Formalising the UML in Structured Temporal Theories

K. Lano, J. Bicarregui
Dept. of Computing, Imperial College
kcl@doc.ic.ac.uk

Abstract

In this paper we describe a possible semantics for a large part of the Unified Modelling Notation (UML), using structured theories in a simple temporal logic. This semantic representation is suitable for modular reasoning about UML models. We show how it can be used to clarify certain ambiguous cases of UML semantics, and how to justify enhancement or refinement transformations on UML models.

1 Introduction

The semantic model of UML used here is based on the set-theoretic Z-based model of Syntropy [3]. A mathematical semantic representation of UML models can be given in terms of *theories* in a suitable logic, as in the semantics presented for Syntropy in [1] and VDM^{++} in [6]. In order to reason about real-time specifications the more general version, Real-time Action Logic (RAL) [6] will be used.

The semantics developed here should complement and be additional to the UML metamodel and OCL constraints on this defined in [8].

A RAL theory has the form:

theory *Name*

types *local type symbols*

attributes *time-varying data, representing instance or class variables*

actions *actions which may affect the data, such as operations, statechart transitions and methods*

axioms *logical properties and constraints between the theory elements.*

Theories can be used to represent classes, instances, associations and general submodels of a UML model. These models are therefore taken as *specifications*: they describe the features and properties which should be supported by any implementation that satisfies the model. In terms of the semantics, theory **S** satisfies theory **T** if there is an interpretation σ of the symbols of **T** into those of **S** under which every property of **T** holds:

$$\mathbf{S} \vdash \sigma(\varphi)$$

for every theorem φ of $\mathbf{T}$.

In addition to standard mathematical notation such as $\mathbb{F}$ for "set of finite sets of", etc, RAL theories can use the following notations:

1. For each classifier or state $\mathbf{X}$ there is an attribute $\overline{\mathbf{X}} : \mathbb{F}(\mathbf{X})$ denoting the set of existing instances of $\mathbf{X}$[1].

2. If α is an action symbol, and $\mathbf{P}$ a predicate, then $[\alpha]\mathbf{P}$ is a predicate which means "every execution of α establishes $\mathbf{P}$ on termination", that is, $\mathbf{P}$ is a *postcondition* of α.

3. For every action α there are functions $\uparrow(\alpha, \mathbf{i})$, $\downarrow(\alpha, \mathbf{i})$, $\leftarrow(\alpha, \mathbf{i})$ and $\rightarrow(\alpha, \mathbf{i})$ of $\mathbf{i} : \mathbb{N}_1$ which denote the activation, termination, request send and request arrival times, respectively, of the $\mathbf{i}$-th invocation of α. These times are ordered as:

$$\leftarrow(\alpha, \mathbf{i}) \ \leq \ \rightarrow(\alpha, \mathbf{i}) \ \leq \ \uparrow(\alpha, \mathbf{i}) \ \leq \ \downarrow(\alpha, \mathbf{i})$$

Also

$$\mathbf{i} \leq \mathbf{j} \ \Rightarrow \ \leftarrow(\alpha, \mathbf{i}) \ \leq \ \leftarrow(\alpha, \mathbf{j})$$

Using these we can define concepts such as "every execution of α coincides with an execution of β" (α calls β):

$$\begin{array}{l} \alpha \supset \beta \ \equiv \\ \quad \forall \mathbf{i} : \mathbb{N}_1 \cdot \exists \mathbf{j} : \mathbb{N}_1 \cdot \\ \quad\quad \uparrow(\alpha, \mathbf{i}) = \uparrow(\beta, \mathbf{j}) \ \wedge \ \downarrow(\alpha, \mathbf{i}) = \downarrow(\beta, \mathbf{j}) \end{array}$$

This corresponds to β being a generalisation of α on a class diagram of *signals* in UML.

We can also define the times that a given condition $\mathbf{G}$ becomes true or false for the $\mathbf{i}$-th time: $\clubsuit(\mathbf{G} := \mathbf{true}, \mathbf{i})$ and $\clubsuit(\mathbf{G} := \mathbf{false}, \mathbf{i})$, for $\mathbf{i} : \mathbb{N}_1$. Temporal operators $\diamond$ (sometime in the future), $\Box$ (always in the future) and $\bigcirc$ (next) are also included.

Although for the sake of conciseness we will use Z-style notation for set comprehension, set unions, etc [9], the OCL [8] notation could be used instead. A systematic translation of OCL notation into Z is given in [5].

Temporal logic makes representation and reasoning about dynamic models (state machines, interaction diagrams, etc) more concise than using a formalism such as pure Z. However it would be possible to work just in Z, by using sequences of states to represent the allowed behaviours of objects over time.

We focus on some areas where formalisation helps to clarify the meaning and consequences of certain UML constructs: aggregation, qualification, statecharts and collaboration diagrams.

2 Object Models

A UML class $\mathbf{C}$ is represented as a theory of the form given in Figure 1. Each instance

[1] Alternative notation for $\overline{\mathbf{X}}$ is $\mathbf{ext}(\mathbf{X})$, the *extension* of $\mathbf{X}$ [10].

theory $\Gamma_{\mathbf{C}}$

types $\mathbf{C}$

attributes

$\overline{\mathbf{C}} : \mathbb{F}(\mathbf{C})$
$\mathbf{self} : \mathbf{C} \rightarrow \mathbf{C}$
$\mathbf{att}_1 : \mathbf{C} \rightarrow \mathbf{T}_1$
$\ldots$

actions

$\mathbf{create}_{\mathbf{C}}(\mathbf{c} : \mathbf{C}) \quad \{\overline{\mathbf{C}}\}$
$\mathbf{kill}_{\mathbf{C}}(\mathbf{c} : \mathbf{C}) \quad \{\overline{\mathbf{C}}\}$
$\mathbf{op}_1(\mathbf{c} : \mathbf{C}, \mathbf{x} : \mathbf{X}_1) : \mathbf{Y}_1$
$\ldots$

axioms

$$\forall \mathbf{c} : \mathbf{C} \cdot$$
$$\mathbf{self}(\mathbf{c}) = \mathbf{c} \ \wedge$$
$$[\mathbf{create}_{\mathbf{C}}(\mathbf{c})](\mathbf{c} \in \overline{\mathbf{C}}) \ \wedge$$
$$[\mathbf{kill}_{\mathbf{C}}(\mathbf{c})](\mathbf{c} \notin \overline{\mathbf{C}})$$

Figure 1: Theory of Class $\mathbf{C}$

attribute $\mathbf{att_i} : \mathbf{T_i}$ of $\mathbf{C}$ gains an additional parameter of type $\mathbf{C}$ in the class theory $\Gamma_{\mathbf{C}}$ and similarly for operations[2]. Class attributes and actions do not gain the additional $\mathbf{C}$ parameter as they are independent of any particular instance. We can denote $\mathbf{att}(\mathbf{a})$ for attribute $\mathbf{att}$ of instance $\mathbf{a}$ by the standard OO notation $\mathbf{a}.\mathbf{att}$, and similarly denote actions $\mathbf{act}(\mathbf{a}, \mathbf{x})$ by $\mathbf{a}!\mathbf{act}(\mathbf{x})$.

Similarly each association $\mathbf{lr}$ can be interpreted in a theory which contains an attribute $\overline{\mathbf{lr}}$ representing the current extent of the association (the set of pairs in it) and actions **add_link** and **delete_link** to add and remove pairs (links) from this set. Axioms define the cardinality of the association ends and other properties of the association. In particular, if $\mathbf{ab}$ is an association between classes $\mathbf{A}$ and $\mathbf{B}$, then $\overline{\mathbf{ab}} \subseteq \overline{\mathbf{A}} \times \overline{\mathbf{B}}$, so membership of $\overline{\mathbf{ab}}$ implies existence for elements of a link.

Normally, subclasses $\mathbf{S}$ of a class $\mathbf{C}$ are assumed to be static (ie, once an object is created as a member of $\mathbf{S}$ it remains in $\mathbf{S}$ for the rest of its life in $\mathbf{C}$):

$$\forall \mathbf{a} : \mathbf{C} \cdot \mathbf{a} \in \overline{\mathbf{S}} \ \Rightarrow \ \Box(\mathbf{a} \in \overline{\mathbf{C}} \Rightarrow \mathbf{a} \in \overline{\mathbf{S}})$$

This is not assumed for states $\mathbf{S}$ in a statechart, or for specifically marked 'dynamic' subclasses. $\Box\mathbf{P}$ means that $\mathbf{P}$ is true at the present time and always in the future.

[2] The class theory can be generated from a theory of a typical $\mathbf{C}$ instance by means of an $\mathbf{A}$-morphism [1].

2.1 Aggregation

There are two main forms of aggregation in UML: simple aggregation, represented by an open diamond at the 'whole' end of the aggregation between classes, and composition aggregation, represented by a filled diamond at the 'whole' end, or by physical containment of the part classes or model elements within the whole, as in Fusion [2].

Simple aggregation is limited only by the constraint that there cannot be aggregation symbols at both ends of an association. It may also be reasonable to assume that a simple aggregation cannot be reflexive at the instance level:

$$\forall \mathbf{a} : \mathbf{A} \cdot \neg ((\mathbf{a}, \mathbf{a}) \in \overline{\mathbf{r}})$$

where $\mathbf{r}$ is an aggregation from $\mathbf{A}$ to $\mathbf{A}$.

In [8] a semantics is suggested for composition aggregation, which we will formalise as follows, using ideas from [3]. If there is a composition aggregation $\mathbf{ab}$ from a whole class $\mathbf{A}$ to a part class $\mathbf{B}$, then:

1. parts $\mathbf{b}$ cannot be deleted or removed from a whole object $\mathbf{a}$, whilst $\mathbf{a}$ continues to exist:

   $$(\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{ab}} \;\Rightarrow\; \Box(\mathbf{a} \in \overline{\mathbf{A}} \;\Rightarrow\; (\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{ab}})$$

2. parts $\mathbf{b}$ cannot be moved from one whole to another:

   $$(\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{ab}} \;\wedge\; \Diamond((\mathbf{a}', \mathbf{b}) \in \overline{\mathbf{ab}}) \;\Rightarrow\; \mathbf{a}' = \mathbf{a}$$

   $\Diamond\mathbf{P}$ means that $\mathbf{P}$ is true at the present time or at some time in the future.

Together, these properties mean that a whole may gain new parts but cannot lose or exchange existing parts.

Transitivity of composition Two composition aggregations can be put together to produce a third one (Figure 2) because

$$(\mathbf{a}, \mathbf{c}) \in \overline{\mathbf{ab}};\ \overline{\mathbf{bc}} \quad\Rightarrow\quad \exists \mathbf{b} : \overline{\mathbf{B}} \cdot (\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{ab}} \wedge (\mathbf{b}, \mathbf{c}) \in \overline{\mathbf{bc}}$$

$((1) \Rightarrow (2))$ and

$$\Diamond((\mathbf{a}', \mathbf{c}) \in \overline{\mathbf{ab}};\ \overline{\mathbf{bc}}) \quad\Rightarrow\quad \Diamond(\exists \mathbf{b}' : \overline{\mathbf{B}} \cdot (\mathbf{a}', \mathbf{b}') \in \overline{\mathbf{ab}} \wedge (\mathbf{b}', \mathbf{c}) \in \overline{\mathbf{bc}})$$

$((3) \Rightarrow (4))$. But then $(1) \wedge (3)$ implies $(2) \wedge (4)$, so by 2 applied to $\mathbf{b}$, $\mathbf{c}$ we have $\mathbf{b}' = \mathbf{b}$. Therefore, applying 2 to $\mathbf{a}$, $\mathbf{b}$ we have $\mathbf{a}' = \mathbf{a}$ as required.

Also if $(\mathbf{a}, \mathbf{c}) \in \overline{\mathbf{ab}};\ \overline{\mathbf{bc}}$ then $\exists \mathbf{b} : \overline{\mathbf{B}} \cdot (\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{ab}} \wedge (\mathbf{b}, \mathbf{c}) \in \overline{\mathbf{bc}}$ and hence $\exists \mathbf{b} : \overline{\mathbf{B}} \cdot \Box(\mathbf{a} \in \overline{\mathbf{A}} \Rightarrow (\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{ab}}) \wedge \Box(\mathbf{b} \in \overline{\mathbf{B}} \Rightarrow (\mathbf{b}, \mathbf{c}) \in \overline{\mathbf{bc}})$. But then $\exists \mathbf{b} : \overline{\mathbf{B}} \cdot \Box(\mathbf{a} \in \overline{\mathbf{A}} \Rightarrow (\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{ab}} \wedge (\mathbf{b}, \mathbf{c}) \in \overline{\mathbf{bc}})$ as required for 2.

Moving Associations into Aggregates Another valid transformation on composition aggregations is to move an association between two part classes into the aggregation (Figure 3). This is a valid transformation because the new model has the extra axiom

$$\forall (\mathbf{b}, \mathbf{c}) \in \overline{\mathbf{r}} \cdot \exists \mathbf{a} : \overline{\mathbf{A}} \cdot (\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{r}_1} \wedge (\mathbf{a}, \mathbf{c}) \in \overline{\mathbf{r}_2}$$

In other words, $\mathbf{r}$ can only relate parts of the same aggregate.

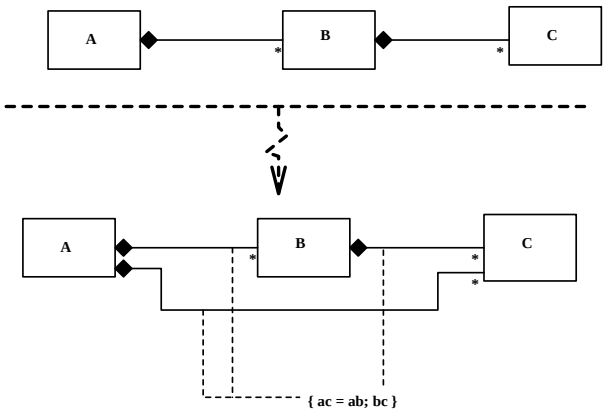

Figure 2: Transitivity of Composition Aggregations

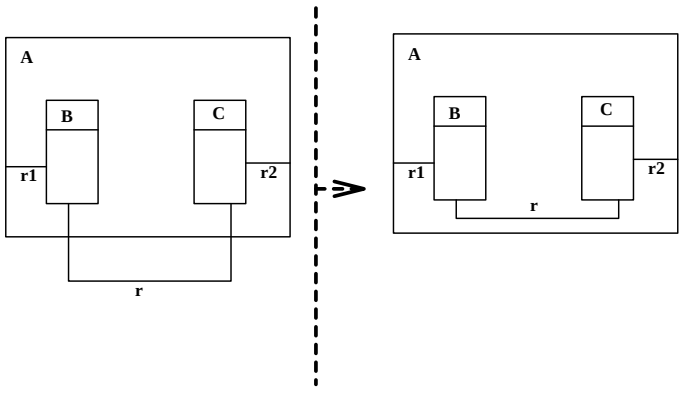

Figure 3: Moving Associations into Aggregates

2.2 Qualifiers

There is a certain ambiguity regarding the meaning of *qualifiers* on associations. In the UML notation guide (Version 1.1, page 58) it is stated "an object of the source class together with a value of the qualifier uniquely select a partition in the set of target class objects." Considering the example then given, of an association between banks and people, qualified by account numbers, we could infer that it is impossible for the same person to have more than one bank, contradicting the cardinalities given on the diagram.

Instead, we could consider qualification as an abbreviation of a particular form of association class (Figure 4). That is, the qualification means that each $\mathbf{a} : \mathbf{A}$ has a

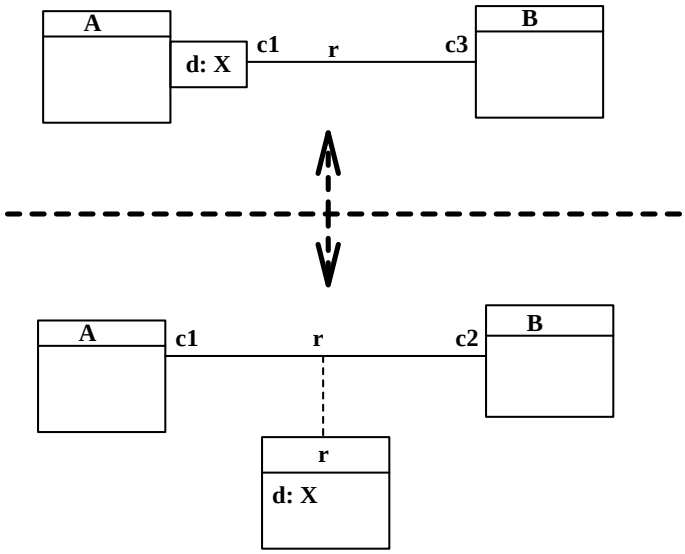

Figure 4: Qualification as Association Classes

number $\mathbf{x}$ of $\mathbf{b} : \mathbf{B}$ elements associated with it, where $\mathbf{x} \in \mathbf{c}_2$:

$$\forall\, \mathbf{a} : \overline{\mathbf{A}} \cdot \mathbf{card}(\{\mathbf{b} : \overline{\mathbf{B}} \mid (\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{r}}\}) \ \in \ \mathbf{c}_2$$

and

$$\forall\, \mathbf{a} : \overline{\mathbf{A}};\ \mathbf{x} : \mathbf{X} \cdot \mathbf{card}(\{\mathbf{b} : \overline{\mathbf{B}} \mid (\mathbf{a}, \mathbf{b}) \in \overline{\mathbf{r}} \wedge \mathbf{d}(\mathbf{a}, \mathbf{b}) = \mathbf{x}\}) \ \in \ \mathbf{c}_3$$

so there is a function $\mathbf{f} : \overline{\mathbf{A}} \times \mathbf{X} \rightarrow \mathbb{P}(\overline{\mathbf{B}})$ where each $\mathbf{f}(\mathbf{a}, \mathbf{x})$ has cardinality in the set $\mathbf{c}_3$.

This equivalence means that a valid transformation is to introduce the qualification instead of the explicit association class – this is a useful transformation if $\mathbf{c}_3$ is more restrictive (strictly smaller as a set) than $\mathbf{c}_2$.

The stronger interpretation suggested in [8] would require the additional constraint

$$\begin{array}{l} \forall\, \mathbf{b} : \overline{\mathbf{B}};\ \mathbf{a}_1, \mathbf{a}_2 : \overline{\mathbf{A}};\ \mathbf{x}_1, \mathbf{x}_2 : \mathbf{X} \cdot \\ \qquad \mathbf{d}(\mathbf{a}_1, \mathbf{b}) = \mathbf{x}_1 \wedge \mathbf{d}(\mathbf{a}_2, \mathbf{b}) = \mathbf{x}_2 \wedge (\mathbf{a}_1, \mathbf{b}) \in \overline{\mathbf{r}} \wedge (\mathbf{a}_2, \mathbf{b}) \in \overline{\mathbf{r}} \ \Rightarrow \\ \qquad\qquad \mathbf{a}_1 = \mathbf{a}_2 \wedge \mathbf{x}_1 = \mathbf{x}_2 \end{array}$$

in the association class version for the two models of Figure 4 to be equivalent. $\mathbf{c}_1$ could then be taken as $0\,..\,1$.

3 Statecharts

A statechart specification of the behaviour of instances of a class $\mathbf{C}$ can be formalised as an extension of the class theory of $\mathbf{C}$, as follows.

1. Each state $\mathbf{S}$ is represented in the same manner as a subclass of $\mathbf{C}$, and in general, nesting of state $\mathbf{S}_1$ in state $\mathbf{S}_2$ is expressed by axioms $\mathbf{S}_1 \subseteq \mathbf{S}_2$ and $\overline{\mathbf{S}_1} \subseteq \overline{\mathbf{S}_2}$ as for class generalisation.

2. Each transition in the statechart and each event for which the statechart defines a response yields a distinct action symbol. Each event $\mathbf{e}$ is the abstract generalisation of the actions $\mathbf{t}_1, \ldots, \mathbf{t_n}$ representing its transitions:

$$\forall\, \mathbf{a} : \mathbf{C} \cdot \mathbf{a}!\mathbf{t}_1 \supset \mathbf{a}!\mathbf{e} \ \wedge \ \ldots \ \wedge \ \mathbf{a}!\mathbf{t_n} \supset \mathbf{a}!\mathbf{e}$$

3. The axiom for the effect of a transition $\mathbf{t}$ from state $\mathbf{S}_1$ to state $\mathbf{S}_2$ with label

$$\mathbf{e(x)[G]/Post} \frown \mathbf{Act}$$

where $\mathbf{G}$ is the guard condition and $\mathbf{Post}$ is some postcondition constraint on the resulting state, is

$$\forall\, \mathbf{a} : \mathbf{C} \cdot \mathbf{a}.\mathbf{G} \wedge \mathbf{a} \in \overline{\mathbf{S}_1} \ \Rightarrow \ [\mathbf{a}!\mathbf{t(x)}](\mathbf{a}.\mathbf{Post} \wedge \mathbf{a} \in \overline{\mathbf{S}_2})$$

4. The transition only occurs if the trigger event occurs whilst the object is in the correct state:

$$\forall\, \mathbf{a} : \mathbf{C} \cdot \mathbf{a} \in \overline{\mathbf{S}_1} \wedge \mathbf{a}.\mathbf{G} \ \Rightarrow \ (\mathbf{a}!\mathbf{e(x)} \supset \mathbf{a}!\mathbf{t(x)})$$

We assume that distinct transitions from the same source state have non-overlapping guard conditions.

5. The generated actions must occur at some future time (after $\mathbf{t}$ has occurred):

$$\mathbf{a}!\mathbf{t(x)} \ \Rightarrow \ \bigcirc \diamond (\mathbf{a}!\mathbf{Act}_1 \wedge \diamond(\ldots \diamond \mathbf{a}!\mathbf{Act_m}) \ldots)$$

where $\mathbf{Act}$ is a list $\mathbf{Act}_1, \ldots, \mathbf{Act_m}$ of action invocations on objects associated to $\mathbf{a}$.

Transitions γ with labels of the form $\mathbf{after(t)}$ from source state $\mathbf{S}$ have an alternative axiom 4 defining their triggering, which asserts that they are triggered $\mathbf{t}$ time units after the most recent entry time $\clubsuit((\mathbf{a} \in \overline{\mathbf{S}}) := \mathbf{true}, \mathbf{j})$ to state $\mathbf{S}$ [7].

Likewise, automatic transitions α from a state $\mathbf{S}$ execute as soon as the activity of that state terminates.

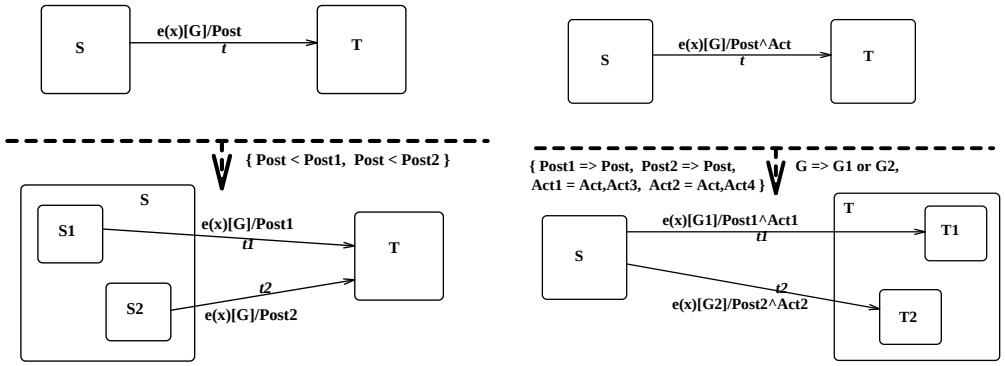

Figure 5: Source and Target Splitting of Transition

Source Splitting This transformation (Figure 5) can be justified in our semantics as follows.

The translation morphism σ from the old to the new model is:

$$\overline{\mathbf{S}} \longmapsto \overline{\mathbf{S}_1} \cup \overline{\mathbf{S}_2}$$
$$\mathbf{t} \longmapsto \mathbf{t}_1 \sqcap \mathbf{t}_2$$

$\mathbf{t} \sqcap \mathbf{t}'$ denotes the binary choice between two actions $\mathbf{t}$ and $\mathbf{t}'$.

Axioms 1 of the new model are $\overline{\mathbf{S}_1} \subseteq \overline{\mathbf{S}}$ and $\overline{\mathbf{S}_2} \subseteq \overline{\mathbf{S}}$, which are additional to the existing axioms of the old model.

Axioms 2 of the new model are (for each object)

$$\mathbf{t}_1 \supset \mathbf{e} \ \wedge \ \mathbf{t}_2 \supset \mathbf{e}$$

Together these establish that $\mathbf{t}_1 \sqcap \mathbf{t}_2 \supset \mathbf{e}$ as required.

Axioms 3 of the new model are

$$(\forall\, \mathbf{a} : \mathbf{C} \cdot \mathbf{a}.\mathbf{G} \wedge \mathbf{a} \in \overline{\mathbf{S}_1} \ \Rightarrow \ [\mathbf{a}!\mathbf{t}_1(\mathbf{x})](\mathbf{a}.\mathbf{Post}_1 \wedge \mathbf{a} \in \overline{\mathbf{T}})) \ \wedge$$
$$(\forall\, \mathbf{a} : \mathbf{C} \cdot \mathbf{a}.\mathbf{G} \wedge \mathbf{a} \in \overline{\mathbf{S}_2} \ \Rightarrow \ [\mathbf{a}!\mathbf{t}_2(\mathbf{x})](\mathbf{a}.\mathbf{Post}_2 \wedge \mathbf{a} \in \overline{\mathbf{T}}))$$

These imply the translation under σ of the axiom 3 of the old model because

$$\mathbf{a}.\mathbf{G} \wedge \sigma(\mathbf{a} \in \overline{\mathbf{S}}) \ \Rightarrow$$
$$\mathbf{a}.\mathbf{G} \wedge (\mathbf{a} \in \overline{\mathbf{S}_1} \vee \mathbf{a} \in \overline{\mathbf{S}_2}) \ \Rightarrow$$
$$[\mathbf{a}!\mathbf{t}_1(\mathbf{x})](\mathbf{a}.\mathbf{Post}_1 \wedge \mathbf{a} \in \overline{\mathbf{T}}) \ \vee \ [\mathbf{a}!\mathbf{t}_2(\mathbf{x})](\mathbf{a}.\mathbf{Post}_2 \wedge \mathbf{a} \in \overline{\mathbf{T}})$$

But by axiom 4 of the new model, $\mathbf{a}!\mathbf{e}(\mathbf{x}) \supset \mathbf{a}!\mathbf{t}_1(\mathbf{x})$ in the first case where $\mathbf{a} \in \overline{\mathbf{S}_1}$, so, by the property $(\alpha \supset \beta) \ \Rightarrow \ ([\beta]\mathbf{P} \Rightarrow [\alpha]\mathbf{P})$ of $\supset$ we have $[\mathbf{a}!\mathbf{e}(\mathbf{x})](\mathbf{a}.\mathbf{Post}_1 \wedge \mathbf{a} \in \overline{\mathbf{T}})$ in the first case (ie, if $\mathbf{a} \in \overline{\mathbf{S}_1}$).

But we already know that $\mathbf{t}_1 \sqcap \mathbf{t}_2 \ \supset \ \mathbf{e}$, so $[\mathbf{a}!(\mathbf{t}_1 \sqcap \mathbf{t}_2)(\mathbf{x})](\mathbf{a}.\mathbf{Post}_1 \wedge \mathbf{a} \in \overline{\mathbf{T}})$ in the first case, and hence, since $\mathbf{Post}_1 \Rightarrow \mathbf{Post}$, we have $[\mathbf{a}!(\mathbf{t}_1 \sqcap \mathbf{t}_2)(\mathbf{x})](\mathbf{a}.\mathbf{Post} \wedge \mathbf{a} \in \overline{\mathbf{T}})$.

By similar reasoning the same holds in the second case. But this means the conclusion is exactly σ of the conclusion of axiom 3 of the original model for $\mathbf{e}$, as required.

Axioms 4 in the new model are:

$$\mathbf{a} \in \overline{\mathbf{S}_1} \wedge \mathbf{a}.\mathbf{G} \ \Rightarrow \ (\mathbf{a}!\mathbf{e}(\mathbf{x}) \supset \mathbf{a}!\mathbf{t}_1(\mathbf{x}))$$
$$\mathbf{a} \in \overline{\mathbf{S}_2} \wedge \mathbf{a}.\mathbf{G} \ \Rightarrow \ (\mathbf{a}!\mathbf{e}(\mathbf{x}) \supset \mathbf{a}!\mathbf{t}_2(\mathbf{x}))$$

But because $\mathbf{t}_1 \supset \mathbf{t}_1 \sqcap \mathbf{t}_2$ and $\mathbf{t}_2 \supset \mathbf{t}_1 \sqcap \mathbf{t}_2$, this means that $\mathbf{a}!\mathbf{e}(\mathbf{x}) \supset \mathbf{a}!(\mathbf{t}_1 \sqcap \mathbf{t}_2)(\mathbf{x})$ in both cases, which establishes the translation of axiom 4 of the old model.

Target Splitting Dual to source splitting, we can replace a single transition $\mathbf{t}$ from $\mathbf{S}$ to $\mathbf{T}$ by two or more transitions for the same event which go to distinct substates of $\mathbf{T}$ in particular cases, and may have additional postconditions and generations (left hand side of Figure 5).

The refinement mapping in this case is that $\overline{\mathbf{T}}$ is interpreted by $\overline{\mathbf{T}_1} \cup \overline{\mathbf{T}_2}$, and $\mathbf{t}$ by $\mathbf{t}_1 \sqcap \mathbf{t}_2$.

4 Sequence Diagrams

There is a mechanical translation of sequence diagrams into assertions on the sending times $\leftarrow(\mathbf{m},\mathbf{i})$ of the $\mathbf{i}$-th instance of a message $\mathbf{m}$, the arrival times $\rightarrow(\mathbf{m},\mathbf{i})$ of this message instance, and the initiation $\uparrow(\mathbf{m},\mathbf{i})$ and termination $\downarrow(\mathbf{m},\mathbf{i})$ times of this message instance.

For example, Figure 6 translates to the following assertions, where each message execution lifeline is interpreted by a particular message instance:

$$
\begin{array}{l}
\forall\, \mathbf{i} : \mathbb{N}_1 \cdot \exists\, \mathbf{j},\mathbf{k},\mathbf{l},\mathbf{l}' : \mathbb{N}_1 \cdot \\
\qquad \rightarrow(\mathbf{Op},\mathbf{i}) \;=\; \uparrow(\mathbf{create}_{\mathbf{C1}}(\mathbf{ob1}),\mathbf{l}) \\
\qquad \downarrow(\mathbf{create}_{\mathbf{C1}}(\mathbf{ob1}),\mathbf{l}) \;\leq\; \leftarrow(\mathbf{ob3!bar}(\mathbf{x}),\mathbf{j}) \;=\; \rightarrow(\mathbf{ob3!bar}(\mathbf{x}),\mathbf{j}) \\
\qquad\qquad\qquad \leq\; \leftarrow(\mathbf{ob4!do}(\mathbf{w}),\mathbf{k}) \;=\; \rightarrow(\mathbf{ob4!do}(\mathbf{w}),\mathbf{k}) \\
\qquad \downarrow(\mathbf{ob4!do}(\mathbf{w}),\mathbf{k}) \;\leq\; \downarrow(\mathbf{ob3!bar}(\mathbf{x}),\mathbf{j}) \\
\qquad\qquad\qquad \leq\; \downarrow(\mathbf{kill}_{\mathbf{C1}}(\mathbf{ob1}),\mathbf{l}') \;=\; \downarrow(\mathbf{Op},\mathbf{i})
\end{array}
$$

These assertions can then be checked for consistency against detailed implementation level statecharts.

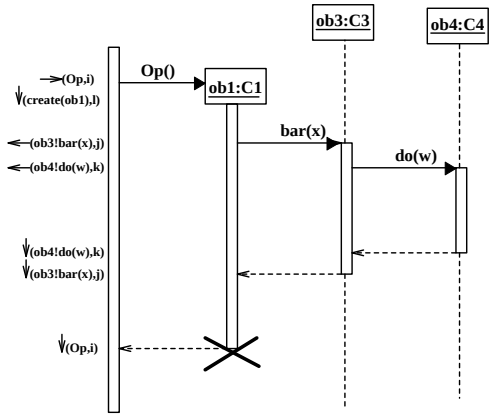

Figure 6: Example Sequence Diagram with Annotations

5 Collaboration Diagrams

The interaction elements of collaboration diagrams can also be interpreted as constraints on (generic) instances of objects and action invocations. Each message label in an interaction corresponds to a particular invocation instance of an action on an object. For

example, a message $\mathbf{label}_1 : \mathbf{m}_1(\mathbf{x}_1)$ sent to object $\mathbf{obj}_1$ yields the association of $\mathbf{label}_1$ to an invocation instance $(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}_1), \mathbf{i}_1)$ for some $\mathbf{i}_1 : \mathbb{N}_1$. The lexicographical ordering of labels determines the ordering of the executions of these invocation instances:

1. If $\mathbf{label}_2 : \mathbf{m}_2(\mathbf{x}_2)$ sent to object $\mathbf{obj}_2$ is an immediate successor of the $\mathbf{label}_1$ message, ie: $\mathbf{label}_1 = \mathbf{label}.\mathbf{x}[\mathbf{Name}]$ for some integer $\mathbf{x}$ and optional string $[\mathbf{Name}]$, and $\mathbf{label}_2 = \mathbf{label}.\mathbf{y}[\mathbf{Name}']$ where $\mathbf{y} > \mathbf{x}$, then the sending of $(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}_2), \mathbf{i}_2)$ strictly succeeds that of $(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}_1), \mathbf{i}_1)$:

   $$\leftarrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}_1), \mathbf{i}_1) \ < \ \leftarrow(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}_2), \mathbf{i}_2)$$

   If the first message send is synchronous, then the second send cannot occur until the first action instance has terminated:

   $$\downarrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}_1), \mathbf{i}_1) \ \leq \ \leftarrow(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}_2), \mathbf{i}_2)$$

2. If the $\mathbf{label}_1$ message is the immediate caller of a set of messages including $\mathbf{label}_2$, ie, $\mathbf{label}_2$ has the form $\mathbf{label}_1.\mathbf{x}[\mathbf{Name}]$ for integer $\mathbf{x}$ and optional string $[\mathbf{Name}]$, then if the $\mathbf{label}_1$ message has procedural control flow (filled solid arrowhead), then the $\mathbf{label}_2$ message must terminate before the $\mathbf{label}_1$ message:

   $$\begin{array}{l} \uparrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}_1), \mathbf{i}_1) \ \leq \ \leftarrow(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}_2), \mathbf{i}_2) \quad \wedge \\ \downarrow(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}_2), \mathbf{i}_2) \ \leq \ \downarrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}_1), \mathbf{i}_1) \end{array}$$

   However if the control flow of $\mathbf{label}_1$ is asynchronous or flat (half stick or stick arrowheads), then the second constraint is replaced by:

   $$\downarrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}_1), \mathbf{i}_1) \ \leq \ \leftarrow(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}_2), \mathbf{i}_2)$$

   That is, the calling message terminates (at least from the viewpoint of the caller) before the subordinate messages are sent.

Synchronisation constraints place additional restrictions on the start time of messages. If we have a message $\mathbf{label}_1/\mathbf{label} : \mathbf{m}(\mathbf{x})$ sent to $\mathbf{obj}$, then this message cannot commence until the message labelled by $\mathbf{label}_1$ has terminated:

$$\downarrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}_1), \mathbf{i}_1) \ \leq \ \leftarrow(\mathbf{obj}!\mathbf{m}(\mathbf{x}), \mathbf{i})$$

If we assume that the interaction diagram indicates the expected processing to be carried out for *every* invocation of a calling message (rather than just providing an *example* of what might happen), then for each calling message index $\mathbf{i}$ we can assert the existence of appropriate indecies for the subordinate messages. In addition, for conditional messages, the condition must be true at the time the message is sent. For example, Figure 7 is formalised as:

$$\begin{array}{l} \forall \mathbf{i} : \mathbb{N}_1 \cdot \exists \mathbf{i}_1 : \mathbb{N}_1 \cdot \\ \qquad (\uparrow(\mathbf{obj}!\mathbf{m}(\mathbf{x}), \mathbf{i}) \ \leq \ \leftarrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}), \mathbf{i}_1) \ \wedge \\ \qquad \ (\mathbf{x} > 0) \circledcirc \leftarrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}), \mathbf{i}_1) \ \wedge \\ \qquad \ \downarrow(\mathbf{obj}_1!\mathbf{m}_1(\mathbf{x}), \mathbf{i}_1) \ \leq \ \downarrow(\mathbf{obj}!\mathbf{m}(\mathbf{x}), \mathbf{i})) \ \vee \\ \qquad (\uparrow(\mathbf{obj}!\mathbf{m}(\mathbf{x}), \mathbf{i}) \ \leq \ \leftarrow(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}), \mathbf{i}_1) \ \wedge \\ \qquad \ (\mathbf{x} \leq 0) \circledcirc \leftarrow(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}), \mathbf{i}_1) \ \wedge \\ \qquad \ \downarrow(\mathbf{obj}_2!\mathbf{m}_2(\mathbf{x}), \mathbf{i}_1) \ \leq \ \downarrow(\mathbf{obj}!\mathbf{m}(\mathbf{x}), \mathbf{i})) \end{array}$$

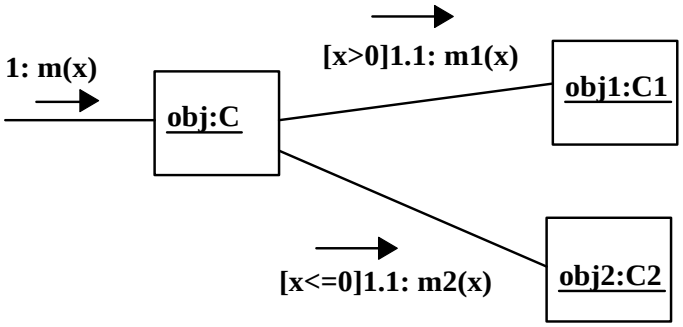

Figure 7: Example Conditional Interaction

$\mathbf{P}\circledcirc\mathbf{t}$ denotes that $\mathbf{P}$ holds at time $\mathbf{t}$. Time labels can be placed on message arrows as for sequence diagrams. The label $\mathbf{A}$ at the tail of a message arrow indicates the send time $\mathbf{A} = \leftarrow(\mathbf{obj!m(x)}, \mathbf{i})$ of the corresponding message instance. The corresponding dashed label $\mathbf{A}'$ denotes the receive time of this message.

In the case that all message sends in an interaction are procedural, then we can simplify this semantic representation by translating the interaction into an abstract psuedocode using composite actions such as α; β (sequential composition), **if e then** α **else** β (conditionals), **for all** (unordered iteration), etc.

We illustrate this case with a proof of the correctness of the 'state' pattern (Figure 8).

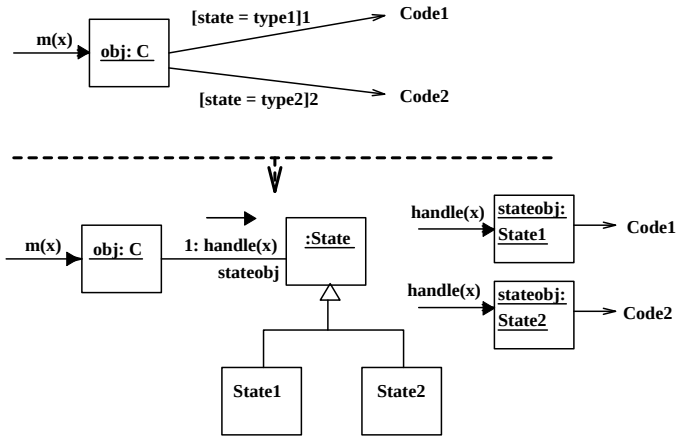

Figure 8: Introduction of State Pattern

In the original model we have the behaviour axiom:

$$\begin{aligned}&\mathbf{obj} \in \overline{\mathbf{C}} \;\Rightarrow\\ &\qquad \mathbf{obj!m(x)} \;\supset\\ &\qquad\qquad \mathbf{if\ state = type1}\\ &\qquad\qquad \mathbf{then\ Code1}\\ &\qquad\qquad \mathbf{else\ Code2}\end{aligned}$$

In the new model we have

$$\mathbf{obj} \in \overline{\mathbf{C}} \;\Rightarrow\; \mathbf{obj!m(x)} \supset \mathbf{obj.stateobj!handle(x)}$$

and

$$\mathbf{stateobj} \in \overline{\mathbf{State}1} \quad \Rightarrow \quad \mathbf{stateobj!handle(x)} \supset \mathbf{Code}1$$

and similarly for **State**2.

Therefore, interpreting **obj.state** by

if obj.stateobj $\in \overline{\mathbf{State}1}$
then type1
else
  if obj.stateobj $\in \overline{\mathbf{State}2}$
  then type2
  else nil

we obtain that the original behaviour axiom is valid in the new model.

The semantics of sequence and interaction diagrams are defined in a similar way. Indeed there is considerable overlap in the expressiveness of these diagrams, suggesting that they may not be optimal choices for models of dynamic behaviour.

6 Relationship with Other Work

Representation of UML semantics in Z has been developed in [4]. This representation is a formalisation of the UML metamodel, whilst our approach shows how to associate theories to specific UML models. The set-theoretic models are similar: [4] represents each class by a set of instances, and each association by a set of links consisting of object tuples, as in this paper. Our notion of theory satisfaction agrees with the $\models$ relationship of [4]: if $\sigma : \Gamma \rightarrow \Gamma'$ is a theory interpretation then every model $\mathbf{M}$ of Γ' has a *reduct* $\mathbf{M}' = \mathbf{M}_\sigma$ of Γ with the interpretation $[\![\overline{\mathbf{C}}]\!]_{\mathbf{M}'}$ of the extension of classes $\mathbf{C}$ of Γ in $\mathbf{M}'$ being defined as $[\![\overline{\sigma(\mathbf{C})}]\!]_{\mathbf{M}}$. But this means that $\Gamma' \models \Gamma$ in the terms of [4].

It is trivial to show that simple transformations such as adding a class or association *are* theory extensions in our formalism. Our approach has been chosen to simplify the task of proving transformations correct, but could be further formalised and generalised by using the Z semantics given in [4].

7 Conclusions

We have proposed an axiomatic semantics of the UML notation, together with examples where this semantics helps to clarify issues of the meaning of UML constructs. Other areas which we have addressed elsewhere include dynamic classification (page 69 of [8]), the meaning of interface specifications and the interpretation of OCL in conventional mathematical notation [5].

Areas of UML with quite vague and incomplete semantics include Use Cases and Implementation diagrams. A detailed mathematical semantics would probably be inappropriate and irrelevant for such notations, given their roles in development. However the semantics could be useful to validate models – ie., to check that desired properties are true in particular models.

Object models and statecharts together seem to be sufficient to provide a fully detailed model of an implementation of a system, hence sequence and collaboration diagrams are in a sense redundant. There is also considerable redundancy between the sequence diagram, interaction and activity diagram notations, and some unification of these would be potentially beneficial for methods based on the UML.

References

[1] J C Bicarregui, K C Lano, T S E Maibaum, *Objects, Associations and Subsystems: a hierarchical approach to encapsulation*, ECOOP 97, LNCS, 1997.

[2] D Coleman, P Arnold, S Bodoff, C Dollin, H Gilchrist, F Hayes, and P Jeremaes. *Object-oriented Development: The FUSION Method*. Prentice Hall Object-oriented Series, 1994.

[3] Cook S., Daniels J., *Designing Object Systems: Object-oriented Modelling with Syntropy*, Prentice Hall, 1994.

[4] Evans A., France R., Lano K., Rumpe B., *The UML as a Formal Modelling Notation*, PUML working group, 1998.

[5] Lano K., Bicarregui J., *UML Refinement and Abstraction Transformations*, ROOM 2 Workshop, Bradford University, 1998.

[6] K Lano, *Logical Specification of Reactive and Real-Time Systems*, to appear in *Journal of Logic and Computation*, 1998.

[7] K. Lano, *Transformations on Syntropy and UML Models*, Technical Report, "Formal Underpinnings for Object Technology" project, Dept. of Computing, Imperial College, 1997.

[8] Rational Software et al, *UML Notation Guide*, Version 1.1, http://www.rational.com/uml, 1997.

[9] M. Spivey, *The Z Notation: A Reference Manual*, Prentice Hall, 1992.

[10] R Wieringa, W. de Jonge, P. Spruit, *Roles and Dynamic Subclasses: A Model Logic Approach*, IS-CORE report, Faculty of Mathematics and Computer Science, Vrije Universiteit, Amsterdam, 1993.