0, V, T_0), s_0) \rightarrow \\
X = o_1 \land P_0 = (7, 6) \land V = \text{vel}(belt_2) \land T_0 = 0 \lor \\
X = U \land P_0 = (7, 2) \land V = \text{vel}(belt_2) \land T_0 = 0]
\]

The planning problem of getting \( o_1 \) into \( a \) has no solution since there is no provably legal situation which includes the action \( \text{put}(a, 5) \).
Zeno's Paradox

If infinitely many natural actions happen in a finite time interval, then no legal situation exists beyond that interval.

Let $\mathcal{F}$ consist of the formulas

\[
\begin{align*}
\text{natural}(A) & \leftrightarrow (\exists T) A = a(T) \\
\text{time}(a(T)) & = T \\
\text{poss}(a(T)) & \leftrightarrow (\exists N : \mathbb{N}) T = 1 - 2^{-N}
\end{align*}
\]

along with the foundational axioms $\mathcal{F}_{\text{start}}$ and $\mathcal{F}_{\text{legal}}$. Then,

$\mathcal{F} \models \text{legal}(s_0) \land \text{legal}(\text{do}(a(1/2), s_0)) \land \text{legal}(\text{do}(a(3/4), \text{do}(a(1/2), s_0))) \land \ldots$

but there is no $S$ such that

$\mathcal{F} \models \text{legal}(S) \land \text{start}(S) > 1$

[156] Cognitive Robotics: Continuous Change
Trajectories in the Fluent Calculus

▷ Idea: Each situation has its own trajectory, which describes how the state evolves according to the expected natural actions.

\[ t_1 < \text{time}(a_1) = t'_0 = \text{start}(s_1) < t_2 \]

[157] Cognitive Robotics: Continuous Change
The General Approach (1)

- We adopt sort `TIMEPOINT`, predicate `natural`, and function `time`.
- Fluent `start_time(T)` denotes the start time of the state in which it holds true.
- `succ : ACTION \times STATE \mapsto STATE` defines the successor state after a natural action.
- `expect(A, Z) :⇔` natural action `A` is expected to happen in state `Z`
  if no earlier natural action prevents this.
- `trajectory(Z, Z') :⇔` state `Z'` lies on the trajectory rooted in state `Z`
- State constraints on start times `F_{st}`:
  \[ (\exists T) \text{holds}_\text{in}(\text{start}_\text{time}(T), \text{state}(S)) \]
  \[ \text{holds}_\text{in}(\text{start}_\text{time}(T_1), \text{state}(S)) \land \text{holds}_\text{in}(\text{start}_\text{time}(T_2), \text{state}(S)) \rightarrow T_1 = T_2 \]
- Foundational axioms `F_{traj}`:
  \[ \text{trajectory}(Z, Z) \]
  \[ \text{trajectory}(Z, Z') \land \text{next}_\text{nat}_\text{action}(A, Z') \rightarrow \text{trajectory}(Z, \text{succ}(A, Z')) \]
  where `next_natural_action(A, Z)` abbreviates the formula
  \[ \text{expect}(A, Z) \land (\forall A')[ \text{natural}(A') \land \text{expect}(A', Z) \land \text{time}(A') \leq \text{time}(A) \rightarrow A = A'] \]
Useful Macros for Precondition and Effect Specification

$\text{after}(T, Z) \overset{\text{def}}{=} (\forall T_0) \left[ \text{holds\_in}(\text{start\_time}(T_0), Z) \rightarrow T > T_0 \right]$

$\text{after}(T, S) \overset{\text{def}}{=} \text{after}(T, \text{state}(S))$

$\text{actual\_state}(S, T, Z) \overset{\text{def}}{=} (\forall T_0) \left[ \text{holds\_in}(\text{start\_time}(T_0), Z) \rightarrow 
\begin{array}{l}
T_0 \leq T \land \text{trajectory}(\text{state}(S), Z) \land \\
(\forall A) \left( \text{next\_nat\_action}(A, Z) \rightarrow \text{time}(A) > T \right) \right]$

$\text{holds}(F, S, T) \overset{\text{def}}{=} (\forall Z) \left[ \text{actual\_state}(S, T, Z) \rightarrow \text{holds\_in}(F, Z) \right]$
General Form of Precondition and Effect Specifications

- If $a(\overline{X}, T)$ is a deliberative action, then the precondition axiom takes this form:

$$\text{poss}(a(\overline{X}, T), S) \iff \Phi_a(\overline{X}, S) \land \text{after}(T, S)$$

and the state update axioms take this form:

$$\text{poss}(a(\overline{X}, T), S) \rightarrow \text{actual}_\text{state}(S, T, Z) \rightarrow \Delta(Z) \rightarrow$$

$$(\exists T_0) \text{state}(\text{do}(a(\overline{X}, T), S)) \circ \vartheta^- \circ \text{start}_\text{time}(T_0) = Z \circ \vartheta^+ \circ \text{start}_\text{time}(T)$$

- If $a(\overline{X}, T)$ is a natural action, then the precondition axiom takes this form:

$$\text{expect}(a(\overline{X}, T), Z) \iff \Phi_a(\overline{X}, Z) \land \text{after}(T, Z)$$

and the update axioms take this form:

$$\text{expect}(a(\overline{X}, T), Z) \rightarrow \Delta(Z) \rightarrow$$

$$(\exists T_0) \text{succ}(a(\overline{X}, T), Z) \circ \vartheta^- \circ \text{start}_\text{time}(T_0) = Z \circ \vartheta^+ \circ \text{start}_\text{time}(T)$$
The precondition of \textit{put} is to have the object ready and that no other object happens to be at location \((0, 0)\) and that the action is performed after the arising of the situation:

\[
\text{poss(put}(X, T), S) \leftrightarrow \\
\text{holds(has}(X), S, T) \land \text{after}(T, S) \land \\
\neg(\exists Y, P_0, V, T_0) [\text{holds(movement}(Y, P_0, V, T_0), S, T) \land P_0 + V \cdot (T - T_0) = (0, 0)]
\]

The effect of \textit{put} is to lose possession of the object and to initiate a certain movement in the actual state:

\[
\text{poss(put}(X, T), S) \rightarrow \\
\text{actual\_state}(S, T, Z) \rightarrow \\
(\exists T_0) \text{state(do(put}(X, T), S)) \circ \text{has}(X) \circ \text{start\_time}(T_0) = \\
Z \circ \text{movement}(X, (7, 0), \text{vel}(\text{belt}_1), T) \circ \text{start\_time}(T)
\]
Precondition and Effect in the Production Line Domain (2)

The precondition of \textit{falls} is that the object reaches the end of the belt:

\[
\text{expect}(\text{falls}(X, T), Z) \leftrightarrow \\
(\exists P_y, T_0) \left[ \text{holds\_in}(\text{movement}(X, (7, P_y), \text{vel}(belt_2), T_0), Z) \land \\
(7, P_y) + \text{vel}(belt_2) \cdot (T - T_0) = (7, 0) \land \text{after}(T, Z) \right]
\]

The effect of \textit{falls} is to terminate the current movement and to initiate a new one:

\[
\text{expect}(\text{falls}(X, T), Z) \rightarrow \\
(\exists T_0, T'_0, P_0, V) \left[ \text{succ}(\text{falls}(X, T), Z) \circ \text{movement}(X, P_0, V, T'_0) \circ \text{start\_time}(T_0) = \\
Z \circ \text{movement}(X, (7, 0), \text{vel}(belt_1), T) \circ \text{start\_time}(T') \right]
\]
Planning with Incomplete Initial Knowledge (Revisited)

Consider this initial specification:

\[
(\exists Z) \left[ \text{state}(s_0) = \text{start\_time}(0) \circ \text{has}(a) \circ \text{movement}(o_1, (7, 6), \text{vel(belt}_2), 0) \circ Z \land \\
(\exists U)(\forall) \left[ \text{holds\_in}(\text{movement}(X, P_0, V, T_0), Z) \right) \rightarrow X = U \land P_0 = (7, 2) \land V = \text{vel(belt}_2) \land T_0 = 0 \right]
\]

Then the foundational axioms of the Fluent Calculus with continuous change along with the axioms for the Production Line Domain entail,

\[
(\exists Z) \left[ \text{trajectory}\left(\text{state}(\text{do}(\text{put}(a, 5), s_0)), Z\right) \land \text{holds\_in}(\text{start\_time}(12), Z) \land \\
\text{holds\_in}(\text{movement}(a, (0, 0), (1, 0), 5), Z) \land \text{holds\_in}(\text{movement}(o_1, (7, 0), (1, 0), 12), Z) \right]
\]

\[\text{Model (a)}\]

\[\text{Model (b)}\]
Zeno's Paradox (Revisited)

Let $\mathcal{F}$ consist of the formulas

\begin{align*}
\text{natural}(A) & \leftrightarrow (\exists T) \ A = a(T) \\
\text{time}(a(T)) &= T \\
\text{expect}(a(T), Z) & \leftrightarrow (\exists N : \mathbb{N}) \ T = 1 - 2^{-N} \land \text{after}(T, Z) \\
\text{expect}(a(T), Z) & \rightarrow (\exists T_0) \ \text{succ}(a(T), Z) \circ \text{start\_time}(T_0) = Z \circ \text{start\_time}(T)
\end{align*}

along with the foundational axioms of the Fluent Calculus with continuous change plus this foundational second-order axiom on limits:

\[
(\forall \psi : \mathbb{N} \mapsto \mathbb{R}) \left[ (\forall N : \mathbb{N}) \text{trajectory}(Z, Z' \circ \text{start\_time}(\psi(N))) \rightarrow \text{trajectory}(Z, Z' \circ \text{start\_time}(\lim_{N \to \infty} \psi(N))) \right]
\]

Then,

\[
\text{state}(s_0) = Z \circ \text{start\_time}(0) \rightarrow T \geq 1 \rightarrow \text{actual\_state}(s_0, T, Z \circ \text{start\_time}(1))
\]
Modeling Continuous Change with the Event Calculus

▷ Events can be triggered automatically.

▷ Distinguish the usual discrete fluents (e.g., movement) from continuous fluents, which constantly change as time goes by (e.g., position).

\[
\text{trace}(F_1, T, F_2, D) \iff \text{if the discrete fluent } F_1 \text{ is initiated at time } T, \\
\text{then the continuous fluent } F_2 \text{ holds at time } T + D.
\]

Foundational axioms for continuous fluents:

\[
\text{holds\_at}(F_2, D) \leftarrow \\
(\exists F_1) \text{ initially}(F_1) \land \text{trace}(F_1, 0, F_2, D) \land D > 0 \land \neg \text{clipped}(0, F_1, D)
\]

\[
\text{holds\_at}(F_2, T_2) \leftarrow \\
(\exists E, F_1, T_1, D) \text{ happens}(E, T_1) \land \text{initiates}(E, F, T_1) \land \text{trace}(F_1, T_1, F_2, D) \\
\land D > 0 \land T_2 = T_1 + D \land \neg \text{clipped}(T_1, F_1, T_2)
\]
The Conveyor Belt Robot in the Event Calculus (1)

event \( \text{put}(X) \) :⇔ put \( X \) onto conveyor \( \text{belt}_1 \)

event \( \text{falls}(X) \) :⇔ \( X \) falls from \( \text{belt}_2 \) onto \( \text{belt}_1 \)

discrete fluent \( \text{has}(X) \) :⇔ the robot is in possession of \( X \)

discrete fluent \( \text{movement}(X, V) \) :⇔ \( X \) moves with constant two-dimensional velocity \( V \)

discrete fluent \( \text{position}(X, P) \) :⇔ \( P \) is the current (stable) position of \( X \)

continuous fluent \( \text{position}(X, P) \) :⇔ \( P \) is the current (constantly changing) position of \( X \)

Let \( I \) be,

\[
\begin{align*}
\text{initially}(\text{has}(a)) & \land \text{initially}(\text{has}(b)) \land \\
\text{initially}(\text{position}(a, (0, 0))) & \land \text{initially}(\text{position}(b, (0, 0))) \land \\
\text{initially}(\text{movement}(o_1, \text{vel}(\text{belt}_2))) & \land \text{initially}(\text{movement}(o_2, \text{vel}(\text{belt}_2)))
\end{align*}
\]

Consider, in addition,

\[
\text{holds_at}(\text{position}(o_1, (7, 6)), 0) \land \text{holds_at}(\text{position}(o_2, (7, 4)), 0)
\]

Finally, let \( N \) be,

\[
\text{happens}((\text{put}(a), 1)) \land \text{happens}((\text{put}(b), 5))
\]

[166] Cognitive Robotics: Continuous Change
The Conveyor Belt Robot in the Event Calculus (2)

\[
\text{terminates}(\text{falls}(X), \text{movement}(X, (7, P_y), \text{vel}(\text{belt}_2)), T) \\
\text{initiates}(\text{falls}(X), \text{movement}(X, (7, 0), \text{vel}(\text{belt}_1)), T)
\]

\[
\text{terminates}(\text{put}(X), \text{has}(X), T) \leftarrow \text{holds}\_at(\text{has}(X), T) \\
\text{initiates}(\text{put}(X), \text{movement}(X, \text{vel}(\text{belt}_1)), T) \leftarrow \text{holds}\_at(\text{has}(X), T) \\
\text{terminates}(\text{put}(X), \text{position}(X, (0, 0)), T) \leftarrow \text{holds}\_at(\text{has}(X), T)
\]

\[
\text{trace}(\text{movement}(X, V), T, \text{position}(X, P), D) \leftarrow \text{holds}\_at(\text{position}(X, P_0), T) \land P = P_0 + V \cdot D
\]

\[
\text{happens}(\text{falls}(X), T) \leftarrow \\
\text{holds}\_at(\text{movement}(X, \text{vel}(\text{belt}_2)), T) \land \text{holds}\_at(\text{position}(X, (7, 0)), T)
\]

▷ The very last formula indicates difficulties with implementing in PROLOG the Event Calculus with continuous change, because queries of the form holds\_at(f, t) diverge in straightforward implementations.
Literature


Agent Programming Languages

GOLOG and the Situation Calculus

Complex Actions in the Fluent Calculus

GOLEX: GOLOG and Real Robots

Literature
GOLOG — Repetition

▷ Agent programming language for reasoning about the situations of the world and considering the effects of various possible plans.

▷ \( do(\delta, S, S') \) holds, whenever \( S' \) is a terminating situation of an execution of a complex action \( \delta \) starting in situation \( S \).

▷ Primitive actions:
\[
do(A, S, S') \overset{def}{=} poss(A, S) \land S' = do(A, S).
\]

▷ Test actions:
\[
do(\Phi?, S, S') \overset{def}{=} holds(\Phi, S) \land S = S'.
\]

▷ Sequence:
\[
do([\delta_1; \delta_2], S, S') \overset{def}{=} (\exists S^*)(do(\delta_1, S, S^*) \land do(\delta_2, S^*, S')).
\]
Complex Actions: Nondeterministic Choice

▷ Nondeterministic choice of two actions:

\[ \text{do}([\delta_1 \mid \delta_2], S, S') \overset{\text{def}}{=} \text{do}(\delta_1, S, S') \lor \text{do}(\delta_2, S, S') \]

〜〜 Conditionals:

\[ \text{if } \Phi \text{ then } \delta_1 \text{ else } \delta_2 \text{ endIf } \overset{\text{def}}{=} [\Phi?; \delta_1]\|[\neg\Phi?; \delta_2] \]

\[ \text{if car\textunderscore in\textunderscore driveway then drive else walk endIf} \]

▷ Nondeterministic choice of action arguments:

\[ \text{do}(\pi X \delta(X), S, S') \overset{\text{def}}{=} (\exists X) \text{ do}(\delta(X), S, S') \]

\[ (\pi X) \text{ remove}(X) \]
Complex Actions: Nondeterministic Iteration

▷ Nondeterministic Iteration: Execute δ zero or more times.

\[ do(\delta^*, S, S') \]

The macro is defined by a second order formula (see [Levesque et al, 1997]).

↝ While statements:

\[ \text{while } \Phi \text{ do } \delta \text{ endwhile } \equiv [[\Phi; \delta]^*; \neg \Phi] \]

\[ \text{while (\exists B) ontable(B) do (\pi X) remove(X) endwhile} \]
Complex Actions: Procedures

Procedure calls:

$$do(p(t_1, \ldots, t_n)), S, S') \overset{\text{def}}{=} p(t_1[S], \ldots, t_n[S], S, S')$$

\(\sim\) Call by value

GOLOG programs:

$$do(\{\text{proc } p_1(V_1)\delta_1\text{ endProc}; \ldots; \text{proc } p_n(V_n)\delta_n\text{ endProc}; \delta_0\}, S, S')$$

where \(p_i(V_i)\) is a procedure declaration with formal parameters \(V_i\) and \(\delta_i\) is its body, \(1 \leq i \leq n\), and \(\delta_0\) is the main program, i.e. a complex action.

The macro is defined by a second order formula (see [Levesque et al, 1997]).

Move an elevator down \(N\) floors:

$$\text{proc } d(N) [ (N = 0)? [d(N - 1); \text{ down}] \text{ endProc},$$

where \(\text{ down}\) moves an elevator down one floor.
An Elevator Controller: Primitive Actions and Fluents

▷ Primitive actions:

- $up(N)$ denotes the movement of the elevator up to floor $N$.
- $down(N)$ denotes the movement of the elevator down to floor $N$.
- $turnoff$ denotes the turning off of the call button $N$.
- $open$ denotes the opening of the elevator door.
- $close$ denotes the closing of the elevator door.

▷ Fluents:

- $current_floor(N)$ denotes that the elevator is at floor $N$.
- $on(N)$ denotes that call button $N$ is on.
- $next_floor(N)$ denotes that the next floor to be served is $N$.

▷ Primitive action preconditions:

- $poss(up(N), S) \leftrightarrow (\exists M) [holds(current_floor(M), S) \land M < N]$.
- $poss(down(N), S) \leftrightarrow (\exists M) [holds(current_floor(M), S) \land M > N]$.
- $poss(open, S) \leftrightarrow \top$.
- $poss(close, S) \leftrightarrow \top$.
- $poss(turnoff(N), S) \leftrightarrow holds(on(N), S)$.
An Elevator Controller: Successor State Axioms

\[\text{Successor state axioms:}\]
\[
\text{poss}(A, S) \rightarrow [\text{holds}(\text{current\_floor}(M), \text{do}(A, S)) \leftrightarrow \\
A = \text{up}(M) \lor A = \text{down}(M) \\
\lor \text{holds}(\text{current\_floor}(M), S) \land \neg(\exists N) \ A = \text{up}(N) \land \neg(\exists N) \ A = \text{down}(N)].
\]
\[
\text{poss}(A, S) \rightarrow [\text{holds}(\text{on}(M), \text{do}(A, S)) \leftrightarrow \text{holds}(\text{on}(M), S) \land A \neq \text{turnoff}(M)].
\]

\[\text{A defined fluent:}\]
\[
\text{holds}(\text{next\_floor}(N), S) \leftrightarrow \text{holds}(\text{on}(N), S) \land \\
(\forall M, L) [\text{holds}(\text{on}(M), S) \land \text{holds}(\text{current\_floor}(L), S) \rightarrow |M - L| \geq |N - L|].
\]
An Elevator Controller: The Procedures

▷ The procedures:

\[
\text{proc } \text{serve}(N) \text{ go\_floor}(N)\text{; turnoff}(N)\text{; open; close endProc.} \\
\text{proc } \text{current\_floor}(N)\?up(N)\text{; down}(N) \text{ endProc.} \\
\text{proc } \text{serve\_a\_floor}(\pi N)[\text{next\_floor}(N)\?\text{serve}(N)] \text{ endProc.} \\
\text{proc } \text{control} [\text{while } (\exists N)\text{on}(N) \text{ do } \text{serve\_a\_floor} \text{ endWhile; park endProc.} \\
\text{proc } \text{park} \text{ if } \text{current\_floor}(0) \text{ then } \text{open else } \text{down}(0)\text{; open endif endProc.}
\]

▷ Initial situation:

\[
\text{holds(current\_floor}(4), s_0) \land \text{holds(on}(5), s_0) \land \text{holds(on}(3), s_0)
\]
Reasoning about the Elevator Controller

Let $\mathcal{F} = \mathcal{F}_{ex} \cup \mathcal{F}_{ss} \cup \mathcal{F}_{ap} \cup \mathcal{F}_{uns} \cup \mathcal{F}_{vs}$, then e.g.

$$\mathcal{F} \models (\exists S) \ do(\Pi; \text{control}, s_0, S),$$

where $\Pi$ is the sequence of procedure definitions.

A successful proof might return the substitution

$$S = do(open, do(down(0), do(close, do(open, do(turnoff(5),
  do(up(5), do(close, do(open, do(turnoff(3), do(down(3), s_0))))))))))).$$
Complex Actions are also available in the fluent calculus including
- conditional actions,
- nondeterministic choice of action arguments and
- recursive procedures.

▷ For the details see [Hölldobler, Störr: 99].
GOLEX (1)

- Bridging the gap between cognitive robotics and real robots, in particular, between GOLOG and RHINO.

- Decompose primitive actions specified in GOLOG into a sequence of directives for the low-level robot control system.

```prolog
exec(go(L)) :- position(L, (X, Y)),
             pan_tilt_set_track_point((X, Y)),
             target_message(L, M),
             speech_talk_text(["please follow me to", L]),
             sound_play(horn),
             robot_drive_path([(X, Y)]),
             robot_turn_to_point((X, Y)).
```
GOLEX (2)

▷ Execution monitoring:
  – stops the execution of an action if timed out,
  – verifies that primitive actions have been successively carried out,
  – ensures that the world is consistent with GOLOG’s model
  – etc.

▷ Simple forms of sensing and acting:
  – simple forms of speech,
  – accepts confirmations,
  – accepts simple forms of user input
  – etc.

▷ Applications:
  – Museum tour guide,
  – Coffee delivery agent.
Literature