A Theory of Complex Actions for Enterprise Modelling

Michael Grüninger
Department of Industrial Engineering
University of Toronto
Toronto, Canada M5S 1A4
gruninger@ie.utoronto.ca

Javier A. Pinto
Departamento de Ciencia de la Computacion
Pontificia Universidad Catolica de Chile
Casilla 306, Santiago 22, CHILE
javier@ing.puc.cl

Introduction

Enterprise modelling is an essential component in defining an enterprise, such as a manufacturing facility, an end-item distribution company, a financial institution, or a university department. The goal of our enterprise modelling research is to create generic representations of enterprise knowledge that can be reused across a variety of enterprises. Towards this end, we have been developing the TOVE (TOronto Virtual Enterprise) ontology [Fox and Grüninger 94]. TOVE provides a rich and precise representation of generic knowledge, such as activities, resources, time, and of more enterprise oriented knowledge such as cost, quality, products, and organization structure.

In this paper, we present enterprise modelling as a new area of application for theoretical work in reasoning about action. We give an overview of the problems encountered in enterprise modelling and the requirements that they impose on any theory of action. We outline our approach to these problems using an extension to the situation calculus.

Motivation for Complex Actions

In enterprise modelling, we want to define the actions performed within an enterprise, and define constraints for plans and schedules which are constructed to satisfy the goals of the enterprise. Enterprise modelling therefore requires several classes of complex actions. First of all, it is necessary to represent actions that have duration, in the sense that some fluent must hold over some time interval associated with a complex action. For example, we need to represent complex actions such as the use of a machine, or the consumption of a resource such as fuel. We will refer to this class of complex actions as duration-based actions, since the interval associated with the action is defined by the occurrence of an action that initiates the use or consumption of a resource, and the future occurrence of an action that terminates the use or consumption of the resource. Within the TOVE framework, such complex actions are referred to as activities.

Some complex actions are composed of duration-based actions. For example, an activity such as *fabricate_switch* may be composed of the subactions that *consume_wire*, *consume_plug*, and *use_inject_mold*. Within the TOVE framework, such complex actions are referred to as aggregate activities.

Enterprises are designed in a hierarchical fashion, so that we need to represent the notion of aggregate activities. For example, the activity of assembling a lamp may be composed of the activities that fabricate the components and then finally assemble the components to form the lamp.

Reasoning about activities in enterprise modelling requires the representation of temporal constraints. For example, in a steel manufacturing enterprise we may need to represent the constraint that if we do not fold the metal within 5 minutes of fabrication, we must reheat it. There may be activities that must be performed periodically, such as every 10 minutes, or activities that must be performed at nondeterministic intervals, such as transporting a load of steel ingots as soon as the truck is full.

Within enterprise modelling, we need to reason about plans and schedules. This can be naturally done if we consider plans and schedules to be complex actions. In this approach, plans and schedules can be constructed by combining complex actions, and then defining additional ordering and temporal constraints over the new set of subactions.We want to be able to compare alternative plans and schedules, and reason about hypothetical scenarios with respect to these alternatives. In particular, in scheduling we need to reason about the availability of resources to support multiple activities.

We also want to define properties of complex actions, such as duration or cost, and reason about these properties within the language. A major challenge facing many enterprises today is to be able to rapidly respond to customer demand, and in this way reduce the lead time from receiving a request to fulfilling the request. This requires a characterization of schedules with the shortest cycle time, which in turn correspond to complex actions with minimum duration.

Preliminaries

Our theory of complex actions is developed using an extension to the situation calculus. The axiomatization is

based on the discrete situation calculus [Reiter 91]. The situation calculus is a sorted second order language with equality. There are several domain sorts $\mathcal{A}, \mathcal{S}, \mathcal{F}, \mathcal{T}, \mathcal{D}$ for action types, situations, fluents, time, and arbitrary domain objects.

We use the notions of an actual line of situations, linear time and the notion of action occurrences as defined by [Pinto 94]. The axiomatization is as follows:

$$actual(S_0) \tag{1}$$

$$(\forall a, \sigma)\ actual(do(a, \sigma)) \supset actual(\sigma) \wedge Poss(a, \sigma) \tag{2}$$

$$(\forall a_1, a_2, \sigma)\ actual(do(a_1, \sigma)) \wedge actual(do(a_2, \sigma)) \supset a_1 = a_2 \tag{3}$$

$$end(\sigma, a) = start(do(a, \sigma)) \tag{4}$$

$$(\forall a, \sigma)\ start(\sigma) < end(\sigma, a) \tag{5}$$

$$start(S_0) = 0 \tag{6}$$

$$(\forall a, \sigma)\ occurs(a, \sigma) \equiv actual(do(a, \sigma)) \tag{7}$$

$$occurs_T(a, t) \equiv (\exists \sigma) occurs(a, \sigma) \wedge start(do(a, \sigma)) = t \tag{8}$$

$$holds_T(f, t) \equiv (\exists \sigma)\ actual(\sigma) \wedge start(\sigma) < t \wedge$$
$$(\forall a)\ [actual(do(a, \sigma)) \supset end(\sigma, a) \geq t] \wedge holds(f, \sigma) \tag{9}$$

We utilize the predicate $Poss(a, \sigma)$ that is true whenever the action a can be performed in a situation σ. This is used to inductively define an ordering over situations:

$$(\forall \sigma)\ \neg(\sigma < S_0)$$
$$(\forall a, \sigma, \sigma')\ \sigma < do(a, \sigma') \equiv (Poss(a, \sigma') \wedge \leq \sigma \leq \sigma')$$

In order to address the frame problem, we make use of Reiter's approach [Reiter 91]. The basic idea behind this solution is to derive successor state axioms for each fluent, which provide necessary and sufficient conditions for a fluent to be true in situation $do(a, \sigma)$ given the state in situation σ. The successor state axioms have the form

$$(\forall a, \sigma)\ Poss(a, \sigma) \supset [holds(R, do(a, \sigma)) \equiv$$
$$\gamma_R^+(a, \sigma) \vee holds(R, \sigma) \wedge \neg\gamma_R^-(a, \sigma)$$

where $\gamma_R^+(a, \sigma)$ and $\gamma_R^-(a, \sigma)$ are simple formulae which are used to provide conditions under which an action a produces an effect on a fluent R.

Complex Actions and Occurrence Theories

Within our approach to enterprise modelling, complex actions are represented as objects in the language. The predicate $Do(a, \sigma, \sigma')$ denotes that if action a is done in situation σ, then σ' is one of the possible situations reached.

We define the relation $subaction(a, a')$ to denote that action a is a subaction of action a'. We define primitive actions to be those with no subactions:

$$(\forall a)\ primitive(a) \equiv \neg(\exists a')\ subaction(a', a) \tag{10}$$

$$(\forall a, \sigma, \sigma')\ primitive(a) \supset$$
$$(Do(a, \sigma, \sigma') \equiv Poss(a, \sigma) \wedge \sigma' = do(a, \sigma)) \tag{11}$$

A complex action is defined by specifying its subactions and constraints over the occurrence of these subactions.

Definition 1 *A formula $\mathcal{O}_<(x_1, ..., x_n)$ is an ordering formula if it only mentions the terms $x_1, ..., x_n$, does not include any quantifiers, and all its literals are $<$ literals.*

Definition 2 *A complex action A is defined by*

$$(\forall \sigma, \sigma')\ Do(A, \sigma, \sigma') \equiv \Sigma_{Do}(A, \sigma, \sigma')$$

where $\Sigma_{Do}(A, \sigma, \sigma')$ is a formula whose only free variables are σ, σ' and which contains only subaction literals, Do literals for these subactions, and ordering formulae.

We will refer to $\Sigma_{Do}(A, \sigma, \sigma')$ as an occurrence theory.

In addition, we will need to reason about external actions and their role in complex actions. For example, an action may require the occurrence of an external event to establish the fluents to satisfy its precondition axioms. The occurrence of such actions is entailed by the occurrence theory $\Sigma_{Do}(A, \sigma, \sigma')$.

Our approach is to reason about complex actions by reasoning about the occurrence theories that define the complex actions. In particular, various classes of complex actions, such as those defined in [Lesperance et al. 94] (conditionals, nondeterministic choice, iteration), as well as concurrent actions can be represented using occurrence theories.

Within enterprise modelling, we are especially concerned with the class of duration-based complex actions. Such an action is composed of two primitive subactions, such that the second subaction occurs some time after the first subaction. Furthermore, the first subaction initiates a fluent that is the precondition for the second subaction. Intuitively, this fluent represents the interval over which the complex action is executing. For example, if the complex action is *assemble_lamp*, then there is a fluent associated with the interval, say *assembling_lamp*, which is initiated by the first subaction, and terminated by the second subaction of *assemble_lamp*. The duration of the complex action would be the amount of time between the occurrence of the two subactions. In addition, there may be state constraints relating the interval fluent with fluents for resources required by the action.

Definition 3 *A duration-based complex action A is represented by the following sentences:*

$$(\forall \sigma, \sigma')\ Do(A, \sigma, \sigma') \equiv$$
$$(\exists \sigma_1, \sigma_2)\ subaction(A_1, A) \wedge subaction(A_2, A)$$
$$\wedge primitive(A_1) \wedge primitive(A_2)$$
$$\wedge \sigma_1 = do(A_1, \sigma) \wedge \sigma' = do(A_2, \sigma_2) \wedge \sigma_1 \leq \sigma_2$$

where A_1, A_2 are ground action terms that satisfy the following successor state and precondition axioms:

$$holds(F, do(a, \sigma)) \equiv a = A_1 \vee holds(F, \sigma) \wedge a \neq A_2$$

$$Poss(A_2, \sigma) \equiv holds(F, \sigma)$$

We will refer to F as an interval fluent.

In order to evaluate and justify our approach to complex actions, we use the problem of temporal projection:

Problem 1 *Given a ground formula* $\Sigma_{Do}(A, S_0, S_1)$, *determine*

$$\Sigma_{Do}(A, S_0, S_1) \models Q(S_1)$$

for some ground simple formula $Q(\sigma)$.

We show that our theory of complex action is sufficient to characterize the solutions to this problem.

The results of [Reiter 91] prove that the theory of primitive actions proposed there is justified with respect to temporal projection. This paper generalizes these results from primitive actions to complex actions. We show that the definition of the effects of complex actions is correct given the definition of successor state axioms T_{succ} and precondition axioms T_{pre} for primitive actions in [Reiter 91]. We state conditions under which a complex action is complete with respect to temporal projection.

Related to the problem of temporal projection is the qualification problem. The intuition formalized in [Lin and Reiter 94] is that the specification of preconditions for action prevents the violation of qualification constraints by the effects of the action. We show that this intuition is preserved by our definition of complex actions.

Effects of Complex Actions

In order to justify our theory of complex action with respect to temporal projection, we are first faced with the problem of defining the effects of a complex action.

In general, $Do(a, \sigma, \sigma')$ defines a set of branches in the tree of situations. These branches arise from different total orderings consistent with the partial ordering of subactions within an action, or from different alternative subactions in a nondeterministic complex action. This branch structure of a complex action must be made explicit; intuitively, each branch must correspond to a legal action sequence.

In specifying the effects of a complex action defined by $\Sigma_{Do}(A, \sigma, \sigma')$, we first show that these effects are reducible to the effects of the primitive actions in $\Sigma_{Do}(A, \sigma, \sigma')$.

Theorem 1 *Let* T_{succ} *be the set of successor state axioms for the primitive actions and* T_{pre} *be the set of precondition axioms for the primitive actions. Given an occurrence theory* $\Sigma_{Do}(A, \sigma, \sigma')$ *for a complex action* A, *we have*

$$\Sigma_{Do}(A, \sigma, \sigma') \cup T_{succ} \cup T_{pre} \models$$
$$(\forall a, \sigma, \sigma', \sigma^*)\ Do(A, \sigma, \sigma') \wedge \sigma < \sigma^* < \sigma' \supset [holds(R, \sigma^*) \equiv$$
$$(\exists a', \sigma'')\ primitive(a') \wedge \gamma_R^+(a', \sigma'')$$
$$\wedge(\sigma \leq do(a', \sigma'') \leq \sigma^*)$$
$$\wedge\neg(\exists a'', \sigma''')\ primitive(a'') \wedge \gamma_R^-(a'', \sigma''')$$
$$\wedge(do(a', \sigma'') \leq do(a'', \sigma''') \leq \sigma^*)$$
$$\vee holds(R, \sigma) \wedge \neg(\exists a', \sigma'')\ primitive(a') \wedge \gamma_R^-(a', \sigma'')$$
$$\wedge(\sigma \leq do(a', \sigma'') \leq \sigma^*)$$

Thus, a fluent R holds in a situation iff either there is an earlier situation in which a primitive action establishes the fluent and no action occurs to falsify the fluent, or the fluent initially held, and no action occurred to falsify it.

This definition of the effects of Do depends on the occurrence of primitive actions. These actions may not be subactions of A; the occurrence of some of these actions is entailed by the subactions of A. However, other actions that are not required by A may also occur; in what sense would the effects of these actions be related to the effects of A? It is therefore necessary to define the effects of a complex action using a minimization policy with respect to the occurrence of actions on a branch.

Another motivation for the minimization policy arises from issues in enterprise modelling. Recall that we wish to combine actions together to form new complex actions. We may have one action A that consists of subactions A_1 followed by A_2 and another action A' that consists of subactions A_3 followed by A_4. If we want to combine these actions together with the ordering A_1, A_3, A_2, A_4, then we cannot have A_2 occurring in the successor situation for A_1, since A_3 must be able to occur in between them. We thus want to represent complex actions by orderings over the occurrence of their subactions, but define their effects with respect to the assumption that no unnecessary actions occur on the same branch.

To characterize the effects of a complex action along a branch, we define a set of ordering formulae $\mathcal{O}(a, \sigma, \sigma')$ over the subactions of a, and show that by using the minimization policy of [Pinto 94], minimal models of $\Sigma_{Do}(A, \sigma, \sigma') \cup \mathcal{O}(A, \sigma, \sigma')$ correspond to legal action sequences.

Definition 4 *Let* $\mathcal{O}(A, \sigma, \sigma')$ *be a set of ordering formulae over a set of subactions of* A *as defined by* $\Sigma_{Do}(A, \sigma, \sigma')$.

1. *The orderings in* $\mathcal{O}(A, \sigma, \sigma')$ *are consistent with the definition of the complex action.*

$$\Sigma_{Do}(A, \sigma, \sigma') \not\models \neg\mathcal{O}(A, \sigma, \sigma')$$

2. $\mathcal{O}(A, \sigma, \sigma')$ *defines a total ordering.*

$$\mathcal{O}(A, \sigma, \sigma') \models subaction(A_i, A) \wedge subaction(A_j, A)$$
$$\supset (\forall \sigma'', \sigma''', \sigma^*, \sigma^{**})(Do(A_i, \sigma'', \sigma''') \wedge Do(A_j, \sigma^*, \sigma^{**})$$
$$\supset (\sigma''' < \sigma^{**} \vee \sigma''' = \sigma^{**} \vee \sigma^{**} < \sigma'''))$$

3. *The orderings in* $\mathcal{O}(A, \sigma, \sigma')$ *define a maximal ordering:*

$$\Sigma_{Do}(A, \sigma, \sigma') \cup \mathcal{O}(A, \sigma, \sigma') \models (\exists \sigma^*)\ \sigma < \sigma^* < \sigma'$$

iff

$$\mathcal{O}(A, \sigma, \sigma') \models (\exists \sigma^*)\ \sigma < \sigma^* < \sigma'$$

Given this set of ordering formulae we can show that models of $Circ(\Sigma_{Do}(A, S_0, S_n) \cup \mathcal{O}(A, S_0, S_n); Do; start)$ define legal action sequences.

Theorem 2 *Suppose* $\Sigma_{Do}(A, S_1, S_n) \cup \mathcal{O}(A, S_0, S_n)$ *entails a sentence of the form*

$$subaction(A_1, A) \wedge ... \wedge subaction(A_n, A) \wedge S_0 < do(A_1, S_1)$$
$$\wedge ... \wedge S_0 < do(A_n, S_n) \wedge S_1 < ... < S_n$$

Then every model of $Circ(\Sigma_{Do}(A, S_0, S_n) \cup \mathcal{O}(A, S_0, S_n); Do; start)$ *satisfies*

$$S_n = do(A_n, S_{n-1}) \wedge ... \wedge S_2 = do(A_1, S_1) \wedge S_0 = S_1$$

In [Reiter 91], temporal projection is equivalent to computing the effects of a legal action sequence using regression. By the preceding theorem, we can use regression to compute the effects of complex actions using the legal action sequences defined in the models of $\Sigma_{Do}(A, S_1, S_n) \cup \mathcal{O}(A, \sigma, \sigma')$.

Furthermore, our definition of effects of a complex action on each branch is complete with respect to temporal projection. That is, all models of $Circ(\Sigma_{Do}(A, S_0, S_n) \cup \mathcal{O}(A, \sigma, \sigma'); Do; start)$ agree on the extension of the predicate *holds*.

Theorem 3 *For any two models* $\mathcal{M}, \mathcal{M}'$ *of*

$$Circ(\Sigma_{Do}(A, \sigma, \sigma') \cup \mathcal{O}(A, \sigma, \sigma'); Do; start)$$

and any variable assignment ν, *suppose* $\mathcal{M}, \nu \models Do(A, \sigma, \sigma')$ *and* $\mathcal{M}', \nu \models Do(A, \sigma, \sigma')$. *Then*

$$\mathcal{M}, \nu \models holds(f, \sigma) \; iff \; \mathcal{M}', \nu \models holds(f, \sigma)$$

iff

$$\mathcal{M}, \nu \models holds(f, \sigma') \; iff \; \mathcal{M}', \nu \models holds(f, \sigma')$$

Complex Action Precondition Axioms

The theorems in the preceding section defined the relationship between complex actions and legal action sequences. If a sequence of actions corresponding to a complex action is not legal, that is, if it violates some state constraint, then the complex action must be defined in such a way that we prevent this sequence from occurring. Recall that all effects of a complex action are effects of the primitive subactions. Therefore, if a complex action A violates a state constraint, there exists a primitive subaction whose effects violate the state constraint.

In order to characterize the notion of precondition axioms for complex actions, we need to show how the definition of a complex action A prevents the effects of A from violating state constraints. In this way we extend the characterization of precondition axioms for primitive actions in [Reiter and Lin 94].

Theorem 4 *Suppose* $\Sigma_{Do}(A, \sigma, \sigma')$ *is consistent. Let* T_{qual} *be a set of qualification constraints. For every qualification constraint* $(\forall \sigma)\, QC(\sigma) \in T_{qual}$

$$T_{succ} \cup T_{pre} \cup \Sigma_{Do}(A, \sigma, \sigma') \models$$
$$Do(A, \sigma, \sigma') \supset ((\forall \sigma'')\, \sigma \leq \sigma'' \leq \sigma' \wedge Q(\sigma) \supset Q(\sigma''))$$

It is very important not to confuse violated state constraints with actions that prevent the action from achieving its intended effects. If it is possible for the constraint to be violated, then it is not a state constraint; this will be discussed in the section on actual line constraints.

Recall that we reason about complex actions by reasoning about the occurrence theories that define them. In this approach we want to characterize the set of legal action sequences defined by the complex action A in terms of occurrence constraints over the subactions of A. What we want is a set of occurrence sentences Φ such that $T_{succ} \cup T_{pre} \cup \Sigma_{Do}(A, \sigma, \sigma)$ is consistent iff

$$\Sigma_{Do}(A, \sigma, \sigma) \models \Phi$$

Consider the special class of complex actions composed of duration-based complex actions. The preconditions for the subactions may require that a fluent hold over the interval corresponding to the complex action. For example, we need to be able to specify that a machine must be functional over the entire interval that it is being used, or that some quantity of a resource that is being consumed must be available over the entire interval associated with the action. How are these constraints on the set of legal action sequences defined by duration-based complex actions?

First of all, it is important to realize that a legal action sequence corresponding to a duration-based complex action may contain actions that falsify the interval fluent. In such cases, there exist actions that occur later in the sequence to reestablish the fluent so that the final subaction is $Poss$. For example, a car needs gas in the tank for the duration of the trip; if the tank becomes empty, an action must occur to refill the tank before the car can reach its destination. Thus, if

$$T_{succ} \models holds(F_1, do(a, \sigma)) \equiv a = A_1 \vee holds(F_1, \sigma) \wedge a \neq A_2$$

then any duration-based complex action composed of A_1 and A_2 must satisfy

$$(do(A_1, \sigma') \leq \sigma \leq do(A_2, \sigma'') \wedge \neg holds(F, \sigma)$$
$$\wedge ((\forall \sigma^*)\, Poss(A_2, \sigma^*) \equiv holds(F, \sigma^*)) \supset$$
$$(\exists a, \sigma_1)\, \sigma < do(a, \sigma_1) \leq \sigma'' \wedge holds(F, do(a, \sigma_1))$$

There may be interval fluents that cannot be achieved once they have been falsified. In this case, any action that falsifies the interval fluent cannot occur during the complex action:

$$\neg holds(F, do(a, \sigma)) \wedge \neg(\exists \sigma')\, do(a, \sigma) \leq \sigma' \wedge Poss(A_1, \sigma')$$
$$\supset \neg(\exists \sigma^*, \sigma_1, \sigma_2)\, do(A_1, \sigma_1) \leq do(a, \sigma^* \leq do(A_2, \sigma_2)$$

Finally, there may be qualification constraints of the form

$$(\forall \sigma, \sigma_1, \sigma_2)\, \neg(holds(F_1, \sigma) \wedge holds(F_2, \sigma))$$

for two interval fluents F_1, F_2. This arises in problems concerning concurrent complex actions. If an action requires exclusive use of a machine over the interval, then two actions cannot concurrently use the machine,

and hence the intervals cannot be overlapping. If A_2 is the action that terminates F_1 and A_3 is the action that initiates F_2, then the complex actions must satisfy the sentence

$$(\forall \sigma, \sigma_1, \sigma_2)\ \neg(holds(F_1, \sigma) \wedge holds(F_2, \sigma))$$
$$\supset do(A_2, \sigma_1) < do(A_3, \sigma_2)$$

We are currently working on a generalization of these intuitions for a complete characterization of the legal action sequences corresponding to duration-based complex actions.

For complex actions which are composed of duration-based actions, the characterization of legal action sequences depends on the ordering of the subactions. In general, an activity is executing if the fluents associated with the appropriate duration-based subactions hold. The problem lies in determining this set of subactions, since the definitions depend on the particular ordering and occurrence of actions in the activity. For example, an activity may require the use of an inject mold machine between times T_1 and T_2, and the use of a lathe between times T_3 and T_4, where $T_1 < T_3 < T_2 < T_4$. If the lathe is broken at some time between T_1 and T_2, it does not affect the possibility of the activity, since the use of the lathe is not yet required. Similarly, if the inject mold machine breaks down between T_3 and T_4, the activity should still be able to be executing, since the use of the inject mold machine is no longer required. The characterization of legal action sequences for activities is also a focus for future work.

Actual Line Constraints

In addition to defining complex actions in terms of preconditions and successor state axioms, it is also necessary to introduce a new class of constraints on action occurrences.

For example, an important problem for end-item distribution enterprises is reasoning about the shelf life of perishable products. Consider the following sentences, which state that a resource, such as milk, will spoil if any quantity of the resource still exists at the expiry date:

$$(\forall r, q, t)\ expire\_date(r, t) \wedge holds_T(quantity(r, q), t) \wedge q > 0$$
$$\supset occurs_T(spoilage(r), t)$$
$$(\forall r, a, \sigma)\ holds(spoiled(r), do(a, \sigma)) \equiv$$
$$a = spoilage(r) \vee holds(spoiled(r), \sigma)$$

and the constraint

$$(\forall r, \sigma)\ \neg holds(spoiled(r), \sigma)$$

This cannot be a qualification constraint, since there can exist situations where spoilage occurs; however, we do not want spoilage to occur. This is similar to the notion of deontic constraints. Our approach is that these must be constraints on actual situations – branches that violate these constraints cannot be actual. Thus, the above spoilage constraint should be written as:

$$(\forall \sigma)\ actual(\sigma) \supset \neg holds(spoiled(r), \sigma)$$

This allows spoilage to occur on nonactual branches, but not in any actual situation.

Another example illustrates the distinction between intended effects of an action and possible effects of an action. Suppose that we have the successor state axioms:

$$holds(edible(x), do(a, \sigma)) \equiv$$
$$a = bake(x) \wedge (\exists y)\ holds(temp(Oven, y), \sigma) \wedge y = 300$$
$$\vee holds(edible(x), \sigma)$$

It is possible to bake a cake in the oven at a temperature different than 300, but in such a case, the cake would not be edible. The intended effects of the *bake* action is that the cake is edible. We may enforce this with the actual line constraint

$$(\forall x, \sigma)\ actual(do(bake(x), \sigma))$$
$$\supset holds(edible(x), do(bake(x), \sigma))$$

from which we want to infer

$$(\forall x, y, \sigma)\ actual(do(bake(x), \sigma))$$
$$\wedge holds(temp(Oven, y), \sigma) \supset y = 300$$

Actual line constraints arise in many applications of enterprise modelling including integrated supply chain management (purchases as forward expectation occurrence axioms), inventory management (preventing spoilage of perishable products), workflow management (the distinction between conditional actions and events triggered by state), the definition of the quality of products in manufacturing, and the representation of design requirements for products.

Analogous to the compilation of qualification constraints into precondition axioms, we can define the compilation of actual line constraints axioms defining the preconditions for actions that occur on the actual line. These are sentences that must be satisfied in a situation in order to guarantee that all actual line constraints will be satisfied in the successor situation.

We therefore need to generate a set of actual line precondition axioms of the form

$$actual(do(A, \sigma)) \equiv \Phi_A$$

where Φ is a simple state formula.

In this paper, we present the special case for primitive actions only, such that Σ_{occ} is empty and we are dealing only with actual line constraints that are simple formulae.

Let T_{actual} be a set of actual line constraints of the form

$$actual(\sigma) \supset AC(\sigma)$$

where AC is a simple state formula with a unique free variable σ. Using the regression operator $\mathcal{R}$ defined in [Reiter 91], we can obtain a set $T_{allowed}$ of actual line precondition axioms of the form

$$actual(do(A(x_1, ..., x_n), \sigma)) \equiv \bigwedge \Pi_{AC}$$

These axioms are characterized by the following results:

Theorem 5 *Let $T_{closure}$ be the following domain closure axiom for actions:*

$$(\forall a)((\exists \overline{x})\ a = A_1(\overline{(}x)) \vee ... \vee (\exists \overline{y})\ a = A_1(\overline{(}y))$$

For every actual line constraint $(\forall \sigma) actual(\sigma) \supset AC(\sigma)$ in T_{actual},

$$T_{una} \cup T_{succ} \cup T_{allowed} \cup T_{closure} \models$$
$$AC(S_0) \supset (\forall \sigma)\, actual(\sigma) \supset AC(\sigma)$$

Corollary 1

$$T_{una} \cup T_{succ} \cup T_{allowed} \cup T_{closure} \cup T^{S_0}_{allowed} \models T_{actual}$$

where $T^{S_0}_{allowed}$ is the set of actual line constraints restricted to the initial state:

$$T^{S_0}_{allowed} = \{AC(S_0) | (\forall \sigma)\, actual(\sigma) \supset AC(\sigma) \in T_{actual}\}$$

This corollary is analogous to that for qualification constraints in [Lin and Reiter 94]– provided that the initial state S_0 satisfies all actual line constraints, then these constraints are satisfied by the theory

$$T_{una} \cup T_{succ} \cup T_{allowed} \cup T_{closure} \cup T^{S_0}_{allowed}$$

so that the actual line constraints have been "compiled" into the actual line precondition axioms.

We can then define the notion of allowable action sequences, analogous to legal action sequences, along which all actions can occur on the actual line.

Definition 5 *Suppose $A_1, ..., A_n$ is a sequence of ground action terms. This sequence is allowable with respect to T iff*

$$T \models actual(do([A_1, ..., A_n], S_0)$$

We briefly discuss some of the issues involved in extending the above results. First, we need to consider actual line constraints that are not simple formulae. Goals and deadlines are represented by existential actual line constraints:

$$(\exists \sigma)\, actual(\sigma) \wedge holds(F, \sigma) \wedge start(\sigma) < T_1$$

This allows us to reason about hypothetical scenarios with nonactual branches where the goal is not achieved by the deadline. However, in these cases, every action in a sequence may satisfy the actual line constraints, yet the entire sequence violates the constraint since the goal or deadline is not satisfied.

We must also incorporate occurrence theories when compiling actual line precondition axioms. For example, suppose we have the sentences

$$(\forall t)\, occurs_T(A_1, t) \supset occurs_T(A_2, t + 5)$$
$$(\forall a, \sigma)\, holds(F_1, do(a, \sigma)) \equiv a = A_2 \vee holds(F_1, \sigma)$$
$$(\forall \sigma)\, actual(\sigma) \supset \neg holds(F_1, \sigma)$$

We intuitively want to derive the actual line preconditions:

$$(\forall \sigma)\ \neg actual(do(A_1, \sigma)$$
$$(\forall \sigma)\ \neg actual(do(A_2, \sigma)$$

Although the second precondition axiom can be derived using the compilation defined above, the first precondition axiom can only be derived using the occurrence sentence. We thus require a procedure for compiling the occurrence sentences into the actual line precondition axioms.

It is important to realize that although we are reasoning about complex actions using occurrence theories, the definition of a complex action is distinct from the occurrence sentences that define actual line constraints. For example, conditional actions are distinct from actual line constraints defining the triggering of an action by fluents in a state. In the former case we have

$$(\forall \sigma, \sigma')\, Do(A, \sigma, \sigma') \equiv (holds(F_1, \sigma) \supset Do(A_1, \sigma, \sigma'))$$
$$\wedge (holds(F_2, \sigma) \supset Do(A_2, \sigma, \sigma'))$$

while in the latter case we have

$$(\forall \sigma)\, holds(F_1, \sigma) \supset occurs(A_1, \sigma)$$
$$(\forall \sigma)\, holds(F_2, \sigma) \supset occurs(A_2, \sigma)$$

This constraint can be violated on non-actual branches. In addition, on the actual line, if the fluent F_1 holds in any situation, then action A_1 will occur; intuitively, the action A_1 is triggered by the fluent. In the case of the conditional action, however, A_1 will occur if only if we are performing the action A and F_1 holds in that situation.

References

[1] Fox, M.S. and Grüninger, M. (1994). Ontologies for enterprise integration, *Proceedings of the Second International Conference on Cooperative Information Systems*, pages 82-89.

[2] Lesperance, Y., Levesque, L., Lin, F., Marcu, D., Reiter, R., and Scherl, R. A Logical Approach to High-Level Robot Programming – A Progress Report. In Benjamin Kuipers, editor, Control of the Physical World by Intelligent Systems, Papers from the 1994 AAAI Fall Symposium, pages 79-85, New Orleans, LA, November, 1994.

[3] Lin, F. and Reiter, R. (1994) State constraints revisited, to appear in the *Journal of Logic and Computation*.

[4] Pinto, J. (1994). *Temporal Reasoning in the Situation Calculus*, Technical Report KRR-TR-94-1, Department of Computer Science, University of Toronto.

[5] Reiter, R. (1991). The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418-440. Academic Press, San Diego.