Lecture 1: Introduction to Situation Calculus

Mikhail Soutchanski

(Acknowledgement: includes a few slides prepared by Ray Reiter for his course CSC2532 at the University of Toronto)

TMU, Department of Computer Science
https://www.cs.torontomu.ca/ mes/

July 9, 2025

From First Order Logic to Actions and Change

In this tutorial, we'll not talk about first order logic (FOL) in general or about reasoning with arbitrary FOL formulas.

Instead, we focus on a particular class of FOL axioms that represent discrete change: the situation calculus. This case is sufficiently large, it is interesting from the perspective of applications.

By shifting our focus from arbitrary FOL formulas, to reasoning about discrete change we make a step towards discussion how to reason about discrete and continuous change efficiently, and how to develop computationally efficient implementations.

Our next step will be how to solve planning problems efficiently without making strong assumptions such as the domain closure assumption, when the names of all objects are explicitly given in advance. Also, how to plan when the action parameters vary over an infinite continuous space ?

But before we get to planning, we have to see how to compute individual effects efficiently and how to compute what does not change after doing an action.

Language of the situation calculus (John McCarthy)

A *first order* language for representing dynamically changing worlds; all changes are the result of named *actions*. Actions are denoted by function symbols. Actions may be parameterized, e.g. the term $move(x, y)$ might stand for the action of moving object x on object y.

A possible world history, which is simply a sequence of actions, is represented by a first order term called a *situation*.

S_0 denotes the initial situation, where no actions have yet occurred. $do(a, s)$ denotes the successor situation to s resulting from performing the action a.

Example: $do(move(B_1, B_2), s)$ denotes that situation resulting from placing B_1 on B_2 when the world is in situation s.

Fluents are those relations (or functions) whose truth values may vary from situation to situation. They are denoted by predicate (or function) symbols taking a situation term as one of their arguments.
(John McCarthy borrowed the word "fluent" from Isaac Newton.)

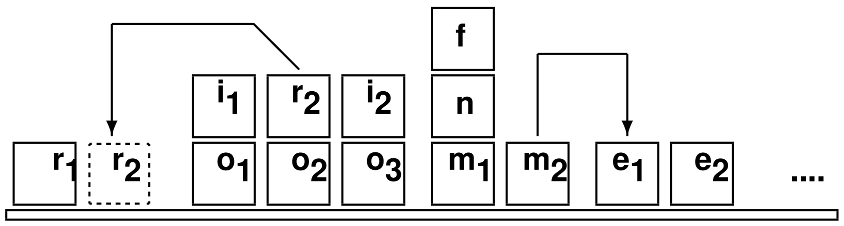
Example. In a world, in which it is possible to paint objects, we might have a functional fluent $colour(x, s)$ that denotes the colour of object x when the world is in situation s. For example, $colour(B_9, S_0) = Blue$ means that colour of block B_9 is *Blue* in the initial situation S_0.

Notation. Use the dot "**.**" to indicate the scope of quantifier: $Qx.F$ means Q, which is either $\forall$ or $\exists$ over x, applies everywhere in F.

The Blocks World (BW)

We consider mostly relational fluents in the beginning of this tutorial.

Fluents: $ontable(x, s)$ means x is on the table, $clear(x, s)$ means there is nothing on the top of x, $on(x, y, s)$ means block x is on the top of block y.



A basic version of BW includes the following 2 different actions:

$move(x, y)$: move block x from its current position to the top of block y

$moveToTable(x)$: move block x from its current position to the table.

Later, we'll consider a different set of actions for BW.

Preconditions: the qualification problem

Actions have preconditions: necessary conditions which a world situation must satisfy if the action can be performed in this situation. We write them as formulas using the predicate symbol $Poss(a,s)$ or $poss(a,s)$.

Example. If it is possible for a robot to move an object x to the table in situation s, then there is nothing on the top of x in s, and x is not already on the table in situation s:

$\forall s \forall x. poss(moveToTable(x),\ s) \rightarrow clear(x,s) \wedge \neg ontable(x).$

Example. Whenever it is possible for a robot to move an object x on top of y in situation s, then both objects must be clear, and they must be distinct:

$\forall s \forall x \forall y. poss(move(x,y),\ s) \rightarrow clear(x,s) \wedge clear(y,s) \wedge x \neq y.$

Can we infer when a *moveToTable* or a *move* is possible?

What about reversing the implication?

$\forall s \forall x. clear(x,s) \wedge \neg ontable(x) \rightarrow poss(moveToTable(x), s).$

$\forall s \forall x \forall y. clear(x,s) \wedge clear(y,s) \wedge x \neq y \rightarrow poss(move(x,y),\ s).$

Our approach: ignore all the “minor" qualifications and assume that all preconditions are explicitly specified. For each action $A(\vec{x})$, we have an axiom

$$\forall \vec{x}\, \forall s.\ poss(A(\vec{x}), s) \leftrightarrow \Pi_A(\vec{x}, s),$$

where $\Pi_A(\vec{x}, s)$ is a first order formula with free variables $\vec{x}, s$ which does not mention *do*. We shall call these *action precondition axioms*.

Preconditions as Conjunctive Queries in DBs

Usually, the RHS $\Pi_A(\vec{x}, s)$ is a conjunctive query with $\vec{x}$ as distinguished variables, but sometimes negations and numerical built-in are allowed.

An extended conjunctive query (ECQ) is of the form $\exists \vec{x} \phi(\vec{x}, \vec{y})$, where ϕ is a conjunction of fluents and safe disequalities $\neq$, that is, disequalities between variables or variables and constants such that each variable has to appear in at least one fluent.

The problem of evaluating ECQ over a finite relational database is NP-complete.

For this reason, in Automated Planning, there is an interest to an important class of conjunctive queries that have polynomial-time combined complexity: the *acyclic* conjunctive queries.

The connections between Reasoning About Actions (this course) and Data Bases are very important for developing efficient implementations. They are also important conceptually, if one considers data bases with incomplete information.

Effect axioms

World dynamics are specified by *effect axioms* which specify the effect of a given action on the truth value of a given fluent.

The effect of a robot moving an object x from object y to the table on the relational fluent $clear(y,s)$ /* |x| */

$\forall x \forall y \forall s.\ on(x,y,s) \rightarrow clear(y, do(moveToTable(x), s)).$ /* |y| */

This is a *positive effect* axiom (PEA) since it asserts when a fluent becomes true after doing an action.

A robot moving an object x on top of an object y causes y not to be clear:

$\forall x \forall y \forall s.\ \neg clear(y, do(move(x,y), s)).$

This is a *negative effect* axiom (NEA) since it asserts when a fluent becomes false after doing an action.

The previous two examples deal with a relational fluent. However, we could write effect axioms also for functional fluents. Skip to simplify.

For functional fluents, we do not distinguish between positive and negative effect axioms. This distinction applies only to axioms for relational fluents. Functional fluents: see R.Reiter’s book “Knowledge in Action", Section 3.2.6.

The frame problem (McCarthy and Hayes)

Axioms other than effect axioms are required for formalizing dynamic worlds. These are called *frame axioms*, and they specify the action invariants of the domain, i.e., those fluents unaffected by the performance of an action.

Example. A positive frame axiom (PFA) – moving uncovered block to the table does not affect the block:

$\forall x \forall s.\ clear(x,s) \rightarrow clear(x, do(moveToTable(x), s)).$

Similarly, an object’s colour remains unchanged as a result of picking things up, opening a door, turning on a light, electing a new prime minister, etc.

Example. A negative frame axiom (NFA) – not uncovering things:

$\forall x \forall y \forall z \forall s. \neg clear(x,s) \wedge on(y,z,s) \wedge (x \neq z) \rightarrow \neg clear(x, do(moveToTable(y), s)).$

Problem: Vast number of frame axioms; only few actions affect a given fluent. All other actions leave the fluent invariant. If we have $\mathcal{A}$ actions, and $\mathcal{F}$ fluents, then in the worst case we have to write about $2 \cdot \mathcal{A} \cdot \mathcal{F}$ frame axioms.

The frame problem (FP): how to deal with the following
- Frame axioms are necessary to reason about actions
- Accuracy. Axiomatizer must think of all these frame axioms; no omissions.
- System must reason efficiently in the presence of so many axioms.
- Modularity. The axiomatizer wants only to add new effect axioms.

Simplifying assumptions to solve the FP

Want a systematic procedure for generating all the frame axioms from effect axioms. If possible, also want a compact representation for these frame axioms, because in their simplest form, there are too many of them.

The special syntactic form of effect axioms considered in the previous slide precludes indeterminate actions such as
$heads(do(flip, s)) \vee tails(do(flip, s))$, $\exists x(holding(x, do(pick\_up\_a\_block, s))$.
Determinate action = action with unique effects only

No complex actions such as conditionals, or loops, or recursive procedures.

No hidden indirect effects. We assume that we managed to compile all indirect effects so that they are explicitly formulated in effect axioms. This is a strong assumption. It is difficult to handle arbitrary constraints on fluents that implicitly specify indirect effects. Special case: acyclic dependencies are OK.

Other limitations: *no time*. Actions are atemporal. We cannot talk about how long actions take, or when they occur. No continuous actions like pushing an object between 2 points. Time and continuous change: the last 2 Lectures.

This is why we say that Reiter's approach solves the FP sometimes, i.e., only for simple, sequential, deterministic, atemporal actions without constraints.

Useful FOL equivalent transformations: review

Recall the propositional and first order logic (FOL) equivalences.

$F(C) \equiv \forall x(\, x = C \rightarrow F(x)\,)$

$f(T) = C \equiv \forall x(x = T \rightarrow f(x) = C)$

$((A \rightarrow C) \wedge (B \rightarrow C)) \equiv \quad (\neg A \vee C) \wedge (\neg B \vee C) \equiv$
$(\neg A \wedge \neg B) \vee C \wedge (\neg A \vee \neg B \vee \top) \equiv$
$\neg(A \vee B) \vee C \equiv ((A \vee B) \rightarrow C)$

$A \rightarrow (B \rightarrow C) \equiv \neg A \vee \neg B \vee C \equiv \neg(A \wedge B) \vee C \equiv (A \wedge B) \rightarrow C$

$(A \wedge \neg B) \rightarrow C \equiv \neg(A \wedge \neg B) \vee C \equiv \neg A \vee B \vee C \equiv \neg(A \wedge \neg C) \vee B \equiv (A \wedge \neg C) \rightarrow B$

$\forall y(\, B(y) \rightarrow C\,) \equiv (\,(\exists y B(y)) \rightarrow C\,)$ /* if y does not occur in C.*/

Reiter's solution: (1) effect axioms in normal form

Reiter's solution to the FP has two parts: (1) write positive and negative effect axioms in normal form, and (2) add causal completeness assumption. Combining these axioms yields a successor state axiom (SSA).

$\forall x \forall y \forall s.\ on(x, y, s) \rightarrow clear(y, do(moveToTable(x), s)) \equiv$ /* Recall the equivalence $F(C) \equiv \forall a(a = C \rightarrow F(a))$*/
$\forall x \forall y \forall s \forall \underline{a}.\ (on(x, y, s) \wedge a = moveToTable(x)) \rightarrow clear(y, do(a, s)) \equiv$
$\forall y \forall s \forall a.\ \underline{\exists x}(on(x, y, s) \wedge a = moveToTable(x)) \rightarrow clear(y, do(a, s))$ /*1*/

$\forall x \forall y \forall z \forall s.\ on(x, y, s) \rightarrow clear(y, do(move(x, z), s)) \equiv$
$\forall x \forall y \forall z \forall s \forall \underline{a}.\ (on(x, y, s) \wedge a = move(x, z)) \rightarrow clear(y, do(a, s)) \equiv$
$\forall y \forall s \forall a.\ \underline{\exists x}(\, on(x, y, s) \wedge \underline{\exists z}(a = move(x, z))\,) \rightarrow clear(y, do(a, s))$ /*2*/

These two positive effect axioms can be combined into the logically equivalent form because $((A \rightarrow C) \wedge (B \rightarrow C)) \equiv ((A \vee B) \rightarrow C)$:
$\forall y \forall s \forall a.\ (\ \exists x(on(x, y, s) \wedge a = moveToTable(x)) \vee$
$\exists x(\, on(x, y, s) \wedge \exists z(a = move(x, z))\,) \rightarrow clear(y, do(a, s))$.
This is a positive effect axiom in normal form (PEA-NF).

Similarly, consider the negative effect axiom for *clear*:
$\forall x \forall y \forall s.\ \neg clear(y, do(move(x, y), s)) \equiv$
$\forall y \forall s \forall \underline{a}.\ (\,\exists x(a = move(x, y))\,) \rightarrow \neg clear(y, do(a, s))$.
This is a negative effect axiom in normal form (NEA-NF).

Reiter's solution: (2) causal completeness

Now appeal to the following Causal Completeness Assumption (=*explanation closure axiom*): *the positive effect axiom in normal form characterizes* ***all*** *the conditions under which action a leads to y being clear* (did not miss anything).

Then if $\neg clear(y, s)$ and $clear(y, do(a, s))$ are both true, the truth value of *clear* must have changed due to
$\exists x(on(x, y, s) \wedge a = moveToTable(x)) \vee \exists x(on(x, y, s) \wedge \exists z(a = move(x, z)))$

This intuition can be formalized, after logically equivalent transform of a version of *Negative Frame Axiom*, by the following explanation closure axiom:
$\forall y \forall s \forall a.\ \neg clear(y, s) \wedge clear(y, do(a, s)) \rightarrow$
$(\,\exists x(on(x, y, s) \wedge a = moveToTable(x)) \vee \exists x(on(x, y, s) \wedge \exists z(a = move(x, z)))\,)$
Exercise: rewrite this as an equivalent negative frame axiom.

Similarly, *the negative effect axiom in normal form summarizes all the cases when actions a result in* $\neg clear(y, do(a, s))$. This yields the following explanation closure axiom (which is a variant of a Positive Frame Axiom):
$\forall y \forall s \forall a.\ clear(y, s) \wedge \neg clear(y, do(a, s)) \rightarrow \exists x(a = move(x, y))$.
Exercise: rewrite this as an equivalent positive frame axiom.

Combining all these after doing logically equivalent simplifications produces the desired *successor state axiom* that solves the FP thanks to $\forall \mathbf{a}$ at front:
$\forall y \forall s \forall \mathbf{a}.\ clear(y, do(\mathbf{a}, s)) \leftrightarrow (\exists x(\mathbf{a} = moveToTable(x) \wedge on(x, y, s)) \vee$
$\exists x \exists z(\mathbf{a} = move(x, z)) \wedge on(x, y, s)\,) \vee$
$clear(y, s) \wedge \neg \exists x(\mathbf{a} = move(x, y))$.

Reiter's solution to the FP: functional fluents

The above example demonstrated Reiter's solution for the relational fluent. For a functional fluent we reason similarly.

First, transform an effect axiom into a normal form:

$$\forall x \forall c \forall s.\ colour(x, do(paint(x,c), s)) = c\ \equiv$$
$$\forall x \forall c \forall s \forall \mathbf{a}.\ \mathbf{a} = paint(x,c) \rightarrow colour(x, do(\mathbf{a}, s)) = c.$$

Second, formulate the causal explanation axiom:

$$\forall x \forall c \forall s \forall \mathbf{a}.\ colour(x,s) = c \wedge \neg(colour(x, do(\mathbf{a}, s)) = c) \rightarrow$$
$$\exists c' (\mathbf{a} = paint(x, c') \wedge c \neq c')$$

Notice this is equivalent to the frame axiom /*since $(A \wedge \neg B) \rightarrow C \equiv (A \wedge \neg C) \rightarrow B$*/

$$\forall x \forall c \forall s \forall \mathbf{a}. colour(x,s) = c \wedge \neg \exists c' (\mathbf{a} = paint(x, c') \wedge c \neq c') \rightarrow colour(x, do(\mathbf{a}, s)) = c.$$

Third, combine axioms, do logical simplifications to get a SSA:

$$\forall x \forall c \forall s \forall \mathbf{a}.\ colour(x, do(\mathbf{a}, s)) = c \leftrightarrow \mathbf{a} = paint(x,c) \vee$$
$$colour(x,s) = c \wedge \neg \exists c' (\mathbf{a} = paint(x, c') \wedge c \neq c').$$

Notice this solves the FP thanks to ($\forall a$) at front: all actions have no effect on *colour* except of painting action. So, the frame axiom is implicitly included in this successor state axiom. We achieved this feat by writing effect axiom in normal form, and then by appealing to the causal completeness assumption, when we said that nothing else can be responsible for change in colour. See Reiter's book "Knowledge in Action", Section 3.2.6, for another example.

Reiter's solution is SSA: conclusion

in a general case, we obtain the successor state axiom (SSA)

$$\forall s \forall \vec{x} \forall a.\ F(\vec{x}, do(a,s)) \leftrightarrow \gamma_F^+(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg \gamma_F^-(\vec{x}, a, s),$$

where $\gamma_F^+(\vec{x}, a, s)$ is a formula summarizing all positive effects on F, and $\gamma_F^-(\vec{x}, a, s)$ is a formula representing all negative effects on F.
Conclusion: *SSA is equivalent to conjunction of effect axioms in normal form and causal explanation axioms* (which are essentially frame axioms with $\forall a$).

We formulated Causal Completeness Assumption under an implicit *Unique Names Axioms* (UNA) for Actions:

- For distinct action names A_1 and A_2, $\forall \vec{x} \forall \vec{y}.\ \neg(A_1(\vec{x}) = A_2(\vec{y}))$
- Identical actions have identical arguments:
  $\forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_n.\ A(x_1, \ldots, x_n) = A(y_1, \ldots, y_n) \rightarrow x_1 = y_1 \wedge \cdots x_n = y_n$

Comment: the consistency assumption that the sentence
$\neg \exists \vec{x} \exists a \exists s. (\gamma_F^-(\vec{x}, a, s) \wedge \gamma_F^+(\vec{x}, a, s))$
should be entailed by the underlying KB simply guarantees the integrity of the positive and negative effect axioms. Under this consistency assumption, it will be impossible for both $F(\vec{x}, do(a,s))$ and $\neg F(\vec{x}, do(a,s))$ to be simultaneously derived. Notice that by the unique names axioms for actions, this condition is satisfied by the example about the fluent *clear*.

Reiter's solution to the FP: summary

Reiter's solution to the frame and qualification problems includes the following groups of axioms.

Successor state axiom (SSA): for each fluent F
$\forall s \forall \vec{x} \forall a.\ F(\vec{x}, do(a,s)) \leftrightarrow \gamma_F^+(\vec{x}, a, s)) \vee F(\vec{x}, s) \wedge \neg \gamma_F^-(\vec{x}, a, s).$
If there are $\mathcal{F}$ fluents, then we need $\mathcal{F}$ axioms of this kind.

For each action A, a single action precondition axiom of the form
$\forall s \forall \vec{x} \forall a.\ poss(A(\vec{x}), s) \leftrightarrow \Pi_A(\vec{x}, s).$
If there are $\mathcal{A}$ actions, then we need $\mathcal{A}$ precondition axioms.

Unique name axioms (UNA) for actions: the order of $\mathcal{A}^2$ axioms. Ignoring UNA, Reiter's axiomatization requires $\mathcal{F} + \mathcal{A}$ axioms in total, compared with the $2 \cdot \mathcal{F} \cdot \mathcal{A}$ explicit frame axioms that would otherwise be required. Moreover, each SSA is relatively short because most actions do not affect F.

The conciseness and effectiveness of this axiomatization relies on

- ▶ Quantification $\forall a$ over actions in normal form effect axioms.
- ▶ The practically confirmed intuition that relatively few actions affect a given fluent, i.e., for each fluent, there are few effect axioms.
- ▶ The causal completeness assumption with ($\forall a$) over actions.

Appendix includes the detailed proof showing how the SSA can be derived in a general case from the effect axioms and explanation closure axioms.

Axioms for situations: analogy with the Peano axioms

We also need axioms to reason about situations. Before we formulate them, recall FOL theory of a single successor function. The Peano axioms for non-negative numbers use second order language based on $0, s, <, =$.

$\forall x \forall y.\ s(x) = s(y) \rightarrow x = y$

$\forall x.\ \neg(x < 0)$

$\forall x \forall y.\ x < s(y) \leftrightarrow x \leq y$, where $x \leq y \overset{def}{=} (x < y \vee x = y)$.

$\forall P.\ \big(P(0) \wedge \forall x (P(x) \rightarrow P(s(x)))\big) \rightarrow \forall x (P(x))$
This axiom characterizing the domain as the smallest set such that 0 is in the set, and whenever x is in the set, so is $s(x)$.

This second order (with $\forall P$) Peano axioms have only one standard model. We cannot stay within FOL if we want non-standard models to be excluded. We know second order logic is computationally intractable, but we shall learn how to circumvent these difficulties.

Details: H.Enderton's book "A Mathematical Introduction to Logic", 2nd Ed., Academic Press, 2001: see Section 3.1 "Natural Numbers with Successors".

The main issue now: each situation has many successors. Each action leads to a new successor situation. Let's divide the universum into 3 disjoint parts each representing a separate sort: *situations*, *objects* and *actions*.

Foundational axioms Σ for situations

The unique initial situation S_0 is like the number 0. Unlike Peano axioms which have a unique successor function, we have a family of successor functions $do : action \times situation \mapsto situation$.

$$\forall a_1 \forall a_2 \forall s_1 \forall s_2.\ do(a_1, s_1) = do(a_2, s_2) \rightarrow a_1 = a_2 \wedge s_1 = s_2$$

$$\forall s.\ \neg(s \sqsubset S_0)$$

$$\forall a \forall s \forall s'.\ s \sqsubset do(a, s') \leftrightarrow s \sqsubseteq s', \text{ where } s \sqsubseteq s' \overset{def}{=} (s \sqsubset s' \vee s = s').$$

$$\forall P.\ (P(S_0) \wedge \forall a \forall s (P(s) \rightarrow P(do(a, s)))) \rightarrow \forall s (P(s))$$

The second order axiom limits the sort *situation* to the smallest set containing S_0 and closed under the application of *do* to an action and a situation.

These axioms say that the set of situations is really a tree. No cycles, no merging. These foundational axioms Σ are domain independent. They provide the basic properties of situations in any application specific axiomatization of particular fluents and actions.

Situations are finite sequences of actions that can be implemented as lists in Prolog. S_0 is like [ ], and $do(A, S)$ adds an action A at front of a list $[A|S]$. Therefore, when situation lists are read from right to left, the relation $s \sqsubset s'$ means that situation s is a proper sub-list (suffix) of the situation s'.

Lesson: Straightforward implementations of situations satisfy axioms in Σ.
Comment: we'll see these axioms are **not** important in automated planning.

The Tree of Situations

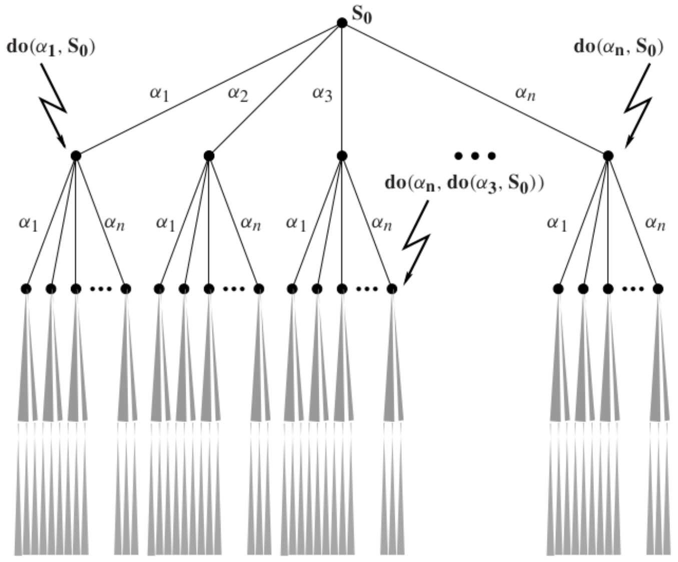


Figure 4.1: The tree of situations for a model with n actions.

Figure: Acknowledgement: Ray Reiter "Knowledge in Action", Section 4.2.2, Page 51

Logical consequences from foundational axioms Σ

The foundational axioms Σ have a number of natural logical consequences.

$\forall s\ (S_0 \sqsubseteq s)$	(S_0 is before any other situation)
$\forall s.\ s = S_0 \vee \exists a \exists s' (s = do(a, s'))$	(Existence of a predecessor)
$\forall a \forall s.\ \neg(S_0 = do(a, s))$	(S_0 is distinct from other situations)
$\forall s_1 \forall s_2.\ s_1 \sqsubset s_2 \rightarrow \neg(s_1 = s_2)$	(Unique names)
$\forall s\ \neg(s \sqsubset s)$	(Anti-reflexivity)
$\forall s \forall s'.\ s \sqsubset s' \rightarrow \neg(s' \sqsubset s)$	(Anti-symmetry)
$\forall s_1 \forall s_2 \forall s_3.\ s_1 \sqsubset s_2 \wedge s_2 \sqsubset s_3 \rightarrow s_1 \sqsubset s_3$	(Transitivity)
$\forall s \forall s'.\ s \sqsubseteq s' \wedge s' \sqsubseteq s \ \rightarrow\ s = s'.$	

Executable Situations: "Green" Edges in the Tree

A situation is a sequence of actions. There are no constraints on the actions entering into such a sequence, so that it may not be possible to actually execute these actions one after the other. Proceed along "green edges" only.

Executable situations: Action histories in which it is actually possible to perform the actions one after the other.

$$s \prec s' \overset{def}{=} s \sqsubset s' \wedge \forall a \forall s^* (s \sqsubset do(a, s^*) \sqsubseteq s' \rightarrow poss(a, s^*)),$$

where $s \prec s'$ means that s is an initial sub-sequence of s', and all the actions occurring between s and s' can be executed one after the other.

We use the convenient abbreviation $s \preceq s' \overset{def}{=} (s \prec s') \vee s = s'$. Subsequently, $executable(s) \overset{def}{=} S_0 \preceq s$. This definition has a few logical consequences.

$$\forall a \forall s.\ executable(do(a, s)) \leftrightarrow executable(s) \wedge poss(a, s).$$

$$\forall s. executable(s) \leftrightarrow s = S_0 \vee \exists a \exists s' (s = do(a, s') \wedge poss(a, s') \wedge executable(s'))$$

Let an abbreviation $do([\alpha_1, \cdots, \alpha_n], S_0))$ represents situation $do(\alpha_n, do(\cdots, do(\alpha_1, S_0) \cdots))$ resulting from execution of ground action terms $\alpha_1, \cdots, \alpha_n$ in S_0. One can prove $executable(do([\alpha_1, \cdots, \alpha_n], S_0)) \leftrightarrow poss(\alpha_1, S_0) \wedge \bigwedge_{i=2}^{n} poss(\alpha_i, do([\alpha_1, \cdots, \alpha_{i-1}], S_0))$.

Basic Action Theory (BAT)

Definition A formula is *uniform* in s iff it mentions neither the predicate *poss*, nor $\sqsubset$, it does not quantify over variables of sort situation, it does not mention equality on situations, and whenever it mentions a term of sort situation inside a fluent, then that term is s.

Let Σ be the foundational axioms for situations

$\mathcal{D}_{\mathbf{ss}}$ be a set of SSA of the form $\forall a \forall s \forall \vec{x}.\ F(\vec{x}, do(a,s)) \leftrightarrow \Phi_F(\vec{x}, a, s)$, where $\Phi_F(\vec{x}, a, s)$ is a formula that is uniform in s. (It includes $\gamma_F^+(\vec{x}, a, s)$, $F(\vec{x}, s)$ and $\gamma_F^-(\vec{x}, a, s)$ as subformulas.)

$\mathcal{D}_{\mathbf{ap}}$ be a set of action precondition axioms of the form

$$\forall s \forall \vec{x}.\ poss(A(\vec{x}), s) \leftrightarrow \Pi_A(\vec{x}, s),$$

where $\Pi_A(\vec{x}, s)$ is a formula uniform in s, and A is an n-ary action function.

$\mathcal{D}_{una}$ be a set of unique name axioms for actions.

$\mathcal{D}_{S_0}$ is a set of FOL formulas whose only situation term is S_0. It specifies the values of all fluents in the initial situation. It also describes all the static facts. In particular, it includes unique name axioms for object constants./*No DCA*/

Basic action theory (BAT) $\mathcal{D}$ is the conjunction of axioms

$$\mathcal{D} = \Sigma \wedge \mathcal{D}_{ss} \wedge \mathcal{D}_{ap} \wedge \mathcal{D}_{una} \wedge \mathcal{D}_{S_0}$$

Theorem (F.Pirri & R.Reiter, J. ACM, 1999, vol 46, N3, p.325–361)
$\mathcal{D}$ is satisfiable iff $\mathcal{D}_{una} \wedge \mathcal{D}_{S_0}$ is satisfiable (i.e., no Σ is needed!).

This result is the key to tractability since $\mathcal{D}_{una} \wedge \mathcal{D}_{S_0}$ are sentences in FOL.

The Blocks World [Reiter,Sections 6.5.2 and 10.2]

To simplify notation, we assume all free vars are implicitly $\forall$-quantified at front.

`Action Precondition Axioms.`

$poss(move(x,y), s) \leftrightarrow clear(x,s) \wedge clear(y,s) \wedge x \neq y.$
$poss(moveToTable(x), s) \leftrightarrow clear(x,s) \wedge \exists y(on(x,y,s)).$

`Successor State Axioms.`

$on(x,y,do(a,s)) \leftrightarrow a = move(x,y) \vee on(x,y,s) \wedge a \neq moveToTable(x) \wedge \neg\exists z(a = move(x,z)).$
$ontable(x, do(a,s)) \leftrightarrow a = moveToTable(x) \vee ontable(x,s) \wedge \neg\exists y(a = move(x,y)).$
$clear(x, do(a,s)) \leftrightarrow \exists y\,(\exists z(a = move(y,z)) \vee a = moveToTable(y) \wedge on(y,x,s)) \vee$
$clear(x,s) \wedge \neg\exists y(a = move(y,x)).$

`Unique Name Axioms (UNA) for Actions.`

$\forall x, y, z.move(x,y) \neq moveToTable(z)$
$\forall x', y', x'', y''.move(x',y') = move(x'',y'') \rightarrow x' = x'' \wedge y' = y''$
$\forall x, y.moveToTable(x) = moveToTable(y) \rightarrow x = y$

`(Incomplete) Initial Theory.` Note: no Domain Closure Axiom (DCA)

$on(B, C, S_0) \wedge ontable(C, S_0) \wedge ontable(D, S_0) \wedge \neg on(P, D, S_0) \wedge$
$\exists x(on(x, D, S_0)) \wedge \underline{\forall y(y \neq D \wedge y \neq C \rightarrow clear(y, S_0))}$ /*$\forall y$ can be over ∞-many*/
$\wedge(\forall x \forall y.on(x,y,S_0) \rightarrow \neg on(y,x,S_0)) \wedge$
$(\forall x \forall y \forall z.on(y,x,S_0) \wedge on(z,x,S_0) \rightarrow y = z) \wedge$
$(\forall x \forall y \forall z.on(x,y,S_0) \wedge on(x,z,S_0) \rightarrow y = z).$ /*also UNA for constants*/

The axioms in red are *state constraints*. One can prove that for any situation s such that *executable*(s) all state constraints hold wrt s.

Initial theory is **using fluent literals only** (no disjunctions). The names of some blocks are not known, and they are $\exists$-quantified. Our **initial knowledge is incomplete**.

The Projection and Executability Problems

Given an action sequence $\alpha_1, \ldots, \alpha_n$ of ground action terms, and a query $Q(s)$ whose only free variable is the situation variable s, what is the answer to Q in that situation resulting from performing this action sequence from S_0 ?

This is the *projection problem*. Define this formally as the problem of determining whether $BAT \models^? Q(do([\alpha_1, \ldots, \alpha_n], S_0))$.

The *executability problem*: Determine whether ground actions $\alpha_1, \ldots, \alpha_n$ can be executed in sequence one after another. Formally, is it the case that

$$BAT \models^? executable(do([\alpha_1, \ldots, \alpha_n], S_0)).$$

We are going to see that both these computational problems can be solved by defining a special purpose *regression* operator $\mathcal{R}$ applicable to a formula with respect to a ground situation term. This is a recursive operator that will reduce a given formula to another formula with respect to S_0 only.

Then, we will see that the projection problem can be reduced to $\mathcal{D}_{una} \wedge \mathcal{D}_{S_0} \models^? \mathcal{R}[Q(do([\alpha_1, \ldots, \alpha_n], S_0))]$. Notice: LHS has only FOL axioms about initial situation and UNAs, while RHS will be a formula about S_0 only.

Similarly the executability problem can be reduced to
$\mathcal{D}_{una} \wedge \mathcal{D}_{S_0} \models^? poss(\alpha_1, S_0) \wedge \bigwedge_{i=2}^{n} \mathcal{R}[poss(\alpha_i, do([\alpha_1, \cdots, \alpha_{i-1}], S_0))]$.

Alternative approach: compute *progression* of $\mathcal{D}_{S_0}$ incrementally through actions $\alpha_1, \cdots, \alpha_n$ and then check if the query is true now. Next Lecture.

Goal Regression Operator: Defined Recursively

Assume W is a *regressable formula*. The essence of W is that each of its ground situation terms is rooted at S_0. Therefore, one can tell, by inspection of such a term, exactly what actions it involves.

For simplicity, let a regressable formula W mention no functional fluents, but only relational fluents.

Assume a background axiomatization that includes a set of successor state and action precondition axioms.

Intuitively, the regression operator eliminates Poss atoms in favour of their definitions as given by action precondition axioms, and replaces fluent atoms about $do(A, S)$ by logically equivalent expressions about S as given by SSA. Moreover, it repeatedly does this until it cannot make such replacements any further. In Prolog: "fluent(Arg1,[A | S])" rule calls recursively fluent(Arg2,S).

Suppose W is an atom. Since W is regressable, there are four possibilities.

(a) W is a situation independent atom. Then, $R[W] = W$.

(b) W is a relational fluent atom of the form $F(\vec{t}, S_0)$. Then, $R[W] = W$.

Regression: suppose W is an atomic formula

(c) W is a regressable poss atom, so it has the form $poss(A(\vec{t}), S)$ for an action term $A(\vec{t})$ and situation S. In BAT, there must be an action precondition axiom for A of the form $\forall s \forall \vec{x}.\ poss(A(\vec{x}), s) \leftrightarrow \Pi_A(\vec{x}, s)$. Then,

$$\mathcal{R}[\, poss(A(\vec{t}), S)\,] = \mathcal{R}[\, \Pi_A(\vec{t}, S)\,].$$

In other words, replace the atom $poss(A(\vec{t}), S)$ by a suitable instance of the RHS of A's action precondition axiom, and regress that expression. If action $A(\vec{t})$ has free variables as arguments, and if there are quantified variables in $\Pi_A(\vec{x}, s)$, we rename these quantified vars to prevent the quantifiers from capturing free variables in $A(\vec{t})$.

(d) W is a relational fluent atom of the form $F(\vec{t}, do(A, S))$. Let F's SSA in BAT be of the form $\forall a \forall s \forall \vec{x}.\ F(\vec{x}, do(a, s)) \leftrightarrow \Phi_F(\vec{x}, a, s)$. Then,

$$\mathcal{R}[\, F(\vec{t}, do(A, S))\,] = \mathcal{R}[\, \Phi_F(\vec{t}, A, S)\,].$$

In other words, replace the atom $F(\vec{t}, do(A, S))$ by a suitable instance of the RHS of the equivalence in F's SSA, and regress this formula.

If there are free variables as arguments in $F(\vec{t}, do(A, S))$, and if there are any quantified variables in $\Phi_F(\vec{x}, a, s)$, then rename them to prevent the quantifiers (if any) in $\Phi_F(\vec{x}, a, s)$ from capturing free variables in the instance $F(\vec{t}, do(A, S))$.

Regression: non-atomic formulas.

For non-atomic formulas W, regression is defined recursively over structure:

$$\mathcal{R}[\neg W] = \neg \mathcal{R}[W],$$
$$\mathcal{R}[W_1 \wedge W_2] = \mathcal{R}[W_1] \wedge \mathcal{R}[W_2],$$
$$\mathcal{R}[\exists v(W)] = \exists v.\, \mathcal{R}[W].$$

Conclusion: Each $\mathcal{R}$-step reduces the depth of nesting of the symbol do in the fluents of W by substituting suitable instances of the right hand side (RHS) of a SSA for each occurrence of a fluent atom of the form $F(t_1, \ldots, t_n, do(A, S))$. This reduces the depth of nesting of $do(\cdot, \cdot)$ by one.

Theorem (The Regression Theorem) Suppose W is a regressable sentence of situation calculus that mentions no functional fluents, and $\mathcal{D}$ is a basic theory of actions. Then, $\mathcal{D} \models W$ iff $\mathcal{D}_{S_0} \wedge \mathcal{D}_{una} \models R[W]$. /* No Σ on LHS */
Proof: F. Pirri and R.Reiter, J. ACM, 1999, vol. 46(3): pages 325-361.

Sometimes, we need a single-step regression operator $\rho[F; A]$, where F is a fluent, A is a specific action. Let SSA be

$\forall a \forall s \forall \vec{x}.\ F(\vec{x}, do(a, s)) \leftrightarrow \gamma_F^+(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg \gamma_F^-(\vec{x}, a, s).$

To compute $\rho[F; A]$ replace the variable a in the RHS of this SSA with the action A and do simplifications using UNA and FOL equivalences. The resulting formula is $\rho[F; A]$. For any formula W, $\rho[W; A]$ is defined similarly.

Example: Student Records Administration

Fluents: $enrolled(st, course, s)$: st is enrolled in $course$ in situation s.
$grade(st, course, grade, s)$: The grade of st in $course$ is $grade$ in situation s.
$preReq(pre, course)$: pre is a prerequisite for $course$ in s.

Initial state: arbitrary FOL sentences, the only restriction being that fluents mention only the initial situation S_0.
$(enrolled(Sue, C100, S_0) \vee enrolled(Sue, C200, S_0)) \wedge \forall p(\neg preReq(p, C100))$
$\exists c(enrolled(Bill, c, S_0)) \wedge \forall p(preReq(p, P300) \leftrightarrow (p = P100 \vee p = M100)) \wedge$
$\forall c(enrolled(Bill, c, S_0) \leftrightarrow (c = M100 \vee c = M110 \vee c = P200)) \wedge$
$enrolled(Mary, C100, S_0) \wedge \neg enrolled(John, M200, S_0) \wedge \ldots$
$grade(Sue, P300, 75, S_0) \wedge grade(Bill, M200, 70, S_0) \wedge \ldots$

Actions, there are 3 actions: $register(st, c)$, $change(st, c, g)$, $drop(st, c)$.

Precondition axioms for actions:
A student can register in a course iff she has obtained a grade of at least 50 in all prerequisites for the course: $\forall st \forall c \forall s.\ poss(register(st, c), s) \leftrightarrow$
$\forall p(preReq(p, c) \rightarrow \exists g(grade(st, p, g, s) \wedge g \geq 50))$.

It is possible to change a student's grade iff an old grade is different:
$\forall st \forall c \forall g \forall s.\ poss(change(st, c, g), s) \leftrightarrow \exists g'(grade(st, c, g', s) \wedge g' \neq g)$.

A student may drop a course iff the student is currently enrolled in that course: $\forall st \forall c \forall s.\ poss(drop(st, c), s) \leftrightarrow enrolled(st, c, s)$.

Example: Effect Axioms and SSAs

$\forall st \forall c \forall s.$	$\neg enrolled(st, c, do(drop(st, c), s))$
$\forall st \forall c \forall s.$	$enrolled(st, c, do(register(st, c), s))$
$\forall st \forall c \forall g \forall s.$	$grade(st, c, g, do(change(st, c, g), s))$
$\forall st \forall c \forall g \forall g' \forall s.$	$g' \neq g \rightarrow \neg grade(st, c, g, do(change(st, c, g'), s))$

First, we have to transform these effect axioms in normal form. How ?

$\forall st \forall c \forall s \forall a.\ \ a = drop(st, c) \rightarrow \neg enrolled(st, c, do(a, s))$
$\forall st \forall c \forall s \forall a.\ \ a = register(st, c) \rightarrow enrolled(st, c, do(a, s))$
$\forall st \forall c \forall g \forall s \forall a.\ a = change(st, c, g) \rightarrow grade(st, c, g, do(a, s))$
$\forall st \forall c \forall g \forall s \forall a.\ \exists g'(a = change(st, c, g') \wedge g' \neq g) \rightarrow \neg grade(st, c, g, do(a, s))$

Solve the frame problem by appealing to the causal completeness assumption: add explanation closure axioms. This yields the following SSAs:

$\forall st \forall c \forall s \forall a.\ enrolled(st, c, do(a, s)) \leftrightarrow a = register(st, c) \vee$
$enrolled(st, c, s) \wedge \neg(a = drop(st, c))$
$\forall st \forall c \forall g \forall s \forall a.\ grade(st, c, g, do(a, s)) \leftrightarrow a = change(st, c, g) \vee$
$grade(st, c, g, s) \wedge \neg \exists g'(a = change(sr, c, g'))$.

Using these precondition and successor state axioms, we can see how regression can help answer projection and executability queries.

Example: Executability Testing

Compute whether the following sequence of transactions is executable: $register(Bill, C100), drop(Bill, C100), drop(Bill, C100)$ which intuitively should fail because the first drop leaves Bill unenrolled in $C100$, so that the precondition for the second drop will be false. By regression theorem,

$$\begin{aligned}&\mathcal{R}[executable([register(Bill, C100), drop(Bill, C100), drop(Bill, C100)], S_0)] =\\ &\mathcal{R}[poss(register(Bill, C100), S_0)]\wedge\\ &\mathcal{R}[poss(drop(Bill, C100), do(register(Bill, C100), S_0))]\wedge\\ &\mathcal{R}[poss(drop(Bill, C100), do(drop(Bill, C100), do(register(Bill, C100), S_0)))].\end{aligned}$$

Using preconditions, occurrences of $poss$ can be equivalently replaced.

$$\begin{aligned}&\mathcal{R}[\forall p.[preReq(p, C100) \rightarrow \exists g(grade(Bill, p, g, S_0) \wedge g \geq 50)]\wedge\\ &\mathcal{R}[enrolled(Bill, C100, do(register(Bill, C100), S_0))]\wedge\\ &\mathcal{R}[enrolled(Bill, C100, do(drop(Bill, C100), do(register(Bill, C100), S_0)))].\end{aligned}$$

Notice, in the top line, regression is applied to implication between atoms with S_0. Therefore, the resulting expression remains same. In the next two lines, since regression applies to fluent atoms, each expression can be simplified using SSAs. In the SSA for $enrolled$, replace the variables st, c with $Bill, C100$, the variable a with $drop(Bill, C100)$ and s with $do(register(Bill, C100), S_0)$. Simplify.

This yields a logically equivalent expression
$[\forall p.[preReq(p, C100) \rightarrow \exists g(grade(Bill, p, g, S_0) \wedge g \geq 50)] \wedge True \wedge False$.
So the transaction sequence is indeed impossible.

Example: regression of a query wrt sequence T

Transaction $\mathbf{T} = [change(Bill, C100, 60), register(Sue, C200), drop(Bill, C100)]$.
Given a query $BAT \models^? Q(do(T, S_0))$, by the Regression Theorem, we regress the query and check if the result is a logical consequence from the axioms for the initial situation together with unique names axioms. Example of a query:

$$\begin{aligned}BAT \models^? \exists st.\ \ &enrolled(st, C200, do(\mathbf{T}, S_0))\wedge\\ &\neg enrolled(st, C100, do(\mathbf{T}, S_0))\wedge\\ &\exists g(grade(st, C200, g, do(\mathbf{T}, S_0)) \wedge g \geq 50.\end{aligned}$$

Since query is the conjunction of fluent atoms, regression of each fluent involves using SSA for the fluent with respect to the given sequence of actions $\mathbf{T}$. For example, for the fluent $\mathcal{R}[enrolled(st, C200, do(\mathbf{T}, S_0))]$ only the action $register(Sue, C200)$ may have any effect. Therefore, regression using SSA for this fluent and UNA for actions yields the following RHS

$$\begin{aligned}&(register(Sue,C200) = register(st,C200) \vee enrolled(st,C200, do(change(Bill,C100, 60),S_0)))\\ &\equiv (st = Sue \vee enrolled(st, C200, S_0)).\end{aligned}$$

Using negation of the RHS of the same SSA, $\mathcal{R}[\neg enrolled(st, C100, do(\mathbf{T}, S_0))]$=

$$\begin{aligned}&drop(Bill,C100) \neq register(st, C100)\wedge\\ &\quad(\neg enrolled(st,C100,S_0) \vee drop(Bill,C100) = drop(st,C100)\end{aligned}$$

Equivalent simplifications assuming that $D_{una} \models C100 \neq C200$ yield

$$\begin{aligned}\exists st.\ \big(\ &(st = Sue \vee enrolled(st, C200, S_0)) \wedge\\ &(st = Bill \vee \neg enrolled(st, C100, S_0)) \wedge\\ &\exists g\,(grade(st, C200, g, S_0) \wedge g \geq 50)\ \big).\end{aligned}$$

The answer to the query is obtained by evaluating this last formula in $\mathcal{D}_{S_0}$.

Appendix: Reiter's Solution in General Case

The following slides are optional . They present a formal proof of how SSA can be obtained from normalized effect axioms and explanation closure axioms in a general case.

Normal Form for Effect Axioms (in the General Case)

Suppose each of the given positive effect axioms has the form:

$$\forall s \forall \underline{\vec{y}}.\ \phi^+_F(s) \rightarrow F(\underline{\vec{t}}, do(A, s)),$$

where A is an action term, $\vec{t}$ is a tuple of terms, $\phi^+_F(s)$ is a FOL formula about situation s representing a context condition when an action A has an effect on a fluent F, $\underline{\vec{y}}$ are object variables, if any, that may occur in A or in ϕ^+_F.

This can be rewritten in the equivalent form /* Recall the logical equivalence $\forall y(B(y) \rightarrow C) \equiv (\exists y B(y)) \rightarrow C$ */

$$\forall s \forall \vec{x} \forall a.\ (\exists \vec{y}(a = A \wedge \vec{x} = \vec{t} \wedge \phi^+_F(s))\,) \rightarrow F(\vec{x}, do(a, s)).$$

We write $\vec{x} = \vec{t}$ as an abbreviation for $x_1 = t_1 \wedge \cdots \wedge x_n = t_n$, where $\vec{x}$ are fresh variables distinct from any $\vec{y}$ occurring in the original effect axiom, if any.
/*Notice we use repeatedly the equivalence $\forall x(x = C \rightarrow F(x)) \equiv F(C)$ and the equivalence $A \rightarrow (B \rightarrow C) \equiv \neg A \vee \neg B \vee \neg C \equiv \neg(A \wedge B) \vee C \equiv (A \wedge B) \rightarrow C$ */

To summarize each of the p positive effect axioms for fluent F can be written as

$$\begin{gathered}\forall s \forall \vec{x} \forall a.\ RHS^F_1(\vec{x}, a, s) \rightarrow F(\vec{x}, do(a, s))\\ \vdots\\ \forall s \forall \vec{x} \forall a.\ RHS^F_p(\vec{x}, a, s) \rightarrow F(\vec{x}, do(a, s))\end{gathered}$$

These p sentences are equivalent to a single *effect axiom in normal form*

$$\forall s \forall \vec{x} \forall a.\ \gamma^+_F(\vec{x}, a, s) \rightarrow F(\vec{x}, do(a, s)),$$

where $\gamma^+_F(\vec{x}, a, s)$ is an abbreviation for $RHS^F_1(\vec{x}, a, s) \vee \cdots \vee RHS^F_p(\vec{x}, a, s)$.

Similarly, compute the normal form for the negative effect axioms for fluent F.

Reiter's Solution to the FP: the General Case

To generalize the previous examples we suppose given, for each fluent F, the following two normal form effect axioms.

Positive Normal Form Effect Axiom (PNFEA) for fluent F

$$\forall s \forall \vec{x} \forall a.\ \gamma_F^+(\vec{x}, a, s) \rightarrow F(\vec{x}, do(a, s)) \quad \text{(PNFEA)}$$

Negative Normal Form Effect Axiom (NNFEA) for fluent F

$$\forall s \forall \vec{x} \forall a.\ \gamma_F^-(\vec{x}, a, s) \rightarrow \neg F(\vec{x}, do(a, s)) \quad \text{(NNFEA)}$$

Causal Completeness Assumption: Axioms (PNFEA) and (NNFEA), respectively, characterize all the conditions under which action a can lead to F becoming true (respectively, false) in the successor situation. They are all the causal laws for the fluent F.

Hence, if F's truth value changes from *false* to *true* as a result of doing a, then $\gamma_F^+(\vec{x}, a, s)$ must be *true*. Similarly, if F's value changes from *true* to *false*.

This informally stated assumption can be represented axiomatically by the following Explanation Closure Axioms (ECA)

$$\forall s \forall \vec{x} \forall a.\ F(\vec{x}, s) \wedge \neg F(\vec{x}, do(a, s)) \rightarrow \gamma_F^-(\vec{x}, a, s) \quad \text{(ECA1)}$$
$$\forall s \forall \vec{x} \forall a.\ \neg F(\vec{x}, s) \wedge F(\vec{x}, do(a, s)) \rightarrow \gamma_F^+(\vec{x}, a, s) \quad \text{(ECA2)}$$

Notice (ECA1) and (ECA2) express compactly all the frame axioms because all relevant action terms are mentioned in $\gamma_F^+(\vec{x}, a, s), \gamma_F^-(\vec{x}, a, s)$.

Reiter's Solution: Equivalent Transformations.

From (ECA1) we can obtain a positive frame axiom (PFA) for F:

$$\begin{aligned} \text{(ECA1)} &\equiv \forall s \forall \vec{x} \forall a.\ \neg F(\vec{x}, s) \vee F(\vec{x}, do(a, s)) \vee \gamma_F^-(\vec{x}, a, s) \\ &\equiv \forall s \forall \vec{x} \forall a.\ \neg F(\vec{x}, s) \vee \neg(\neg\gamma_F^-(\vec{x}, a, s)) \vee F(\vec{x}, do(a, s)) \\ &\equiv \forall s \forall \vec{x} \forall a.\ (\underline{F(\vec{x}, s)} \wedge \neg\gamma_F^-(\vec{x}, a, s)) \rightarrow \underline{F(\vec{x}, do(a, s))} \quad \text{(PFA)} \end{aligned}$$

Similarly, from (ECA2) we can obtain a negative frame axiom (NFA) for F:

$$\begin{aligned} \text{(ECA2)} &\equiv \forall s \forall \vec{x} \forall a.\ F(\vec{x}, s) \vee \neg F(\vec{x}, do(a, s)) \vee \gamma_F^+(\vec{x}, a, s) \\ &\equiv \forall s \forall \vec{x} \forall a.\ \neg F(\vec{x}, do(a, s)) \vee (\ F(\vec{x}, s) \vee \gamma_F^+(\vec{x}, a, s)\ ) \\ &\equiv \forall s \forall \vec{x} \forall a.\ F(\vec{x}, do(a, s)) \rightarrow (F(\vec{x}, s) \vee \gamma_F^+(\vec{x}, a, s)) \quad \text{(NFA)} \end{aligned}$$

Since $(NNFEA) \equiv \forall s \forall \vec{x} \forall a.\ F(\vec{x}, do(a, s)) \rightarrow \neg\gamma_F^-(\vec{x}, a, s)$ and because in propositional logic $((A \rightarrow B) \wedge (A \rightarrow C)) \equiv (A \rightarrow (B \wedge C))$, (NFA) and (NNFEA) yield

$$(\forall s \forall \vec{x} \forall a.\ F(\vec{x}, do(a, s)) \rightarrow \neg\gamma_F^-(\vec{x}, a, s) \wedge (F(\vec{x}, s) \vee \gamma_F^+(\vec{x}, a, s))\ ) \equiv$$
$$\forall s \forall \vec{x} \forall a.\ F(\vec{x}, do(a, s)) \rightarrow F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s) \vee \neg\gamma_F^-(\vec{x}, a, s) \wedge \gamma_F^+(\vec{x}, a, s))\ (*)$$

Consistency assumption: assume underlying KB about actions entails

$$\neg\exists \vec{x} \exists a \exists s.(\gamma_F^-(\vec{x}, a, s) \wedge \gamma_F^+(\vec{x}, a, s)) \equiv \forall s \forall \vec{x} \forall a(\gamma_F^+(\vec{x}, a, s) \rightarrow \neg\gamma_F^-(\vec{x}, a, s))$$

Since $B \wedge A \equiv A$, if $A \models B$, under consistency assumption the sentence $(*) \equiv \forall s \forall \vec{x} \forall a.\ F(\vec{x}, do(a, s)) \rightarrow F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s) \vee \gamma_F^+(\vec{x}, a, s))$

Reiter's Solution to the FP: Conclusion

(PNFEA) and (PFA) together are logically equivalent to
$\forall s \forall \vec{x} \forall a.\ (\gamma_F^+(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s)\ ) \rightarrow F(\vec{x}, do(a, s))$

Therefore, we obtain the successor state axiom (SSA)

$$\forall s \forall \vec{x} \forall a.\ F(\vec{x}, do(a, s)) \leftrightarrow \gamma_F^+(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s)$$

Conclusion: SSA is equivalent to conjunction of effect axioms in normal form and causal explanatory axioms (which are essentially frame axioms).

We formulated Causal Completeness Assumption under an implicit *Unique Names Axioms* (UNA) for Actions:

- For distinct action names A and B, $\forall \vec{x} \forall \vec{y}.\ \neg(A(\vec{x}) = B(\vec{y}))$
- Identical actions have identical arguments:
  $\forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_n.\ A(x_1, \ldots, x_n) = A(y_1, \ldots, y_n) \rightarrow x_1 = y_1 \wedge \cdots x_n = y_n$

Comment: the consistency assumption that the sentence
$\neg\exists \vec{x} \exists a \exists s.(\gamma_F^-(\vec{x}, a, s) \wedge \gamma_F^+(\vec{x}, a, s))$
should be entailed by the underlying KB simply guarantees the integrity of the effect axioms (PNFEA) and (NNFEA). Under this consistency assumption, it will be impossible for both $F(\vec{x}, do(a, s))$ and $\neg F(\vec{x}, do(a, s))$ to be simultaneously derived. Notice that by the unique names axioms for actions, this condition is satisfied by the example about the fluent *broken*.

References

Serge Abiteboul, Richard Hull and Victor Vianu: "Foundations of Databases", Addison-Wesley, 1995. http://webdam.inria.fr/Alice/

H.Enderton: "A Mathematical Introduction to Logic", 2nd Ed., 2001
https://archive.org/download/MathematicalIntroductionToLogicEnderton/MathematicalIntroductionToLogic-Enderton.pdf

H.J. Levesque, F. Pirri, and R. Reiter: "Foundations for the situation calculus". Linköping Electronic Articles in Computer and Information Science, 1998, vol. 3, N 18. https://www.ida.liu.se/ext/epa/cis/1998/018/

Fangzhen Lin, Ray Reiter: How to Progress a Database. Artif. Intell. 92(1-2): 131-167 (1997) https://doi.org/10.1016/S0004-3702(96)00044-6

Sheila A. McIlraith: Integrating actions and state constraints: A closed-form solution to the ramification problem (sometimes). Artif. Intell. 116(1-2): 87-121 (2000)

Fiora Pirri, Raymond Reiter: Some Contributions to the Metatheory of the Situation Calculus. Journal of the ACM 46(3): 325-361 (1999)
https://dl.acm.org/doi/pdf/10.1145/316542.316545

Raymond Reiter: "Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems". The MIT Press 2001. Available at http://cognet.mit.edu/book/knowledge-action