Foundations of
Cognitive Robotics

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

Contents

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

- ▷ applications:
  - ⇝ high level control of robots and industrial processes
  - ⇝ intelligent software agents
  - ⇝ descrete 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.”

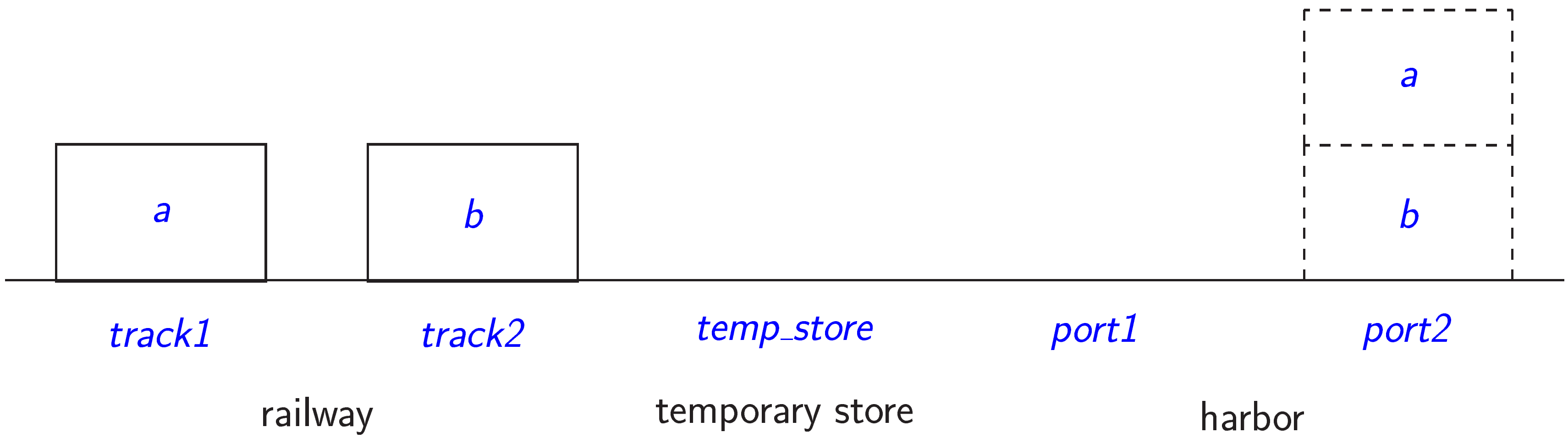
States, Actions and Causality

- ▷ 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.
  - $\rightsquigarrow$ inherent partiality
  - $\rightsquigarrow$ only facts about situations and actions can be specified
  - $\rightsquigarrow$ 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 *port2* .

Example: Fork Lift Truck (1)

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

Initial State: $\xrightarrow{?}$ Goal State:

$$\begin{aligned} I \leftrightarrow\ & \mathit{holds}(\mathit{on}(a, \mathit{track1}), s_o) \\ & \wedge \mathit{holds}(\mathit{on}(b, \mathit{track2}), s_o) \\ & \wedge \mathit{holds}(\mathit{clear}(\mathit{port1}), s_0) \\ & \wedge \mathit{holds}(\mathit{clear}(\mathit{port2}), s_0) \\ & \wedge \mathit{holds}(\mathit{clear}(\mathit{temp\_store}), s_0) \\ & \wedge \mathit{holds}(\mathit{clear}(a), s_0) \\ & \wedge \mathit{holds}(\mathit{clear}(b), s_0) \end{aligned}$$

$$\begin{aligned} & \mathit{holds}(\mathit{on}(a, b), S) \leftrightarrow G \\ & \wedge \mathit{holds}(\mathit{on}(b, \mathit{port2}), S) \\ & \wedge \mathit{holds}(\mathit{clear}(a), S) \\ & \wedge \mathit{holds}(\mathit{clear}(\mathit{track1}), S) \\ & \wedge \mathit{holds}(\mathit{clear}(\mathit{track2}), S) \\ & \wedge \mathit{holds}(\mathit{clear}(\mathit{temp\_store}), S) \\ & \wedge \mathit{holds}(\mathit{clear}(\mathit{port1}), S) \end{aligned}$$

Example: Fork Lift Truck (2)

- How must A be defined such that $\models I \wedge A \rightarrow G$?
- $\mathit{move}(X, Z)$:

$$\begin{aligned}&\mathit{holds}(\mathit{on}(X,Y),S) \wedge \mathit{holds}(\mathit{clear}(X),S) \wedge \mathit{holds}(\mathit{clear}(Z),S) \wedge Z \neq X \\ &\rightarrow\ \mathit{holds}(\mathit{on}(X,Z),S') \wedge \mathit{holds}(\mathit{clear}(Y),S') \\ &\text{where } S' = \mathit{do}(\mathit{move}(X,Z),S)\end{aligned}$$

- Is the stack action sufficient to show $\models I \wedge A \rightarrow G$?

Frame Problem

- Common Assumption: Unless an action explicitly causes a fact to hold or nor to hold the facts are preserved by the action.

  $\leadsto$ philosophical and technical aspects!

- Frame Axioms F:

$$\mathit{holds}(\mathit{on}(Y,X),S) \wedge Y \neq Z \rightarrow \mathit{holds}(\mathit{on}(Y,X),\mathit{do}(\mathit{move}(Z,Z'),S))$$
$$\mathit{holds}(\mathit{clear}(Y),S) \wedge Y \neq Z' \rightarrow \mathit{holds}(\mathit{clear}(Y),\mathit{do}(\mathit{move}(Z,Z'),S))$$

- $\models I \wedge \mathit{move}(X,Y) \wedge F \wedge \mathit{UNA} \rightarrow G$, where 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!

  $\leadsto$ Can we do better?

Qualification, Ramification and Prediction Problem

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

$\leadsto$ *unstack*(X) could also have precondition *not_glued_on*(X).

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

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

▷ Prediction Problem: How long does a fluent hold?

$\leadsto$ How long will an expensive bicyle be standing in front of the office building if it is parked there in the morning?

Positive Effect Axioms

- Positive effect axioms specify the emergence of fluents.
- Two positive effect axioms for the fluent *broken*:

$$\begin{aligned} \mathit{holds}(\mathit{holding}(R,X),S) \land Y = X \land \mathit{fragile}(Y) &\;\rightarrow\; \mathit{holds}(\mathit{broken}(Y), \mathit{do}(\mathit{drop}(R,X),S)) \\ \mathit{holds}(\mathit{next\_to}(B,Y),S) \land \mathit{bomb}(B) &\;\rightarrow\; \mathit{holds}(\mathit{broken}(Y), \mathit{do}(\mathit{explode}(B),S)) \end{aligned}$$

- Equivalent logical form:

$$\begin{aligned} \mathit{poss}(A,S) \;\land\; & [(\exists R,X)\; A = \mathit{drop}(R,X) \land Y = X \land \mathit{fragile}(Y) \\ & \lor\; (\exists B)\; A = \mathit{explode}(B) \land \mathit{holds}(\mathit{next\_to}(B,Y),S)] \\ & \rightarrow\; \mathit{holds}(\mathit{broken}(Y), \mathit{do}(A,S)) \end{aligned}$$

$$\begin{aligned} \mathit{holds}(\mathit{holding}(R,X),S) &\;\rightarrow\; \mathit{poss}(\mathit{drop}(R,X),S) \\ \mathit{bomb}(B) &\;\rightarrow\; \mathit{poss}(\mathit{explode}(B),S) \end{aligned}$$

- 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 disappearence of fluents.
- A negative effect axiom for the fluent *broken*:

$$\mathit{holds}(\mathit{has\_glue}(R), S) \wedge \mathit{holds}(\mathit{broken}(X), S) \wedge Y = X$$
$$\rightarrow\ \neg \mathit{holds}(\mathit{broken}(Y), \mathit{do}(\mathit{repair}(R, X), S))$$

- Equivalent logical form:

$$\mathit{poss}(A, S) \wedge (\exists R, X)\ A = \mathit{repair}(R, X) \wedge Y = X \rightarrow\ \neg \mathit{holds}(\mathit{broken}(Y), \mathit{do}(A, S))$$

$$\mathit{holds}(\mathit{has\_glue}(R), S) \wedge \mathit{holds}(\mathit{broken}(X), S) \rightarrow \mathit{poss}(\mathit{repair}(R, X), S)$$

- 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:

$$\begin{aligned}&\mathit{poss}(A,S) \land \neg \mathit{holds}(\mathit{broken}(Y),S) \land \mathit{holds}(\mathit{broken}(Y),\mathit{do}(A,S))\\ \rightarrow\ & (\exists R,X)\ A = \mathit{drop}(R,X) \land X = Y \land \mathit{fragile}(Y)\\ &\quad \lor\ (\exists B)\ A = \mathit{explode}(B) \land \mathit{holds}(\mathit{next\_to}(B,Y),S)\end{aligned}$$

- Explanation Closure Axiom 2:

$$\begin{aligned}&\mathit{poss}(A,S) \land \mathit{holds}(\mathit{broken}(Y),S) \land \neg \mathit{holds}(\mathit{broken}(Y),\mathit{do}(A,S))\\ \rightarrow\ & (\exists R)\ A = \mathit{repair}(R,Y)\end{aligned}$$

Situation Calculus: The General Approach (1)

- ▷ General positive effect axioms for each fluent f:

$$\mathit{poss}(A,S) \wedge \gamma_f^+(A,S) \rightarrow \mathit{holds}(f, \mathit{do}(A,S))$$

- ▷ General negative effect axioms for fluent f:

$$\mathit{poss}(A,S) \wedge \gamma_f^-(A,S) \rightarrow \neg \mathit{holds}(f, \mathit{do}(A,S))$$

- ▷ Precondition axioms for each action a:

$$\pi_a(S) \rightarrow \mathit{poss}(a,S)$$

- ▷ 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:

$$\begin{aligned} \mathit{poss}(A,S) \wedge \mathit{holds}(f,S) \wedge \neg \mathit{holds}(f, \mathit{do}(A,S)) &\rightarrow \gamma_f^-(A,S) \\ \mathit{poss}(A,S) \wedge \neg \mathit{holds}(f,S) \wedge \mathit{holds}(f, \mathit{do}(A,S)) &\rightarrow \gamma_f^+(A,S) \end{aligned}$$

Situation Calculus: The General Approach (2)

- ▷ Unique names axioms for actions:

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

$$a(X_1, \ldots, X_n) = a(Y_1, \ldots, Y_n) \rightarrow X_1 = Y_1 \wedge \ldots X_n = Y_n$$

- ▷ Unique names axioms for situations $\mathcal{F}_{uns}$:

$$s_0 \neq \textit{do}(A, S)$$

$$\textit{do}(A, S) = \textit{do}(A', S') \rightarrow A = A' \wedge S = S'$$

Successor State Axioms

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

$$\mathit{poss}(A,S) \;\rightarrow\; [\mathit{holds}(f,\mathit{do}(A,S)) \;\leftrightarrow\; \gamma_f^+(A,S) \;\lor\; \mathit{holds}(f,S)\land\neg\gamma_f^-(A,S)].$$

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

▷ The successor state axiom for *broken*:

$$\begin{aligned}\mathit{poss}(A,S) \;\rightarrow\; & [\mathit{holds}(\mathit{broken}(Y),\mathit{do}(A,S)) \;\leftrightarrow \\ & (\exists R,X)\; A=\mathit{drop}(R,Y)\land X=Y\land \mathit{fragile}(Y) \\ & \lor\; (\exists B)\; A=\mathit{explode}(B)\land \mathit{holds}(\mathit{next\_to}(B,Y),S) \\ & \lor\; \mathit{holds}(\mathit{broken}(Y),S)\land\neg(\exists R)\; A=\mathit{repair}(R,Y)]\end{aligned}$$

▷ $n+m$ 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:

$$\mathcal{F}_{ex} = \{\mathit{ex}(S) \quad \leftrightarrow \quad S = s_0 \quad \vee \quad (\exists A, S')\ S = \mathit{do}(A, S') \wedge \mathit{poss}(A, S') \wedge \mathit{ex}(S')\}$$

$\leadsto$ $\mathcal{F} \models (\exists S)\ g(S) \wedge \mathit{ex}(S)$, where $\mathcal{F}$ 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 *poss* or *ex*.
  - There does not exist a *holds*–predicate in $\mathcal{F}$ which contains an occurrence of *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:

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

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

$$(\exists \overline{X})\ A = a_1(\overline{X}) \wedge \pi_{a_1} \quad \vee \ldots \vee \quad (\exists \overline{Z})\ A = a_n(\overline{Z}) \wedge \pi_{a_n}.$$

▷ Successor State Axioms:

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

A Regression Operator

▷ A regression operator R:

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

$$R[W] = W.$$

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

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

then

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

3. Whenever W, W_1, W_2 are formulas then

$$\begin{aligned} R[\neg W] &= \neg R[W] \\ R[(\forall V)\ W] &= (\forall V)\ R[W] \\ R[W_1 \wedge W_2] &= R[W_1] \wedge R[W_2] \end{aligned}$$

and likewise for $\exists$, $\vee$, $\rightarrow$ and $\leftrightarrow$.

Regression

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

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

▷ 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

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

where $\mathcal{F}_{\forall s}$ 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

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

iff for some n

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

GOLOG

- ▷ 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!

  $\leadsto$ Reason about the situations of the world and consider the effects of various possible plans.

Complex Actions

- $\textit{do}(\delta, S, S')$ holds, whenever S' is a terminating situation of an execution of a complex action δ starting in situation S.
- Primitive actions:

$$\textit{do}(A, S, S') \stackrel{def}{=} \textit{poss}(A, S) \wedge S' = \textit{do}(A, S).$$

- Test actions:

$$\textit{do}(\Phi?, S, S') \stackrel{def}{=} \textit{holds}(\Phi, S) \wedge S = S'.$$

- Sequence:

$$\textit{do}([\delta_1; \delta_2], S, S') \stackrel{def}{=} (\exists S^*)\ (\textit{do}(\delta_1, S, S^*) \wedge \textit{do}(\delta_2, S^*, S')).$$

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

Correctness and Termination

▷ Correctness:

$$\mathcal{F} \models (\forall S)\ \textit{do}(\delta, s_0, S) \rightarrow p(S)$$

or, even stronger

$$\mathcal{F} \models (\forall S_0, S)\ \textit{do}(\delta, S_0, S) \rightarrow p(S).$$

▷ Termination:

$$\mathcal{F} \models (\exists S)\ \textit{do}(\delta, s_0, S)$$

or, even stronger

$$\mathcal{F} \models (\forall S_0)\ (\exists S)\ \textit{do}(\delta, S_0, S).$$

An Example: The Tile Crawler

- Primitive action: $\mathit{crawl}(N)$: crawls to tile N.
- Fluents:
  - $\mathit{on}(N)$: the crawler is on tile N,
  - $\mathit{clean}(N)$: tile N is clean.
- Action Precondition Axioms:

$$\mathit{poss}(\mathit{crawl}(N), S) \leftrightarrow \mathit{holds}(\mathit{on}(N-1), S).$$

- Successor State Axiom:

$$\begin{aligned}
\mathit{poss}(A, S) \ &\rightarrow \ [\mathit{holds}(\mathit{on}(N+1), \mathit{do}(A, S)) \leftrightarrow \\
&\quad [A = \mathit{crawl}(N+1) \wedge \mathit{holds}(\mathit{on}(N), S)] \vee [A \neq \mathit{crawl}(N) \wedge \mathit{holds}(\mathit{on}(N+1), S)]]. \\
\mathit{poss}(A, S) \ &\rightarrow \ [\mathit{holds}(\mathit{clean}(N), \mathit{do}(A, S)) \leftrightarrow \mathit{holds}(\mathit{clean}(N), S)].
\end{aligned}$$

- *do*([*crawl*(1);*crawl*(2);*clean*(2)?;*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 $\mathit{do}(\delta, S, S')$ 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 charcterized by n fluents.
  - ⇝ Inferential frame problem remains!

Literature

- J. McCarthy: Situations and Actions and Causal Laws. Stanford Artificial Intelligence Project, Memo 2: 1963.
- J. McCarthy and P. J. Hayes: Some Philosophical Problems from the Standpoint of Artificial Intelligence. In: B. Meltzer and D. Michie (eds.), Machine Intelligence 4, Edinburgh University Press, 463-502: 1969.
- R. Reiter: The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression. In: V. Lifschitz (ed.), Artificial Intelligence and Mathematical Theory of Computation — Papers in Honor of John McCarthy. Academic Press, 359-380: 1991.
- H. Levesque etal: GOLOG: A Logic Programming Language for Dynamic Domains. Journal of Logic Programming 31, 59-83: 1997.

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 poinoning as has beed diagnosed afterwards. According to the witness, the nephew had no oppertunity to poison the tea beforehand. This proves that it was the milk which was poisoned and by which the victim was murdered."

▷ Fluents:

– *poisoned*(X): X is poisoned,

– *alive*(X): X is alive.

▷ Actions:

– *mix*(P, X, Y): agent P mixes X into Y,

– *drink*(P, X): agent P drinks X.

Formalizing the Murder Mystery

- Action precondition axioms: exactly as in the situation calculus

$$\begin{aligned} \mathit{alive}(P) &\rightarrow \mathit{poss}(\mathit{mix}(P, X, Y), S) \\ \mathit{alive}(P) &\rightarrow \mathit{poss}(\mathit{drink}(P, X), S) \end{aligned}$$

- Effect axioms for each action:

$$\begin{aligned} &\mathit{poss}(\mathit{mix}(P, X, Y), S) \wedge \mathit{holds}(\mathit{poisoned}(X), S) \\ &\quad \rightarrow \mathit{holds}(\mathit{poisoned}(Y), \mathit{do}(\mathit{mix}(P, X, Y), S)) \end{aligned}$$

$$\begin{aligned} &\mathit{poss}(\mathit{drink}(P, X), S) \wedge \mathit{holds}(\mathit{alive}(P), S) \wedge \mathit{holds}(\mathit{poisoned}(X), S) \\ &\quad \rightarrow \neg \mathit{holds}(\mathit{alive}(P), \mathit{do}(\mathit{drink}(P, X), S)) \end{aligned}$$

- Initial situation in the situation calculus:

$$\neg \mathit{holds}(\mathit{poisoned}(\mathit{tea}), s_0) \wedge \mathit{holds}(\mathit{alive}(\mathit{nephew}), s_0) \wedge \mathit{holds}(\mathit{alive}(\mathit{aunt}), s_0)$$

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 $\wedge$ and $\top$;
  $\wedge$ is associative, commutative, idempotent and $\top$ is its unit element.

  $\rightsquigarrow$ Each fact is stated only once ($\mathit{holds}(\mathit{car}, s_0) \wedge \mathit{holds}(\mathit{car}, s_0) \leftrightarrow \mathit{holds}(\mathit{car}, s_0)$).
- ▷ Situational fluents are reified ($\mathit{holds}(F, S)$).
- ▷ Initial situations completly specify the initial state, eg.:

$$\mathit{holds}(\mathit{alive}(\mathit{aunt}), s_0) \wedge \mathit{holds}(\mathit{alive}(\mathit{nephew}), s_0) \wedge \neg\mathit{holds}(\mathit{poisoned}(\mathit{tea}), s_0).$$

- ▷ Negative facts are explicitly stated.

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.

  ⇝ Fluents may be stated more than once ($\mathit{car} \circ \mathit{car} \neq \mathit{car}$).
- ▷ Multisets of fluents are reified.
- ▷ Initial situations may be partially specified, eg.:

$$(\exists Z)\ [\mathit{state}(s_0) = \mathit{alive}(\mathit{nephew}) \circ \mathit{alive}(\mathit{aunt}) \circ Z \quad \wedge \quad (\forall Z')\ Z \neq \mathit{poisoned}(\mathit{tea}) \circ Z'].$$

- ▷ 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 contstraint $\text{FLUENT} < \text{STATE}$ $\big((\forall X)\ (\text{FLUENT}(X) \rightarrow \text{STATE}(X))\big)$.

▷ Function symbols $\Sigma_F = \Sigma_A \cup \Sigma_{Sit} \cup \Sigma_{Fl} \cup \Sigma_{St} \cup \Sigma_O$, where

Σ_A is a set of function symbols denoting action names,

Σ_{Sit} $= \{s_0, \mathit{do}\}$,

Σ_{Fl} is a set of function symbols denoting fluent names,

Σ_{St} $= \{\emptyset, \circ, \mathit{state}\}$ and

Σ_O is a set of function symbols denoting objects.

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

▷ All sets are mutually disjoint.

Function Symbols

▷ Special function symbols:

$$\begin{aligned}
s_0 &: \quad \rightarrow \text{SITUATION} \\
\mathit{do} &: \quad \text{ACTION} \times \text{SITUATION} \rightarrow \text{SITUATION} \\
\emptyset &: \quad \rightarrow \text{STATE} \\
\circ &: \quad \text{STATE} \times \text{STATE} \rightarrow \text{STATE} \\
\mathit{state} &: \quad \text{SITUATION} \rightarrow \text{STATE}
\end{aligned}$$

▷ Remaining function symbols:

$$\begin{aligned}
\mathit{nephew} &: \quad \rightarrow \text{OBJECT} \\
\mathit{aunt} &: \quad \rightarrow \text{OBJECT} \\
\mathit{tea} &: \quad \rightarrow \text{OBJECT} \\
\mathit{milk} &: \quad \rightarrow \text{OBJECT} \\
\mathit{mix} &: \quad \text{OBJECT} \times \text{OBJECT} \times \text{OBJECT} \rightarrow \text{ACTION} \\
\mathit{drink} &: \quad \text{OBJECT} \times \text{OBJECT} \rightarrow \text{ACTION} \\
\mathit{alive} &: \quad \text{OBJECT} \rightarrow \text{FLUENT} \\
\mathit{poisoned} &: \quad \text{OBJECT} \rightarrow \text{FLUENT}
\end{aligned}$$

Action, Situation, Fluent, State and Object Terms

- ▷ Object terms: tea, nephew.
- ▷ Action terms: $\mathit{mix}(\mathit{nephew}, \mathit{milk}, \mathit{tea})$, $\mathit{drink}(X, Y)$.
- ▷ Situation terms: s_0, $\mathit{do}(\mathit{mix}(\mathit{nephew}, \mathit{milk}, \mathit{tea}), s_0)$.
- ▷ Fluent terms: $\mathit{poisoned}(\mathit{milk})$, $\mathit{alive}(X)$.
- ▷ State terms: $\emptyset$, $Z_1 \circ Z_2$, $\mathit{state}(s_0)$, $\mathit{alive}(X)$.
  - – A state term is said to be simple if it contains at most one occurrence of a variable of sort *state*.

    $\leadsto$ $\mathit{alive}(\mathit{nephew}) \circ \mathit{alive}(\mathit{aunt}) \circ Z$, $\mathit{state}(\mathit{do}(A, S)) \circ \mathit{alive}(X)$
  - – A state term is said to be a constructor state term if it is built from fluent terms, the constant $\emptyset$, the function symbol $\circ$ and variables of sort STATE only.

    $\leadsto$ $\mathit{alive}(\mathit{nephew}) \circ \mathit{alive}(\mathit{aunt}) \circ Z$, $\mathit{alive}(\mathit{nephew}) \circ \mathit{alive}(\mathit{aunt})$

Formalizing States in the Fluent Calculus

- Some properties of $\circ$: $\mathcal{F}_{AC1}$

  A associative: $(\forall Z_1, Z_2, Z_3 : \text{STATE})\ (Z_1 \circ Z_2) \circ Z_3 = Z_1 \circ (Z_2 \circ Z_3)\,,$

  C commutative: $(\forall Z_1, Z_2 : \text{STATE})\ Z_1 \circ Z_2 = Z_2 \circ Z_1\,,$

  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

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

- $\mathit{holds}(F, S) \stackrel{def}{=} (\exists Z : \text{STATE})\ \mathit{state}(S) = F \circ Z\,.$

Extended Unique Names Assumptions

▷ 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 $\overline{X}$.

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

State Update Axioms

- State update axioms:

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

- Examples:

$$\begin{aligned}&\mathit{poss}(\mathit{do}(\mathit{mix}(P,X,Y),S) \wedge \mathit{holds}(\mathit{poisoned}(X),S) \wedge \neg\mathit{holds}(\mathit{poisoned}(Y),S)\\&\qquad \rightarrow \mathit{state}(\mathit{do}(\mathit{mix}(P,X,Y),S)) = \mathit{state}(S) \circ \mathit{poisoned}(Y)\end{aligned}$$

$$\begin{aligned}&\mathit{poss}(\mathit{do}(\mathit{mix}(P,X,Y),S) \wedge \neg\mathit{holds}(\mathit{poisoned}(X),S) \vee \mathit{holds}(\mathit{poisoned}(Y),S)\\&\qquad \rightarrow \mathit{state}(\mathit{do}(\mathit{mix}(P,X,Y),S)) = \mathit{state}(X)\end{aligned}$$

$$\begin{aligned}&\mathit{poss}(\mathit{do}(\mathit{drink}(P,X),S)) \wedge \mathit{holds}(\mathit{alive}(P),S) \wedge \mathit{holds}(\mathit{poisoned}(X),S)\\&\qquad \rightarrow \mathit{state}(\mathit{do}(\mathit{drink}(P,X),S)) \circ \mathit{alive}(P) = \mathit{state}(S)\end{aligned}$$

$$\begin{aligned}&\mathit{poss}(\mathit{do}(\mathit{drink}(P,X),S)) \wedge \neg\mathit{holds}(\mathit{alive}(P),S) \vee \neg\mathit{holds}(\mathit{poisoned}(X),S)\\&\qquad \rightarrow \mathit{state}(\mathit{do}(\mathit{drink}(P,X),S)) = \mathit{state}(S)\end{aligned}$$

Fluent Calculus: The General Approach (1)

- Positive effect axioms:

$$\mathit{poss}(a(\overline{X}), S) \wedge \epsilon^{+}_{a,f}(\overline{X}, S) \rightarrow \mathit{holds}(f(\overline{Y}), \mathit{do}(a(\overline{X}), S)),$$

  where each variable occurring in $\overline{Y}$ occurs also in $\overline{X}$.

- Negative effect axioms:

$$\mathit{poss}(a(\overline{X}), S) \wedge \epsilon^{-}_{a,f}(\overline{X}, S) \rightarrow \neg \mathit{holds}(f(\overline{Y}), \mathit{do}(a(\overline{X}), S)).$$

  where each variable occurring in $\overline{Y}$ occurs also in $\overline{X}$.

- Precondition axioms for each action a:

$$\pi_a(\overline{X}, S) \rightarrow \mathit{poss}(a(\overline{X}, S)).$$

- 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

$$\neg(\exists \overline{X}, S)\ [\mathit{poss}(a(\overline{X}), S) \wedge \epsilon^{+}_{a,f}(\overline{X}, S) \wedge \epsilon^{-}_{a,f}(\overline{X}, S)].$$

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

$$\Delta(S) \quad \rightarrow \quad \mathit{state}(\mathit{do}(a, S)) \circ \vartheta^{-} = \mathit{state}(S) \circ \vartheta^{+},$$

where ϑ^{-} and ϑ^{+} are the negative and positive effects of action a under condition $\Delta(S)$ respectively.

Another Example: Yale Shooting Problem (1)

▷ Fluents:

– $\mathit{loaded}(X)$: gun X is loaded

– $\mathit{dead}(Y)$: individual Y is dead

▷ Actions:

– $\mathit{shoot}(X, Y)$: gun X is aimed at Y and the trigger is pulled.

▷ For simplicity, actions are always possible.

▷ Effect axioms:

$$\begin{aligned} \mathit{loaded}(X, S) &\rightarrow \mathit{holds}(\mathit{dead}(Y), \mathit{do}(\mathit{shoot}(X, Y), S)) \\ \top &\rightarrow \neg \mathit{holds}(\mathit{loaded}(X), \mathit{do}(\mathit{shoot}(X, Y), S)) \end{aligned}$$

Yale Shooting Problem (2)

▷ State update axioms:

$$\neg \mathit{holds}(\mathit{loaded}(X), S) \\ \quad \rightarrow \mathit{state}(\mathit{do}(\mathit{shoot}(X,Y),S)) = \mathit{state}(S)$$

$$\mathit{holds}(\mathit{dead}(Y), S) \wedge \mathit{holds}(\mathit{loaded}(X), S) \\ \quad \rightarrow \mathit{state}(\mathit{do}(\mathit{shoot}(X,Y),S)) \circ \mathit{loaded}(X) = \mathit{state}(S)$$

$$\mathit{holds}(\mathit{loaded}(X), S) \wedge \neg \mathit{holds}(\mathit{dead}(Y), S) \\ \quad \rightarrow \mathit{state}(\mathit{do}(\mathit{shoot}(X,Y),S)) \circ \mathit{loaded}(X) = \mathit{state}(S) \circ \mathit{dead}(Y)$$

Comments

▷ State update axioms involve only simple state terms.

▷ State axioms should be designed such that they do not violate the axiom

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

It is still crucial, however, in cases of incompletely specified situations.

▷ AC1–unification is decidable; complete unification algorithms exist.

Literature

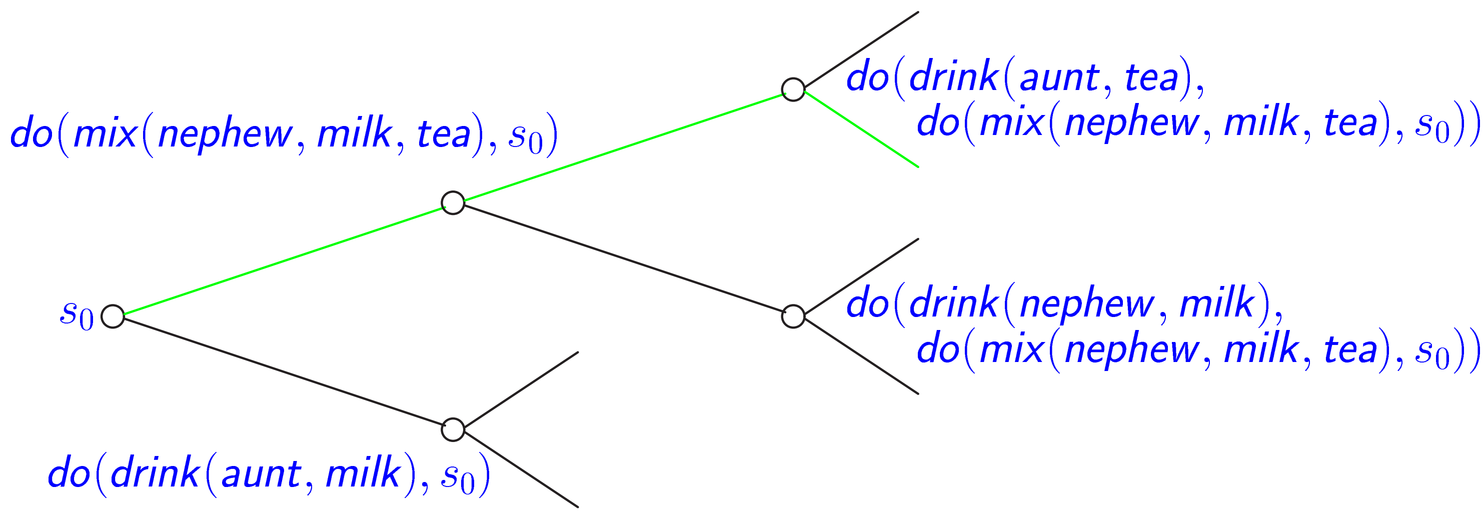
▷ S. Hölldobler and J. Schneeberger: A New Deductive Approach to Planning. New Generation Computing 8, 225-244: 1990.

▷ M. Thielscher: From Situation Calculus to Fluent Calculus: State Update Axioms as a Solution to the Inferential Frame Problem. To appear in: Artificial Intelligence Journal: 1999.

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



Reasoning About Counterfactual Action Sequences

The observation

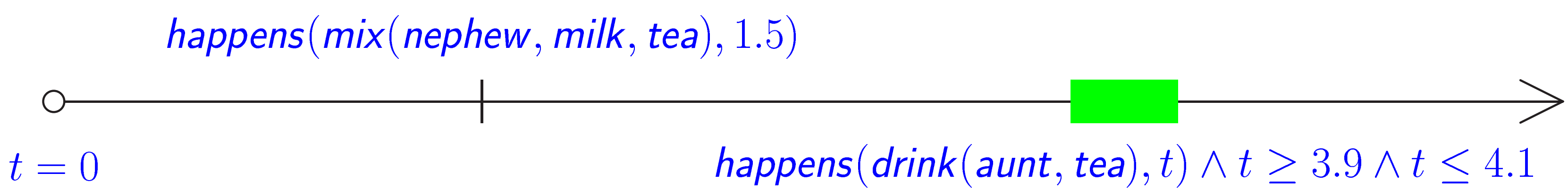
$$\mathit{holds}(\mathit{alive}(\mathit{aunt}), s_0) \wedge \mathit{holds}(\mathit{alive}(\mathit{nephew}), s_0) \wedge \neg \mathit{holds}(\mathit{poisoned}(\mathit{tea}), s_0)$$
$$\wedge \neg \mathit{holds}(\mathit{alive}(\mathit{aunt}), \mathit{do}(\mathit{drink}(\mathit{aunt}, \mathit{tea}), \mathit{do}(\mathit{mix}(\mathit{nephew}, \mathit{milk}, \mathit{tea}), s_0)))$$

entails this statement about a hypothetical course of events:

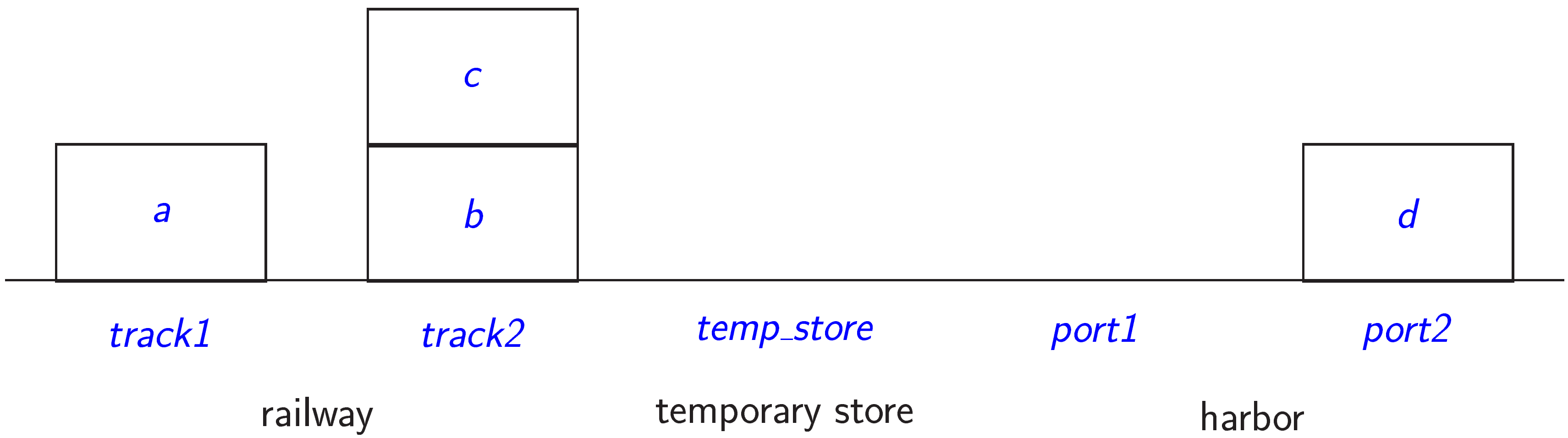
$$\neg \mathit{holds}(\mathit{alive}(\mathit{aunt}), \mathit{do}(\mathit{drink}(\mathit{aunt}, \mathit{milk}), s_0))$$

Linear Time Structure

$\mathit{happens}(E, T)$ $:\Leftrightarrow$ event E happens at time T



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:	$\mathit{on}(X, Y)$	X container
		Y container or location
	$\mathit{clear}(X)$	X container or location
Events:	$\mathit{move}(X, Y)$	X container
		Y container or location

Time points: positive real numbers (incl. 0)

Effect Axioms

$\mathit{holds\_at}(F, T)$:⇔ fluent F holds at time T

$\mathit{initiates}(E, F, T)$:⇔ event E makes fluent F true at time T

$\mathit{terminates}(E, F, T)$:⇔ event E makes fluent F false at time T

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

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

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

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

Initial Situation, Course of Events

$$\mathit{initially}(F) \quad :\Leftrightarrow \quad \text{fluent } F \text{ is initiated at time } 0$$
$$\mathit{n\_initially}(F) \quad :\Leftrightarrow \quad \text{fluent } F \text{ is terminated at time } 0$$

$$\begin{array}{l}
\mathit{initially}(\mathit{on}(a, \mathit{track1})) \wedge \mathit{initially}(\mathit{clear}(a)) \wedge \\
\mathit{initially}(\mathit{on}(b, \mathit{track2})) \wedge \mathit{initially}(\mathit{on}(c, b)) \wedge \mathit{initially}(\mathit{clear}(c)) \\
\mathit{initially}(\mathit{clear}(\mathit{temp_store})) \wedge \mathit{initially}(\mathit{clear}(\mathit{port1})) \wedge \\
\mathit{initially}(\mathit{on}(d, \mathit{port2})) \wedge \mathit{initially}(\mathit{clear}(d)) \\
[\,\mathit{initially}(\mathit{on}(X, Y)) \;\rightarrow\; \mathit{n_initially}(\mathit{clear}(Y))\,] \wedge \\
[\,\mathit{initially}(\mathit{clear}(X)) \;\rightarrow\; \mathit{n_initially}(\mathit{on}(Y, X))\,] \\
[\,\mathit{initially}(\mathit{on}(X, Y)) \wedge Y \neq Z \;\rightarrow\; \mathit{n_initially}(\mathit{on}(X, Z))\,] \wedge \\
[\,\mathit{initially}(\mathit{on}(X, Z)) \wedge X \neq Y \;\rightarrow\; \mathit{n_initially}(\mathit{on}(Y, Z))\,] \wedge
\end{array}$$

$$\begin{array}{l}
\mathit{happens}(\mathit{move}(c, \mathit{temp_store}), 3) \wedge \mathit{happens}(\mathit{move}(b, \mathit{port1}), 5) \wedge \\
\mathit{happens}(\mathit{move}(a, b), 8) \wedge \mathit{happens}(\mathit{move}(d, \mathit{track2}), 11) \wedge \mathit{happens}(\mathit{move}(c, \mathit{port2}), 13)
\end{array}$$

Foundational Axioms of the Event Calculus

$\mathit{clipped}(T_1, F, T_2)$:⇔ fluent F becomes false between time T_1 and time T_2

$\mathit{declipped}(T_1, F, T_2)$:⇔ fluent F becomes true between time T_1 and time T_2

$$\mathit{holds\_at}(F, T) \leftarrow \mathit{initially}(F) \land \neg \mathit{clipped}(0, F, T)$$

$$\mathit{holds\_at}(F, T_2) \leftarrow (\exists E)\ \mathit{happens}(E, T_1) \land \mathit{initiates}(E, F, T_1) \land T_1 < T_2 \land \neg \mathit{clipped}(T_1, F, T_2)$$

$$\neg \mathit{holds\_at}(F, T) \leftarrow \mathit{n\_initially}(F) \land \neg \mathit{declipped}(0, F, T)$$

$$\neg \mathit{holds\_at}(F, T_2) \leftarrow (\exists E)\ \mathit{happens}(E, T_1) \land \mathit{terminates}(E, F, T_1) \land T_1 < T_2 \land \neg \mathit{declipped}(T_1, F, T_2)$$

$$\mathit{clipped}(T_1, F, T_2) \leftrightarrow (\exists E, T)\ \mathit{happens}(E, T) \land \mathit{terminates}(E, F, T) \land T_1 < T \land T < T_2$$

$$\mathit{declipped}(T_1, F, T_2) \leftrightarrow (\exists E, T)\ \mathit{happens}(E, T) \land \mathit{initiates}(E, F, T) \land T_1 < T \land T < T_2$$

Summary: Event Calculus Signatures

- Sorts FLUENT, EVENT, TIMEPOINT
- predicates:

$$
\begin{aligned}
\mathit{happens} &: \text{EVENT} \times \text{TIMEPOINT}\\
\mathit{holds\_at} &: \text{FLUENT} \times \text{TIMEPOINT}\\
\mathit{initially} &: \text{FLUENT}\\
\mathit{n\_initially} &: \text{FLUENT}\\
\mathit{initiates} &: \text{EVENT} \times \text{FLUENT} \times \text{TIMEPOINT}\\
\mathit{terminates} &: \text{EVENT} \times \text{FLUENT} \times \text{TIMEPOINT}\\
\mathit{clipped} &: \text{TIMEPOINT} \times \text{FLUENT} \times \text{TIMEPOINT}\\
\mathit{declipped} &: \text{TIMEPOINT} \times \text{FLUENT} \times \text{TIMEPOINT}\\
< &: \text{TIMEPOINT} \times \text{TIMEPOINT}
\end{aligned}
$$

Still Something Seems Missing . . .

So far the axiomatization does not entail many useful conclusions.

For example, the first event is $\mathit{happens}(\mathit{move}(c, \mathit{temp\_store}), 3)$.
It would be useful to prove that $\mathit{holds\_at}(\mathit{clear}(c), 3)$.

But from

$$\mathit{holds\_at}(\mathit{clear}(c), 3) \leftarrow \mathit{initially}(\mathit{clear}(c)) \wedge \neg \mathit{clipped}(0, \mathit{clear}(c), 3)$$

$$\mathit{initially}(\mathit{clear}(c))$$

$$\mathit{clipped}(0, \mathit{clear}(c), 3) \leftrightarrow (\exists E, T)\ \mathit{happens}(E, T) \wedge \mathit{terminates}(E, \mathit{clear}(c), T) \wedge 0 < T \wedge T < 3$$

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 $p(\overline{X})$.

$A(p/\Phi)$ $:\Leftrightarrow$ replace in A all occurrences of p by Φ

CIRC$[A;p]$ $:\Leftrightarrow$ $A \wedge (\forall \Phi) \{ A(p/\Phi) \wedge [(\forall \overline{X}) \Phi(\overline{X}) \rightarrow p(\overline{X})] \rightarrow [(\forall \overline{X}) p(\overline{X}) \rightarrow \Phi(\overline{X})] \}$

Example

Let N be,

$$\begin{aligned}&\mathit{happens}(\mathit{move}(c, \mathit{temp\_store}), 3) \wedge \mathit{happens}(\mathit{move}(b, \mathit{port1}), 5)\\ &\wedge \mathit{happens}(\mathit{move}(a, b), 8) \wedge \mathit{happens}(\mathit{move}(d, \mathit{track2}), 11)\\ &\wedge \mathit{happens}(\mathit{move}(c, \mathit{port2}), 13)\end{aligned}$$

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

$$\begin{gathered}\mathit{happens}(E, T)\\ \leftrightarrow\\ \begin{aligned}&E = \mathit{move}(c, \mathit{temp\_store}) \wedge T = 3 \ \vee\\ &E = \mathit{move}(b, \mathit{port1}) \wedge T = 5 \ \vee\\ &E = \mathit{move}(a, b) \wedge T = 8 \ \vee\\ &E = \mathit{move}(d, \mathit{track2}) \wedge T = 11 \ \vee\\ &E = \mathit{move}(c, \mathit{port2}) \wedge T = 13\end{aligned}\end{gathered}$$

Unique Names Assumptions

$$\mathsf{UNA}[c_1, \ldots, c_n] \quad :\Leftrightarrow \quad \bigwedge_{\substack{i = 1 \ldots n \\ j = 1 \ldots n \\ i \neq j}} c_i \neq c_j$$

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 $q(\overline{Y})$.

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

$$:\Leftrightarrow$$

$$A \wedge$$

$$(\forall \Phi, \Psi) \left\{ \begin{array}{c} A(p/\Phi)(q/\Psi) \wedge [\,(\forall \overline{X})\,\Phi(\overline{X}) \rightarrow p(\overline{X})\,] \wedge [\,(\forall \overline{Y})\,\Psi(\overline{Y}) \rightarrow q(\overline{Y})\,] \\ \rightarrow \\ [\,(\forall \overline{X})\,p(\overline{X}) \rightarrow \Phi(\overline{X})\,] \wedge [\,(\forall \overline{Y})\,q(\overline{Y}) \rightarrow \Psi(\overline{Y})\,] \end{array} \right\}$$

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

$$\mathrm{CIRC}[N \wedge I; \mathit{happens}] \wedge \mathrm{CIRC}[E; \mathit{initiates}, \mathit{terminates}] \wedge U \wedge EC$$

The Event Calculus in PROLOG

```
initiates(move(X,Y), on(X,Y), T) :-
   holds_at(clear(X), T), holds_at(clear(Y), T), not X=Y.

initiates(move(X,Y), clear(Z), T) :-
   holds_at(clear(X), T), holds_at(clear(Y), T),
   holds_at(on(X,Z), T), not X=Y, not Y=Z.

terminates(move(X,Y), on(X,Z), T) :-
   holds_at(clear(X), T), holds_at(clear(Y), T),
   holds_at(on(X,Z), T), not X=Y, not Y=Z.

terminates(move(X,Y), clear(Y), T) :-
   holds_at(clear(X), T), holds_at(clear(Y), T), not X=Y.


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 clipped(T1, F, T2).

clipped(T1, F, T2) :- happens(E, T), T1<T, T<T2, terminates(E, F, T).
```

The Fork Lift Truck Scenario in PROLOG

```
initially(on(a,track1)).
initially(on(b,track2)).
initially(clear(c)).
initially(clear(port1)).
initially(clear(d)).
initially(clear(a)).
initially(on(c,b)).
initially(clear(temp_store)).
initially(on(d,port2)).

happens(move(c,temp_store),3).
happens(move(a,b),8).
happens(move(c,port2),13).
happens(move(b,port1),5).
happens(move(d,track2),11).
```

```
| ?- 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 charcterized by n fluents.
  - ⇝ Same inferential frame problem as with the Situation 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

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

Literature

- ▷ M. Shanahan: Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. MIT Press 1997.
- ▷ M. Shanahan: Robotics and the Common Sense Informatic Situation. In: W. Wahlster (ed.), Proceedings of the European Conference on AI, pp. 684–688. John Wiley 1996.
- ▷ M. Shanahan: Event Calculus Planning Revisited. In: Proceedings of the European Conference on Planning, pp. 390–402. Springer LNAI 1348, 1997.
- ▷ J. McCarthy: Circumscription—A form of non-monotonic reasoning. Artificial Intelligence Journal 13: 27–39, 1980.

Nondeterministic Actions

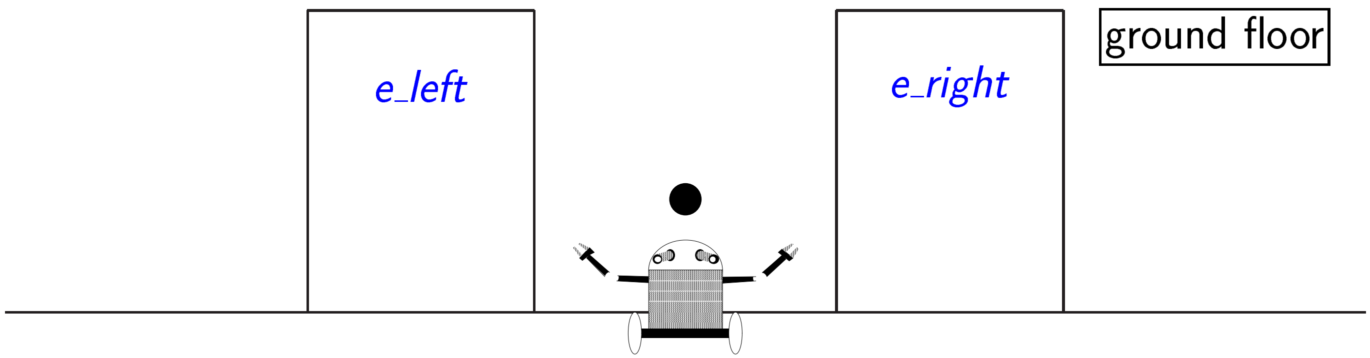
▷ What happens when calling an elevator

▷ Modeling nondeterministic actions in the Situation Calculus

▷ Modeling nondeterministic actions in the Fluent Calculus

▷ Modeling nondeterministic actions in the Event Calculus

Two Elevators



$\mathit{holds}(\mathit{at\_floor}(X,Y),S) \quad :\Leftrightarrow$ elevator X is at floor Y in situation S

$$\mathit{holds}(\mathit{at\_floor}(X,Y),S) \rightarrow X = \mathit{e\_left} \vee X = \mathit{e\_right}$$
$$\mathit{holds}(\mathit{at\_floor}(X,Y),S) \wedge \mathit{holds}(\mathit{at\_floor}(X,Z),S) \rightarrow Y = Z$$
$$\mathit{poss}(\mathit{call\_floor}(Y),S) \leftrightarrow \neg(\exists X)\, \mathit{holds}(\mathit{at\_floor}(X,Y),S)$$

$$(\exists Y,Z)\ \mathit{holds}(\mathit{at\_floor}(\mathit{e\_left},Y),s_0) \wedge \mathit{holds}(\mathit{at\_floor}(\mathit{e\_right},Z),s_0) \wedge Y \neq 1 \wedge Z \neq 1$$

$$\begin{aligned} &\mathit{poss}(\mathit{call\_floor}(Y),S) \rightarrow \\ &\quad \mathit{caused}(\mathit{at\_floor}(\mathit{e\_left},Y),\mathit{true},\mathit{do}(\mathit{call\_floor}(Y),S)) \oplus \\ &\quad \mathit{caused}(\mathit{at\_floor}(\mathit{e\_right},Y),\mathit{true},\mathit{do}(\mathit{call\_floor}(Y),S)) \end{aligned}$$

(where $F \oplus G \stackrel{\text{def}}{=} F \wedge \neg G \vee \neg F \wedge G$)

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.

$\mathit{case}(N, A, S)$:⇔ in situation S, performing the nondeterministic action A results in outcome no. N

$$\mathit{poss}(\mathit{call\_floor}(Y), S) \wedge \mathit{case}(1, \mathit{call\_floor}(Y), S) \rightarrow \\ \mathit{caused}(\mathit{at\_floor}(\mathit{e\_left}, Y), \mathit{true}, \mathit{do}(\mathit{call\_floor}(Y), S))$$

$$\mathit{poss}(\mathit{call\_floor}(Y), S) \wedge \mathit{case}(2, \mathit{call\_floor}(Y), S) \rightarrow \\ \mathit{caused}(\mathit{at\_floor}(\mathit{e\_right}, Y), \mathit{true}, \mathit{do}(\mathit{call\_floor}(Y), S))$$

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

$$\mathit{caused}(\mathit{at\_floor}(X, Y), \mathit{true}, S) \wedge Y \neq Z \rightarrow \mathit{caused}(\mathit{at\_floor}(X, Z), \mathit{false}, S)$$

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

$$\mathit{case}(1, a(\overline{X}), S) \oplus \ldots \oplus \mathit{case}(n, a(\overline{X}), S)$$

The Resulting Successor State Axiom

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

$$
\begin{array}{l}
[\, A = \mathit{call\_floor}(Y) \wedge X = \mathit{e\_left} \wedge \mathit{case}(1, A, S) \,] \vee \\
[\, A = \mathit{call\_floor}(Y) \wedge X = \mathit{e\_right} \wedge \mathit{case}(2, A, S) \,]
\end{array}
$$

$$
\begin{array}{l}
\mathit{poss}(A, S) \rightarrow \\
\quad [\, \mathit{holds}(\mathit{at\_floor}(X, Y), \mathit{do}(A, S)) \leftrightarrow \\
\qquad \Psi(X, Y, A, S) \vee \\
\qquad \mathit{holds}(\mathit{at\_floor}(X, Y), S) \wedge \neg(\exists Z)\, [\, \Psi(X, Z, A, S) \wedge Y \neq Z \,] \,]
\end{array}
$$

Disjunctive State Update Axioms

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

$$\begin{aligned}
&\mathit{holds}(\mathit{at_floor}(X,Y),S) \rightarrow X = \mathit{e_left} \lor X = \mathit{e_right} \\
&\mathit{holds}(\mathit{at_floor}(X,Y),S) \land \mathit{holds}(\mathit{at_floor}(X,Z),S) \rightarrow Y = Z \\
&\mathit{poss}(\mathit{call_floor}(Y),S) \leftrightarrow \neg(\exists X)\, \mathit{holds}(\mathit{at_floor}(X,Y),S)
\end{aligned}$$

$$\begin{aligned}
&\mathit{poss}(\mathit{call_floor}(Y),S) \land \mathit{holds}(\mathit{at_floor}(\mathit{e_left},Z_1),S) \land \mathit{holds}(\mathit{at_floor}(\mathit{e_right},Z_2),S) \rightarrow \\
&\quad [\, \mathit{state}(\mathit{do}(\mathit{call_floor}(Y),S)) \circ \mathit{at_floor}(\mathit{e_left},Z_1) = \mathit{state}(S) \circ \mathit{at_floor}(\mathit{e_left},Y) \,] \lor \\
&\quad [\, \mathit{state}(\mathit{do}(\mathit{call_floor}(Y),S)) \circ \mathit{at_floor}(\mathit{e_right},Z_2) = \mathit{state}(S) \circ \mathit{at_floor}(\mathit{e_right},Y) \,]
\end{aligned}$$

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

Nondeterministic Actions plus Ramifications

$$
\begin{aligned}
&\mathit{causes}(Z_1, E_1, Z_2, E_2) \ \leftrightarrow \\
&\quad \mathit{holds\_in}(\mathit{at\_floor}(X, Y_1), E_1) \wedge \mathit{holds\_in}(\mathit{at\_floor}(X, Y_2), Z_1) \wedge Y_1 \neq Y_2 \wedge \\
&\quad Z_2 \circ \mathit{at\_floor}(X, Y_2) = Z_1 \wedge E_2 = E_1 \circ \neg \mathit{at\_floor}(X, Y_2)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{poss}(\mathit{call\_floor}(Y), S) \ \rightarrow \\
&\quad [\,(\exists Z)\ Z = \mathit{state}(S) \circ \mathit{at\_floor}(\mathit{e\_left}, Y) \wedge \\
&\qquad\qquad \mathit{ramify}(Z, \mathit{at\_floor}(\mathit{e\_left}, Y), \mathit{state}(\mathit{do}(\mathit{call\_floor}(Y), S)))\,] \vee \\
&\quad [\,(\exists Z)\ Z = \mathit{state}(S) \circ \mathit{at\_floor}(\mathit{e\_right}, Y) \wedge \\
&\qquad\qquad \mathit{ramify}(Z, \mathit{at\_floor}(\mathit{e\_right}, Y), \mathit{state}(\mathit{do}(\mathit{call\_floor}(Y), S)))\,]
\end{aligned}
$$

Conditional Plans

▷ action $\mathit{if}(F, A_1, A_2)$ $:\Leftrightarrow$ if fluent F holds then action A_1 else action A_2

▷ Foundational axioms $\mathcal{F}_{\mathit{if}}$:

$$\begin{aligned}
&\mathit{poss}(\mathit{if}(F, A_1, A_2), S) \leftrightarrow \\
&\qquad [\, \mathit{holds}(F, S) \rightarrow \mathit{poss}(A_1, S) \,] \wedge [\, \neg \mathit{holds}(F, S) \rightarrow \mathit{poss}(A_2, S) \,]
\end{aligned}$$

$$\begin{aligned}
&\mathit{poss}(\mathit{if}(F, A_1, A_2), S) \rightarrow \\
&\qquad [\, \mathit{holds}(F, S) \rightarrow \mathit{state}(\mathit{do}(\mathit{if}(F, A_1, A_2), S)) = \mathit{state}(\mathit{do}(A_1, S)) \,] \wedge \\
&\qquad [\, \neg \mathit{holds}(F, S) \rightarrow \mathit{state}(\mathit{do}(\mathit{if}(F, A_1, A_2), S)) = \mathit{state}(\mathit{do}(A_2, S)) \,]
\end{aligned}$$

An Example Conditional Plan

$$\begin{aligned} \mathit{holds}(\mathit{at}(Y), S) \;&:\Leftrightarrow\; \text{the robot is at floor } Y \text{ in situation } S \\ \mathit{holds}(\mathit{inside}(X), S) \;&:\Leftrightarrow\; \text{the robot is inside of elevator } X \text{ in situation } S \\ \text{action } \mathit{enter}(X) \;&:\Leftrightarrow\; \text{the robot enters elevator } X \end{aligned}$$

$$\mathit{poss}(\mathit{enter}(X), S) \;\leftrightarrow\; \neg \mathit{holds}(\mathit{inside}(X), S) \,\wedge\, (\exists Y)\,[\, \mathit{holds}(\mathit{at}(Y), S) \wedge \mathit{holds}(\mathit{at\_floor}(X, Y), S) \,]$$

$$\mathit{poss}(\mathit{enter}(X), S) \;\rightarrow\; \mathit{state}(\mathit{do}(\mathit{enter}(X), S)) = \mathit{state}(S) \circ \mathit{inside}(X)$$

Then, from

$$\begin{aligned} (\exists Y_1, Y_2, Z)\; \mathit{state}(s_0) = \mathit{at}(1) \circ \mathit{at\_floor}(\mathit{e\_left}, Y_1) \circ \mathit{at\_floor}(\mathit{e\_right}, Y_2) \circ Z \,\wedge \\ Y_1 \neq 1 \,\wedge\, Y_2 \neq 1 \,\wedge\, (\forall X, Z')\, Z \neq \mathit{inside}(X) \circ Z' \end{aligned}$$

it follows that

$$\mathit{poss}(\mathit{if}\,(\mathit{at\_floor}(\mathit{e\_left}, 1), \mathit{enter}(\mathit{e\_left}), \mathit{enter}(\mathit{e\_right})), \mathit{do}(\mathit{call\_floor}(1), s_0))$$

Disjunctive Events

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

$$\begin{array}{rcl} \textit{happens}(\textit{arrives\_left}(Y), T) & :\Leftrightarrow & \text{at time } T\text{, the left elevator arrives when calling at floor } Y \\ \textit{happens}(\textit{arrives\_right}(Y), T) & :\Leftrightarrow & \text{at time } T\text{, the right elevator arrives when calling at floor } Y \end{array}$$

$$\begin{array}{c} \textit{happens}(\textit{call\_floor}(Y), T) \wedge (\neg\exists X)\, \textit{holds\_at}(\textit{at\_floor}(X, Y), T) \ \rightarrow \\ \textit{happens}(\textit{arrives\_left}(Y), T) \vee \textit{happens}(\textit{arrives\_right}(Y), T) \end{array}$$

Modeling the Elevator Domain in the Event Calculus

Let E be,

$$\mathit{initiates}(\mathit{at\_floor}(\mathit{e\_left}, Y), \mathit{arrives\_left}(Y), T)$$
$$\mathit{terminates}(\mathit{at\_floor}(\mathit{e\_left}, Y), \mathit{arrives\_left}(Z), T) \leftarrow Y \neq Z$$
$$\mathit{initiates}(\mathit{at\_floor}(\mathit{e\_right}, Y), \mathit{arrives\_right}(Y), T)$$
$$\mathit{terminates}(\mathit{at\_floor}(\mathit{e\_right}, Y), \mathit{arrives\_right}(Z), T) \leftarrow Y \neq Z$$

Let I be,

$$(\exists Y, Z)\ \mathit{initially}(\mathit{at\_floor}(\mathit{e\_left}, Y)) \land \mathit{initially}(\mathit{at\_floor}(\mathit{e\_right}, Z)) \land Y \neq 1 \land Z \neq 1$$

Let N be,

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

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

$$T > 5 \rightarrow \mathit{holds\_at}(\mathit{at\_floor}(\mathit{e\_left}, 1), T) \lor \mathit{holds\_at}(\mathit{at\_floor}(\mathit{e\_right}, 1), T)$$

Literature

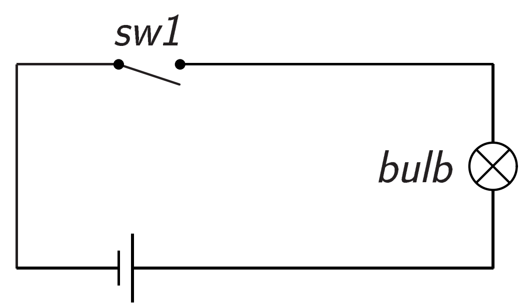
- ▷ F. Lin: Embracing causality in specifying the indeterminate effects of actions. In: B. Clancey and D. Weld (ed.'s), Proceedings of the AAAI National Conference on Artificial Intelligence, pp. 670–676. MIT Press 1996.
- ▷ M. Thielscher: Nondeterministic actions in the Fluent Calculus: disjunctive state update axioms. In: S. Hölldobler (ed.), Intellectics and Computational Logic. Kluwer Academic 1999.
- ▷ M. Shanahan: Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. MIT Press 1997. (Chapter 15).

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.



$\neg \mathit{holds}(\mathit{closed}(\mathit{sw1}), s_0)$
$\neg \mathit{holds}(\mathit{light}(\mathit{bulb}), s_0)$

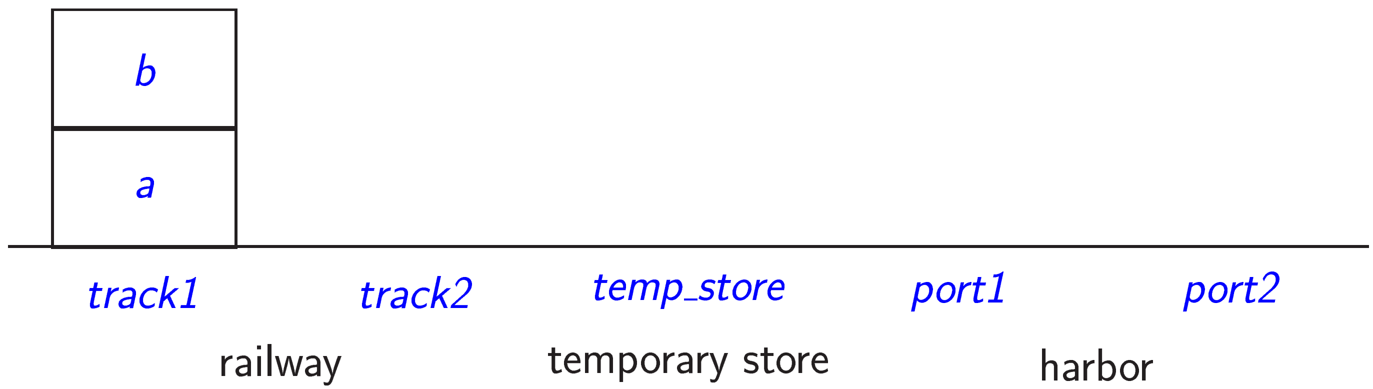
direct effect of $\mathit{toggle}(\mathit{sw1})$: $\mathit{holds}(\mathit{closed}(\mathit{sw1}), \mathit{do}(\mathit{toggle}(\mathit{sw1}), s_0))$

indirect effect of $\mathit{toggle}(\mathit{sw1})$: $\mathit{holds}(\mathit{on}(\mathit{light}), \mathit{do}(\mathit{toggle}(\mathit{sw1}), s_0))$

state constraint : $(\forall S)\ \mathit{holds}(\mathit{on}(\mathit{light}), S) \leftrightarrow \mathit{holds}(\mathit{closed}(\mathit{sw1}), S)$

Another Example: Moving Two Containers Simultaneously

$\mathit{holds}(\mathit{at}(X, Y), S)$:⇔ container X is at location Y in situation S



$$\mathit{holds}(\mathit{on}(b, a), s_0) \wedge \mathit{holds}(\mathit{at}(a, \mathit{track1}), s_0) \wedge \mathit{holds}(\mathit{at}(b, \mathit{track1}), s_0)$$

direct effect : $\mathit{holds}(\mathit{at}(a, \mathit{port2}), \mathit{do}(\mathit{move}(a, \mathit{port2}), s_0))$

indirect effect : $\mathit{holds}(\mathit{at}(b, \mathit{port2}), \mathit{do}(\mathit{move}(a, \mathit{port2}), s_0))$

state constraint : $(\forall X, Y, L, S)\ \mathit{holds}(\mathit{at}(X, L), S) \wedge \mathit{holds}(\mathit{on}(Y, X), S) \rightarrow \mathit{holds}(\mathit{at}(Y, L), S)$

A First Approach: Compiling Away the Problem

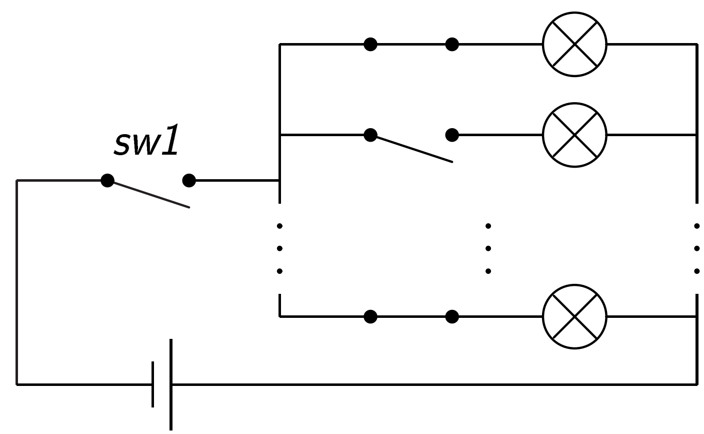
▷ Encode direct and indirect effects in successor state (or: state update, or: effect) axioms.

$$
\begin{aligned}
&\mathit{poss}(A,S) \rightarrow \\
&\quad [\ \mathit{holds}(\mathit{at}(X,L),\mathit{do}(A,S)) \leftrightarrow \\
&\qquad a = \mathit{move}(X,L) \vee \\
&\qquad (\exists Y)\ a = \mathit{move}(Y,L) \wedge \mathit{holds}(\mathit{on}(X,Y),S) \vee \\
&\qquad \mathit{holds}(\mathit{at}(X,L),S) \wedge \neg(\exists L')\, A = \mathit{move}(X,L') \\
&\qquad\qquad \wedge \neg(\exists Y,L')\, A = \mathit{move}(Y,L') \wedge \mathit{holds}(\mathit{on}(X,Y),S)\ ]
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{poss}(A,S) \rightarrow \\
&\quad [\ \mathit{holds}(\mathit{light}(X),\mathit{do}(A,S)) \leftrightarrow \\
&\qquad A = \mathit{toggle}(\mathit{sw1}) \wedge \neg \mathit{holds}(\mathit{closed}(\mathit{sw1}),S) \wedge X = \mathit{bulb} \vee \\
&\qquad \mathit{holds}(\mathit{light}(X),S) \wedge \neg\,(A = \mathit{toggle}(\mathit{sw1}) \wedge \mathit{holds}(\mathit{closed}(\mathit{sw1}),S) \wedge X = \mathit{bulb})\ ]
\end{aligned}
$$

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.



⇝ With ordinary state update axioms, this examples requires 2^{n+1} update axioms for *toggle*(*sw1*).

⇝ Adding just another switch-bulb pair would amount to rewriting the successor state axiom for *light*(X) rather than just adding the corresponding new state constraint.

An Even More Serious Problem

$$holds(on(b, a), s_0) \wedge holds(at(a, track1), s_0) \wedge holds(at(b, track1), s_0) \wedge holds(at(b, track1), s_0)$$

dir. eff. : $holds(at(a, port2), do(move(a, port2), s_0))$

indir. eff. : $holds(at(b, port2), do(move(a, port2), s_0))$; $holds(at(c, port2), do(move(a, port2), s_0))$

constr. : $(\forall X, Y, L, S)\ holds(at(X, L), S) \wedge holds(on(Y, X), S) \rightarrow holds(at(Y, L), S)$

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

A First Solution: The Frame-/Non-Frame Distinction

▷ Idea: Only frame fluents have successor state axioms.

f frame fluent :⇔ f directly manipulated by actions, never an indirect effect

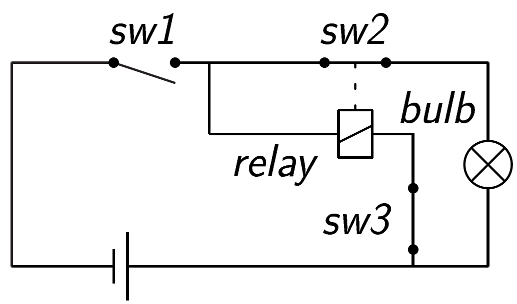
f non-frame fluent :⇔ f never directly manipulated by actions,
f completely determined by all frame fluents

$$\begin{array}{l} \mathit{poss}(A,S) \rightarrow \\ \quad [\ \mathit{holds}(\mathit{closed}(X), \mathit{do}(A,S)) \leftrightarrow \\ \qquad A = \mathit{toggle}(X) \wedge \neg \mathit{holds}(\mathit{closed}(X), S) \vee \\ \qquad \mathit{holds}(\mathit{closed}(X), S) \wedge A \neq \mathit{toggle}(X)\ ] \end{array}$$

$$\mathit{holds}(\mathit{on}(\mathit{light}), S) \leftrightarrow \mathit{holds}(\mathit{closed}(\mathit{sw1}), S)$$

Why this Solution is Restricted

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



$$\neg holds(closed(sw1), s_0)$$
$$holds(closed(sw2), s_0)$$
$$holds(closed(sw3), s_0)$$
$$\neg holds(active(relay), s_0)$$
$$\neg holds(light(bulb), s_0)$$

$$holds(light(bulb), S) \leftrightarrow holds(closed(sw1), S) \wedge holds(closed(sw2), S)$$
$$holds(active(relay), S) \leftrightarrow holds(closed(sw1), S) \wedge holds(closed(sw3), S)$$
$$holds(active(relay), S) \rightarrow \neg holds(closed(sw2), S)$$

Frame fluents are: $closed(sw1)$, $closed(sw3)$
Non-frame fluents are: $active(relay)$, $light(bulb)$

But what about $closed(sw2)$?

Causality in the Situation Calculus: Example

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

$$\mathit{caused}(F, V, S) \quad :\Leftrightarrow \quad \text{fluent } F \text{ is caused to have truth value } V \text{ in situation } S$$

$$\begin{aligned}
&\mathit{poss}(\mathit{toggle}(X), S) \rightarrow \\
&\qquad \neg \mathit{holds}(\mathit{closed}(X), S) \rightarrow \mathit{caused}(\mathit{closed}(X), \mathit{true}, \mathit{do}(\mathit{toggle}(X), S)) \\
&\mathit{poss}(\mathit{toggle}(X), S) \rightarrow \\
&\qquad \mathit{holds}(\mathit{closed}(X), S) \rightarrow \mathit{caused}(\mathit{closed}(X), \mathit{false}, \mathit{do}(\mathit{toggle}(X), S))
\end{aligned}$$

$$\begin{aligned}
&\mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \land \mathit{holds}(\mathit{closed}(\mathit{sw2}), S) \rightarrow \mathit{caused}(\mathit{light}(\mathit{bulb}), \mathit{true}, S) \\
&\neg \mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \lor \neg \mathit{holds}(\mathit{closed}(\mathit{sw2}), S) \rightarrow \mathit{caused}(\mathit{light}(\mathit{bulb}), \mathit{false}, S) \\
&\mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \land \mathit{holds}(\mathit{closed}(\mathit{sw3}), S) \rightarrow \mathit{caused}(\mathit{active}(\mathit{relay}), \mathit{true}, S) \\
&\neg \mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \lor \neg \mathit{holds}(\mathit{closed}(\mathit{sw3}), S) \rightarrow \mathit{caused}(\mathit{active}(\mathit{relay}), \mathit{false}, S) \\
&\mathit{holds}(\mathit{active}(\mathit{relay}), S) \rightarrow \mathit{caused}(\mathit{closed}(\mathit{sw2}), \mathit{false}, S)
\end{aligned}$$

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 ach action a:

$$\textit{poss}(a(\overline{X}), S) \;\rightarrow\; \Phi(S) \;\rightarrow\; \textit{caused}(f(\overline{Y}), v, \textit{do}(a(\overline{X}), S))$$

- ▷ Causal rules $\mathcal{F}_{cr}$:

$$\Phi(S) \wedge \textit{caused}(f_1, v_1, S) \wedge \ldots \wedge \textit{caused}(f_n, v_n, S) \;\rightarrow\; \textit{caused}(f(\overline{X}), v, S)$$

- ▷ CIRC$[\mathcal{F}_{ef} \wedge \mathcal{F}_{cr}; \textit{caused}]$
- ▷ Precondition axioms $\mathcal{F}_{ap}$ for each action a:

$$\pi_a(S) \;\rightarrow\; \textit{poss}(a, S)$$

The General Approach (2)

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

$$\mathit{caused}(F, \mathit{true}, S) \;\rightarrow\; \mathit{holds}(F, S)$$

$$\mathit{caused}(F, \mathit{false}, S) \;\rightarrow\; \neg \mathit{holds}(F, S)$$

$$\mathit{true} \neq \mathit{false} \;\wedge\; (\forall V : truth\_value)\; V = \mathit{true} \;\vee\; V = \mathit{false}$$

▷ Unique names assumptions $\mathcal{F}_{una}$ and $\mathcal{F}_{uns}$

Generating Successor State Axioms

$\mathcal{F}_{cr}$ is stratified $:\Leftrightarrow$ no chain of fluents $f_0, f_1, \ldots, f_n$ exists such that $f_0 \rightsquigarrow f_1 \rightsquigarrow \ldots \rightsquigarrow f_n \rightsquigarrow f_0$
where $f \rightsquigarrow f'$ $:\Leftrightarrow$ $\mathcal{F}_{cr}$ 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

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

and the schema

$$\begin{aligned} &\mathit{poss}(A,S) \rightarrow \\ &\quad [\,\mathit{holds}(f(\overline{X}), \mathit{do}(A,S)) \leftrightarrow \mathit{caused}(f(\overline{X}), \mathit{true}, \mathit{do}(A,S)) \vee \\ &\qquad\qquad \mathit{holds}(f(\overline{X}), S) \wedge \neg\mathit{caused}(f(\overline{X}), \mathit{false}, \mathit{do}(A,S))\,] \end{aligned}$$

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)

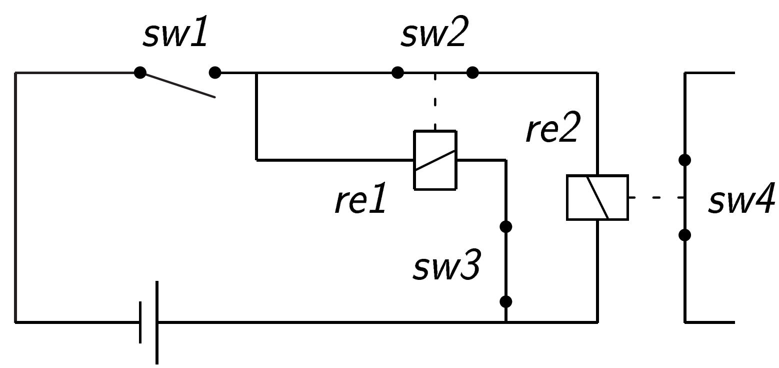
▷ The proposition on slide [93] does not apply if $\mathcal{F}_{cr}$ is not stratified, as in

$$\textit{holds}(\textit{joined}(X,Y),S) \wedge \textit{holds}(\textit{closed}(X),S) \rightarrow \textit{caused}(closed(Y),\textit{true},S)$$
$$\textit{holds}(\textit{joined}(X,Y),S) \wedge \neg\textit{holds}(\textit{closed}(X),S) \rightarrow \textit{caused}(closed(Y),\textit{false},S)$$

```
holds(joined(X,Y), do(A,S)) :- holds(joined(X,Y), S).

holds(closed(X), do(A,S)) :-
   A=toggle(X), not holds(closed(X), S) ;
   holds(joined(Y,X), do(A,S)), holds(closed(Y), do(A,S)) ;
   holds(closed(X), S),
      not A=toggle(X), not ( holds(joined(Y,X), do(A,S)),
                             not holds(closed(Y), do(A,S)) ).

holds(joined(sw1,sw2), s0).        holds(joined(sw2,sw1), s0).
holds(closed(sw1), s0).            holds(closed(sw2), s0).

| ?- holds(closed(X), do(idle,s0)).
```

The query diverges!

Restrictions of this Approach (2)



$\neg \mathit{holds}(\mathit{closed}(\mathit{sw1}), s_0)$
$\mathit{holds}(\mathit{closed}(\mathit{sw2}), s_0)$
$\mathit{holds}(\mathit{closed}(\mathit{sw3}), s_0)$
$\mathit{holds}(\mathit{closed}(\mathit{sw4}), s_0)$
$\neg \mathit{holds}(\mathit{active}(\mathit{re1}), s_0)$
$\neg \mathit{holds}(\mathit{active}(\mathit{re2}), s_0)$

$$\mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \wedge \mathit{holds}(\mathit{closed}(\mathit{sw2}), S) \rightarrow \mathit{caused}(\mathit{active}(\mathit{re2}), \mathit{true}, S)$$
$$\neg \mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \vee \neg \mathit{holds}(\mathit{closed}(\mathit{sw2}), S) \rightarrow \mathit{caused}(\mathit{active}(\mathit{re2}), \mathit{false}, S)$$
$$\mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \wedge \mathit{holds}(\mathit{closed}(\mathit{sw3}), S) \rightarrow \mathit{caused}(\mathit{active}(\mathit{re1}), \mathit{true}, S)$$
$$\neg \mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \vee \neg \mathit{holds}(\mathit{closed}(\mathit{sw3}), S) \rightarrow \mathit{caused}(\mathit{active}(\mathit{re1}), \mathit{false}, S)$$
$$\mathit{holds}(\mathit{active}(\mathit{re1}), S) \rightarrow \mathit{caused}(\mathit{closed}(\mathit{sw2}), \mathit{false}, S)$$
$$\mathit{holds}(\mathit{active}(\mathit{re2}), S) \rightarrow \mathit{caused}(\mathit{closed}(\mathit{sw4}), \mathit{false}, S)$$

▷ The possible indirect effect $\neg \mathit{holds}(\mathit{closed}(\mathit{sw4}), \mathit{do}(\mathit{toggle}(\mathit{sw1}), s_0))$ 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.

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

$$\mathit{holds\_in}(E_1, E) \stackrel{\text{def}}{=} (\exists E')\ E = E_1 \circ E'$$

Two examples of causal relationships:

$$\mathit{causes}(Z \circ \mathit{closed}(\mathit{sw2}),\ E \circ \mathit{active}(\mathit{relay}),\ Z,\ E \circ \mathit{active}(\mathit{relay}) \circ \neg\mathit{closed}(\mathit{sw2}))$$

$$\mathit{holds\_in}(\mathit{on}(X,Y), Z) \rightarrow$$
$$\mathit{causes}(Z \circ \mathit{at}(X,L),\ E \circ \mathit{at}(Y,L'),\ Z \circ \mathit{at}(X,L'),\ E \circ \mathit{at}(Y,L') \circ \neg\mathit{at}(X,L) \circ \mathit{at}(X,L'))$$

The General Approach (1)

▷ Sort EFFECTS such that

- STATE < EFFECTS
- $\Sigma_{Eff} = \{-\}$ where $- : \text{FLUENT} \mapsto \text{EFFECTS}$

▷ State constraints $\mathcal{F}_{sc}$

▷ Causal relationships $\mathcal{F}_{cr}$:

$$\Phi \;\rightarrow\; \mathit{causes}(Z_1, E_1, Z_2, E_2)$$

where Φ is an equational formula

▷ COMP$[\mathcal{F}_{cr}]$

The General Approach (2)

$$\textit{ramify}(Z, E, Z') \quad :\Leftrightarrow \quad \text{state } Z' \text{ is reachable from state } Z \text{ and effects } E$$

▷ Second-order foundational axiom $\mathcal{F}_{ra}$:

$$\textit{ramify}(Z, E, Z') \leftrightarrow \forall \Pi \left\{ \begin{array}{c} (\forall Z_1, E_1)\ \Pi(Z_1, E_1, Z_1, E_1) \\ \wedge \\ \left[\begin{array}{l} (\forall Z_1, E_1, Z_2, E_2, Z_3, E_3) \\ \quad \Pi(Z_1, E_1, Z_2, E_2) \wedge \textit{causes}(Z_2, E_2, Z_3, E_3) \\ \qquad \rightarrow \Pi(Z_1, E_1, Z_3, E_3) \end{array} \right] \\ \rightarrow \\ (\exists E')\ \Pi(Z, E, Z', E') \end{array} \right\}$$

▷ State update axioms $\mathcal{F}_{sua}$:

$$\begin{array}{l} \Delta(S) \rightarrow \\ \quad Z \circ \vartheta^- = \textit{state}(S) \circ \vartheta^+ \rightarrow \\ \qquad \textit{ramify}(Z, \vartheta^+ \circ -\vartheta^-, \textit{state}(\textit{do}(a, S))) \end{array}$$

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

The Joined Switches in the Fluent Calculus

$$\mathcal{F}_{sc} \;:\Leftrightarrow\; \mathit{holds}(\mathit{joined}(X,Y),S) \rightarrow [\, \mathit{holds}(\mathit{closed}(Y),S) \leftrightarrow \mathit{holds}(\mathit{closed}(X),S)\,]$$

$$\mathcal{F}_{ap} \;:\Leftrightarrow\; \mathit{poss}(\mathit{toggle}(X),S)$$

$$\begin{aligned}\mathcal{F}_{sua} \;:\Leftrightarrow\; & \mathit{poss}(\mathit{toggle}(X),S) \wedge \neg \mathit{holds}(\mathit{closed}(X),S) \rightarrow \\ & \quad Z = \mathit{state}(S) \circ \mathit{closed}(X) \rightarrow \\ & \quad\quad \mathit{ramify}(Z, \mathit{closed}(X), \mathit{state}(\mathit{do}(\mathit{toggle}(X),S))) \\ & \mathit{poss}(\mathit{toggle}(X),S) \wedge \mathit{holds}(\mathit{closed}(X),S) \rightarrow \\ & \quad Z \circ \mathit{closed}(X) = \mathit{state}(S) \rightarrow \\ & \quad\quad \mathit{ramify}(Z, -\mathit{closed}(X), \mathit{state}(\mathit{do}(\mathit{toggle}(X),S)))\end{aligned}$$

$$\mathrm{COMP}[\mathcal{F}_{cr}] \;:\Leftrightarrow$$

$$\begin{aligned}& \mathit{causes}(Z_1, E_1, Z_2, E_2) \leftrightarrow \\ & \quad \mathit{holds\_in}(\mathit{closed}(X), E_1) \wedge \mathit{holds\_in}(\mathit{joined}(X,Y), Z_1) \wedge \neg \mathit{holds\_in}(\mathit{closed}(Y), Z_1) \wedge \\ & \quad\quad Z_2 = Z_1 \circ \mathit{closed}(Y) \wedge E_2 = E_1 \circ \mathit{closed}(Y) \vee \\ & \quad \mathit{holds\_in}(-\mathit{closed}(X), E_1) \wedge \mathit{holds\_in}(\mathit{joined}(X,Y), Z_1) \wedge \mathit{holds\_in}(\mathit{closed}(Y), Z_1) \wedge \\ & \quad\quad Z_2 \circ \mathit{closed}(Y) = Z_1 \wedge E_2 = E_1 \circ -\mathit{closed}(Y)\end{aligned}$$

The Double Relay Circuit in the Fluent Calculus (1)

$$
\begin{aligned}
\mathcal{F}_{sc} \;:\Leftrightarrow\; & \mathit{holds}(\mathit{active}(\mathit{re1}), S) \leftrightarrow \mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \wedge \mathit{holds}(\mathit{closed}(\mathit{sw3}), S) \\
& \mathit{holds}(\mathit{active}(\mathit{re2}), S) \leftrightarrow \mathit{holds}(\mathit{closed}(\mathit{sw1}), S) \wedge \mathit{holds}(\mathit{closed}(\mathit{sw2}), S) \\
& \mathit{holds}(\mathit{active}(\mathit{re1}), S) \rightarrow \neg \mathit{holds}(\mathit{closed}(\mathit{sw2}), S) \\
& \mathit{holds}(\mathit{active}(\mathit{re2}), S) \rightarrow \neg \mathit{holds}(\mathit{closed}(\mathit{sw4}), S)
\end{aligned}
$$

$$
\mathcal{F}_{ap} \;:\Leftrightarrow\; \mathit{poss}(\mathit{toggle}(X), S)
$$

$$
\begin{aligned}
\mathcal{F}_{sua} \;:\Leftrightarrow\; & \mathit{poss}(\mathit{toggle}(X), S) \wedge \neg \mathit{holds}(\mathit{closed}(X), S) \rightarrow \\
& \qquad Z = \mathit{state}(S) \circ \mathit{closed}(X) \rightarrow \\
& \qquad\qquad \mathit{ramify}(Z, \mathit{closed}(X), \mathit{state}(\mathit{do}(\mathit{toggle}(X), S))) \\
& \\
& \mathit{poss}(\mathit{toggle}(X), S) \wedge \mathit{holds}(\mathit{closed}(X), S) \rightarrow \\
& \qquad Z \circ \mathit{closed}(X) = \mathit{state}(S) \rightarrow \\
& \qquad\qquad \mathit{ramify}(Z, -\mathit{closed}(X), \mathit{state}(\mathit{do}(\mathit{toggle}(X), S)))
\end{aligned}
$$

The Double Relay Circuit in the Fluent Calculus (2)

COMP$[\mathcal{F}_{cr}]$ $:\Leftrightarrow$

$$
\begin{aligned}
& \mathit{causes}(Z_1, E_1, Z_2, E_2) \ \leftrightarrow \\
& \quad \mathit{holds\_in}(\mathit{closed}(\mathit{sw1}), E_1) \wedge \mathit{holds\_in}(\mathit{closed}(\mathit{sw3}), Z_1) \wedge \neg \mathit{holds\_in}(\mathit{active}(\mathit{re1}), Z_1) \wedge \\
& \qquad Z_2 = Z_1 \circ \mathit{active}(\mathit{re1}) \wedge E_2 = E_1 \circ \mathit{active}(\mathit{re1}) \ \vee \\
& \quad \mathit{holds\_in}(\mathit{closed}(\mathit{sw3}), E_1) \wedge \mathit{holds\_in}(\mathit{closed}(\mathit{sw1}), Z_1) \wedge \neg \mathit{holds\_in}(\mathit{active}(\mathit{re1}), Z_1) \wedge \\
& \qquad Z_2 = Z_1 \circ \mathit{active}(\mathit{re1}) \wedge E_2 = E_1 \circ \mathit{active}(\mathit{re1}) \ \vee \\
& \quad \mathit{holds\_in}(-\mathit{closed}(\mathit{sw1}), E_1) \wedge \mathit{holds\_in}(\mathit{active}(\mathit{re1}), Z_1) \wedge \\
& \qquad Z_2 \circ \mathit{active}(\mathit{re1}) = Z_1 \wedge E_2 = E_1 \circ -\mathit{active}(\mathit{re1}) \ \vee \\
& \quad \mathit{holds\_in}(-\mathit{closed}(\mathit{sw3}), E_1) \wedge \mathit{holds\_in}(\mathit{active}(\mathit{re1}), Z_1) \wedge \\
& \qquad Z_2 \circ \mathit{active}(\mathit{re1}) = Z_1 \wedge E_2 = E_1 \circ -\mathit{active}(\mathit{re1}) \ \vee \\
& \quad \ldots
\end{aligned}
$$

The Double Relay Circuit in the Fluent Calculus (3)

$$
\begin{aligned}
&\ldots\\
&\mathit{holds\_in}(\mathit{closed}(\mathit{sw1}), E_1) \wedge \mathit{holds\_in}(\mathit{closed}(\mathit{sw2}), Z_1) \wedge \neg \mathit{holds\_in}(\mathit{active}(\mathit{re2}), Z_1) \wedge\\
&\quad Z_2 = Z_1 \circ \mathit{active}(\mathit{re2}) \wedge E_2 = E_1 \circ \mathit{active}(\mathit{re2}) \vee\\
&\mathit{holds\_in}(\mathit{closed}(\mathit{sw2}), E_1) \wedge \mathit{holds\_in}(\mathit{closed}(\mathit{sw1}), Z_1) \wedge \neg \mathit{holds\_in}(\mathit{active}(\mathit{re2}), Z_1) \wedge\\
&\quad Z_2 = Z_1 \circ \mathit{active}(\mathit{re2}) \wedge E_2 = E_1 \circ \mathit{active}(\mathit{re2}) \vee\\
&\mathit{holds\_in}(-\mathit{closed}(\mathit{sw1}), E_1) \wedge \mathit{holds\_in}(\mathit{active}(\mathit{re2}), Z_1) \wedge\\
&\quad Z_2 \circ \mathit{active}(\mathit{re2}) = Z_1 \wedge E_2 = E_1 \circ -\mathit{active}(\mathit{re2}) \vee\\
&\mathit{holds\_in}(-\mathit{closed}(\mathit{sw2}), E_1) \wedge \mathit{holds\_in}(\mathit{active}(\mathit{re2}), Z_1) \wedge\\
&\quad Z_2 \circ \mathit{active}(\mathit{re2}) = Z_1 \wedge E_2 = E_1 \circ -\mathit{active}(\mathit{re2}) \vee
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{holds\_in}(\mathit{active}(\mathit{re1}), E_1) \wedge \mathit{holds\_in}(\mathit{closed}(\mathit{sw2}), Z_1) \wedge\\
&\quad Z_2 \circ \mathit{closed}(\mathit{sw2}) = Z_1 \wedge E_2 = E_1 \circ -\mathit{closed}(\mathit{sw2})\\
&\mathit{holds\_in}(\mathit{active}(\mathit{re2}), E_1) \wedge \mathit{holds\_in}(\mathit{closed}(\mathit{sw4}), Z_1) \wedge\\
&\quad Z_2 \circ \mathit{closed}(\mathit{sw4}) = Z_1 \wedge E_2 = E_1 \circ -\mathit{closed}(\mathit{sw4})
\end{aligned}
$$

Comments

- ▷ 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

- F. Lin: Embracing causality in specifying the indirect effects of actions. In: C. S. Mellish (ed.), Proceedings of the International Joint Conference on AI, pp. 1985–1991. Morgan Kaufmann 1995.
- M. Thielscher: Computing Ramifications by Postprocessing. In: C. S. Mellish (ed.), Proceedings of the International Joint Conference on AI, pp. 1994–2000. Morgan Kaufmann 1995.
- M. Thielscher: Ramification and causality. Artificial Intelligence Journal 89(1–2): 317–364, 1997.
- M. Thielscher: Reasoning about actions: steady versus stabilizing state constraints. Artificial Intelligence Journal 104: 339–355, 1998.

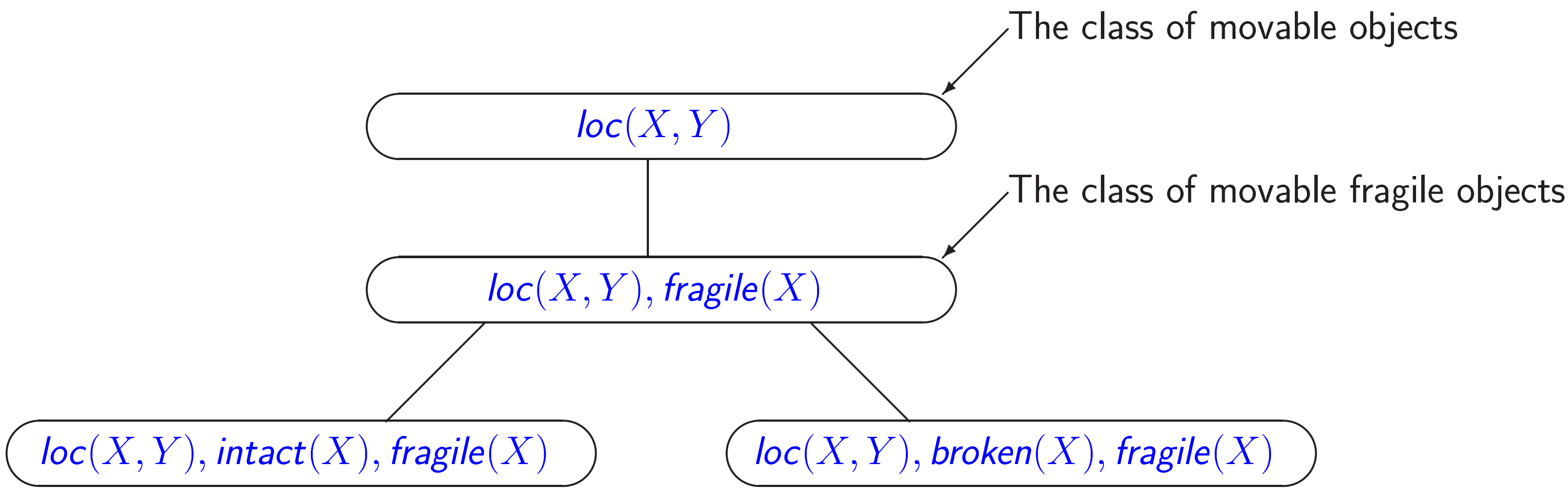
Specificity

- ▷ 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 overriden if more specific methods are defined.

An Example Hierarchy of Classes



$loc(X, Y)$: object X is at location Y
$fragile(X)$: object X is fragile
$broken(X)$: object X is broken
$intact(X)$: object X is intact

Formalizing Objects and Classes

- An object is a ground constructor state term, eg.
  $\mathit{loc}(\mathit{nugget}, \mathit{table})$ or
  $\mathit{loc}(\mathit{vase}, \mathit{table}) \circ \mathit{fragile}(\mathit{vase})$ or
  $\mathit{loc}(\mathit{vase}, \mathit{floor}) \circ \mathit{fragile}(\mathit{vase}) \circ \mathit{broken}(\mathit{vase})$.

- A class is a constructor state term without occurrences of constants, eg.
  $\mathit{loc}(X, Y)$ or
  $\mathit{loc}(X, Y) \circ \mathit{fragile}(X)$ or
  $\mathit{loc}(X, Y) \circ \mathit{fragile}(X) \circ \mathit{broken}(X)$.

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

- The initial state is a ground constructor state term and is denoted by the situation s_0, eg.
  $\mathit{state}(s_0) = \mathit{loc}(\mathit{nugget}, \mathit{table})$ or
  $\mathit{state}(s_0) = \mathit{loc}(\mathit{nugget}, \mathit{table}) \circ \mathit{loc}(\mathit{vase}, \mathit{table}) \circ \mathit{fragile}(\mathit{vase})$.

Formalizing Methods

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

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

- Inheritance comes for free: Let $\mathit{state}(s_0) = \mathit{loc}(\mathit{vase},\mathit{table}) \circ \mathit{fragile}(\mathit{vase})$ then
  $\mathit{state}(\mathit{do}(\mathit{move}(\mathit{vase},\mathit{table},\mathit{cupboard}),s_0)) = \mathit{loc}(\mathit{vase},\mathit{cupboard}) \circ \mathit{fragile}(\mathit{vase})$.

Dropping Objects

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

▷ For the class of movable objects we define

$$\begin{aligned}&\mathit{holds}(\mathit{loc}(X,Y),S)\\ &\to\ \mathit{state}(\mathit{do}(\mathit{drop}(X),S))\circ \mathit{loc}(X,Y)=\mathit{state}(S)\circ \mathit{loc}(X,\mathit{floor})\end{aligned}\tag{1}$$

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

$$\mathit{state}(\mathit{do}(\mathit{drop}(\mathit{nugget}),s_0))=\mathit{loc}(\mathit{nugget},\mathit{floor}).$$

How about Overriding?

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

$$\begin{aligned}&\mathit{holds}(\mathit{loc}(X,Y),S) \wedge \mathit{holds}(\mathit{fragile}(X),S)\\ \rightarrow\;& \mathit{state}(\mathit{do}(\mathit{drop}(X),S)) \circ \mathit{loc}(X,Y) = \mathit{state}(S) \circ \mathit{loc}(X,\mathit{floor}) \circ \mathit{broken}(X)\end{aligned} \tag{2}$$

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

$$\mathit{state}(\mathit{do}(\mathit{drop}(\mathit{vase}),s_0)) = \mathit{loc}(\mathit{vase},\mathit{floor}) \circ \mathit{fragile}(\mathit{vase}) \circ \mathit{broken}(\mathit{vase}).$$

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

$$\mathit{state}(\mathit{do}(\mathit{drop}(\mathit{vase}),s_0)) = \mathit{loc}(\mathit{vase},\mathit{floor}) \circ \mathit{fragile}(\mathit{vase}).$$

⇝ There is a contradiction concerning the fluent $\mathit{broken}(\mathit{vase})$.

⇝ We would like to block the application of (1).

Specificity

▷ Let $\mathit{cond}(1, \mathit{drop}(X), X, Y, S) \leftrightarrow \mathit{holds}(\mathit{loc}(X, Y), S)$
$\mathit{cond}(2, \mathit{drop}(X), X, Y, S) \leftrightarrow \mathit{holds}(\mathit{loc}(X, Y), S) \wedge \mathit{holds}(\mathit{fragile}(X), S)$

$\leadsto$ $\mathit{cond}(1, \mathit{drop}(X), X, Y, S)$ subsumes $\mathit{cond}(2, \mathit{drop}(X), X, Y, S)$.

▷ A conditional is an equivalence of the form $\mathit{cond}(N, a, \overline{X}, S) \leftrightarrow \Delta(\overline{X}, S)$, where N is a natural number, a a term of sort ACTION, $\overline{X}$ a list of variables of sort OBJECT, S a variable of sort $\mathrm{SITUATION}$ 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 *cond*.

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

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

$\leadsto$ $\mathit{cond}(2, \mathit{drop}(X), X, Y, S)$ is strictly more specific $\mathit{cond}(1, \mathit{drop}(X), X, Y, S)$.

Most Specific Actions

- $\mathit{sms}(N, M, a, \overline{X}, S)$
  $\stackrel{def}{=} \; [\mathit{cond}(N, a, \overline{X}, S) \rightarrow \mathit{cond}(M, a, \overline{X}, S)] \wedge \neg[\mathit{cond}(M, a, \overline{X}, S) \rightarrow \mathit{cond}(N, a, \overline{X}, S)]$

  $\rightsquigarrow$ $\mathit{sms}(2, 1, \mathit{drop}(X), X, Y, S)$

- Idea: Apply only most specific applicable action.

- Transform (1) into

$$\begin{aligned} &\mathit{cond}(1, \mathit{drop}(X), X, Y, S) \wedge \neg(\exists N)\; \mathit{sms}(N, 1, \mathit{drop}(X), X, Y, S) \\ &\rightarrow\; \mathit{state}(\mathit{do}(\mathit{drop}(X), S)) \circ \mathit{loc}(X, Y) = \mathit{state}(S) \circ \mathit{loc}(X, \mathit{floor}) \end{aligned} \tag{3}$$

and (2) into

$$\begin{aligned} &\mathit{cond}(2, \mathit{drop}(X), X, Y, S) \wedge \neg(\exists N)\; \mathit{sms}(N, 1, \mathit{drop}(X), X, Y, S) \\ &\rightarrow\; \mathit{state}(\mathit{do}(\mathit{drop}(X), S)) \circ \mathit{loc}(X, Y) = \mathit{state}(S) \circ \mathit{loc}(X, \mathit{floor}) \circ \mathit{broken}(X). \end{aligned} \tag{4}$$

Circumscription

- Let

$$\mathit{state}(s_0) = \mathit{loc}(\mathit{vase}, \mathit{table}) \circ \mathit{fragile}(\mathit{vase}).$$

  - ⇝ (3) is blocked!
  - ⇝ (4) is also blocked!
  - ⇝ We have to minimize the extension of *cond* to ensure that (4) is not blocked.

- Let $\mathcal{F} = \{(3),(4)\} \cup \mathcal{F}_C$ and consider $\mathrm{CIRC}(\mathcal{F}; \mathit{cond})$ instead of $\mathcal{F}$.
  - ⇝ (3) is still blocked!
  - ⇝ (4) is applicable and yields

$$\mathit{state}(\mathit{do}(\mathit{drop}(\mathit{vase}), s_0)) = \mathit{loc}(\mathit{vase}, \mathit{floor}) \circ \mathit{fragile}(\mathit{vase}) \circ \mathit{broken}(\mathit{vase}).$$

  - ⇝ The vase is broken.

Specificity: The General Approach

- Let $\mathcal{F}_{sua}$ be the set of successor state axioms of the form

$$\Delta(\overline{X}, S) \rightarrow \mathit{state}(\mathit{do}(a, S)) \circ \vartheta^- = \mathit{state}(S) \circ \vartheta^+ . \tag{5}$$

- Let $\mathcal{F}_C$ be the set of conditionals for $\mathcal{F}$, ie. for each element (5) in $\mathcal{F}$ the set $\mathcal{F}_C$ contains an element of the form

$$\mathit{cond}(N, a, \overline{X}, S) \leftrightarrow \Delta(\overline{X}, S).$$

- Let $\mathcal{F}_E$ be the set of extended successor state axioms, which is obtained by replacing each element (5) of $\mathcal{F}$ by

$$\begin{array}{l} \mathit{cond}(N, a, \overline{X}, S) \wedge \neg(\exists M)\ \mathit{sms}(M, N, a, \overline{X}, S) \\ \rightarrow\ \mathit{state}(\mathit{do}(a, S)) \circ \vartheta^- = \mathit{state}(S) \circ \vartheta^+ \end{array}$$

  if $\mathit{cond}(N, a, \overline{X}, S) \leftrightarrow \Delta(\overline{X}, S) \in \mathcal{F}_C$.

- Let $\mathcal{F}_{euna}$ be the extended unique names assumptions.

$\rightsquigarrow$ Consider $\mathcal{F}_{euna} \cup \mathcal{F}_C \cup \mathrm{CIRC}(\mathcal{F}_E; cond)$.

Specificity and Ramification

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

$$\neg(\exists X, S, Z)\ \mathit{intact}(X) \circ \mathit{broken}(X) \circ Z = \mathit{state}(S).$$

- ▷ Let

$$\mathit{state}(s_0) = \mathit{loc}(\mathit{vase}, \mathit{table}) \circ \mathit{fragile}(\mathit{vase}) \circ \mathit{intact}(\mathit{vase})$$

- ▷ Applying (4) yields

$$\mathit{state}(\mathit{do}(\mathit{drop}(\mathit{vase}), s_0)) = \mathit{loc}(\mathit{vase}, \mathit{floor}) \circ \mathit{fragile}(X) \circ \mathit{intact}(\mathit{vase}) \circ \mathit{broken}(X)$$

which is inconsistent wrt the state constraint.

$\leadsto$ An additional ramification step yields

$$\mathit{state}(\mathit{do}(\mathit{drop}(\mathit{vase}), s_0)) = \mathit{loc}(\mathit{vase}, \mathit{floor}) \circ \mathit{fragile}(X) \circ \mathit{broken}(X).$$

Specificity and the Need to Add Methods

- In the presence of state constraints it may be necessary to add methods.
- Consider the state constraint

$$(\forall F, S, Z)\ F \circ F \circ Z \neq \mathit{state}(S).$$

- Let

$$\mathit{state}(s_0) = \mathit{loc}(\mathit{vase}, \mathit{table}) \circ \mathit{fragile}(\mathit{vase}) \circ \mathit{broken}(\mathit{vase})$$

- Applying (4) yields

$$\mathit{state}(\mathit{do}(\mathit{drop}(\mathit{vase}), s_0)) = \mathit{loc}(\mathit{vase}, \mathit{floor}) \circ \mathit{fragile}(X) \circ \mathit{fragile}(\mathit{vase}) \circ \mathit{broken}(X)$$

  which is inconsistent wrt the state constraint.

- Adding the state update axioms

$$\begin{array}{l} \mathit{holds}(\mathit{loc}(X, Y), S) \wedge \mathit{holds}(\mathit{fragile}(X), S) \wedge \mathit{holds}(\mathit{broken}(X)) \\ \rightarrow\ \mathit{state}(\mathit{do}(\mathit{drop}(X), S)) \circ \mathit{loc}(X, Y) = \mathit{state}(S) \circ \mathit{loc}(X, \mathit{floor}) \end{array}$$

  will remedy this problem. This can be done automatically.

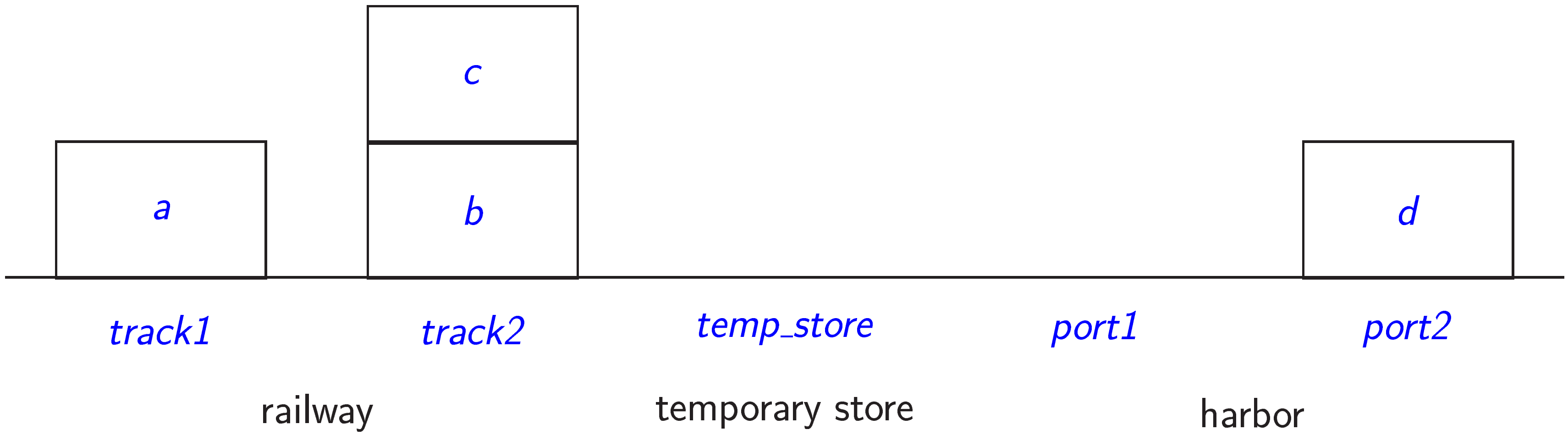
Literature

▷ S. Hölldobler and M. Thielscher: Computing Change and Specificity with Equational Logic Programs. Annals of Mathematics and Artificial Intelligence 14, 99-133: 1995.

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



event $\mathit{move}(F, X, Y)$ $:\Leftrightarrow$ fork lift truck F moves X to Y

$$\mathit{happens}(\mathit{move}(\mathit{fork1}, a, \mathit{port1}), 1) \wedge \mathit{happens}(\mathit{move}(\mathit{fork2}, d, \mathit{temp\_store}), 1)$$

The New Effect Axioms

$$\mathit{initiates}(\mathit{move}(F,X,Y),\mathit{on}(X,Y),T) \leftarrow \\ \quad \mathit{holds\_at}(\mathit{clear}(X),T) \wedge \mathit{holds\_at}(\mathit{clear}(Y),T) \wedge X \neq Y$$

$$\mathit{initiates}(\mathit{move}(F,X,Y),\mathit{clear}(Z),T) \leftarrow \\ \quad \mathit{holds\_at}(\mathit{clear}(X),T) \wedge \mathit{holds\_at}(\mathit{clear}(Y),T) \wedge \mathit{holds\_at}(\mathit{on}(X,Z),T) \wedge X \neq Y \wedge Y \neq Z$$

$$\mathit{terminates}(\mathit{move}(F,X,Y),\mathit{on}(X,Z),T) \leftarrow \\ \quad \mathit{holds\_at}(\mathit{clear}(X),T) \wedge \mathit{holds\_at}(\mathit{clear}(Y),T) \wedge \mathit{holds\_at}(\mathit{on}(X,Z),T) \wedge X \neq Y \wedge Y \neq Z$$

$$\mathit{terminates}(\mathit{move}(F,X,Y),\mathit{clear}(Y),T) \leftarrow \\ \quad \mathit{holds\_at}(\mathit{clear}(X),T) \wedge \mathit{holds\_at}(\mathit{clear}(Y),T) \wedge X \neq Y$$

Cancellation

$\mathit{cancels}(E_1, E_2, T) \quad :\Leftrightarrow \quad$ event E_1 cancels the effects of event E_2 at time T

$$\mathit{cancels}(\mathit{move}(F_1, X_1, Y_1),\ \mathit{move}(F_2, X_2, Y_2),\ T) \leftarrow$$
$$F_1 \neq F_2 \wedge (X_1 = X_2 \vee Y_1 = Y_2) \ \vee\ F_1 = F_2 \wedge (X_1 \neq X_2 \vee Y_1 \neq Y_2)$$

$$\mathit{holds\_at}(F, T) \ \leftarrow\ \mathit{initially}(F) \wedge \neg \mathit{clipped}(0, F, T)$$

$$\mathit{holds\_at}(F, T_2) \leftarrow$$
$$(\exists E)\ \mathit{happens}(E, T_1) \wedge \mathit{initiates}(E, F, T_1) \wedge T_1 < T_2 \wedge \neg \mathit{clipped}(T_1, F, T_2) \wedge$$
$$\neg(\exists E')\,[\,\mathit{happens}(E', T_1) \wedge \mathit{cancels}(E', E, T_1)\,]$$

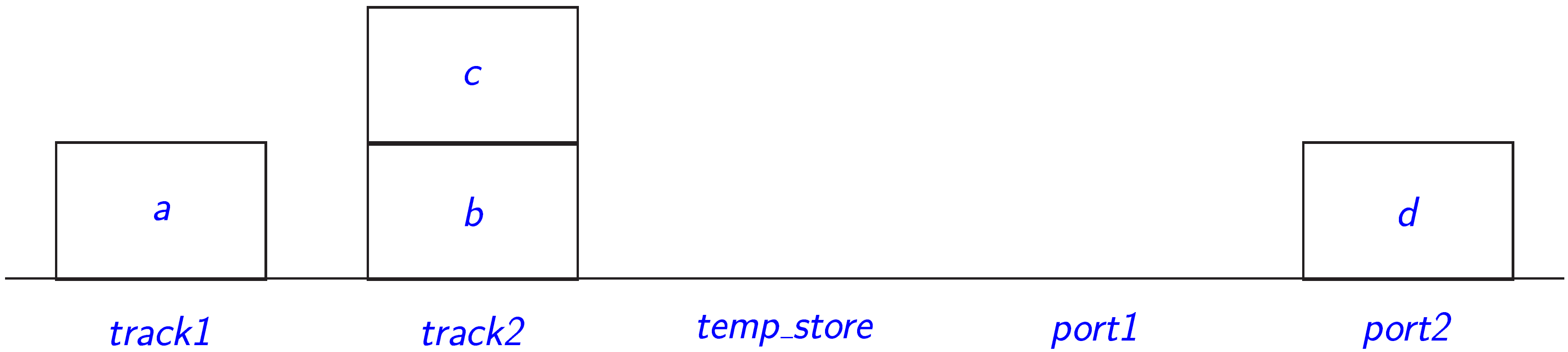
$$\mathit{clipped}(T_1, F, T_2) \leftrightarrow$$
$$(\exists E, T)\ \mathit{happens}(E, T) \wedge \mathit{terminates}(E, F, T) \wedge T_1 < T \wedge T < T_2 \wedge$$
$$\neg(\exists E')\,[\,\mathit{happens}(E', T) \wedge \mathit{cancels}(E', E, T)\,]$$

Cancellation (continued)

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

$$\mathit{declipped}(T_1, F, T_2) \leftrightarrow$$
$$(\exists E, T)\ \mathit{happens}(E, T) \wedge \mathit{initiates}(E, F, T) \wedge T_1 < T \wedge T < T_2 \wedge$$
$$\neg(\exists E')\,[\,\mathit{happens}(E', T) \wedge \mathit{cancels}(E', E, T)\,]$$

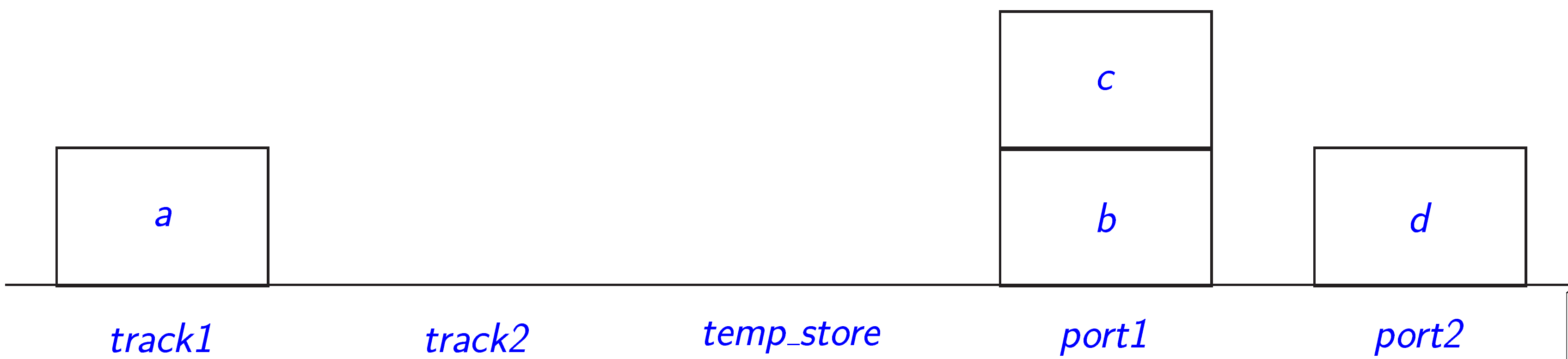
Collaboration



The concurrent events,

$$happens(move(fork1, b, port1), 1) \wedge happens(move(fork2, b, port1), 1)$$

shall now be possible, resulting in this state:



A New Event Type

$\mathit{happens}(E_1 \,\&\, E_2, T)$:⇔ events E_1 and E_2 happen concurrently at time T.

$$\mathit{happens}(E_1 \,\&\, E_2, T) \leftarrow \mathit{happens}(E_1, T) \wedge \mathit{happens}(E_2, T) \wedge E_1 \neq E_2$$

$$\begin{aligned}&\mathit{initiates}(\mathit{move}(F_1, X, Y) \,\&\, \mathit{move}(F_2, X, Y),\ \mathit{on}(X, Y),\ T) \leftarrow \\ &\quad (\exists U)\ \mathit{holds\_at}(\mathit{on}(U, X), T) \wedge \mathit{holds\_at}(\mathit{clear}(U), T) \wedge \mathit{holds\_at}(\mathit{clear}(Y), T) \wedge U \neq Y\end{aligned}$$

$$\begin{aligned}&\mathit{initiates}(\mathit{move}(F_1, X, Y) \,\&\, \mathit{move}(F_2, X, Y),\ \mathit{clear}(Z),\ T) \leftarrow \\ &\quad (\exists U)\ \mathit{holds\_at}(\mathit{on}(U, X), T) \wedge \mathit{holds\_at}(\mathit{clear}(U), T) \wedge \\ &\qquad \mathit{holds\_at}(\mathit{clear}(Y), T) \wedge \mathit{holds\_at}(\mathit{on}(X, Z), T) \wedge U \neq Y \wedge Y \neq Z\end{aligned}$$

$$\begin{aligned}&\mathit{terminates}(\mathit{move}(F_1, X, Y) \,\&\, \mathit{move}(F_2, X, Y),\ \mathit{on}(X, Z),\ T) \leftarrow \\ &\quad (\exists U)\ \mathit{holds\_at}(\mathit{on}(U, X), T) \wedge \mathit{holds\_at}(\mathit{clear}(U), T) \wedge \\ &\qquad \mathit{holds\_at}(\mathit{clear}(Y), T) \wedge \mathit{holds\_at}(\mathit{on}(X, Z), T) \wedge U \neq Y \wedge Y \neq Z\end{aligned}$$

$$\begin{aligned}&\mathit{terminates}(\mathit{move}(F_1, X, Y) \,\&\, \mathit{move}(F_2, X, Y),\ \mathit{clear}(Y),\ T) \leftarrow \\ &\quad (\exists U)\ \mathit{holds\_at}(\mathit{on}(U, X), T) \wedge \mathit{holds\_at}(\mathit{clear}(U), T) \wedge \mathit{holds\_at}(\mathit{clear}(Y), T) \wedge U \neq Y\end{aligned}$$

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

$$\mathrm{CIRC}[N \wedge I; \mathit{happens}] \wedge \mathrm{CIRC}[E; \mathit{initiates}, \mathit{terminates}, \mathit{cancels}] \wedge U \wedge EC$$

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)

```
initiates(move(F,X,Y), on(X,Y), T) :-
   holds_at(clear(X), T), holds_at(clear(Y), T), not X=Y.

initiates(move(F,X,Y), clear(Z), T) :-
   holds_at(clear(X), T), holds_at(clear(Y), T),
   holds_at(on(X,Z), T), not X=Y, not Y=Z.

terminates(move(F,X,Y), on(X,Z), T) :-
   holds_at(clear(X), T), holds_at(clear(Y), T),
   holds_at(on(X,Z), T), not X=Y, not Y=Z.

terminates(move(F,X,Y), clear(Y), T) :-
   holds_at(clear(X), T), holds_at(clear(Y), T), not X=Y.
```

The extended Event Calculus in PROLOG (3)

```
happens(E1 & E2, T) :- E1 = move(_,_,_), E2 = move(_,_,_),
                       happens(E1, T), happens(E2, T), not E1=E2.

initiates(move(F1,X,Y) & move(F2,X,Y), on(X,Y), T) :-
   holds_at(on(U,X), T), holds_at(clear(U), T),
   holds_at(clear(Y), T), not U=Y.

initiates(move(F1,X,Y) & move(F2,X,Y), clear(Z), T) :-
   holds_at(on(U,X), T), holds_at(clear(U), T),
   holds_at(clear(Y), T), holds_at(on(X,Z), T), not U=Y, not Y=Z.

terminates(move(F1,X,Y) & move(F2,X,Y), on(X,Z), T) :-
   holds_at(on(U,X), T), holds_at(clear(U), T),
   holds_at(clear(Y), T), holds_at(on(X,Z), T), not U=Y, not Y=Z.

terminates(move(F1,X,Y) & move(F2,X,Y), clear(Y), T) :-
   holds_at(on(U,X), T), holds_at(clear(U), T),
   holds_at(clear(Y), T), not U=Y.
```

The extended Event Calculus in PROLOG (4)

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

initially(clear(a)).
initially(on(c,b)).
initially(clear(temp_store)).
initially(on(d,port2)).

happens(move(fork1,a,port1),1).
happens(move(fork1,b,port2),3).
happens(move(fork1,d,track2),6).

happens(move(fork2,d,temp_store),1).
happens(move(fork2,b,port2),3).
```

```
| ?- holds_at(F, 7).

F = clear(a)
F = clear(c)
F = on(a,port1)
F = on(d,track2)
F = on(b,port2)

F = on(c,b)
F = clear(d)
F = clear(track1)
F = clear(temp_store)
```

Extending the Situation Calculus

- ▷ New sort $\mathrm{SET\_OF\_ACTION}$ $:\Leftrightarrow$ sets of actions
- ▷ *do* : $\mathrm{SET\_OF\_ACTION} \times \mathrm{SIT} \mapsto \mathrm{SIT}$
  poss : $\mathrm{SET\_OF\_ACTION} \times \mathrm{SIT}$
- ▷ Standard set functions and relations $\{\}$, $\{a_1, \ldots, a_n\}$, and $\in$.

Two Fork Lift Trucks: Precondition Axioms

$$
\begin{aligned}
&\mathit{poss}(C,S) \leftrightarrow \\
&\quad (\exists F,X,Y)\,[\,C = \{\mathit{move}(F,X,Y)\} \wedge \mathit{holds}(\mathit{clear}(X),S) \wedge \mathit{holds}(\mathit{clear}(Y),S)\,] \\
&\ \vee \\
&\quad (\exists F_1,F_2,X_1,X_2,Y_1,Y_2) \\
&\qquad [\,C = \{\mathit{move}(F_1,X_1,Y_1), \mathit{move}(F_2,X_2,Y_2)\} \wedge F_1 \neq F_2 \wedge \\
&\qquad\quad (\,X_1 \neq X_2 \rightarrow Y_1 \neq Y_2 \wedge \mathit{holds}(\mathit{clear}(X_1),S) \wedge \mathit{holds}(\mathit{clear}(Y_1),S) \wedge \\
&\qquad\qquad \mathit{holds}(\mathit{clear}(X_2),S) \wedge \mathit{holds}(\mathit{clear}(Y_2),S)\,) \\
&\qquad\quad (\,X_1 = X_2 \rightarrow Y_1 = Y_2 \wedge (\exists Z)\,\mathit{holds}(\mathit{on}(Z,X_1),S) \wedge \mathit{holds}(\mathit{clear}(Z),S) \wedge \\
&\qquad\qquad \mathit{holds}(\mathit{clear}(Y_1),S)\,)\,]
\end{aligned}
$$

Two Fork Lift Trucks: Effect Axioms

$$\begin{aligned}
&\mathit{holds}(\mathit{on}(X,Y),\mathit{do}(C,S)) \leftarrow \\
&\qquad \mathit{poss}(C,S) \wedge (\exists F)\, \mathit{move}(F,X,Y) \in C
\end{aligned}$$

$$\begin{aligned}
&\neg\mathit{holds}(\mathit{on}(X,Y),\mathit{do}(C,S)) \leftarrow \\
&\qquad \mathit{poss}(C,S) \wedge (\exists F,Z)\, (\, \mathit{move}(F,X,Z) \in C \wedge Y \neq Z \,)
\end{aligned}$$

$$\begin{aligned}
&\mathit{holds}(\mathit{clear}(X),\mathit{do}(C,S)) \leftarrow \\
&\qquad \mathit{poss}(C,S) \wedge (\exists F,Y,Z)\, (\, \mathit{move}(F,Y,Z) \in C \wedge \mathit{holds}(\mathit{on}(Y,X),S) \wedge X \neq Z \,)
\end{aligned}$$

$$\begin{aligned}
&\neg\mathit{holds}(\mathit{clear}(X),\mathit{do}(C,S)) \leftarrow \\
&\qquad \mathit{poss}(C,S) \wedge (\exists F,Y)\, \mathit{move}(F,Y,X) \in C
\end{aligned}$$

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)
   ).
```

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(on(d,track2), s0).
holds(clear(d), s0).               holds(clear(temp_store), s0).
holds(on(a,port1), s0).            holds(clear(a), s0).
holds(on(c,b), s0).                holds(on(b,port2), s0).
holds(clear(c), 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 $\textsc{concurrent} > \textsc{action}$
- ▷ $\epsilon : \textsc{concurrent}$
- ▷ $\circ : \textsc{concurrent} \times \textsc{concurrent} \mapsto \textsc{concurrent}$
- ▷ EUNA for $\circ; \epsilon$ (wrt. $\textsc{concurrent}$)
- ▷ foundational axioms $\mathcal{F}_{co}$:

$$(\forall A : \textsc{action}, C : \textsc{concurrent}, S : \textsc{sit}) \;\; \neg \mathit{poss}(A \circ A \circ C, S)$$

$$(\forall S : \textsc{sit}) \;\; \mathit{poss}(\epsilon, S) \wedge \mathit{state}(\mathit{do}(\epsilon, S)) = \mathit{state}(S)$$

- ▷ $\mathit{cancels}(C, C_1, S)$ $:\Leftrightarrow$ concurrent actions C cancels concurrent actions C_1 in situation S
- ▷ State update axioms $\mathcal{F}_{sua}$ (only deterministic actions, only direct effects):

$$\begin{array}{l} \mathit{poss}(C_1 \circ C, S) \wedge \neg \mathit{cancels}(C, C_1, S) \wedge \Delta(S) \\ \quad \rightarrow \;\; \mathit{state}(\mathit{do}(C_1 \circ C, S)) \circ \vartheta^- = \mathit{state}(\mathit{do}(C, S)) \circ \vartheta^+ \end{array}$$

- ▷ macro: $\mathit{holds\_in}(C_1, C) \stackrel{\text{def}}{=} (\exists C')\; C = C_1 \circ C'$

Two Fork Lift Trucks: Precondition Axiom

$$
\begin{array}{l}
\textit{poss}(C,S) \leftrightarrow \\
\quad (\forall F,X,Y) \\
\qquad [\, \textit{holds\_in}(\textit{move}(F,X,Y),C) \rightarrow \\
\qquad\quad \neg(\exists X_1,Y_1)\ \textit{holds\_in}(\textit{move}(F,X_1,Y_1),C) \wedge (X_1 \neq X \vee Y_1 \neq Y) \\
\qquad\quad \wedge \\
\qquad\quad \textit{holds}(\textit{clear}(Y),S) \\
\qquad\quad \wedge \\
\qquad\quad [\, \neg(\exists F_1,X_1,Y_1)\,(\, \textit{holds\_in}(\textit{move}(F_1,X_1,Y_1),C) \wedge F_1 \neq F \wedge (X_1 = X \vee Y_1 = Y)\,) \\
\qquad\qquad \rightarrow\ \textit{holds}(\textit{clear}(X),S)\,] \\
\qquad\quad \wedge \\
\qquad\quad [\,(\exists F_1,X_1,Y_1)\,(\, \textit{holds\_in}(\textit{move}(F_1,X_1,Y_1),C) \wedge F_1 \neq F \wedge (X_1 = X \vee Y_1 = Y)\,) \\
\qquad\qquad \rightarrow\ X_1 = X \wedge Y_1 = Y \wedge (\exists Z)\, \textit{holds}(\textit{on}(Z,X),S) \wedge \textit{holds}(\textit{clear}(Z),S)\,] \\
\qquad]
\end{array}
$$

Two Fork Lift Trucks: State Update Axioms

$$
\begin{aligned}
&(\exists F_1, X_1, Y_1)\,(\, \textit{holds\_in}(\textit{move}(F_1, X_1, Y_1), C) \land F_1 \neq F \land X_1 = X \land Y_1 = Y\,) \\
&\quad \rightarrow\; \textit{cancels}(C, \textit{move}(F, X, Y), S)
\end{aligned}
$$

$$
\begin{aligned}
&\textit{poss}(\textit{move}(F, X, Y) \circ C, S) \;\land\; \neg\textit{cancels}(C, \textit{move}(F, X, Y), S) \;\land\; \textit{holds}(\textit{on}(X, Z), S) \\
&\quad \rightarrow\; \textit{state}(\textit{do}(\textit{move}(F, X, Y) \circ C, S)) \circ \textit{on}(X, Z) \circ \textit{clear}(Y) \\
&\qquad\quad = \textit{state}(\textit{do}(C, S)) \circ \textit{on}(X, Y) \circ \textit{clear}(Z)
\end{aligned}
$$

$$
\begin{aligned}
&\textit{poss}(\textit{move}(F_1, X, Y) \circ \textit{move}(F_2, X, Y) \circ C, S) \;\land\; \textit{holds}(\textit{on}(X, Z), S) \\
&\quad \rightarrow\; \textit{state}(\textit{do}(\textit{move}(F_1, X, Y) \circ \textit{move}(F_1, X, Y) \circ C, S)) \circ \textit{on}(X, Z) \circ \textit{clear}(Y) \\
&\qquad\quad = \textit{state}(\textit{do}(C, S)) \circ \textit{on}(X, Y) \circ \textit{clear}(Z)
\end{aligned}
$$

Literature

▷ M. Shanahan: Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. MIT Press 1997. (Chapter 15).

▷ 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.

▷ M. Thielscher: Solving the inferential frame problem for concurrent actions. 1999.

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

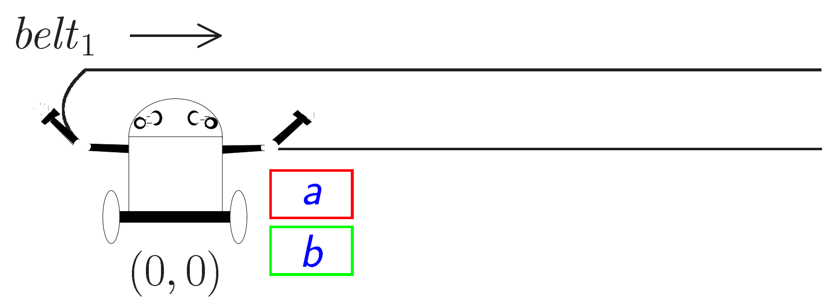
- ▷ Idea: A fluent, which itself is stable, represents a process of continuous change.

$$\textit{holds}(movement(X, \vec{P}_0, \vec{V}, T_0), S)$$
$$:\Leftrightarrow$$
In situation S, object X moves with constant (spatial) velocity $\vec{V}$.
The object started at time T_0 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)



$\textit{holds}(has(X), S)$:⇔ the robot is in possession of X in situation S

$put(X, T)$:⇔ the action of putting X onto $belt_1$ at time T

$$vel(belt_1) = (1, 0)$$

$$\textit{holds}(has(\textsf{a}), s_0) \wedge \textit{holds}(has(\textsf{b}), s_0) \wedge \neg(\exists)\, \textit{holds}(movement(X, P_0, V, T_0), s_0)$$

Precondition and Effect Axioms

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

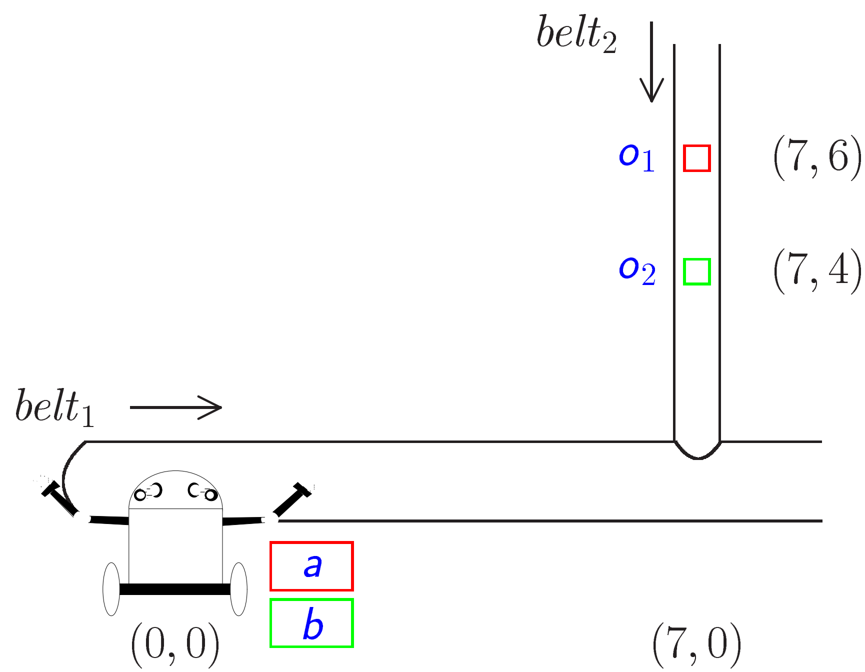
$$
\begin{aligned}
&\textit{poss}(put(X,T),S) \leftrightarrow \\
&\qquad \textit{holds}(has(X),S) \land \\
&\qquad \neg(\exists Y,P_0,V,T_0)\,[\,\textit{holds}(movement(Y,P_0,V,T_0),S) \land P_0 + V \cdot (T - T_0) = (0,0)\,]
\end{aligned}
$$

The effect of put is to lose possession of the object and to initiate a certain movement:

$$\textit{poss}(put(X,T),S) \rightarrow \neg\textit{holds}(has(X),\textit{do}(put(X,T),S))$$

$$\textit{poss}(put(X,T),S) \rightarrow \textit{holds}(movement(X,(0,0),vel(belt_1),T),\textit{do}(put(X,T),S))$$

A Detail of a Production Line (2)



$$vel(belt_1) = (1,0) \wedge vel(belt_2) = (0,-0.5)$$

$$\begin{aligned}
&\textit{\textbf{holds}}(has(\textit{a}), s_0) \wedge \textit{\textbf{holds}}(has(\textit{b}), s_0) \wedge \\
&(\forall X, P_0, V, T_0)\,[\,\textit{\textbf{holds}}(movement(X, P_0, V, T_0), s_0) \leftrightarrow \\
&\qquad X = o_1 \wedge P_0 = (7,6) \wedge V = vel(belt_2) \wedge T_0 = 0 \vee \\
&\qquad X = o_2 \wedge P_0 = (7,4) \wedge V = vel(belt_2) \wedge T_0 = 0\,]
\end{aligned}$$

A Natural Action and its Specification

$falls(X,T)$:⇔ the natural action of X falling down onto $belt_1$

The precondition of $falls$ is that the object reaches the end of the belt:

$$\begin{aligned} \textit{poss}(falls(X,T),S) \ \leftrightarrow\ (\exists P_y,T_0)\,[\, &\textit{holds}(movement(X,(7,P_y),vel(belt_2),T_0),S) \land \\ &(7,P_y)+vel(belt_2)\cdot(T-T_0)=(7,0)\,] \end{aligned}$$

The effect of $falls$ is to terminate the current movement and to initiate a new one:

$$\textit{poss}(falls(X,T),S) \ \rightarrow\ \neg\textit{holds}(movement(X,P_0,vel(belt_2),T_0),\textit{do}(falls(X,T),S))$$

$$\textit{poss}(falls(X,T),S) \ \rightarrow\ \textit{holds}(movement(X,(7,0),vel(belt_1),T),\textit{do}(falls(X,T),S))$$

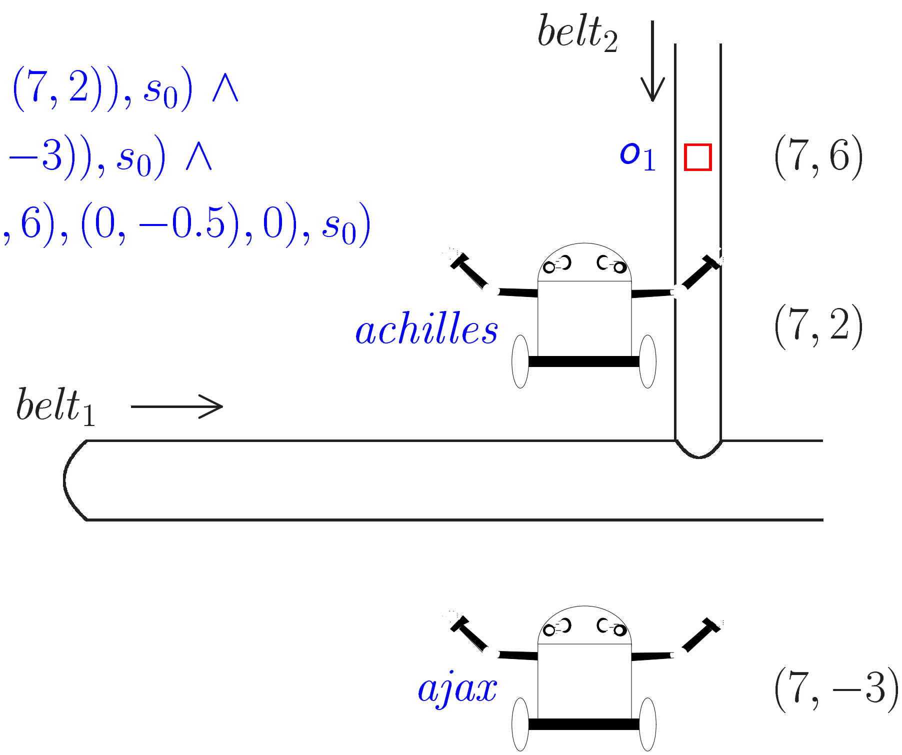
The Successor State Axioms

$$
\begin{aligned}
&\mathit{poss}(A,S) \rightarrow \\
&\quad \mathit{holds}(has(X), \mathit{do}(A,S)) \leftrightarrow \\
&\quad\quad \mathit{holds}(has(X), S) \wedge \neg(\exists T)\, A = put(X,T)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{poss}(A,S) \rightarrow \\
&\quad \mathit{holds}(movement(X,P_0,V,T_0), \mathit{do}(A,S)) \leftrightarrow \\
&\quad\quad A = put(X,T_0) \wedge P_0 = (0,0) \wedge V = vel(belt_1) \\
&\quad\quad \vee \\
&\quad\quad A = falls(X,T_0) \wedge P_0 = (7,0) \wedge V = vel(belt_1) \\
&\quad\quad \vee \\
&\quad\quad \mathit{holds}(movement(X,P_0,V,T_0), S) \wedge \neg(\exists T)\,[\, A = falls(X,T) \wedge V = vel(belt_2)\,]
\end{aligned}
$$

Why Natural Actions Require Special Treatment

$\mathit{holds}(\mathit{location}(achilles, (7,2)), s_0) \wedge$
$\mathit{holds}(\mathit{location}(ajax, (7,-3)), s_0) \wedge$
$\mathit{holds}(movement(o_1, (7,6), (0,-0.5), 0), s_0)$



$$\mathit{poss}(grab(R,X,T),S) \leftrightarrow$$
$$\quad \mathit{holds}(\mathit{location}(R,P),S) \wedge \mathit{holds}(movement(X,P_0,V,T_0),S) \wedge P = P_0 + V \cdot (T - T_0)$$

$$\models \mathit{poss}(grab(achilles, o_1, 8), s_0) \wedge \mathit{poss}(falls(o_1, 12), s_0) \wedge \mathit{poss}(grab(ajax, o_1, 18), s_0)$$

The General Approach (1)

predicate	*natural*(A)	$:\Leftrightarrow$	A is a natural action
predicate	*legal*(S)	$:\Leftrightarrow$	situation S respects the property of natural actions that they must occur at their predicted times, provided no earlier actions prevent them from occurring
function	*start*(S)	$:\Leftrightarrow$	start time of situation S
function	*time*(A)	$:\Leftrightarrow$	time of action A

The sort TIMEPOINT ranges over the real numbers.
We assume a standard interpretation of the reals and their usual operations.

▷ Execution time for each action $A(\overline{X}, T)$:

$$\mathit{time}(A(\overline{X}, T)) = T$$

▷ Start time of a situation (axiom $\mathcal{F}_{start}$):

$$\mathit{start}(s_0) = 0 \quad \wedge \quad \mathit{start}(\mathit{do}(A, S)) = \mathit{time}(A)$$

The General Approach (2)

- Natural action condition:

$$\mathit{natural}(A) \leftrightarrow \Psi(A)$$

where Ψ is a formula with free variable A.

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

- Foundational axioms $\mathcal{F}_{legal}$ for legal situations:

$$\begin{aligned} & \mathit{legal}(s_0) \wedge \\ & \mathit{legal}(\mathit{do}(A, S)) \leftrightarrow \mathit{legal}(S) \wedge \mathit{poss}(A, S) \wedge \mathit{start}(S) \leq \mathit{time}(A) \wedge \\ & \qquad (\forall A')\,[\, \mathit{natural}(A') \wedge \mathit{poss}(A', S) \rightarrow A = A' \vee \mathit{time}(A) < \mathit{time}(A') \,] \end{aligned}$$

- 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

$$\mathcal{F} \models g(s) \wedge \mathit{legal}(s)$$

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]).
```

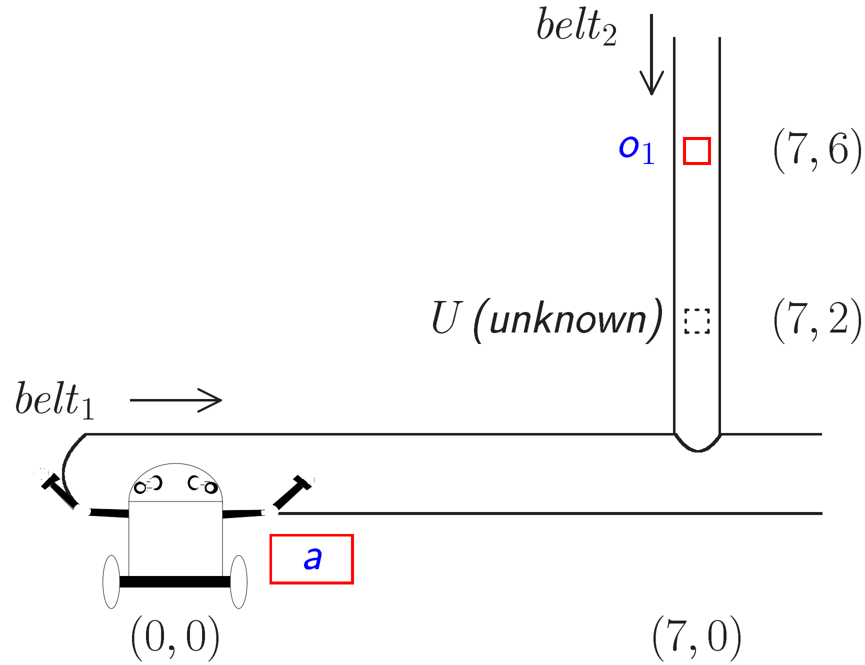
The Conveyor Belt Robot in ECLIPSE (2)

```
poss(put(X,T), S) :- holds(has(X), S), not occupied([0,0], T, S).

occupied([Px,Py], T, S) :-
   holds(movement(_,[Px0,Py0],[Vx,Vy],T0), S),
   { Px0 + Vx*(T-T0) = Px, Py0 + Vy*(T-T0) = Py }.

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

holds(has(X), do(A,S)) :-
   holds(has(X), S), not A=put(X,_).

holds(movement(X,P0,V,T0), do(A,S)) :-
   A=put(X,T0), P0=[0.0,0.0], vel(belt1, V) ;
   A=falls(X,T0), P0=[7.0,0.0], vel(belt1, V) ;
   holds(movement(X,P0,V,T0), S), not ( A=falls(X,_), vel(belt2, V) ).
```

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



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

The planning problem of getting o_1 into a has no solution since there is no provably legal situation which includes the action $put(a, 5)$.

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

$$\begin{aligned}
&\mathit{natural}(A) \leftrightarrow (\exists T)\, A = a(T)\\
&\mathit{time}(a(T)) = T\\
&\mathit{poss}(a(T)) \leftrightarrow (\exists N : \mathbb{N})\, T = 1 - 2^{-N}
\end{aligned}$$

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

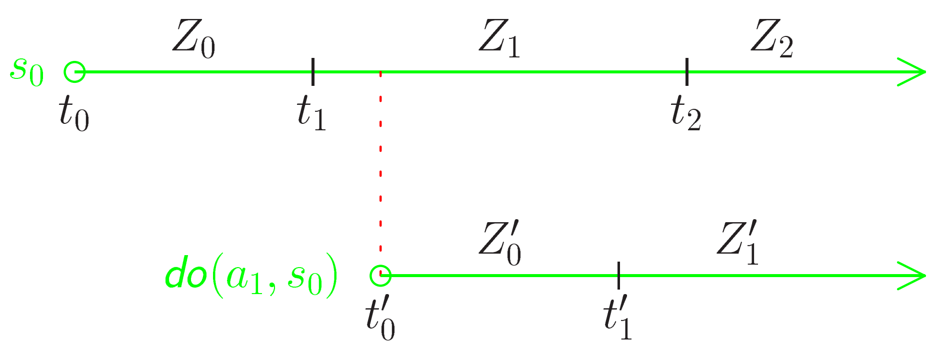
$$\mathcal{F} \models \mathit{legal}(s_0) \wedge \mathit{legal}(\mathit{do}(a(1/2), s_0)) \wedge \mathit{legal}(\mathit{do}(a(3/4), \mathit{do}(a(1/2), s_0))) \wedge \ldots$$

but there is no S such that

$$\mathcal{F} \models \mathit{legal}(S) \wedge \mathit{start}(S) > 1$$

Trajectories in the Fluent Calculus

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



trajectory for s_0 : $Z_0, Z_1, Z_2, \ldots$

trajectory for $do(a_1, s_0)$: $Z'_0, Z'_1, \ldots$

$$t_1 < \mathit{time}(a_1) = t'_0 = \mathit{start}(s_1) < t_2$$

The General Approach (1)

- We adopt sort $\mathrm{TIMEPOINT}$, predicate *natural*, and function *time*.
- Fluent $\mathit{start\_time}(T)$ denotes the start time of the state in which it holds true.
- $\mathit{succ} : \mathrm{ACTION} \times \mathrm{STATE} \mapsto \mathrm{STATE}$ defines the successor state after a natural action.
- $\mathit{expect}(A, Z)$ $:\Leftrightarrow$ natural action A is expected to happen in state Z
  if no earlier natural action prevents this.
- $\mathit{trajectory}(Z, Z')$ $:\Leftrightarrow$ state Z' lies on the trajectory rooted in state Z
- State constraints on start times $\mathcal{F}_{st}$:

$$(\exists T)\, \mathit{holds\_in}(\mathit{start\_time}(T), \mathit{state}(S))$$
$$\mathit{holds\_in}(\mathit{start\_time}(T_1), \mathit{state}(S)) \wedge \mathit{holds\_in}(\mathit{start\_time}(T_2), \mathit{state}(S)) \rightarrow T_1 = T_2$$

- Foundational axioms $\mathcal{F}_{traj}$:

$$\mathit{trajectory}(Z, Z)$$
$$\mathit{trajectory}(Z, Z') \wedge \mathit{next\_nat\_action}(A, Z') \rightarrow \mathit{trajectory}(Z, \mathit{succ}(A, Z'))$$

where $\mathit{next\_nat\_action}(A, Z)$ abbreviates the formula

$$\mathit{expect}(A, Z) \wedge (\forall A')\,[\, \mathit{natural}(A') \wedge \mathit{expect}(A', Z) \wedge \mathit{time}(A') \leq \mathit{time}(A) \rightarrow A = A' \,]$$

Useful Macros for Precondition and Effect Specification

$$\mathit{after}(T, Z) \stackrel{\text{def}}{=} (\forall T_0)\,[\,\mathit{holds\_in}(\mathit{start\_time}(T_0), Z) \;\rightarrow\; T > T_0\,]$$

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

$$\begin{aligned}\mathit{actual\_state}(S, T, Z) \stackrel{\text{def}}{=} (\forall T_0)\,[\,&\mathit{holds\_in}(\mathit{start\_time}(T_0), Z) \;\rightarrow \\ &T_0 \leq T \wedge \mathit{trajectory}(\mathit{state}(S), Z) \wedge \\ &(\forall A)\,(\,\mathit{next\_nat\_action}(A, Z) \;\rightarrow\; \mathit{time}(A) > T\,)\,]\end{aligned}$$

$$\mathit{holds}(F, S, T) \stackrel{\text{def}}{=} (\forall Z)\,[\,\mathit{actual\_state}(S, T, Z) \;\rightarrow\; \mathit{holds\_in}(F, Z)\,]$$

General Form of Precondition and Effect Specifications

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

$$\mathit{poss}(a(\overline{X}, T), S) \leftrightarrow \Phi_a(\overline{X}, S) \wedge \mathit{after}(T, S)$$

and the state update axioms take this form:

$$\begin{aligned}
&\mathit{poss}(a(\overline{X}, T), S) \rightarrow \\
&\quad \mathit{actual_state}(S, T, Z) \rightarrow \\
&\quad\quad \Delta(Z) \rightarrow \\
&\quad\quad\quad (\exists T_0)\ \mathit{state}(\mathit{do}(a(\overline{X}, T), S)) \circ \vartheta^- \circ \mathit{start_time}(T_0) = Z \circ \vartheta^+ \circ \mathit{start_time}(T)
\end{aligned}$$

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

$$\mathit{expect}(a(\overline{X}, T), Z) \leftrightarrow \Phi_a(\overline{X}, Z) \wedge \mathit{after}(T, Z)$$

and the update axioms take this form:

$$\begin{aligned}
&\mathit{expect}(a(\overline{X}, T), Z) \rightarrow \\
&\quad \Delta(Z) \rightarrow \\
&\quad\quad (\exists T_0)\ \mathit{succ}(a(\overline{X}, T), Z) \circ \vartheta^- \circ \mathit{start_time}(T_0) = Z \circ \vartheta^+ \circ \mathit{start_time}(T)
\end{aligned}$$

Precondition and Effect in the Production Line Domain (1)

The precondition of 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:

$$\begin{aligned}
&\textit{poss}(put(X,T),S) \leftrightarrow \\
&\quad \textit{holds}(has(X),S,T) \wedge \textit{after}(T,S) \wedge \\
&\quad \neg(\exists Y,P_0,V,T_0)\,[\,\textit{holds}(movement(Y,P_0,V,T_0),S,T) \wedge P_0 + V \cdot (T - T_0) = (0,0)\,]
\end{aligned}$$

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

$$\begin{aligned}
&\textit{poss}(put(X,T),S) \rightarrow \\
&\quad \textit{actual_state}(S,T,Z) \rightarrow \\
&\qquad (\exists T_0)\,\textit{state}(\textit{do}(put(X,T),S)) \circ has(X) \circ \textit{start_time}(T_0) = \\
&\qquad\qquad Z \circ movement(X,(7,0),vel(belt_1),T) \circ \textit{start_time}(T)
\end{aligned}$$

Precondition and Effect in the Production Line Domain (2)

The precondition of $falls$ is that the object reaches the end of the belt:

$$\begin{aligned}
&\textit{expect}(falls(X,T),Z) \leftrightarrow \\
&\quad (\exists P_y, T_0)\,[\,\textit{holds_in}(movement(X,(7,P_y),vel(belt_2),T_0),Z) \wedge \\
&\qquad (7,P_y) + vel(belt_2)\cdot(T-T_0) = (7,0) \wedge \textit{after}(T,Z)\,]
\end{aligned}$$

The effect of $falls$ is to terminate the current movement and to initiate a new one:

$$\begin{aligned}
&\textit{expect}(falls(X,T),Z) \rightarrow \\
&\quad (\exists T_0, T_0', P_0, V)\,[\,\textit{succ}(falls(X,T),Z) \circ movement(X,P_0,V,T_0') \circ \textit{start_time}(T_0) = \\
&\qquad Z \circ movement(X,(7,0),vel(belt_1),T) \circ \textit{start_time}(T)\,]
\end{aligned}$$

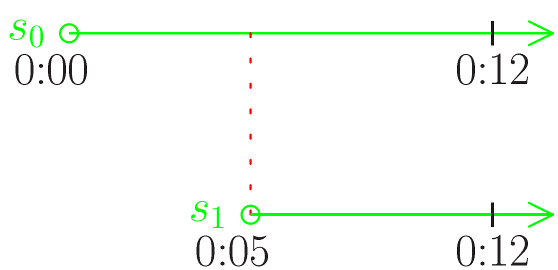
Planning with Incomplete Initial Knowledge (Revisited)

Consider this initial specification:

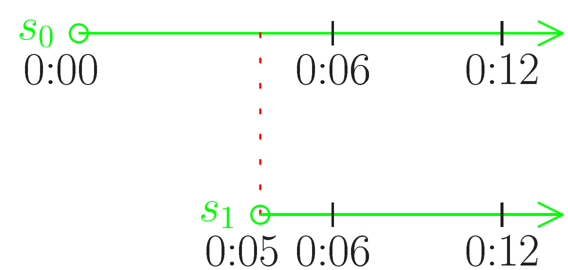
$$
\begin{aligned}
&(\exists Z)\,[\,\textit{state}(s_0) = \textit{start\_time}(0) \circ has(\mathsf{a}) \circ movement(\mathsf{o}_1, (7,6), vel(belt_2), 0) \circ Z \;\wedge \\
&\qquad (\exists U)(\forall)\,[\,\textit{holds\_in}(movement(X, P_0, V, T_0), Z) \\
&\qquad\qquad \rightarrow\; X = U \wedge P_0 = (7,2) \wedge V = vel(belt_2) \wedge T_0 = 0\,]
\end{aligned}
$$

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

$$
\begin{aligned}
&(\exists Z)\,[\,\textit{trajectory}(\textit{state}(\textit{do}(put(\mathsf{a},5), s_0)),\; Z) \;\wedge\; \textit{holds\_in}(\textit{start\_time}(12), Z) \;\wedge \\
&\qquad \textit{holds\_in}(movement(\mathsf{a}, (0,0), (1,0), 5),\; Z) \;\wedge\; \textit{holds\_in}(movement(\mathsf{o}_1, (7,0), (1,0), 12),\; Z)\,]
\end{aligned}
$$



Model (a)



Model (b)

Zeno's Paradox (Revisited)

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

$$\begin{aligned}
&\mathit{natural}(A) \leftrightarrow (\exists T)\, A = a(T) \\
&\mathit{time}(a(T)) = T \\
&\mathit{expect}(a(T), Z) \leftrightarrow (\exists N : \mathbb{N})\, T = 1 - 2^{-N} \wedge \mathit{after}(T, Z) \\
&\mathit{expect}(a(T), Z) \rightarrow (\exists T_0)\, \mathit{succ}(a(T), Z) \circ \mathit{start_time}(T_0) = Z \circ \mathit{start_time}(T)
\end{aligned}$$

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

$$(\forall \psi : \mathbb{N} \mapsto \mathbb{R}) \left[\begin{array}{l} (\forall N : \mathbb{N})\, \mathit{trajectory}(Z, Z' \circ \mathit{start\_time}(\psi(N))) \\ \quad \rightarrow \mathit{trajectory}(Z, Z' \circ \mathit{start\_time}(\lim_{N \to \infty} \psi(N))) \end{array} \right]$$

Then,

$$\mathit{state}(s_0) = Z \circ \mathit{start\_time}(0) \rightarrow T \geq 1 \rightarrow \mathit{actual\_state}(s_0, T, Z \circ \mathit{start\_time}(1))$$

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., $position$).

$$\textit{trace}(F_1, T, F_2, D) \;:\Leftrightarrow\; \text{if the discrete fluent } F_1 \text{ is initiated at time } T, \text{ then the continuous fluent } F_2 \text{ holds at time } T + D.$$

Foundational axioms for continuous fluents:

$$\textit{holds\_at}(F_2, D) \leftarrow$$
$$(\exists F_1)\ \textit{initially}(F_1) \wedge \textit{trace}(F_1, 0, F_2, D) \wedge D > 0 \wedge \neg\textit{clipped}(0, F_1, D)$$

$$\textit{holds\_at}(F_2, T_2) \leftarrow$$
$$(\exists E, F_1, T_1, D)\ \textit{happens}(E, T_1) \wedge \textit{initiates}(E, F, T_1) \wedge \textit{trace}(F_1, T_1, F_2, D)$$
$$\wedge\ D > 0 \wedge T_2 = T_1 + D \wedge \neg\textit{clipped}(T_1, F_1, T_2)$$

The Conveyor Belt Robot in the Event Calculus (1)

event	$put(X)$	$:\Leftrightarrow$	put X onto conveyor $belt_1$
event	$falls(X)$	$:\Leftrightarrow$	X falls from $belt_2$ onto $belt_1$
discrete fluent	$has(X)$	$:\Leftrightarrow$	the robot is in possession of X
discrete fluent	$movement(X, V)$	$:\Leftrightarrow$	X moves with constant two-dimensional velocity V
discrete fluent	$position(X, P)$	$:\Leftrightarrow$	P is the current (stable) position of X
continuous fluent	$position(X, P)$	$:\Leftrightarrow$	P is the current (constantly changing) position of X

Let I be,

$$\begin{array}{l} \textit{initially}(has(a)) \wedge \textit{initially}(has(b)) \wedge \\ \textit{initially}(position(a,(0,0))) \wedge \textit{initially}(position(b,(0,0))) \wedge \\ \textit{initially}(movement(o_1, vel(belt_2))) \wedge \textit{initially}(movement(o_2, vel(belt_2))) \end{array}$$

Consider, in addition,

$$\textit{holds\_at}(position(o_1,(7,6)),0) \wedge \textit{holds\_at}(position(o_2,(7,4)),0)$$

Finally, let N be,

$$\textit{happens}(put(a),1) \wedge \textit{happens}(put(b),5)$$

The Conveyor Belt Robot in the Event Calculus (2)

$$\textit{\textbf{terminates}}(falls(X), movement(X,(7,P_y),vel(belt_2)),T)$$
$$\textit{\textbf{initiates}}(falls(X), movement(X,(7,0),vel(belt_1)),T)$$

$$\textit{\textbf{terminates}}(put(X), has(X),T) \leftarrow \textit{\textbf{holds\_at}}(has(X),T)$$
$$\textit{\textbf{initiates}}(put(X), movement(X,vel(belt_1)),T) \leftarrow \textit{\textbf{holds\_at}}(has(X),T)$$
$$\textit{\textbf{terminates}}(put(X), position(X,(0,0)),T) \leftarrow \textit{\textbf{holds\_at}}(has(X),T)$$

$$\begin{aligned}\textit{\textbf{trace}}(movement(X,V),T,position(X,P),D) \leftarrow\ & \textit{\textbf{holds\_at}}(position(X,P_0),T) \wedge \\ & P = P_0 + V \cdot D\end{aligned}$$

$$\begin{aligned}&\textit{\textbf{happens}}(falls(X),T) \leftarrow \\ &\quad \textit{\textbf{holds\_at}}(movement(X,vel(belt_2)),T) \wedge \textit{\textbf{holds\_at}}(position(X,(7,0)),T)\end{aligned}$$

▷ 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

- 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.
- M. Thielscher: Fluent Calculus planning with continuous change. In: S. Biundo (ed.), Proceedings of the European Conference on Planning. Springer LNAI, 1999.
- M. Shanahan: Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia. MIT Press 1997. (Chapter 13).

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.
- $\mathit{do}(\delta, S, S')$ holds, whenever S' is a terminating situation of an execution of a complex action δ starting in situation S.
- Primitive actions:

$$\mathit{do}(A, S, S') \stackrel{def}{=} \mathit{poss}(A, S) \wedge S' = \mathit{do}(A, S).$$

- Test actions:

$$\mathit{do}(\Phi?, S, S') \stackrel{def}{=} \mathit{holds}(\Phi, S) \wedge S = S'.$$

- Sequence:

$$\mathit{do}([\delta_1; \delta_2], S, S') \stackrel{def}{=} (\exists S^*)\ (\mathit{do}(\delta_1, S, S^*) \wedge \mathit{do}(\delta_2, S^*, S')).$$

Complex Actions: Nondeterministic Choice

▷ Nondeterministic choice of two actions:

$$\mathit{do}([\delta_1|\delta_2], S, S') \stackrel{def}{=} \mathit{do}(\delta_1, S, S') \vee \mathit{do}(\delta_2, S, S')$$

⇝ Conditionals:

$$\texttt{if}\ \Phi\ \texttt{then}\ \delta_1\ \texttt{else}\ \delta_2\ \texttt{endIf} \stackrel{\texttt{def}}{=} [\Phi?; \delta_1]|[\neg\Phi?; \delta_2]$$

`if` *car_in_driveway* `then` *drive* `else` *walk* `endIf`

▷ Nondeterministic choice of action arguments:

$$\mathit{do}((\pi X)\ \delta(X), S, S') \stackrel{def}{=} (\exists X)\ \mathit{do}(\delta(X), S, S')$$

$$(\pi X)\ \mathit{remove}(X)$$

Complex Actions: Nondeterministic Iteration

▷ Nondeterministic Iteration: Execute δ zero or more times.

$$\mathit{do}(\delta^*, S, S')$$

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

⇝ While statements:

$$\texttt{while}\ \Phi\ \texttt{do}\ \delta\ \texttt{endWhile} \stackrel{\mathrm{def}}{=} [[\Phi?;\delta]^*;\neg\Phi?]$$

$$\texttt{while}\ (\exists \mathrm{B})\ \mathit{ontable}(\mathrm{B})\ \texttt{do}\ (\pi \mathrm{X})\ \mathit{remove}(X)\ \texttt{endWhile}$$

Complex Actions: Procedures

▷ Procedure calls:

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

⇝ Call by value

▷ GOLOG programs:

$$\mathit{do}(\{\texttt{proc}\ \mathrm{p}_1(\overline{\mathtt{V}_1})\delta_1\ \texttt{endProc}; \ldots; \texttt{proc}\ \mathrm{p}_\mathrm{n}(\overline{\mathtt{V}_\mathrm{n}})\delta_\mathrm{n}\ \texttt{endProc}; \delta_0\}, \mathtt{S}, \mathtt{S}')$$

where $p_i(\overline{V_i})$ is a procedure declaration with formal parameters $\overline{V_i}$ and δ_i is its body, $1 \leq i \leq n$, and δ_0 is the main program, ie. a complex action.

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

▷ Move an elevator down N floors:

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

where *down* moves an elevator down one floor.

An Elevator Controller: Primitive Actions and Fluents

- Primitive actions:

  up(N) denotes the 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:

  $\mathit{poss}(\mathit{up}(N), S) \leftrightarrow (\exists M)\ [\mathit{holds}(\mathit{current\_floor}(M), S) \wedge M < N]$.
  $\mathit{poss}(\mathit{down}(N), S) \leftrightarrow (\exists M)\ [\mathit{holds}(\mathit{current\_floor}(M), S) \wedge M > N]$.
  $\mathit{poss}(\mathit{open}, S) \leftrightarrow \top$.
  $\mathit{poss}(\mathit{close}, S) \leftrightarrow \top$.
  $\mathit{poss}(\mathit{turnoff}(N), S) \leftrightarrow \mathit{holds}(\mathit{on}(N), S)$.

An Elevator Controller: Successor State Axioms

▷ Successor state axioms:

$$\begin{gathered} \mathit{poss}(A,S) \rightarrow [\mathit{holds}(\mathit{current\_floor}(M), \mathit{do}(A,S)) \;\leftrightarrow \\ A = \mathit{up}(M) \vee A = \mathit{down}(M) \\ \vee\ \mathit{holds}(\mathit{current\_floor}(M), S) \wedge \neg(\exists N)\ A = \mathit{up}(N) \wedge \neg(\exists N)\ A = \mathit{down}(N)]. \end{gathered}$$

$$\mathit{poss}(A,S) \rightarrow [\mathit{holds}(\mathit{on}(M), \mathit{do}(A,S)) \;\leftrightarrow \mathit{holds}(\mathit{on}(M), S) \wedge A \neq \mathit{turnoff}(M)].$$

▷ A defined fluent:

$$\begin{gathered} \mathit{holds}(\mathit{next\_floor}(N), S) \leftrightarrow \mathit{holds}(\mathit{on}(N), S) \wedge \\ (\forall M, L)\ [\mathit{holds}(\mathit{on}(M), S) \wedge \mathit{holds}(\mathit{current\_floor}(L), S) \rightarrow |M - L| \geq |N - L|. \end{gathered}$$

An Elevator Controller: The Procedures

- The procedures:

  $\texttt{proc}\ \textit{serve}(\mathtt{N})\ \textit{go\_floor}(\mathtt{N}); \textit{turnoff}(\mathtt{N}); \textit{open}; \textit{close}\ \texttt{endProc}.$

  $\texttt{proc}\ \textit{current\_floor}(\mathtt{N})?|\textit{up}(\mathtt{N})|\textit{down}(\mathtt{N})\ \texttt{endProc}.$

  $\texttt{proc}\ \textit{serve\_a\_floor}\ (\pi\mathtt{N})[\textit{next\_floor}(\mathtt{N})?; \textit{serve}(\mathtt{N})]\ \texttt{endProc}.$

  $\texttt{proc}\ \textit{control}\ [\texttt{while}\ (\exists\mathtt{N})\textit{on}(\mathtt{N})\ \texttt{do}\ \textit{serve\_a\_floor}\ \texttt{endWhile}; \textit{park}\ \texttt{endProc}.$

  $\texttt{proc}\ \textit{park}\ \texttt{if}\ \textit{current\_floor}(0)\ \texttt{then}\ \textit{open}\ \texttt{else}\ \textit{down}(0); \textit{open}\ \texttt{endIf}\ \texttt{endProc}.$

- Initial situation:

$$\textit{holds}(\textit{current\_floor}(4), s_0) \wedge \textit{holds}(\textit{on}(5), s_0) \wedge \textit{holds}(\textit{on}(3), s_0)$$

Reasoning about the Elevator Controller

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

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

where Π is the sequence of procedure definitions.

⇝ A successful proof might return the substitution

$$\begin{aligned} S \; = \; & \textit{do}(\textit{open}, \textit{do}(\textit{down}(0), \textit{do}(\textit{close}, \textit{do}(\textit{open}, \textit{do}(\textit{turnoff}(5), \\ & \textit{do}(\textit{up}(5), \textit{do}(\textit{close}, \textit{do}(\textit{open}, \textit{do}(\textit{turnoff}(3), \textit{do}(\textit{down}(3), s_0)))))))))). \end{aligned}$$

Complex Actions in the Fluent Calculus

- 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.

```
exec(go(L)) :- positition(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

- ▷ H. Levesque etal: GOLOG: A Logic Programming Language for Dynamic Domains. Journal of Logic Programming 31, 59-83: 1997.
- ▷ S. Hölldobler and H.-P. Störr: Complex Plans in the Fluent Calculus. In: Intellectics and Computational Logic: Papers in Honor of Wolfgang Bibel (Hölldobler, ed.), Kluwer Academic Publishers: 1999.
- ▷ D. Hähnel etal: GOLEX — Bridging the Gap between Logic (GOLOG) and a Real Robot. In: KI–98: Advances in Artificial Intelligence (Herzog and Günter, eds.), Springer, Lecture Notes in Artificial Intelligence 1504, 165-176: 1998.