A Modal Logic of Durative Actions(*)

I.Nunes
J.L.Fiadeiro
Department of Informatics
Faculty of Sciences, University of Lisbon
Campo Grande
1700 Lisboa, Portugal
{in,llf}@di.fc.ul.pt

W.M.Turski
Institute of Informatics
Warsaw University
ul. Banacha 2
02-097 Warsaw, Poland
wmt@mimuw.edu.pl

Having in mind the development of formal support for programming in transactional and multiprocessor systems, a modal logic is presented which is based on durative actions, i.e. actions for which the state in which they finish executing is not necessarily the same in which they started. For this purpose, dyadic action modalities are introduced which allow us to relate three states: the launching, the accepting and the resulting states of every action execution. The characterisation of the new modalities is given both through an extension of the traditional notion of transition system and an axiomatisation that is proved to be complete for the class of extended transition systems. Finally, we show how the proposed logic can be used to express and verify properties of specifications in the new behavioural paradigm introduced by W.Turski.

1. Introduction

Traditionally, modal logics have been based on actions that are *atomic* in two senses: (1) they execute without any interference from the environment, i.e. the results that are produced by the execution of an action are a function of the starting state; (2) they have no duration, i.e. the state of the system does not change while an action is executing. These two kinds of atomicity have led to models of system behaviour based on transition systems where actions are modelled as functions (or relations in the case of non-deterministic actions) on a set of worlds, and global system behaviour is modelled through sequences of worlds (runs).

In this paper, we extend this model by relaxing the second assumption of atomicity. That is to say, while assuming that actions compute their effects without interference from the environment by executing on a local copy of the state, we develop a logic in which actions can have duration, i.e. we admit that the global state of the system can change while an action is executing. Notice that we use the term durative not to

(*) This work was partially supported by JNICT through contracts PRAXIS XXI 2/2.1/MAT/46/94 (ESCOLA), PCSH/OGE/1038/95 (MAGO), PRAXIS XXI 2/2.1/TIT/1662/95 (SARA) and PCEX/P/MAT/46/96 (ACL).

express the real-time duration of action execution but, instead, the fact that the system state can change while actions execute.

The motivation for this generalisation is twofold.

On the one hand, several application domains, namely in Artificial Intelligence (e.g. for agent-based programming) and Databases (e.g. for transaction management), require logical support for actions that are durative.

On the other hand, the continuous development of multiprocessor and distributed systems give us the means for truly parallel and concurrent process execution, which prompts the need for new methods and supporting logics.

A new paradigm of programming that is representative of such extended use of multiprocessing capabilities has been recently proposed by W.Turski. A behaviour specification [9] is a collection of actions which describe the reactions to the possible states of the system and such that:

- associated with each action there are two conditions – the pre-guard (desirability condition) and the post-guard (acceptability condition) – which define the system states in which an action should be undertaken and accepted, respectively;
- actions are performed by agents, of which there is an unspecified number, that are equipped with memories capable of storing a copy of the system state. Agents can copy a state into memory – loading – and copy values from memory to system objects – discharging;
- associated with each action there is a loading list which consists of the objects whose values are to be copied into the agent's memory when the action is launched, and a discharging list which consists of the objects of the system which are to be updated at a discharge with the values of the identically named objects in the agent's memory;
- loading and discharging are instantaneous; action execution is not;
- when the system is in a state that satisfies the pre-guard of an action *a*:
  - this action is assigned to an unoccupied agent, which loads to its memory the values of the objects in the loading list of *a* and starts execution;
  - the agent performs action *a* in its memory, without interference from the environment;
  - when the execution of action *a* terminates and if the post-guard of *a* is satisfied by the system state, then the agent discharges the values of the objects referred in the discharging list of *a* into the system, updating it. Testing the post-guard and discharging the local state is performed atomically *(test and set)*.

As an example the following is the specification of the "dining philosophers" (c.f. [8]) as seen from an individual philosopher.

$$
\begin{aligned}
(\neg H, \neg H) &\rightarrow [\ \ |\ \text{think}\ |\ H] \\
(H \wedge PR \wedge PL,\ H \wedge PR \wedge PL) &\rightarrow [\ \ |\ \text{eat}\ |\ H, PR, ROT, PL, LOT] \\
(H \wedge PL \wedge \neg PR \wedge \neg ROT,\ \neg ROT) &\rightarrow [\ \ |\ \text{rel_L}\ |\ PL, LOT] \\
(H \wedge PR \wedge \neg PL \wedge \neg LOT,\ \neg LOT) &\rightarrow [\ \ |\ \text{rel_R}\ |\ PR, ROT] \\
(H \wedge LOT,\ H \wedge LOT) &\rightarrow [\ \ |\ \text{take_L}\ |\ PL, LOT] \\
(H \wedge ROT,\ H \wedge ROT) &\rightarrow [\ \ |\ \text{take_R}\ |\ PR, ROT]
\end{aligned}
$$

where

H – the philosopher is hungry

LOT – the left fork is on the table

ROT – the right fork is on the table

PL – the philosopher possesses the left fork

PR – the philosopher possesses the right fork

Each *dga* takes the form

(pre-guard,postguard) → [loading | program | discharging]

In this example all loading lists are empty. This means that the computations performed locally by each action do not depend on the state in which the action is launched. The actual transformations performed by the actions are specified separately.

The intended semantics of the *rel_R* action specification, for example, is the following. When a philosopher is hungry (H), possesses the right fork (PR) and does not possess the left fork (¬PL), and the left fork is not on the table (¬LOT), he will release the right fork (the action is launched because the pre-guard is true). This is so specified in order to prevent deadlock. If, while our philosopher is in the process of releasing its right fork, the philosopher at his left releases his right fork (our philosopher's left fork), then our philosopher should not bring about the action. This is expressed through the post-guard: if the left fork is on the table (LOT) when the philosopher finishes executing the release, the action is rejected.

Programming in this new paradigm requires logical support that cannot be adequately given using current modal logics. Whereas action execution in the behavioural paradigm involves three states – the launching, accepting and resulting states – the action modalities that are currently available allow us to relate only two states – the launching and the resulting state. Indeed, in the absence of state changes during action execution, the launching and accepting states are the same, thus reducing the characterisation of action execution to two states.

Hence, the main challenge raised by durative actions, and one of the main contributions of the paper, is the characterisation of action modalities that relate three

states. This characterisation is given both through an extension of the traditional notion of transition system and an axiomatisation that is proved to be complete for the class of extended transition systems. Due to the limitations on the size of the paper, we shall concentrate on the proof-theoretic aspects of the logic, namely the completeness proof and the way the new action modalities characterise the behavioural paradigm. Details on the underlying computational model and model-theory can be found in [8,9] and [5,6], respectively.

2. Syntax and semantics of DTL

Labelled transition systems are usually defined (e.g. as in [4] and [7]) in terms of pairs $(\Sigma,\{\xrightarrow{a} | a\in \mathcal{A}\})$ where Σ is a non-empty set (of states), $\mathcal{A}$ is a non-empty set (of actions/labels) and, for each $a\in \mathcal{A}$, $\xrightarrow{a}\subset\Sigma\times\Sigma$ is the transition relation. Given $\sigma,\sigma'\in \Sigma$, $\sigma\xrightarrow{a}\sigma'$ means that action *a* takes the system from state σ to state σ'. If we are modelling systems in which actions are deterministic, each $\xrightarrow{a}$ is a partial function.

In this context, each transition within the system is completely defined by an action and two states, the launching and the resulting ones. When we consider durative actions, we are faced with the possibility of state changes occurring during action execution due to the effects of other agents executing concurrently. Hence, the traditional notion of transition system is no longer adequate.

In order to provide a model that takes action duration into account, we present what we call durative transition systems.

Definition 2.1: A durative transition system is a pair $(\Sigma,\{\xrightarrow{\!\!\!

where $p \in \mathcal{P}$ and $a \in \mathcal{A}$. ♦

The language $\mathcal{L}(\mathcal{P},\mathcal{A})$ is interpreted on durative transition systems as follows:

Definition 2.3: An interpretation structure I for a language $\mathcal{L}(\mathcal{P},\mathcal{A})$ is a pair $(\mathcal{D},\mathcal{V})$ where $\mathcal{D} = (\Sigma, \{\xrightarrow{a} | a \in \mathcal{A}\})$ is a durative transition system and $\mathcal{V}: \Sigma \to 2^{\mathcal{P}}$, the valuation function, maps each state $\sigma \in \Sigma$ to a subset of $\mathcal{P}$ (the subset that consists of the atomic sentences that are defined to be true at σ). We say that I is an interpretation structure based on $\mathcal{D}$. ♦

Definition 2.4: Given an interpretation structure $I=(\mathcal{D},\mathcal{V})$ for a language $\mathcal{L}(\mathcal{P},\mathcal{A})$ we define the satisfaction relation $\models_I \subseteq \Sigma \times \mathcal{L}(\mathcal{P},\mathcal{A})$ as follows (where $\sigma \in \Sigma$ and $p \in \mathcal{P}$):

$\sigma \models_I p$ iff $p \in \mathcal{V}(\sigma)$

$\sigma \models_I \neg A$ iff $\sigma \not\models_I A$

$\sigma \models_I A_1 \supset A_2$ iff $\sigma \models_I A_1$ implies $\sigma \models_I A_2$

$\sigma \models_I A_1[a]A_2$ iff for every $\sigma',\sigma'' \in \Sigma$,
$\sigma \xrightarrow{a} (\sigma' \to \sigma'')$ and $\sigma' \models_I A_1$ implies $\sigma'' \models_I A_2$

$\sigma \models_I A_1[\overline{a}]A_2$ iff for every $\sigma',\sigma'' \in \Sigma$,
$\sigma' \xrightarrow{a} (\sigma \to \sigma'')$ and $\sigma' \models_I A_1$ implies $\sigma'' \models_I A_2$

We further say that A is *true* in I, ($\models_I A$), iff $\sigma \models_I A$ for all states $\sigma \in \Sigma$. ♦

We will use the traditional propositional connectives as abbreviations, together with the "possibility" operators $\langle a \rangle$ and $\langle \overline{a} \rangle$ defined as follows:

$A_1 \langle a \rangle A_2$ is an abbreviation for $\neg(A_1[a]\neg A_2)$

$A_1 \langle \overline{a} \rangle A_2$ is an abbreviation for $\neg(A_1[\overline{a}]\neg A_2)$

It can easily be checked that

$\sigma \models_I A_1 \langle a \rangle A_2$ iff there is $\sigma',\sigma'' \in \Sigma$, $\sigma \xrightarrow{a} (\sigma' \to \sigma'')$ and $\sigma' \models_I A_1$ and $\sigma'' \models_I A_2$

$\sigma \models_I A_1 \langle \overline{a} \rangle A_2$ iff there is $\sigma',\sigma'' \in \Sigma$, $\sigma' \xrightarrow{a} (\sigma \to \sigma'')$ and $\sigma' \models_I A_1$ and $\sigma'' \models_I A_2$

Notice how the new modal operators allow us to refer to the three critical moments of the life of a given action *a*: the sentence $(A_1 \supset A_2[a]A_3)$ is true if whenever *a* is launched in a state satisfying A_1 and accepted in a state *refusing* A_2, the resulting state satisfies A_3.

In particular, the sentence $(ff\langle a \rangle tt)$ allows us to identify *launching* states for action *a*. Hence, for instance, the fact that all states satisfying A are launching states for action *a* corresponds to the truth of $(A \supset ff\langle a \rangle tt)$.

The first argument of the modal operator is negated for technical reasons: it simplifies the presentation of the axiomatisation. From an intuitive point of view, it corresponds to a condition for the *rejection* of the effects of the action.

The operator $\overline{[a]}$ is a kind of reverse of [a] in the sense that it points "backwards" in the transition relation [7]: the accepting state is the state of evaluation of the formula $(A_1 \supset A_2\overline{[a]}A_3)$ and the launching state is the state where A_2 is evaluated.

For instance, the sentence $(ff\overline{<a>}tt)$ allows us to identify *accepting* states for action *a*. Hence, the fact that all accepting states for *a* satisfy property A corresponds to the truth of $(ff\overline{<a>}tt \supset A)$.

In section 4, we shall analyse other kind of properties that are of interest in the context of the behavioural paradigm.

Definition 2.5: Given a set $\mathcal{M}$ of interpretation structures, an interpretation I, a sentence $A \in \mathcal{L}(\mathcal{P}, \mathcal{A})$ and a set Γ of sentences, we say that :

– *A is true in* $\mathcal{M}$, $\models_{\mathcal{M}} A$, iff for all $I \in \mathcal{M}$, $\models_I A$;

– *Γ entails A in I*, $(\Gamma \models_I A)$ iff for all states $\sigma \in \Sigma$, if $\sigma \models_I \Gamma$ then $\sigma \models_I A$;

– *Γ entails A in* $\mathcal{M}$, $(\Gamma \models_{\mathcal{M}} A)$ iff

for all $I \in \mathcal{M}$ and for all $\sigma \in \Sigma$, if $\sigma \models_I \Gamma$ then $\sigma \models_I A$. ♦

Notice that we have defined a *local* semantic consequence relation [7].

3. A complete and decidable axiomatisation

In this section, we present an axiomatisation for the proposed operators and generalise the typical canonical model style of completeness proof for unary operators to the binary case.

Definition 3.1: The logic DTL over the language $\mathcal{L}(\mathcal{P},\mathcal{A})$ is given by the following proof system:

Axioms:

GA1.	Any instance of a propositional tautology
GKR.	$A_1\ [a]\ (A_2 \supset A_3) \supset (A_1[a]A_2 \supset A_1[a]A_3)$
G$\overline{\text{K}}$R.	$A_1\ \overline{[a]}\ (A_2 \supset A_3) \supset (A_1\overline{[a]}A_2 \supset A_1\overline{[a]}A_3)$
GKL.	$(A_1 \supset A_2)\ [a]\ A_3 \supset (A_1[a]A_3 \supset A_2[a]A_3)$
G$\overline{\text{K}}$L.	$(A_1 \supset A_2)\ \overline{[a]}\ A_3 \supset (A_1\overline{[a]}A_3 \supset A_2\overline{[a]}A_3)$

Rules:

MP.	if $\vdash A$ and $\vdash A \supset B$ then $\vdash B$
GNecR.	if $\vdash A$ then $\vdash B\ [a]\ A$
GNecL.	if $\vdash A$ then $\vdash A\ [a]\ B$
GR1.	$\vdash A_1 \supset (A_2[a]A_3)$ iff $\vdash \neg A_2 \supset (\neg A_1\overline{[a]}A_3)$ ♦

The axiom schemes express a principle of distributivity of the modal operators with respect to the conditional. They are a generalisation of the axiom K of normal modal logics. The first two, GKR and G$\overline{\text{K}}$R, express distributivity in what concerns post-conditions. Axiom GKL gives distributivity over rejection conditions, while G$\overline{\text{K}}$L gives distributivity over pre-conditions.

The rule of necessitation "at the right", GNecR, states that a theorem is necessarily true at all resulting states. The rule of necessitation "at the left", GNecL, states that a theorem used as a rejection condition for an action, leads to no accepting state and, hence, no transition. Rule GR1 establishes the relationship between each modal operator and its "reverse": it exchanges between the roles of the launching and accepting states.

We are able to derive within DTL several rules and theorems that are expected to hold when we are dealing with modal operators, such as:

$(A_1[a]A_2 \vee A_1[a]A_3) \supset A_1\ [a]\ (A_2 \vee A_3)$

$(A_1[a]A_3 \vee A_2[a]A_3) \supset (A_1 \vee A_2)\ [a]\ A_3$

$(A_1[a]A_2 \wedge A_1[a]A_3) \equiv A_1\ [a]\ (A_2 \wedge A_3)$

$(A_1[a]A_3 \wedge A_2[a]A_3) \equiv (A_1 \wedge A_2)\ [a]\ A_3$

$(A_1[a]A_2 \wedge A_1\langle a\rangle A_3) \supset A_1\ \langle a\rangle\ (A_2 \wedge A_3)$

if $\vdash (A_1 \wedge ... \wedge A_n) \supset A$ then $\vdash (B[a]A_1 \wedge .... \wedge B[a]A_n) \supset B[a]A$ (GRkR)

if $\vdash (A_1 \wedge ... \wedge A_n) \supset A$ then $\vdash (A_1[a]B \wedge .... \wedge A_n[a]B) \supset A[a]B$ (GRkL)

and also the corresponding ones for the operator $\overline{[a]}$.

We will now establish soundness and completeness of the axiomatic system just given with respect to the class $\mathcal{M}$ of interpretation structures as defined in 2.3.

Proposition 3.2: The axiomatic system defined in 3.1 is sound with respect to the class $\mathcal{M}$ of interpretation structures, that is, if $\Gamma \subseteq \mathcal{L}(\mathcal{P},\mathcal{A})$ then

if $\Gamma \vdash A$ then $\Gamma \models_{\mathcal{M}} A$ ♦

Completeness is proved using canonical models. For simplicity, we consider the case of the "forward" operators [a] only.

The definition used for canonical models extends the one that is typically found in the literature for unary modal operators.

Definition 3.3: The canonical model for DTL is the structure

$I_{DTL} = (\Sigma_{DTL}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, \mathcal{V}_{DTL})$

where:

a) $\Sigma_{DTL} = \{\sigma \subseteq \mathcal{L}(\mathcal{P},\mathcal{A}) \mid \sigma \text{ is maximal DTL-consistent}\}$

b) $\sigma \xrightarrow{a} (\sigma' \rightarrow \sigma'')$ iff, for every $B \notin \sigma'$, $\{A \mid (B[a]A) \in \sigma\} \subseteq \sigma''$

c) $\mathcal{V}_{DTL}(\sigma) = \{p \in \mathcal{P} \mid p \in \sigma\}$, for $\sigma \in \Sigma_{DTL}$ ♦

Notice that this is a good definition because, DTL being sound, Σ_{DTL} is not empty.

Proposition 3.4: Durative transitions in a canonical model satisfy,

$\sigma \xrightarrow{a} (\sigma' \to \sigma'')$ iff, for every $B \notin \sigma'$, $\{(B\langle g\rangle A) | A \in \sigma''\} \subseteq \sigma$ ♦

Theorem 3.5: The axiomatization given in 3.1 is complete with respect to class $\mathcal{M}$ of interpretation structures, that is,

if $\models_{\mathcal{M}} A$ then $\vdash A$

It is also strongly complete, that is, if $\Gamma \subseteq \mathcal{L}(\mathcal{P},\mathcal{A})$ then

if $\Gamma \models_{\mathcal{M}} A$ then $\Gamma \vdash A$ ♦

Proof. The proof relies on a few lemmas. Let $A,B \in \mathcal{L}(\mathcal{P},\mathcal{A})$ and $\sigma \in \Sigma_{DTL}$.

P1. $A[a]B \in \sigma$ iff for every $\sigma',\sigma'' \in \Sigma_{DTL}$, $\sigma \xrightarrow{a} (\sigma' \to \sigma'')$ and $A \notin \sigma'$ implies $B \in \sigma''$

P2. $\sigma \models_{I_{DTL}} A$ iff $A \in \sigma$

P3. $\models_{I_{DTL}} A$ implies $\vdash A$

P4. $\Gamma \models_{I_{DTL}} A$ implies $\Gamma \vdash A$

We prove completeness of DTL, as usual, by using P3: $\models_{\mathcal{M}} A$ implies $\models_{I_{DTL}} A$ which, by P3, implies $\vdash A$.

For proving strong completeness we can argue in a similar way using P4. ♦

The only lemma that is really interesting, because it is new in that it has to deal with the dyadic operators, is P1. Hence, we will sketch its proof:

Lemma 3.6: Given $A,B \in \mathcal{L}(\mathcal{P},\mathcal{A})$ and $\sigma \in \Sigma_{DTL}$, $A[a]B \in \sigma$ iff, for every $\sigma',\sigma'' \in \Sigma_{DTL}$, $\sigma \xrightarrow{a} (\sigma' \to \sigma'')$ and $A \notin \sigma'$ implies $B \in \sigma''$.

Proof:

$\boxed{\Rightarrow}$ Suppose $A[a]B \in \sigma$ and take arbitrary $\sigma',\sigma'' \in \Sigma_{DTL}$. Suppose also that $\sigma \xrightarrow{a} (\sigma' \to \sigma'')$ and $A \notin \sigma'$. By the definition of canonical model we have

$\sigma \xrightarrow{a} (\sigma' \to \sigma'')$ iff, for every $D \notin \sigma'$, $\{C | (D[a]C) \in \sigma\} \subseteq \sigma''$

Because $A \notin \sigma'$ and $A[a]B \in \sigma$, it follows that $B \in \sigma''$

$\boxed{\Leftarrow}$ We have to prove that

$\forall \sigma',\sigma'' \in \Sigma_{DTL}.(\sigma \xrightarrow{a} (\sigma' \to \sigma'') \wedge A \notin \sigma' \Rightarrow B \in \sigma'') \Rightarrow A[a]B \in \sigma$

That is to say

$\forall \sigma'' \in \Sigma_{DTL}.(\exists \sigma' \in \Sigma_{DTL}.(\sigma \xrightarrow{a} (\sigma' \to \sigma'') \wedge A \notin \sigma') \Rightarrow B \in \sigma'') \Rightarrow A[a]B \in \sigma$

Suppose then, that

$\forall \sigma'' \in \Sigma_{DTL}.(\exists \sigma' \in \Sigma_{DTL}.(\sigma \xrightarrow{a} (\sigma' \to \sigma'') \wedge A \notin \sigma') \Rightarrow B \in \sigma'')$

By lemma A.1 (in the Appendix),

$\forall \sigma'' \in \Sigma_{DTL}.(\{C | (A[a]C) \in \sigma\} \subseteq \sigma'' \Rightarrow \exists \sigma' \in \Sigma_{DTL}.(\sigma \xrightarrow{a} (\sigma' \to \sigma'') \wedge A \notin \sigma'))$

Then, we have that

$\forall\sigma'' \in \Sigma_{DTL}.(\{C|(A[a]C) \in \sigma\} \subseteq \sigma'' \Rightarrow B \in \sigma'')$

which is equivalent to

$\{C|(A[a]C) \in \sigma\} \vdash B$

and which implies that there are $A_1, \ldots, A_m \in \{C|(A[a]C) \in \sigma\}$ s.t. $\vdash (A_1 \wedge \ldots \wedge A_m) \supset B$.

If m=0, then B is a theorem of DTL which, through GNecR, implies that A[a]B is a theorem. Because σ is maximal DTL-consistent, $A[a]B \in \sigma$.

If $m \neq 0$, then we can infer $\vdash (A[a]A_1 \wedge .... \wedge A[a]A_m) \supset A[a]B$.
Because $A_1, \ldots, A_m \in \{C|(A[a]C) \in \sigma\}$, we can conclude that $A[a]A_1, \ldots, A[a]A_m \in \sigma$ which, because σ is maximal DTL-consistent, implies $A[a]B \in \sigma$. ♦

We can further prove:

Proposition 3.7: The DTL-system is decidable. ♦

This result is proved by showing that DTL has the finite model property: if $\not\vdash A$ then there is a finite model I s.t. $\not\models_I A$, that is, every non-theorem of DTL is false in some finite model of DTL. The finite model property itself is proved by a straightforward adaptation of filtration techniques.

4. Application to the behavioural paradigm

In this section, we show how the proposed logic can be put to use in expressing and verifying properties of behavioural specifications.

4.1. Locality and frame restrictions

As motivated in the introduction, actions in the behavioural paradigm have two associated sets of propositional symbols, which account for the "attributes" or "system variables" that the actions need to *read* and *write*, respectively. The first set consists of the attributes that are loaded to the local memory of any agent that engages in executing an action – loading list. The second consists of the attributes that are discharged from the local state of the agent into the global state of the system – discharging list. The discharging list is particularly important in determining the effect of an action in the state of the system: attributes of the discharging list get the values computed locally by the executing agent (locality condition); attributes not in the discharging list remain unchanged, i.e. get the same values that they had in the accepting state (frame condition).

These restrictions can be brought into our logic by assigning, to every action $a \in \mathcal{A}$, two sets of propositional symbols – L_a and D_a – accounting for the loading and discharging lists, respectively.

In order to formalise the locality and frame restrictions over durative transition systems, we need the following auxiliary definitions:

Definition 4.1: Given a set $\Omega \subseteq \mathcal{P}$ of propositional symbols and an interpretation structure $I=(\Sigma,\{\xrightarrow{a} | a \in \mathcal{A}\},\mathcal{V})$, we say that two states σ and σ' are Ω-equivalent – written $\sigma \underset{\Omega}{\equiv} \sigma'$ – iff the values of all symbols in Ω are the same in both states:

$$\sigma \underset{\Omega}{\equiv} \sigma' \text{ iff } \mathcal{V}(\sigma) \cap \Omega = \mathcal{V}(\sigma') \cap \Omega \quad \blacklozenge$$

Definition 4.2: Let $\Omega \subseteq \mathcal{P}$ and $\Gamma, \Gamma' \subseteq \mathcal{L}(\mathcal{P},\mathcal{A})$. We say that Γ Ω-implies Γ' – written $\Gamma \underset{\Omega}{\supseteq} \Gamma'$ – iff all states where Γ holds have an Ω-equivalent state where Γ' holds,

$\Gamma \underset{\Omega}{\supseteq} \Gamma'$ iff for every I and σ, $\sigma \models_I \Gamma$ implies that there is σ' s.t. $\sigma \underset{\Omega}{\equiv} \sigma'$ and $\sigma' \models_I \Gamma'$ $\blacklozenge$

Definition 4.3: An interpretation structure is said to be *behavioural* iff:

a) For every state $\sigma_1,\sigma_2,\sigma_1',\sigma_2',\sigma_1'',\sigma_2''$ and action a,

$$(\sigma_1 \xrightarrow{a} (\sigma_1' \rightarrow \sigma_1'') \text{ and } \sigma_2 \xrightarrow{a} (\sigma_2' \rightarrow \sigma_2'') \text{ and } \sigma_1 \underset{L_a}{\equiv} \sigma_2 \text{ imply } \sigma_1'' \underset{D_a}{\equiv} \sigma_2'')$$

b) For every σ, σ', σ'', and action a, $(\sigma \xrightarrow{a} (\sigma' \rightarrow \sigma'')$ implies $\sigma' \underset{\mathcal{P} \setminus D_a}{\equiv} \sigma'')$. $\blacklozenge$

Restriction a) says that the values of attributes in the discharging list (D_a) in a resulting state of a transition only depend on the values that the attributes in the loading list (the ones in L_a) had in the corresponding launching state. This restriction reflects the notion of local execution of actions. We are modelling deterministic actions, which is the model adopted in the behavioural paradigm.

Restriction b) says that the values for attributes other than the ones in the discharging list only depend on the accepting state σ', not on the launching state. This restriction reflects the fact that actions are durative. It is called the *frame* condition because it states that attributes outside the discharging list remain unchanged.

Notice that we are modelling deterministic actions, which is the model adopted in the behavioural paradigm: a durative transition is a function because, given a launching and an accepting states, the resulting state is unique, and it is partial because not all states are necessarily launching states, and not every state is necessarily an accepting state for a given action *a*.

Definition 4.4: The logic DTL^{++} is obtained from DTL through the addition of the axioms

GSy. $\neg P[a]P$ where P is p or $\neg$p for p$\notin D_a$

and the rules

GLo. $A \vdash ff\langle a \rangle P$ implies $A' \vdash ff[a]P$ where $A' \underset{L_a}{\supseteq} A$ and P is

p or $\neg$p for p$\in D_a$ ♦

Proposition 4.5: DTL^{++} is sound and complete with respect to the class of behavioural interpretation structures. ♦

Proof:

We will only prove completeness here. We need to prove that the canonical model for DTL^{++} is behavioural, that is, that it satisfies a) and b) of def. 4.3. We will only prove that it satisfies a) here.

Consider states $\sigma_1,\sigma_2,\sigma_1',\sigma_2',\sigma_1'',\sigma_2'' \in \Sigma_{DTL}$ such that

1. $\sigma_1 \xrightarrow{a} (\sigma_1' \rightarrow \sigma_1'')$
2. $\sigma_2 \xrightarrow{a} (\sigma_2' \rightarrow \sigma_2'')$
3. $\sigma_1 \underset{L_a}{\equiv} \sigma_2$

So, we want to prove that $\sigma_1'' \underset{D_a}{\equiv} \sigma_2''$, that is, $\mathcal{V}_{DTL}(\sigma_1'') \cap D_a = \mathcal{V}_{DTL}(\sigma_2'') \cap D_a$. Because σ_1'', σ_2'' are maximal DTL-consistent, then we have $\sigma_1'' \models_{I_{DTL}} A$ iff $A \in \sigma_1''$ and $\sigma_2'' \models_{I_{DTL}} A$ iff $A \in \sigma_2''$. By definition, $\sigma \models_{I_{DTL}} p$ iff $p \in \mathcal{V}_{DTL}(\sigma)$. So, we want to prove $\sigma_1'' \cap D_a = \sigma_2'' \cap D_a$.

Now let,

4. $P \in \sigma_1''$ where P is p or $\neg$p for p$\in D_a$

We want to prove $P \in \sigma_2''$.

5. for every $B \notin \sigma_2'$ $\{A \mid (B[g]A) \in \sigma_2\} \subseteq \sigma_2''$ From 2 and Def.3.3
6. for every $B \notin \sigma 1'$ $\{(B\langle g\rangle A) \mid A \in \sigma_1''\} \subseteq \sigma_1$ From 1 and Prop.3.4
7. $(ff\langle g\rangle P) \in \sigma_1$ From 4 and 6
8. $\sigma_1 \models_{I_{DTL}} ff\langle g\rangle P$ From 7 and maximal DTL-consistency of σ_1
9. $\sigma_1 \vdash ff\langle g\rangle P$ By strong completeness
10. $\sigma_2 \models \sigma_2$ and $\sigma_1 \models \sigma_1$
11. $\sigma_2 \underset{L_a}{\supseteq} \sigma_1$ From 3, 10 and Def. 4.2
12. $\sigma_2 \vdash ff[g]P$ From 9, 11 and rule Glo
13. $\sigma_2 \models_{I_{DTL}} ff[g]P$ By strong soundness
14. $\sigma_2'' \models_{I_{DTL}} P$ From 2 and 13
15. $P \in \sigma_2''$ Because σ_2'' is maximal DTL-consistent ♦

4.2. Properties of behavioural specifications

A behavioural specification can be identified with a collection of DTL-sentences for a choice of propositional symbols (program "variables" or actions). Indeed, we have seen in section 2 how the new modalities allow us to express properties of the three states that characterise action execution – launching, accepting and resulting states.

We are now interested in typical properties of system behaviour in the new paradigm.

Definition 4.6: Given a set Γ of sentences corresponding to a behavioural specification, we denote by $\vdash_\Gamma$ the extension of DTL^{++} with the sentences of Γ as axioms. ♦

Definition 4.7: Given a behavioural specification Γ, a sentence A expresses a *stability* property of the system iff

$$\vdash_\Gamma \bigwedge_{a\in\mathcal{A}}(\neg A[a]A)$$ ♦

That is to say, all actions accepted in A states, preserve A.

Stability can be seen, as in UNITY[1], as a particular case of *unless* properties:

Definition 4.8: Given a behavioural specification Γ and sentences A and B, the system satisfies the property A *unless* B iff

$$\vdash_\Gamma \bigwedge_{a\in\mathcal{A}}((A\supset B)[a](A\vee B))$$ ♦

That is to say, all actions accepted in states that satisfy A but not B, either preserve A or establish B.

Notice that, because action execution is not atomic, relating the launching state with the resulting state as in UNITY, would not work. That is, defining A *unless* B through

$$\bigwedge_{a\in\mathcal{A}}((A\supset B)\overline{[a]}(A\vee B))$$

would not ensure that, once A is established, either B never holds and A continues to hold forever, or B holds eventually and A continues to hold at least until B holds. This has to do with the possible existence of pending actions launched in states not satisfying A and not B. These pending actions could establish resulting states where neither A nor B hold. If, as proposed, we ensure that all actions *accepted* when A is true and B is false either preserve A or establish B, then, because the "transition" from the accepting to the resulting state is atomic, we have the desired effect.

Example 4.9: A typical safety property expressed through unless properties is, for the dining philosophers presented in section 1, $\neg$PL *unless* H, that is, a philosopher cannot possess his left fork unless he is hungry:

We have to prove that $\bigwedge_{a\in\mathcal{A}}((\neg PL\supset H)[a](\neg PL\vee H))$ is a DTL consequence of the behavioural specification Γ. We will only prove for the rel_L and take_R actions here.

Assume that, from the behavioural specification Γ, we can infer

1. $H\wedge PL\wedge\neg PR\wedge\neg ROT\equiv ff\langle rel_L\rangle tt$ and
2. $H\wedge PL\wedge\neg PR\wedge\neg ROT\vdash_\Gamma ff\langle rel_L\rangle\neg PL$

So,

3. $H\wedge PL\wedge\neg PR\wedge\neg ROT\vdash_\Gamma ff\ [rel_L]\ \neg PL$ from 2 and rule GLo ($PL\in D_{rel_L}$)
4. $H\wedge PL\wedge\neg PR\wedge\neg ROT\supset ff\ [rel_L]\ \neg PL$ From 3

5. $ff\langle rel_L\rangle tt \supset ff\ [rel_L]\ \neg PL$ — From 1, 4
6. $\neg(ff\langle rel_L\rangle tt) \vee ff\ [rel_L]\ \neg PL$ — From 5
7. $ff\ [rel_L]\ ff \vee ff\ [rel_L]\ \neg PL$ — From 6
8. $(ff[rel_L]ff \vee ff[rel_L]\neg PL) \supset ff[rel_L](ff \vee \neg PL)$ — Theorem
9. $ff\ [rel_L]\ (ff \vee \neg PL)$ — From 7, 8, MP
10. $ff\ [rel_L]\ \neg PL$ — From 9
11. $(ff \supset (\neg PL \supset H))[rel_L]\ \neg PL$ — GNecL
12. $((ff \supset (\neg PL \supset H))[rel_L]\neg PL) \supset ((ff[rel_L]\neg PL) \supset (\neg PL \supset H)[rel_L]\neg PL)$ — GKL
13. $((ff[rel_L]\neg PL) \supset (\neg PL \supset H)[rel_L]\neg PL)$ — 11, 12, MP
14. $(\neg PL \supset H)[rel_L]\ \neg PL$ — 10, 13, MP
15. $(\neg PL \supset H)[rel_L]\ (\neg PL \vee H)$ — From 14

In what concerns action take_R:

1. $\neg\neg PL\ [take_R]\neg PL$ — Gsy ($PL \notin D_{take_R}$)
2. $(\neg\neg PL \vee H)[take_R]\neg PL$ — From 1
3. $(\neg PL \supset H)[take_R]\neg PL$ — From 2
4. $(\neg PL \supset H)[take_R](\neg PL \vee H)$ — From 3 ♦

In [5] we present behavioural specifications where the locality conditions are expressed by rules that allow us to infer DTL-formulas from formulas written in the language of *local* logics. These *local* formulas express local behaviour of actions, therefore allowing us to separate computation from coordination concerns.

Stability guarantees that a system never visits states outside a certain set (captured by the sentence). An extreme case is the one in which a system never leaves a certain state. Such states are called *stationary*. We can classify stationary states in two classes:

Alive stationary – states where no action is accepted.

Dead stationary – states where no action is either accepted or desired.

An *Alive* stationary state is so called because actions can continue to be launched although they cannot be accepted. In terms of program termination we cannot say that a program terminates when it reaches such a state, because it still "lives", but we can guarantee that, for all purposes, it will not evolve because no action can be accepted to take it to another state. On the other hand, a *Dead* stationary state corresponds to complete termination of activity.

Definition 4.10: The following predicates capture the notions of stationarity just discussed:

$$Alive \equiv \bigwedge_{a \in \mathcal{A}} (\neg(ff\langle\overline{a}\rangle tt)) \qquad Dead \equiv \bigwedge_{a \in \mathcal{A}} (\neg(ff\langle\overline{a}\rangle tt \wedge \neg(ff\langle a\rangle tt)) \quad \blacklozenge$$

Notice the use, as discussed in section 2, of $ff\langle\overline{a}\rangle tt$ to identify accepting states, and of $ff\langle a\rangle tt$ to identify launching states of action *a*.

When we think about the classical notion of deadlock, we see that *Alive* stationary states can correspond to deadlock situations: actions that are desired (and, hence, launched) cannot be accepted. As an example, the *rel_R* action specification presented in section 1, together with its left version, *rel_L*, which are included in the specification given in [8] for the "dining philosophers" in order to prevent deadlock, cause the predicate *Alive* to be false, that is, the system never deadlocks because in all states there is always an action that can be accepted.

Another class of properties is concerned with the possibility of reaching certain states.

Definition 4.11: Given a behaviour specification Γ and sentences A and B, the system satisfies the property A *reachable from* B iff

$$\vdash_{\Gamma} B \supset \bigvee_{a \in \mathcal{A}} (ff\langle a\rangle A \vee ff\langle\overline{a}\rangle A) \quad \blacklozenge$$

That is to say, A is reachable from B if any state satisfying B is either a launching state of some action which causes an A-resulting state, or the accepting state of some action that causes an A-resulting state.

Reachability can be used to express properties similar to *ensures* and *leads-to* properties of UNITY [1].

5. Concluding remarks

We presented a modal logic supporting the behavioural paradigm introduced in [8] in which actions execute atomically but are durative, i.e. the global state of the system can change while they are executing. The proposed logic is based on an extended notion of transition system in which action execution is characterised not by two but three states – the state in which actions are launched, the state in which they are considered for acceptance, and the resulting state. The corresponding modal operator is, accordingly, dyadic. An axiomatisation was proposed for the new class of operators. This axiomatisation was proved to be complete by adapting from the usual proofs for unary operators based on canonical models. Finally, the logic was applied to the behavioural paradigm by restricting the class of models to those that satisfy locality of execution and a frame condition on attributes outside the discharging list of actions. The expressive power of the logic was illustrated by showing how properties like stability, stationarity and reachability, in the context of a given behavioural specification, can be formalised.

Other developments of this formalism exist that, because of space limitations, could not be presented, namely bisimilarity notions for durative transition systems and corresponding equivalences between specifications. More details on the semantics of the behavioural paradigm can be found in [5,6]. Methodological aspects of the paradigm and worked examples can also be found in [8,9].

As future work, we intend to study the relationship between DTL and interval-based temporal logics, e.g. [3] and [10] (where we also find a completeness proof for an axiom system that involves operators based on ternary relations) as well as the real-time aspects that can be associated with durative actions [2].

References

[1] K.M.Chandy and J.Misra, *Parallel Program Design*, Addison-Wesley 1988.

[2] Z.Chaochen, C.A.R.Hoare and A.P.Ravn, "A Calculus of Durations", *Information Processing Letters* 40, pp. 269-276, North-Holland 1991.

[3] G.Kutty, L.E.Moser, P.M.Melliar-Smith, L.K.Dillon and Y.S.Ramakrishna, "First-Order Future Interval Logic", in D.Gabbay and H.J.Ohlbach (eds) *Temporal Logic*, LNAI 827, pp. 195-209, Springer-Verlag 1994.

[4] Z.Manna and A.Pnueli, *The Temporal Logic of Reactive and Concurrent Systems*, Springer-Verlag 1991.

[5] I.Nunes, J.L.Fiadeiro and W.M.Turski, "Semantics of Behavioural Specifications", Research Report, Department of Informatics, Faculty of Sciences, University of Lisbon, December 1996.

[6] I.Nunes, J.L.Fiadeiro and W.M.Turski, "Coordinating Durative Actions", in *Coordination'97*, LNCS, Springer-Verlag, in print, 1997.

[7] C.Stirling, "Modal and Temporal Logics", in S.Abramsky, D.Gabbay and T.Maibaum(eds), *Handbook of Logic in Computer Science* 2, pp. 477-563, Clarendon Press, Oxford,1992.

[8] W.M.Turski, "On Specification of Multiprocessor Computing", *Acta Informatica* 27, pp. 685-696, 1990.

[9] W.M.Turski, "Extending the Computing Paradigm", *Structured Programming* 13, pp. 1-9, 1992.

[10] Y.Venema, "A Modal Logic for Chopping Intervals", *Journal of Logic and Computation* 1 (4), pp. 453-476, 1991.

Appendix

Lemma A.1: $\forall \sigma'' \in \Sigma_{DTL}$.

$$(\{C \mid (A[a]C) \in \sigma\} \subseteq \sigma'' \Rightarrow \exists \sigma' \in \Sigma_{DTL}.(\sigma \xrightarrow{a} (\sigma' \rightarrow \sigma'') \wedge A \notin \sigma')).$$

Proof:

By definition of canonical model we know that:

$$\sigma \xrightarrow{a} (\sigma' \rightarrow \sigma'') \text{ iff } \forall B \notin \sigma'.\{D \mid (B[a]D) \in \sigma\} \subseteq \sigma''$$

So, we have to show that , for every σ'',

$$\forall C.((A[a]C) \in \sigma \Rightarrow C \in \sigma'') \Rightarrow \exists \sigma'.(\forall D.\forall B \notin \sigma'.((B[a]D) \in \sigma \Rightarrow D \in \sigma'') \wedge A \notin \sigma')$$

Assume given an arbitrary σ'' for which $\forall C.((A[a]C) \in \sigma \Rightarrow C \in \sigma'')$ holds.

Let B_0, B_1, ..., B_n, ... be an enumeration of $\mathcal{L}(\mathcal{P},\mathcal{A})$. Let

$$\sigma'_0 = \{\neg A\}$$

$$\sigma'_{n+1} = \begin{cases} \sigma'_n \cup \{B_n\} & \text{if } \sigma'_n \vdash B_n \\ \sigma'_n \cup \{\neg B_n\} & \text{otherwise} \end{cases}$$

$$\sigma' = \cup_{n \geq 0} \sigma'_n$$

We know that σ' is a maximal DTL-consistent set. Moreover, $A \notin \sigma'$ because $\neg A \in \sigma'$. We still have to prove that, for this specific σ',

$$\forall D.\forall B \notin \sigma'.((B[a]D) \in \sigma \Rightarrow D \in \sigma'')$$

Assume that, for every σ'_i,

$$\forall D.((((\bigvee_{B \in \sigma'_i} \neg B)[a]D) \in \sigma) \Rightarrow D \in \sigma'') \qquad (*)$$

We then have,

$$\forall D.((\bigvee_{B \in \sigma'_i} (\neg B[a]D) \in \sigma) \Rightarrow D \in \sigma'')$$

and also,

$$\forall D.(\bigvee_{B \in \sigma'_i} ((\neg B[a]D) \in \sigma) \Rightarrow D \in \sigma'')$$

that is,

$$\forall D.(\bigwedge_{B \in \sigma'_i} ((\neg B[a]D) \in \sigma \Rightarrow D \in \sigma''))$$

So, for all formula D and for all σ'_i we have, for every $B \in \sigma'_i$, $((\neg B[a]D) \in \sigma \Rightarrow D \in \sigma'')$

We know that σ' is maximal DTL-consistent. So, by definition, for all formula X, either $X \in \sigma'$ or $\neg X \in \sigma'$. Hence we have that:

$$\forall B \notin \sigma'.((B[a]D) \in \sigma \Rightarrow D \in \sigma'')$$

That is, a maximal DTL-consistent set σ' exists such that

$\forall D.\forall B \notin \sigma'.((B[a]D) \in \sigma \Rightarrow D \in \sigma'') \wedge A \notin \sigma'$

It remains to prove (*), i.e. for every σ'_i,

$\forall D.((((\bigvee_{B \in \sigma'_i} \neg B)[a]D) \in \sigma) \Rightarrow D \in \sigma'')$

The proof is done by induction.

– induction base:

$\sigma'_0 = \{\neg A\}$ so (*) reduces to $\forall C((A[a]C) \in \sigma \Rightarrow C \in \sigma'')$ which was assumed to be true.

– induction step: assume that (*) holds for σ'_i;

Let us abbreviate $(\bigvee_{B \in \sigma'_i} \neg B)$ by $\neg\sigma'_i$ instead. Because $\neg\sigma'_{i+1} \Leftrightarrow \neg\sigma'_i \vee \neg B'_i$, where B'_i is B_i or $\neg B_i$, we have that $\forall D.(((\neg\sigma'_{i+1})[a]D) \in \sigma \Rightarrow D \in \sigma'')$ is equivalent to $\forall D.(((\neg\sigma'_i \vee \neg B'_i)[a]D) \in \sigma \Rightarrow D \in \sigma'')$. So, this is what we are going to prove now.

We distinguish three cases:

i) if $\sigma'_i \vdash B'_i$ then $\neg\sigma'_{i+1} \Leftrightarrow \neg\sigma'_i \vee \neg B'_i \Leftrightarrow \neg\sigma'_i$

The result then follows from the induction hypothesis.

ii) if $\sigma'_i \nvdash B'_i$ and $\sigma'_i \vdash \neg B'_i$ the reasoning is similar to i).

iii) in the case where ($\sigma'_i \nvdash B'_i$ and $\sigma'_i \nvdash \neg B'_i$) we show that, for our purposes, adding B'_i to σ'_i or adding $\neg B'_i$ is equally correct. If not, suppose we have:

$((\neg\sigma'_i \vee \neg B'_i)[a]D_1) \in \sigma$ and $D_1 \notin \sigma''$ and $((\neg\sigma'_i \vee B'_i)[a]D_2) \in \sigma$ and $D_2 \notin \sigma''$

Because σ'' is maximal DTL-consistent, if $D_1 \notin \sigma''$ and $D_2 \notin \sigma''$ then $\neg D_1 \in \sigma''$ and $\neg D_2 \in \sigma''$, and so $(\neg D_1 \wedge \neg D_2) \in \sigma''$, hence $(D_1 \vee D_2) \notin \sigma''$. Moreover, $((\neg\sigma'_i \vee \neg B'_i)[a](D_1 \vee D_2)) \in \sigma$ and $((\neg\sigma'_i \vee B'_i)[a](D_1 \vee D_2)) \in \sigma$ and $(D_1 \vee D_2) \notin \sigma''$ which implies $(((\neg\sigma'_i \vee \neg B'_i) \wedge (\neg\sigma'_i \vee B'_i))[a](D_1 \vee D_2)) \in \sigma$ and $(D_1 \vee D_2) \notin \sigma''$, that is, $\neg\sigma'_i[a](D_1 \vee D_2)) \in \sigma$ and $(D_1 \vee D_2) \notin \sigma''$.

By the induction hypothesis, we reach a contradiction. ♦