Towards a Modal Logic of Durative Actions

Stuart Kent
sjk@doc.ic.ac.uk
Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ, United Kingdom

Abstract

This paper proposes an extension of modal action logics, which typically make the assumption that an action is atomic, to include durative actions. These logics have been developed to support the formal specification of information systems: we argue, with particular reference to object oriented systems, that assuming atomicity is too restrictive to express many kinds of temporal constraint. In consequence, we propose that actions be regarded as durative, and encode this by assuming that an action occurs over a sequence of atomic transitions, or interval, rather than a single transition. With this as a pre-requisite, the paper continues to redefine and extend operators of atomic action logics to fit the durative case.

1 Introduction

This paper describes work whose aim is to support the formal specification, characterisation and development of object oriented systems. Specifically, we propose an extension to action logics [16, 18, 7, 15] which have already been used with some success in providing axiomatic semantics to object oriented specification languages [8, 9] and thereby providing a basis for reasoning about object oriented systems and their development. The core of our proposal is a reworking of the semantics of such logics to admit *durative* actions.

An action logic distinguishes between terms denoting actions performed in the system and terms denoting values. The value-denoting terms are used to represent the state of the "abstract machine" being specified. Actions identify transitions to change that state. Typically, the semantics of these logics force actions to be atomic:[1] they are regarded as occurring over single atomic transitions and, with regard to concurrency, it is only possible to express restrictions on their synchronisation. However, when constructing an object oriented model of a system one usually makes the assumption that methods have duration, and, in general, may operate concurrently not just synchronously –ie. one method may start whilst another is in progress, and only parts of a method need to synchronise where conflict avoidance is required. This is reflected in the recent development of OO programming languages, such as DRAGOON [4] and POOL [3], in which concurrent method execution is the norm and language

[1] Perhaps an exception to this is [17], which admits compound durative actions (eg. actions constructed using a sequential combinator). However, primitive durative actions (ie. actions whose structure is not determined) are not allowed.

constructs are provided to restrict this concurrency. Furthermore, objects call on the services of other objects by invoking their methods. Such invocations force the invoking method to run concurrently with the invoked method.[2]

In order to specify such systems, it would be desirable to have a specification language in which methods are assumed to be durative, and which has language constructs for imposing restrictions on the concurrency of methods. In providing an axiomatic semantics for such a language, two options present themselves:

- Attempt to model methods in terms of atomic actions, and specify concurrency constraints through synchronisation of these atomic actions.
- Develop an action logic which admits durative actions and has appropriate language constructs for expressing their concurrent behaviour.

We favour the latter approach, for the same reasons that motivated the use of modal logics for specification in the first place: they are "engineered" to provide a range of operators, which allows the same language to be used both to express behaviour and reason about it.

The paper is organised as follows. Section 2 introduces the time point and interval structures and gives a semantic characterisation of durative actions. This prepares the incremental definition of the logic in Sections $3-6$, introducing, respectively, the core logic, action combinators for constructing actions from component actions, operators for transforming action terms to formulae and vice-versa (eg. to express when an action is in progress), and a range of temporal operators for relating occurrences of different actions. A summary of results and indication of future work is provided in Section 7.

2 Temporal Structures and Actions

This section introduces the mathematical machinery used in the remainder of the paper. The semantics of action logics, which assume atomicity of actions, may be given in terms of a structure of states or *time points*, and an action is characterised as a collection of atomic transitions (its occurrences) –ie. transitions between neighbouring points.[3] In order to admit durative actions, an occurrence of an action will be regarded as a *sequence* of atomic transitions, alternatively an *interval* comprising at least two time points. To formulate this, we begin by introducing a time-point structure, an interval structure derived from a point structure, and the basic mathematical 'tools' required to manipulate and relate points and intervals. This will be followed by a semantic characterisation of actions along the lines just outlined.

A time point structure and the interval structure derived from it are given by Definition 1.

---

[2] Even in sequential programming languages, such as Smalltalk, the invoked method must be performed within the body of the invoking method.

[3] Note that a transition may observe more than one action.

Definition 1 (point and interval structures)

(a) $\mathcal{T} = \langle T, < \rangle$ is a time point structure, where T is a set of time points and $<$ a discrete, linear ordering with an initial time point t_0.

(b) $\mathcal{I}(\mathcal{T}) = \langle I_{\mathcal{T}}, \lhd_{\mathcal{T}} \rangle$ is the interval structure derived from $\mathcal{T}$ where $I_{\mathcal{T}}$ is the set of intervals and $\lhd_{\mathcal{T}}$ is the precedence ordering on intervals such that:

i. for any $i, j \in I_{\mathcal{T}}, i \lhd_{\mathcal{T}} j$ if and only if for every $t \in i$ and $t' \in j, t < t'$

ii. $I_{\mathcal{T}} = \left\{ i \mid \begin{array}{c} i \in \wp(T) \text{ and for all } t, t' \in i \text{ if there exists} \\ t'' \in T \text{ s.t. } t < t'' < t' \text{ then } t'' \in i \end{array} \right\}$

The point structures used to interpret the action logics referred to earlier have generally been discrete and have an initial time point. To keep things simple, we have also assumed here a linear ordering.[4] From now on, when we write $\mathcal{T}$ we will mean the point structure $\langle T, < \rangle$. Similarly, we use the shorthand $\mathcal{I}$ for $\mathcal{I}(\mathcal{T})$, and assume this to be defined as $\langle I, \lhd \rangle$. Some functions and notation for manipulating points, intervals, sets of points and sets of intervals are given by Definition 2, which assumes a point structure $\mathcal{T}$ and associated interval structure $\mathcal{I}$, according to the conventions set out above.

Definition 2 For any $i \in I, s, t \in T, U \subseteq T$ and $J \subseteq I$,

(a) $\mathrm{l}(i)$ and $\mathrm{r}(i)$ represent, respectively, the *left* and *right bounds* of the interval i, where

$$\mathrm{l}(i) = \max\{t' \in T \mid \text{ for every } t'' \in i, t' \leq t''\}$$

and

$$\mathrm{r}(i) = \min\{t' \in T \cup \infty \mid \text{ for every } t'' \in i, t' \leq t''\}$$

for ∞ s.t. for every $t' \in T, t' < \infty$;

(b) $\mathrm{i}(i) = i - \{\mathrm{l}(i), \mathrm{r}(i)\}$ represents the *interior* of the interval i;

(c) $[s,t], (s,t), [s,t)$ and $(s,t]$ are the intervals defined by,

$$\begin{array}{rclrcl} [s,t] & = & \{t' \mid s \leq t' \leq t\} & (s,t) & = & \{t' \mid s < t' < t\} \\ [s,t) & = & \{t' \mid s \leq t' < t\} & (s,t] & = & \{t' \mid s < t' \leq t\} \end{array}$$

(d) $\mathrm{succ}(t) \in T$ is the unique point succeeding t where $\mathrm{succ}(t) > t$ and there exists no $t' \in T$ s.t. $t < t' < \mathrm{succ}(t)$; $\mathrm{pred}(t)$ is the point immediately preceeding t and is defined similarly;

(e) $\mathrm{ev}_t(t) = [t, \mathrm{succ}(t)]$, is the atomic transition or *event* observed from t;

(f) $\mathrm{ev}_U(U) = \{\mathrm{ev}_t(t) \mid t \in U\}$, is the set of events observed from points in the set of points U (note that the set of all events is given by $\mathrm{ev}_U(T)$);

(g) $\mathrm{intvls}(U) = \{i \in I \mid i \subseteq U\}$, is the set of intervals generated from the set of points U;

(h) $\max_{intvls}(J) = \{i \in J \mid \text{for every } j \in J, i \not\subset j\}$ is the set of maximal intervals in the set of intervals J;

[4] A possible use of branching time is discussed in the conclusions.

(i) $\max_{ev}(E) = \left\{ i \in I \mid \begin{array}{c} \mathrm{ev}_U(i) \subseteq E \text{ and for every } j \in I, \\ \text{if } j \supset i \text{ then } \mathrm{ev}_U(j) \not\subseteq E \end{array} \right\}$,

for any $E \subseteq \mathrm{ev}_U(T)$, returning the largest intervals whose events are in E;

(j) $\mathrm{insts}(J) = \{[t] \mid t \in T \text{ and } [t] \in J\}$

As a number of authors have observed [19, 2, 20, 5], there are thirteen primitive relations between intervals: six original relations, their inverses and equality. All relations are disjoint and partition the universal relation. The notation used for the six original relations and their inverses is given by Definition 3, which also assumes an interval structure $\mathcal{I}$.

Definition 3 (interval relations) $\lhd$, $\rhd$, $[-(-]-)$, $(-[-)-]$, $)[$, $]($, $([-]-)$, $[(-)-]$, $(-[-])$, $[-(-)]$, $(-[-]-)$ and $[-(-)-]$ are relations on $I \times I$, such that, for any $i, j \in I$,

(a) $i \lhd j$ iff $\mathrm{r}(i) < \mathrm{l}(j)$ iff $j \rhd i$

(b) $i \; [-(-]-) \; j$ iff $\mathrm{l}(i) < \mathrm{l}(j) < \mathrm{r}(i) < \mathrm{r}(j)$ iff $j \; (-[-)-] \; i$

(c) $i \;)[\; j$ iff $\mathrm{r}(i) = \mathrm{l}(j)$ iff $j \;](\; i$

(d) $i \; ([-]-) \; j$ iff $\mathrm{l}(i) = \mathrm{l}(j) < \mathrm{r}(i) < \mathrm{r}(j)$ iff $j \; [(-)-] \; i$

(e) $i \; (-[-]) \; j$ iff $\mathrm{l}(j) < \mathrm{l}(i) < \mathrm{r}(i) = \mathrm{r}(j)$ iff $j \; [-(-)] \; i$

(f) $i \; (-[-]-) \; j$ iff $\mathrm{l}(j) < \mathrm{l}(i) < \mathrm{r}(i) < \mathrm{r}(j)$ iff $j \; [-(-)-] \; i$

These relations may be compounded in many different ways. For example, the usual set theoretic relations $\subset$ and $\subseteq$ are defined to be $(-[-]-) \cup (-[-]) \cup ([-]-)$ and $\subset \cup =$, respectively. Although we do not define all these compounds explicitly here, when later we refer to an arbitrary relation between intervals, we will mean a primitive or compound relation.

2.1 Actions

As indicated earlier, an occurrence of an action is to be characterised as a sequence of atomic transitions or events. Intuitively, an action must be subject to the condition that one occurrence may not begin either whilst another of its occurrences is in progress or at the same time as another of its occurrences begins.[5] In linear time, this restriction is simply encoded by ensuring that the interiors of occurrences must be disjoint, here assuming that an action is in progress on all interior states of one of its occurrences. This leads to a formulation of the set of actions, as follows:

Definition 4 (actions) For any point structure $\mathcal{T}$,

(a) $Act(\mathcal{T}) = \left\{ a \mid \begin{array}{c} a \in \wp(I), \mathrm{insts}(a) = \emptyset \text{ and} \\ \text{for every } i, j \in a, \mathrm{i}(i) \cap \mathrm{i}(j) = \emptyset \end{array} \right\}$

(b) $Act_{atom}(\mathcal{T}) = \{a \mid a \subseteq \mathrm{ev}_U(T)\}$

The restriction $\mathrm{insts}(a) = \emptyset$ in (a) ensures that the occurrences of an action have at least two time points: ie. one in which it starts, and one in which it either finishes or is in progress.[6] $Act_{atom}(\mathcal{T})$ distinguishes the set of atomic actions, ie. those actions which have only events as occurrences. We will write Act and Act_{atom} as shorthands for $Act(\mathcal{T})$ and $Act_{atom}(\mathcal{T})$, respectively.

[5] See [14] for a more detailed discussion on the nature of actions.

[6] Occurrences of an action may have no right bound. Thus an action may be in progress for ever.

Definition 5 provides some functions for constructing various subcomponents of an action. The function *ev* results in all those events performed during any occurrence of an action a. These include the begin events and the end events, which are given by *beg* and *end*, respectively. *prog* identifies those points in time when an action a is said to be in progress, namely those instants which are contained in the interior of any of its occurrences. The definition assumes a point structure $\mathcal{T}$.

Definition 5 (action subcomponents) For any $a \in Act$,
(a) $\mathrm{ev}(a) = \{e \mid e \in \mathrm{ev}_U(i - \{\mathrm{r}(i)\}),\ \text{for some } i \in a\}$
(b) $\mathrm{beg}(a) = \{e \mid e \in \mathrm{ev}_U(T) \text{ and there exists } i \in a \text{ s.t. } e\ ([-]-)\ i\}$
(c) $\mathrm{end}(a) = \{e \mid e \in \mathrm{ev}_U(T) \text{ and there exists } i \in a \text{ s.t. } e\ (-[-])\ i\}$
(d) $\mathrm{prog}(a) = \bigcup_{i \in a} \mathrm{i}(i)$

3 The Core Logic

Before considering action combinators and temporal operators, we describe the core part of the logic, comprising the first order component and action modalities.

3.1 Syntax

The syntax of the core logic is no different to that used in the action logics mentioned earlier. Our presentation conforms to that of [15]. We begin with the definition of a signature for some arbitrary theory. We distinguish value from action terms, and accordingly distinguish value from action symbols. There are three types of value symbol: logical variables, functions and attributes. Semantically, functions are *rigid*, as their denotation will not change over time, and attributes are *non-rigid*, as their denotation may change with time.

Definition 6 (signature) $\Theta = \langle \Sigma, \Omega, \mathcal{V}, \mathcal{AT}, \mathcal{AC} \rangle$ is a *signature* where:
(a) Σ is a set of sorts
(b) Ω is a $\Sigma^* \times \Sigma$-indexed family of function symbols,[7]
(c) $\mathcal{V}$ is a collection of variables over the sorts,
(d) $\mathcal{AT}$ is a $\Sigma^* \times \Sigma$-indexed family of attribute symbols,
(e) $\mathcal{AC}$ is an Σ^*-indexed family of action symbols.

We will write Θ as a shorthand for the signature $\langle \Sigma, \mathcal{V}, \mathcal{AT}, \mathcal{AC} \rangle$. Value terms are variables, or composed from an attribute or function symbol combined with an appropriate number of value term arguments. An action term is either composed from an action symbol combined with an appropriate number of value term arguments, or is formed from an action term or terms in combination with one of a number of combinators. In this section, we only consider uncombinated action terms.

[7] Constants are simply function symbols with zero arity. Predicates are effectively Boolean functions

Definition 7 (terms) The set of terms $Term(\Theta) = VTerm(\Theta) \cup AcTerm(\Theta)$ where:
(a) $VTerm(\Theta) = \bigcup_{S \in \Sigma} VTerm'(\Theta, S)$
(b) for any $\mathsf{x} : S \in \mathcal{V}, \mathsf{x} \in VTerm'(\Theta, S)$
(c) for any $\mathsf{f} : S_1 \times \ldots \times S_n \times S \in \Omega$ and $t_i \in VTerm'(\Theta, S_i)$, $\mathsf{f}(t_1, \ldots, t_n) \in VTerm'(\Theta, S)$
(d) for any $\mathsf{A} : S_1 \times \ldots \times S_n \times S \in \mathcal{AT}$ and $t_i \in VTerm'(\Theta, S_i)$, $\mathsf{A}(t_1, \ldots, t_n) \in VTerm'(\Theta, S)$
(e) for any $\mathsf{a} : S_1 \times \ldots \times S_n \in \mathcal{AC}$ and $t_i \in VTerm'(\Theta, S_i)$, $\mathsf{a}(t_1, \ldots, t_n) \in AcTerm(\Theta)$

As usual, formulae of the logic are obtained by combining terms with various logical operators in a prescribed way. The formulae of the core logic comprise:

- the usual first order formulae, where equalities between value terms are the atomic formulae;
- action modalities, $[\,]-$ and $[-]-$, familiar to modal action logics and used for expressing change.

They are given by Definition 8.

Definition 8 (formulae) For any signature Θ, the set of Θ-formulae is $Form(\Theta)$, where:

$$Form(\Theta) \supset \left\{ \begin{array}{c} (t_1 = t_2), \neg\alpha, (\alpha \wedge \beta), \\ (\alpha \vee \beta), (\alpha \rightarrow \beta), (\alpha \leftrightarrow \beta), \\ \forall \mathsf{x} \cdot \alpha, \exists \mathsf{x} \cdot \alpha, [\,]\alpha, [at]\alpha \end{array} \middle| \begin{array}{c} t_1, t_2 \in VTerm(\Theta), \\ \mathsf{x} \in \mathcal{V}, at \in AcTerm(\Theta) \\ \text{and } \alpha, \beta \in Form(\Theta) \end{array} \right\}$$

3.2 Semantics

The semantics is given in terms of an interpretation structure and an interpretation function. An interpretation structure for a signature Θ (Definition 9), is a tuple comprising a point structure $\mathcal{T}$, an algebra of sorts for interpreting the sorts and functions of Σ, and a function f assigning appropriate denotations to the actions and attribute symbols. Specifically, f maps an action symbol to a function from the sorts of its arguments to actions, and maps an attribute symbol to a function from the sorts of its arguments to intensions.

Definition 9 (interpretation structure) For any signature Θ, a Θ-*interpretation structure* is a tuple $\Im = \langle \mathcal{T}, \mathcal{U}, f \rangle$, where:
(a) $\mathcal{T}$ is a time point structure,
(b) $\mathcal{U}$ is a tuple $\langle \Sigma_U, \Omega_\mathcal{U} \rangle$ s.t.

$$\begin{array}{rcl} \Sigma_U & = & \{S_u \mid S \in \Sigma\} \\ \Omega_\mathcal{U} & = & \{\mathsf{h}_\mathcal{U} : S_{1_u} \times \ldots \times S_{n_u} \times T \rightarrow S_u \mid \mathsf{h} : S_1 \times \ldots \times S_n \times S \in \Omega\} \end{array}$$

(c) f is a function mapping

$$\begin{array}{rcl} \mathsf{a} : S_1 \times \ldots \times S_n \in \mathcal{AC} & \text{to} & f(\mathsf{a}) : S_{1_u} \times \ldots \times S_{n_u} \rightarrow Act(\mathcal{T}) \\ \mathsf{A} : S_1 \times \ldots \times S_n \times S \in \mathcal{AT} & \text{to} & f(\mathsf{A}) : S_{1_u} \times \ldots \times S_{n_u} \times T \rightarrow S_u \end{array}$$

From now on, we will use $\Im$ as a shorthand for the interpretation structure $\langle \mathcal{T}, \mathcal{U}, f \rangle$.

An interpretation function assigns denotations to terms and formulae at a given point in time.[8] Definition 10 defines the interpretation function over value terms, uncombinated action terms, and formulae of the core logic. The definition will be extended to consider other terms and formulae in later sections.

Definition 10 (interpretation function) For any signature Θ, Θ-interpretation structure $\Im$ and Θ-variable assignment A, $[\![\ ]\!]^{\Im,A}$ is a $\langle\Theta, \Im, A\rangle$-*interpretation function* which maps

$$\begin{array}{rcl} VTerm(\Theta) \times T & \text{to} & \bigcup_{S_{\mathcal{U}} \in \mathcal{S}} S_{\mathcal{U}} \\ AcTerm(\Theta) \times T & \text{to} & Act(\mathcal{T}) \\ Form(\Theta) \times T & \text{to} & \{true, false\} \end{array}$$

such that, for any $t \in T, \mathsf{a} : S_1 \times \ldots \times S_n \in \mathcal{AC}, \mathsf{A} : S_1 \times \ldots \times S_n \times S \in \mathcal{AT}$, $\mathsf{f} : S_1 \times \ldots \times S_n \times S \in \Omega, t_1, \ldots, t_n \in VTerm(\Theta), \mathsf{x} \in \mathcal{V}, at \in AcTerm(\Theta)$ and $\alpha, \beta \in Form(\Theta)$,

$$\begin{array}{rcl} [\![\, \mathsf{x} \,]\!]^{\Im,A}(t) & = & A(\mathsf{x}) \\ [\![\, \mathsf{f}(t_1, \ldots, t_n) \,]\!]^{\Im,A}(t) & = & \mathsf{f}_{\mathcal{U}}([\![\, t_1 \,]\!]^{\Im,A}(t), \ldots, [\![\, t_n \,]\!]^{\Im,A}(t)) \\ [\![\, \mathsf{A}(t_1, \ldots, t_n) \,]\!]^{\Im,A}(t) & = & f(\mathsf{A})([\![\, t_1 \,]\!]^{\Im,A}(t), \ldots, [\![\, t_n \,]\!]^{\Im,A}(t), t) \\ [\![\, \mathsf{a}(t_1, \ldots, t_n) \,]\!]^{\Im,A}(t) & = & f(\mathsf{a})([\![\, t_1 \,]\!]^{\Im,A}(t), \ldots, [\![\, t_n \,]\!]^{\Im,A}(t)) \\ [\![\, t_1 = t_2 \,]\!]^{\Im,A}(t) & = & true \text{ iff } [\![\, t_1 \,]\!]^{\Im,A}(t) = [\![\, t_2 \,]\!]^{\Im,A}(t) \\ [\![\, [at]\alpha \,]\!]^{\Im,A}(t) & = & true \text{ iff for every } i \in [\![\, at \,]\!]^{\Im,A}(t), \\ & & \text{if } \mathrm{l}(i) = t \text{ then } [\![\, \alpha \,]\!]^{\Im,A}(\mathrm{r}(i)) = true \\ [\![\, [\,]\alpha \,]\!]^{\Im,A}(t) & = & true \text{ iff if } t = t_0 \text{ then } [\![\, \alpha \,]\!]^{\Im,A}(t) = true \end{array}$$

and where the interpretation of the classical logical connectives is as usual.

Intuitively, an action modality is interpreted as for an atomic action logic, in that the post condition must hold at the end of any occurrence of the action starting in the state in which the formula is true. The difference, of course, is that here occurrences of an action may occur over a sequence of transitions (interval) rather than just a single transition. The usual notions of validity, satisfiability etc. for modal logics apply here.

4 Action Combinators

Action combinators are operators used to construct new actions from component actions. In logics where actions are treated atomically (see eg. [15]), the combinators are parallel ($at \parallel bt$), choice ($at + bt$) and complement ($\overline{at}$). The occurrences of a parallel combination of actions is the intersection of the occurrences of the component actions, those of a choice of actions is the union of the occurrences of the components, and the complement of an action comprises all those transitions which do not observe that action.[9] These definitions do not

[8] The alternative is to use intervals of time as the units of evaluation. This point is returned to in the concluding section.

[9] That is, performing the complement of an action a means performing any action other than a.

carry over directly to the durative framework, because the result of combining the component actions in the way described may not themselves be actions according to the restrictions defined earlier. In particular, problems arise with the choice combinator, when the interiors of occurrences of the component actions overlap, and with complementation.

In this section we reinterpret the atomic combinators to fit the durative case,[10] and indicate how new 'temporal' combinators could be defined. Looking first at complementation, directly porting the definition from the atomic case would mean that every interval, which is not an occurrence of the component action, would be an occurrence of its complementation. Clearly, such a collection of intervals would not be an action according to the restrictions set out earlier. An alternative interpretation is to use complementation to identify the largest intervals during which the action is not being performed. Formally, extending Definitions 7 and 10:

- $AcTerm(\Theta) \supset \{\overline{at} \mid at \in AcTerm(\Theta)\}$
- $[\![\, \overline{at} \,]\!]^{\Im,A}(t) = \max_{ev}(\mathrm{ev}_U(T) - \mathrm{ev}([\![\, at \,]\!]^{\Im,A}(t)))$

We re-interpret the parallel combinator in a similar way to complementation, with the idea that $(at_1 \parallel at_2)$ denotes the action whose occurrences are the maximal periods during which both component actions are being performed. However, here we have to be slightly more careful about what we mean by maximality. Specifically, we would like equivalence between at and $(at \parallel at)$.[11] This will not be the case if $\parallel$ is interpreted as just taking the maximal intervals during which both actions are being performed. The problem arises when two occurrences of at meet. By the definition just suggested, these occurrences will be merged into one maximal occurrence of the action $(at \parallel at)$, thereby blocking the required equivalence. The problem can be avoided by first defining a weaker notion of merging: merge is a function merging a set of intervals into an action, by only merging those intervals whose interiors intersect.

Definition 11 (merge) For any $a \in Act$,
(a) $\mathrm{merge}(a) = \max_{ev}(\mathrm{ev}(a - \mathrm{action}(a))) \cup \mathrm{action}(a)$
(b) $\mathrm{action}(a) = \{i \in a \mid$ for all $j \in a, j = i$ or $\mathrm{i}(j) \cap i(i) = \emptyset\}$

The parallel combinator may now be defined as an extension of Definitions 7 and 10:

- $AcTerm(\Theta)\{\supset (at_1 \parallel at_2) \mid at_1, at_2 \in AcTerm(\Theta)\}$
- $[\![\, (at_1 \parallel at_2) \,]\!]^{\Im,A}(t) = \mathrm{merge}([\![\, at_1 \,]\!]^{\Im,A}(t) \cap [\![\, at_2 \,]\!]^{\Im,A}(t))$

It is a simple matter to show that this definition yields the desired property of $\parallel$. A similar definition may be given to the choice combinator, and this too ensures that at is equivalent to $(at + at)$.

- $AcTerm(\Theta) \supset \{(at_1 + at_2) \mid at_1, at_2 \in AcTerm(\Theta)\}$
- $[\![\, (at_1 + at_2) \,]\!]^{\Im,A}(t) = \mathrm{merge}([\![\, at_1 \,]\!]^{\Im,A}(t) \cup [\![\, at_2 \,]\!]^{\Im,A}(t))$

[10] The interpretations given here are similar to those given to operators proposed by [5], who develop an interval algebra in the Z-like specification language GLIDER for the expression of interval constraints. The differences are twofold: firstly, they assume a rational model of time; secondly, they have a difference combinator instead of complementation –this would be defined here as $at_1 \parallel \overline{at_2}$.

[11] By the definition we have given, it is not the case that $\overline{\overline{at}}$ is equivalent to at, for the reason that the second complementation will merge any occurrences of at which meet. This, perhaps, is also an undesirable property, but, unfortunately, there seems to be no way of avoiding it.

The introduction of durative actions also provides scope for introducing new action combinators:

- $\mathsf{beg}, \mathsf{end}$ and ev, for identifying atomic component actions, as characterised by the functions beg, end and ev, introduced in Section 2.
- *Temporal combinators* such as $at_1 \Leftarrow_R at_2$, for identifying those occurrences of at_1 which are in relation R to an occurrence of at_2, and $at_1 \Uparrow_R at_2$, for "merging" occurrences of at_1 and at_2 which are in relation R to each other.

These are not defined here largely for reasons of space. Nevertheless, it would be wrong to claim that their definitions are obvious. Specifically, it is not entirely clear how $at_1 \Uparrow_R at_2$ should be defined. Consider, for example, a sequential combination of actions, which might be represented as $at_1 \Uparrow_{)[} at_2$. A naive interpretation would result in the action whose occurrences were those intervals exactly containing an occurrence of at_1 meeting an occurrence of at_2. However, this might not be an action, for example, if at_1 and at_2 denoted the same action, which had more than one occurrence in sequence. Apart from finding an alternative interpretation, one possible solution would be to distinguish a type of "underdetermined" actions, where such an action is the union of more than one "determined" action, hence may have occurrences which overlap. As this would lead to at least three types of action (atomic, determined, underdetermined) and possibly more,[12] it would perhaps be better to reconstruct the logic to include a sort of actions, with different subsorts, which would allow for greater flexibility when defining combinators and other operators. The use of a sort of actions has been suggested before in eg. [17, 9].

5 prog and occ

In the semantic characterisation of actions presented earlier, we introduced the notion of an action in progress. This can be reflected in the logic, by introducing an operator prog taking an action as argument and resulting in a formula which is true at those points in time when an action is in progress. Formally, extending Definitions 8 and 10:

- $Form(\Theta) \supset \{\mathsf{prog}(at) \mid at \in AcTerm(\Theta)\}$
- $[\![\, \mathsf{prog}(at) \,]\!]^{\Im,A}(t) = true$ iff $t \in \mathrm{prog}([\![\, at \,]\!]^{\Im,A}(t))$

A natural compliment to this is an operator which takes formulae and returns an action. $\mathsf{occ}(\alpha)$ returns the 'action' whose occurrences are the maximal occurrences over which the formula is true, and may be defined as an extension of Definitions 7 and 10:

- $AcTerm(\Theta) \supset \{\mathsf{occ}(\alpha) \mid \alpha \in Form(\Theta)\}$
- $[\![\, \mathsf{occ}(\alpha) \,]\!]^{\Im,A}(t) = \max_{intvls}(\mathrm{intvls}(\{t' \mid [\![\, \alpha \,]\!]^{\Im,A}(t') = true\}))$

Under this interpretation, $\mathsf{prog}(\mathsf{occ}(\alpha)) \leftrightarrow \alpha$ is not an axiom, because $\mathsf{prog}(\mathsf{occ}(\alpha))$ is true only on those time points in the *interior* of an occurrence of $\mathsf{occ}(\alpha)$. The consequences of this on the logic (eg. whether inverses of these operators will be required to provide a complete deduction calculus) still need to be investigated.

[12] For example, [14] discusses a number of different types of action expressible in natural language.

6 Temporal Operators

A number of action logics [16, 18, 7, 15] have deontic operators for prescribing when, in normal circumstances, an action occurs with respect to other actions and with respect to the truth or not of formulae. Deontic operators are used to distinguish between normal and abnormal behaviour,[13] and are of two kinds: *permission* operators, for stipulating when actions may occur, and *obligation* operators for stipulating when actions must occur. Temporal operators can be defined to achieve a similar effect as deontic operators, provided one is willing to lose the normative/non-normative distinction. [6] presents a detailed formal analysis of the relationship between the two views, under the assumption that actions are atomic.

With durative actions, it is possible to define temporal operators which make use of the ability to relate occurrences of actions using interval relations. For example, extending Definitions 8 and 10:[14]

- $Form(\Theta) \supset \{(at_1 \supset_R at_2) \mid at_1, at_2 \in AcTerm(\Theta)\}$, where R is a relation on $I \times I$
- $[\![(at_1 \supset_R at_2)]\!]^{\Im,A}(t) = true$ iff for every $i \in [\![at_1]\!]^{\Im,A}(t)$, if $\mathrm{l}(i) = t$ then there exists $j \in [\![at_2]\!]^{\Im,A}(t)$ s.t. iRj

This family of operators could be further generalised by specifiying the number of occurrences of at_2 that must be in relation R to at_1, for example as in the formula $(at_1 \supset_{R,n} at_2)$ where n is a term denoting a natural number.

A problem[15] with this definition is that the temporal operators may not be iterated, as they are binary operators on action terms which result in formulae. It is possible that the problem could be circumvented by the introduction and judicious use of 'temporal' action combinators. A more radical approach, would be to interpret formulae over intervals of time, thereby allowing action terms to be treated as formulae, and define interval temporal operators such as those in [20]; however, this moves further away from the action logic style of expressing behaviour. The ideal solution might be to adopt a dual logic approach – an interval logic and a logic with action combinitors and modalities, formally relating theories expressed in the two logics by using the same interpretation structures and signatures.

7 Summary and Further Work

We have considered how a modal action logic might be extended to include durative actions. A logic was developed by reinterpreting existing operators, based on the assumption that actions occur over intervals rather than atomic transitions, and by introducing new 'temporal' operators. Some improvements and possible extensions were also indicated:

[13] If a specification does not meet its deontic constraints, then it is viewed as performing abnormally and some form of error recovery will need to be instigated. See eg. [12, 15] for further discussion.

[14] This is an extends the definition of the 'subsumption' operator ($\supset$) of [7], where $at \supset bt$ is true at a point in time provided that if at is observed by the next transition, bt is also.

[15] For example, suppose we want to express a bounded obligation (see [15]), where an action at must be performed during the next future period during which the bounding formula α is true. Without iterating temporal operators, we are only able to express that at must be performed within a period starting at the current time during which the formula α is true – ie. by $\mathrm{occ}(\alpha) \supset_{\supseteq} at$.

- A stable set of combinators, in particular 'temporal' combinators, needs to be defined. As discussed in Section 4, this would be related to the introduction of a *sort* of actions.
- A dual logic approach – an interval logic and a logic with action combinitors and modalities – could be adopted. In particular, the interval logic would allow temporal operators to be iterated. The two logics could be formally related by using the same interpretation structures and signatures.
- A linear time structure has been supposed throughout the paper. Preliminary investigation reveals that a branching time structure could be used to distinguish between the atomic transitions when an action stops and those when it completes as expected (ends).[16]
- Deontic operators could be introduced to make the distinction between normal and abnormal behaviour. By combining these with temporal operators, it should then be possible to express mixed temporal/deontic constraints, such as those described in [15].

As well as changes to the logic, there is still much work to be done for the logic to be a useful tool for formal specification. The suitability of the logic in interpreting OO specification languages and for reasoning about specifications and their developments needs to be tested. In this respect, we are considering using the logic to give an axiomatic semantics to an object oriented extension of VDM [1]. In particular, one problem that will need to be addressed is the handling of different granularities of time, both between objects at one level of specification and between different levels of a specification. Finally, we believe it important to develop tools to support the specification process, in particular the substantial proof effort that is involved. To this end, we are in the considering instantiating proof tools, such as the Mural proof assistant [11], with various action logics.

Acknowledgments

Thanks to José Fiadeiro, Stephen Goldsack, Tom Maibaum, Dimitrios Raptis and Mark Ryan for useful comments and discussion. This work was partially supported by ESPRIT Project 6500, Afrodite.

References

[1] Afrodite Language Group. *Language Reference Manual.* Technical Report R.1., ESPRIT Project 6500 Afrodite, CAP Gemini Innovation, Holland, 1993.

[2] J.F. Allen. Towards a General Theory of Action and Time. In *Journal of Artificial Intelligence*, Vol. 23, 1983.

[3] P. America & J. Rutten. A Parallel Object-Oriented Language: Design and Semantic Foundations. PhD thesis, Free University of Amsterdam, 1989.

[4] C. Atkinson, S.J. Goldsack, A. Di Maio & R. Bayan. Object-Oriented Concurrency & Distribution in DRAGOON. In *Journal of Object-oriented Programming*, March 91.

[5] M. Celiktin & A. Lamsweerde. *Specification of Time Constraints with Interval Sequences.* Research Report RR-92-46, Unité d'Informatique, Faculté des Sciences Appliquées, Université Catholique de Louvain, 1992.

[16] See [14] for a detailed discussion of this distinction.

[6] J. Fiadeiro & T.S.E. Maibaum. Temporal Reasoning over Deontic Specifications. In *Journal of Logic and Computation 1(3)*, pp357-395, 1991.

[7] J. Fiadeiro & T.S.E. Maibaum. Towards Object Calculi. In *Information Systems Correctness and Reusability (Selected Papers)*, G. Saake & A. Senadas (eds.), IS-CORE 91 Workshop, 1991.

[8] J. Fiadeiro, C. Sernadas, T.S.E. Maibaum & G. Saake. Proof-Theoretic Semantics of Object-Oriented Specification Constructs. In *Object-oriented Databases: Analysis Design and Construction*, R. Meersman & W. Kent (eds), North Holland, 1991.

[9] J. Fiadeiro, C. Sernadas, T.S.E. Maibaum & A. Sernadas. Describing and Structuring Objects for Conceptual Schema Development. In *Conceptual Modelling, Databases and CASE*, P. Loucopoulos & R. Zicari (eds), John Wiley, 1992.

[10] C.B. Jones. *Systematic Software Development using VDM (second edition)*. Prentice Hall, 1990.

[11] C.B. Jones, K.D. Jones, P.A. Lindsay & R. Moore. *Mural: A Formal Development Support System*. Springer-Verlag, 1991.

[12] A. Jones and M. Sergot. On the role of Deontic Logic in the Characterization of Normative Systems. In *Proc. First International Workshop on Deontic Logic in Computer Science - DEON 91*, Springer-Verlag, 1991.

[13] S.J.H. Kent. *A Deduction Calculus for Modal Action Logic with Action Combinators*. FOREST Research Deliverable Report WP3.R2, Imperial College, London, 1991.

[14] S.J.H. Kent. *Modelling Events from Natural Language*. PhD Thesis, Dept. of Computing, Imperial College, London, to be submitted July, 1993.

[15] S.J.H. Kent, T.S.E. Maibaum & W.J. Quirk. Formally Specifying Temporal Constraints and Error Recovery. In Proc. of IEEE First Int. Symposium on Requirements Engineering, San Diego, 1993.

[16] T.S.E. Maibaum. *A Logic for the Formal Requirements Specification of Real-Time, Embedded Systems*. Alvey FOREST Report R3, Imperial College, London, 1986.

[17] J. Meyer & R.J. Wieringa. *Actors, Actions, and Initiative in Normative System Specification*. Technical report no. IR-257, Faculteit der Wiskunde en Informatica, Free University of Amsterdam, 1991.

[18] M. Ryan, J. Fiadeiro & T.S.E. Maibaum. Sharing Actions and Attributes in Modal Action Logic. In *Proceedings of the International Conference on Theoretical Aspects of Computer Science (TACS 91)*, T. Ito & A. Meyer (eds), Springer Verlag, 1991.

[19] J. van Benthem. *The Logic of Time*. Reidel, Dordrecht, 1983.

[20] Y. Venema. Expressiveness and Completeness of an Interval Tense Logic. In *Notre Dame Journal of Formal Logic*, Vol. 31, 1990.