Real-Time Object Specification Logic

Klemens Böhm, Amílcar Sernadas

Klemens Böhm
GMD-IPSI
Postfach 10 01 38, 64201 Darmstadt, GERMANY

Phone: 49-6151-869-939
Fax: 49-6151-869-818
E-mail:kboehm@darmstadt.gmd.de

Amílcar Sernadas
INESC
Apartado 10105, 1017 Lisboa Codex, PORTUGAL

Phone: 351-1-3523870
Fax: 351-1-525843
E-mail: acs@inesc.pt

Abstract

A formal specification method called RTOSL ('Real-Time Object Specification Logic') for the object-oriented approach is presented: We have extended the object specification logic OSL ([SernadasA *et al.* 92b]) so that real-time constraints can be included in a specification in an elegant way. Concepts of real-time temporal logic have been applied to this end. Characteristics of RTOSL (different from OSL) are that every position is mapped to a concrete instant of time, that there is a differentiation between actions and action instances, and that events with a duration of time are replaced with an instantaneous begin event and an instantaneous end event. A deductive system is given. Automated verification is "within reach".

Key Words

System specification and verification, object-orientation, real-time, temporal logic.

Table of Contents

1. Introduction
2. The Formalism
   2.1 Basic Notations
   2.2 Life Cycles
   2.3 Syntactic Reasoning
3. An Example of RTOSL and the Deductive System Taken from the 'Package-Router'-Problem
4. Conclusions

Acknowledgements

References

1. Introduction

Current research interest is on both the object-oriented approach (OOA) and real-time. Object-oriented specification methods are widely accepted. The importance of the issue of real-time results from the broad variety of applications. A framework to describe objects with real-time features is very much of practical interest. Even though in both fields the number of formalisms has exploded in the recent past, the number of contributions for both object-orientation and real-time is relatively small.

In this paper we introduce a logic to specify objects with real-time features ('RTOSL' (='Real-Time Object Specification Logic')). On the one hand, it is an extension of the object specification logic (OSL) developped in [SernadasA *et al.* 92b]. (OSL does not have real-time features.) On the other hand, we have applied concepts from real-time temporal logic. (An overview of real-time temporal logics can be found in [Emerson 90] or [Alur and Henzinger 92], together with further references.) We are convinced of the usefulness of a formal method. Descriptions with strict techniques tend to be disambiguous. With a logic-based formalism it is possible to reason about specifications. Automated verification is within reach. The calculus is a "natural supplement" of our framework.

In general a system is called a real-time system if the duration or the begin or end instant of at least one action is subjected to a constraint. Besides that, the correctness of a real-time system may also depend on temporal consistency of the data. (For instance consider an application that continuously takes down measurements. A correctness criterion of that type might be that consecutive readings may be apart five seconds at the most.) RTOSL provides the means to express both timing constraints on actions and constraints on the temporal consistency of the data. - There is a main distinction between hard and soft real-time constraints. Given a hard real-time constraint for a task the task's execution has a positive value iff it is over within the time limit given by the constraint. 'Switching may last at most two seconds.' is an example of a hard real-time constraint. It does not matter how long switching lasts - as long as it is less than two seconds. On the other hand with soft real-time constraints the value of a task's execution (as a function of the task's end instant) changes gradually.

The principles of the OOA are briefly summarized. We intend to establish a common basis before going into details. In both OSL and RTOSL an object has several so-called aspects. For instance, consider aspects `person`, `student` and `professor`. The object 'Klemens_Böhm' has aspects `person` and `student`, but not `professor`. The object 'Amílcar_Sernadas' has aspects `person` and `professor` (but not `student`). (The notion of 'aspect' has been brought up in [SernadasA and Ehrich 90] and addressed in [SernadasA *et al.* 92].) An aspect is in part specified by its observations and actions. Observations are like variables. They can only be modified by actions of the same aspect. An aspect's specification consists of observations and actions and a set of formulae called (proper) axioms. The axioms constrain the aspect behavior and describe the effects of the actions on the observations. It is a methodological principle that aspects with the same specification are comprised in so-called aspect sorts. Because actions may only alter observations of their aspect instance interaction between aspects is the triggering of actions by

other aspect's actions ("message passing"). 'Specialization inheritance' and 'aggregation inheritance' take up a large part of [SernadasA *et al.* 92b]. These concepts have not been modified within the step from OSL to RTOSL. They are not further described in this introduction.

To understand RTOSL it should not be necessary to know OSL. (At least, this report has been arranged with this objective.) However, in this paragraph the main distinctions between OSL and RTOSL are casually pointed out. If you are not familiar with OSL this might look to you as an arbitrary emphasis of particular features of RTOSL.

- In OSL there is a sequence of points of time. They are called positions. In RTOSL - and this is new with regard to OSL - every position is mapped to a real instant of time. Taking this for granted it is nearby to introduce formulae meaning that 'Action c($\underline{x}$) happens at instant t.'. (Here and in the following paragraphs c is an action symbol and $\underline{x}$ a possible list of parameters for c.)
- In RTOSL it has turned out necessary to refine the notion of 'action'. There is a differentiation between 'actions' and 'action instances'. To give an example , if battle is an action then 'Battle of Waterloo', 'Battle of Little Big Horn' or 'Battle of Königgrätz' are possible instances of this action. By saying 'battle happened on July 3rd, 1866.' we would retain information (- which battle?). On the contrary, 'The Battle of Königgrätz happened on July 3rd, 1866.' ressembles a formula in RTOSL - except for the fact that in RTOSL action instances are not identified by names but by (consecutive) numbering: 'battle$_n$' is the notation referring to the n-th instance of battle. The notion of 'action instance' can be used to express that an action happens (at least/at most) n times within a certain time interval.
- In OSL no presuppositions are made with regard to the duration of actions. In reality actions are either instantaneous or have a duration of time. We have taken this into account by introducing formulae that read "Instance n of action c($\underline{x}$) begins." (or "Instance n of action c($\underline{x}$) begins at instant t_b.") or "Instance n of action c($\underline{x}$) ends." (or "Instance n of action c($\underline{x}$) ends at instant t_e.") instead of "Action c($\underline{x}$) happens." (as in OSL). If actions are instantaneous the begin instants and the end instants are identical. For that case there will be formulae that read "Instance n of action c($\underline{x}$) happens instantaneously." (or "Instance n of action c($\underline{x}$) happens instantaneously at instant t_p.").

We have proven that RTOSL is an institution. An institution is a formal definition of (the imprecise notion of) a logical system. (Institutions have been introduced in [Goguen and Burstall 84], examples are in [SernadasA *et al.* 92a].) Casually spoken, having shown that RTOSL is an institution we know that (the reduct of) a heir's model fulfills the specification of any ancestor. Aggregation inheritance is used to describe aspect interaction. That result also applies to specialization inheritance. We feel that this is the wrong platform to present the proof of the institution. Besides, it is relatively long. The proof together with exact definitions is in [Böhm and Sernadas 93a].

We are not aware of a logic-based specification technique for objects with real-time features in the literature so far. Furthermore, within this paper a deductive system for RTOSL is presented. There are postulates for the object- and for the real-time facet. (Most of the axioms of the proof system of OSL (presented in [SernadasA *et al.* 92b]) are also part of our system. Naturally, they

do not cover the real-time part.) A deductive system for objects with real-time features seems to be original, too.

2. The Formalism

In this section we describe the syntax and semantics of RTOSL. The fundamental notion of a life-cycle of an aspect is introduced. We also present a deductive system for RTOSL.

Most of the examples in this report are taken from a problem that is known as the 'Package-Router'-problem. The package distribution system (PDS) is a mechanical system for the transportation of packages along a binary tree of pipes towards a set of destination bins. The packages go from the root to a bin. The internal nodes of the tree are switches capable of directing packages to the right branch or the left one. The leaf nodes are just the bins. For our momentary purposes it is not important how the switches are steered and by which physical actuator the packages are driven. It is possible that there is more than one package in a pipe or a switch at a time.

2.1 Basic Notations

The notions introduced in this paragraph are directly taken over from OSL: An *aspect signature* consists of a set of *observation* symbols and a set of *action* symbols. (In the literature observations are also entitled *slots* and actions are entitled *methods*.) In a description of the PDS there might be aspect sorts `pipe`, `switch` and `package`. `pipe` might have observations succeeding_switch, first_package_in_pipe etc. and actions is_entered_by, is_left_by. `switch` might have observations switch_state, number_of_packages_in_switch etc. and actions is_entered_by, is_left_by. (Since the observation and action names are local to the aspect signatures it is O.K. that the same action names are used for both `pipe` and `switch`.) An *event* symbol is either an observation symbol or an action symbol. Each event symbol has a parameter list (that may be empty, though). For instance in our solution of the 'Package-Router'-problem the `switch`-event switch_state is completed with a parameter of sort '`enum: left, right`', the `pipe`-action is_entered_by is completed with a parameter of sort `package`. It stands for the package that is entering the pipe. In this context we assume as given plain data sorts such as `integer`, `bool`, `real` together with the usual operations. Event parameters may be of such a sort. For instance number_of_packages_in_switch has a parameter of sort `integer`. The expression *sort* contains both plain data sorts as well as aspect sorts. In addition to observation symbols and action symbols an aspect signature contains the sets of all sorts that are relevant in the context (basically the sorts that can be in an event's parameter sort list).

The *(schema) language* of RTOSL consists of the following constructs, among others:

- $t_1=t_2$
  t_1 and t_2 are terms of the same sort. The formula evaluates to 'true' iff t_1 has the same meaning as t_2. A term whose sort is a plain data sort is either a variable of that sort or an

operator symbol with licit arguments. 'not(true)', to give an example, is a term of sort `bool` because the operator not is of this sort. It is well-typed because the argument of not must be of sort `bool`. This is the case in this example. true is an operator symbol of sort `bool` with no arguments (i.e. a constant symbol). A term whose sort is an aspect sort is either a variable of that sort or a variable of a subsort. If `dictionary` is a subsort of `book` then a term of sort `dictionary` is also of sort `book`. The meaning of a variable depends on the value it is mapped to by some *assignment*. So the evaluation of a formula depends on the assignment of the variables.

- s(t)
  t is a term, and s is a sort symbol. s(t) means that t is of sort s. For instance '`package`(package19)' is a true formula, '`integer`(3.1415926)' is not. (package19 is a term of sort `package`.)
- $\Diamond e(\underline{x})$
  e is an event symbol. $\underline{x}$ is a list of parameters of the appropriate sort. The formula states that the event e with parameters $\underline{x}$ may occur at the current instant. To provide an example, let us look at the formula '$\Diamond$is_entered_by(package19)'. is_entered_by is an action symbol of the aspect sort `pipe` with a parameter of sort `package`. '$\Diamond$is_entered_by(package19)' signifies that at the current moment the package package19 can enter a specific pipe. We will point out later on how that specific pipe is identified. Namely, that formula might be true for one pipe, but not for another. In general in a formula '$\Diamond e(\underline{x})$' the concrete instance of the aspect sort rests to be singled out.

The formulae types presented so far are as in OSL.

- $\nabla_b c_n(\underline{x})$, $\nabla_e c_n(\underline{x})$, $\nabla_p c_n(\underline{x})$
  (c is an action symbol. $\underline{x}$ is a list of parameters, as before.) We have explained that the diamond ('$\Diamond$') signifies the enabling of an event. The triangle ('∇'), on the other hand, stands for the actual occurrence of an event. (The distinction between event enabling and event occurrence has been taken over from OSL.) '$\nabla_p c_n(\underline{x})$' means that action c with parameters $\underline{x}$ currently happens for the n-th time. The index 'p' indicates that the event has no duration of time. ('p' stands for 'pointwise'.) To continue the example above, '∇_pis_entered_by$_n$(package19)' means that that a pipe (that rests to be specified) is currently being entered by the package package19 for the n-th time. A real action might have a duration of time. In that case there are two instantaneous actions associated with it: A so-called 'begin action' and a so-called 'end action'. Obviously the begin action happens at the begin instant of the real action and the end action at the end instant. (The notation is '$\nabla_b c_n(\underline{x})$' and '$\nabla_e c_n(\underline{x})$', respectively.) - The counterpart in OSL is '$\nabla c(\underline{x})$': Neither is there an 'action instance'-index n nor an index $\theta \in \{b, e, p\}$.
- $\nabla_\theta c_n(\underline{x})$:t (with $\theta \in \{b, e, p\}$)
  Again, c is an action symbol and $\underline{x}$ an appropriate parameter list. The formula means that the n-th instance of the begin/end/instantaneous action c occurs at instant t. t is a term of sort `real`. `real` is our model of time. Within this framework we want a dense time domain: Frequently, in real-time the system to be specified must be embedded in a physical system that already exists. To describe that physical system in general a dense time domain is compulsory. Please note that a formula of kind '$\nabla_\theta c_n(\underline{x})$:t' is either always true or always false. On the other hand a formula of kind '$\nabla_\theta c_n(\underline{x})$' is true at most at one position. It is the position at which the n-th instance of $c(\underline{x})$ happens. 'The Battle of Königgrätz happened on July 3rd, 1866.' is a

formula of the first kind. (Note that it is true in the past, present or future.) "The Battle of Königgrätz is currently happening.' is of the second kind. (It is true at exactly one particular date.) - There is no formula kind in OSL corresponding to '$\nabla_\theta c_n(\underline{x})$:t'.

The rest of the language is as OSL (on the conceptual level):

- The language contains the logical connectives negation, conjunction, disjunction, implication etc. (denoted by '$\neg$', '$\wedge$', '$\vee$', '$\Rightarrow$', respectively).
- Besides that, we also use the well-known quantors from first-order logic ('$\exists$' and '$\forall$'). Their meaning is as usual. It is possible to quantify both over aspect sort variables and plain data sort variables.
- Finally, there is a broad variety of temporal operators ('F', 'G', 'X', 'P', 'H', 'Y', 'U', 'S'). 'F' is read 'sometime in (the present and) the future'. The formula 'F f' is true at the current instant iff the formula 'f' is true at a later instant. In the 'Package-Router'-context one might want to postulate that '∇_pis_entered_by$_n$(package19) $\Rightarrow$ (F ∇_pis_left_by$_n$(package19)' always holds. (If that package enters a particular pipe it will eventually leave it.) Analogously, 'G' reads 'always in (the present and) the future'. 'X' is the 'next'-operator. '(X f)' is currently true iff f is true in the next position. The corresponding operators for the past are P, H and Y ('P' being the 'sometime-in-the-past'-operator, 'H' the 'always-in-the-past'-operator and 'Y' the operator referring to the previous position). 'f U g' means that f is true until g becomes true; 'f S g' means that f has been true since an instant at which g was true.
- A couple of abbreviations turned out to be useful. Here we only mention that '(M f)' is short for '(P f) $\vee$ (F f)' because it is used in the sequel.

As mentioned before the enabling or actual occurrence of an event is to be pinned to an instance of the aspect sort. A formula '$\nabla_\theta c_n(\underline{x})$' of the schema language, to give an example, is transformed to the formula 'z.$\nabla_\theta c_n(\underline{x})$'. z is a term whose (aspect) sort is the one that has c as an action symbol. As before, z can be a variable or a constant symbol. 'pipe38.∇_pis_entered_by$_n$(package19)' would be a formula of the *aspect language* (supposing pipe38 is an instance of aspect sort `pipe`). The formula means that pipe38 is currently entered by package19 for the n-th time. The language where the instance is identified will be referred to as aspect language. (The opposite, with no such references, is the schema language.)

2.2 Life Cycles

The notion of 'life cycle' is the fundamental notion on the semantic level of both OSL and RTOSL. The life cycle of an aspect is a sequence of points of time. They are called "positions". Each position is mapped to a set of instantaneous action instances and a set of events. These are the instances that occur at that position and the events that are enabled at that position. (This holds for OSL and RTOSL.) Besides that, in RTOSL each position is mapped to a point of real time. Position n is after position n+k, i.e. $\tau(n)<\tau(n+k)$. τ is the function mapping each position to an instant of time. So τ is of type $\mathbf{Z}\rightarrow\mathbf{R}$. ($\mathbf{R}$ is our model of time, $\mathbf{Z}$ the sequence of positions.) Furthermore, in OSL and RTOSL each sort symbol relates to a set of instances. $\mathcal{U}$(s) is the notation for the set of instances of sort s. $\mathcal{U}$(`movie`)={"Casablanca", "E.T.", ...}, to give an example. s' being a subsort of s implies that $\mathcal{U}(s')\subseteq\mathcal{U}(s)$, so $\mathcal{U}$(`science_fiction`)={"Aliens", "Dune",

...}⊆$\mathcal{U}$(movie). For illustration purposes, cuts of two possible life-cycles of the instance pipe38 of class pipe are depicted in diagram 2.2.1.

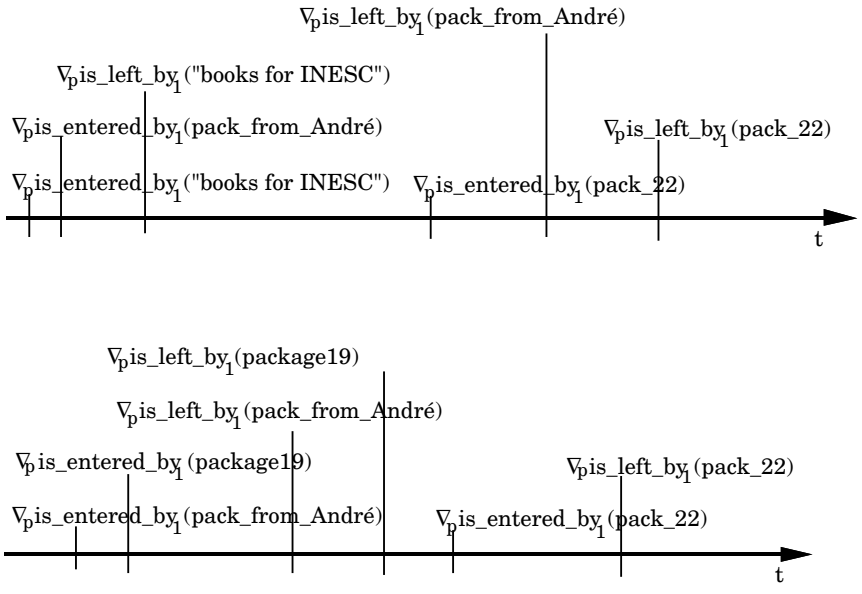

diagram 2.2.1

Clearly the truth of an aspect formula depends on the underlying life-cycle. For instance, the formula '∃t:(pipe38.∇_pis_entered_by$_n$(package19):t)' does not hold for the first life-cycle. It holds for the second one, though.

A life-cycle is subject to a couple of restrictions. These restrictions will be called *general assertions* of a life-cycle. They have been formulated in the language. (In the sequel we will refer to them with the abbreviations in brackets behind the assertions.) Most of these assertions do not have a counterpart in OSL. (They supplement conceptions that are not part of OSL.) It is more convenient to just point out the equivalents, the rest is new.

- z.$\nabla_b c_n(\underline{x})$ ⇒ F z.$\nabla_e c_n(\underline{x})$ (*BEA* - for each action symbol c, and n is a term of sort nat): If an action instance begins then it ends eventually. Thus, we do not allow actions that last forever.
- z.$\nabla_e c_n(\underline{x})$ ⇒ P z.$\nabla_b c_n(\underline{x})$ (*EBA* - dto.): Analogously the occurrence of the end of an action instance implies that that action has begun at an earlier (or the same) point of time.
- z.$\nabla_\theta c_n(\underline{x})$ ⇒ X G ¬ z.$\nabla_\theta c_n(\underline{x})$ (*UniqA* - dto., and $\theta \in \{b, e, p\}$): If an action instance happens at some point of time it may not happen again. This axiom is called 'uniqueness axiom'. Note that the action instance is unique. However, another instance of the same action may happen at another point of time. (The action instances are identified by the index behind the action symbol, as mentioned before.)
- z.$\nabla_b c_n(\underline{x})$ ⇒ z.◊c($\underline{x}$) ($\nabla\Diamond AA_c$ - for each action symbol c, and n is a term of sort nat): An action can only occur when it is enabled. In other words, if a begin action occurs the action is currently enabled. (This property corresponds to an OSL-property, let aside that the 'begin instant' is an RTOSL-specific notion.)

- $(z.\Diamond o(\underline{x}, x) \,\&\, z.\Diamond o(\underline{x}, x')) \Rightarrow (x=x')$ ($FuncA_o$ - for each observation symbol o): This so-called 'functionality axiom' states that observations are functional. We want to rule out formulae like 'switch17.◊switch_state(left) ∧ switch17.◊switch_state(right)'. (switch17 is an instance of sort `switch`.) The formula states that at the same time switch17 is in the 'left'- and in the 'right'-position. (The parameter list of switch_state is of sort '`enum: left, right`'.) - The functionality axiom is also in OSL.
- $(z.\nabla_b c_n(\underline{x}) \wedge 1 \le n' < n) \Rightarrow Y\ P\ z.\nabla_b c_{n'}(\underline{x})$ (*ContA* - for each action symbol c, and n is a term of sort nat): This axiom is called 'continuity axiom' because we want the numbering of the action instances to be continuous.
- $\exists(\neg z.\Diamond e(\underline{x}) \wedge X\ z.\Diamond e(\underline{x})) \Rightarrow ((\exists\ z.\nabla_e c^1_n(\underline{x}_1)) \vee .... \vee (\exists\ z.\nabla_e c^m_n(\underline{x}_m)))$ ($FrmEA_e$ - for each event symbol e, assuming that $\{c^1,...,c^m\}$ is the set of all action symbols); $\exists(z.\Diamond e(\underline{x}) \wedge X \neg z.\Diamond e(\underline{x})) \Rightarrow ((\exists\ z.\nabla_b c^1_n(\underline{x}_1)) \vee .... \vee (\exists\ z.\nabla_b c^m_n(\underline{x}_m)))$ ($FrmBA_e$ - dto.): The so-called frame-rules states that any modification of an aspect instance (of its observations, for instance) can only be caused by an action. This is a principle of object-orientation. (Another variant of) the frame rule is part of OSL.
- $z.\nabla_p c_n(\underline{x}) \Leftrightarrow (z.\nabla_b c_n(\underline{x}) \wedge z.\nabla_e c_n(\underline{x}))$ (*PointA* - for each action symbol c): The instantaneous action ("pointwise") as well as the begin and the end action have been introduced. The 'pointwise'-action occurs iff begin and end action happen at the same instant of time.
- $z.\nabla_\theta c_n(\underline{x}) \Rightarrow n \ge 1$ (*PosIndA* - for each action symbol c): Though trivial, this axiom is needed in the sequel.

2.3 Syntactic Reasoning

We have developped a deductive system for RTOSL, taking pattern from the proof system for OSL ([SernadasA *et al.* 92b]). In more detail, there are axioms of first-order temporal logic, the general axioms presented in chapter 2.2 and RTOSL-specific axioms. (Alltogether there are about 50 axioms. The complete list is in [Böhm 93].) The axioms of first-order temporal logic have been taken over from OSL's deductive system. However, OSL only has operators for the future. This calculus also contains axioms for the past. They have been obtained by "mechanically" replacing the operators for the future in those axioms with their counterparts for the past. Most of the general assertions from the definition of 'life-cycle' are not in OSL. (We have already stated that in chapter 2.2.) The RTOSL-specific axioms are new, naturally. Some of the RTOSL-specific axioms that we apply in the sequel are presented next:

- $M\ z.\nabla_\theta c_n(\underline{x}) \Leftrightarrow \exists t: (z.\nabla_\theta c_n(\underline{x}):t)$ *DefA*
  This axiom is the 'definedness axiom'. There are two equivalent ways to express that the n-th instance of action $c(\underline{x})$ occurs at some time in the past, present or future.
- $(z.\nabla_\theta c_n(\underline{x}):t \wedge z.\nabla_\theta c_n(\underline{x}):t') \Rightarrow t=t'$ *UITA*
  Every instance of the begin/end/instantaneous action $c(\underline{x})$ occurs at a unique instant of time.

- M $(z.\nabla_{\theta} c_n(\underline{x}) \wedge F\ z.\nabla_{\theta'} c'_{n'}(\underline{x}')) \Leftrightarrow \exists t,t'\colon (z.\nabla_{\theta} c_n(\underline{x}){:}t \wedge z.\nabla_{\theta'} c'_{n'}(\underline{x}'){:}t' \wedge t \leq t')$ *FTA*
  The temporal order of two actions can either be expressed using the sometime-in-the-future-operator ('F') or by comparing the actions' real instants of time. (Analogously there exists an axiom PTA with the sometime-in-the-past operator ('P').)
- M $(z.\nabla_{\theta} c_n(\underline{x}) \wedge X\ F\ z.\nabla_{\theta'} c'_{n'}(\underline{x}')) \Leftrightarrow \exists t,t'\colon (z.\nabla_{\theta} c_n(\underline{x}){:}t \wedge z.\nabla_{\theta'} c'_{n'}(\underline{x}'){:}t' \wedge t < t')$ *XFTA*
  There are two equal ways to express that two distinct positions are pinned to two different points of time. In axiom FTA 'F' corresponds to the relation '≤' between the instants of time. In XFTA there are 'X F' (read as 'in the next state sometime in the future') and '<'. (Analogously there exists an axiom YPTA with the operator 'Y' and 'F' on the one hand and '≥' on the other hand.)
- M $(z.\nabla_{\theta} c_n(\underline{x}) \wedge z.\nabla_{\theta'} c'_{n'}(\underline{x}')) \Leftrightarrow \exists t\colon (z.\nabla_{\theta} c_n(\underline{x}){:}t \wedge z.\nabla_{\theta'} c'_{n'}(\underline{x}'){:}t)$ *STA*
  There are two ways to express that two actions happen at the same instant of time.

The so-called 'congruence axiom' is also part of the OSL-calculus. It is used extensively in the examples coming up:

- $x=y \Leftrightarrow f \Rightarrow g$ *CongA*
  This is a postulate provided that f and g are atomic and g is obtained from f by replacing zero or more occurrences of x by y.

Finally, the 'detachment rule' (DetR) states that the formulae 'f', 'f ⇒ g' imply the formula 'g'. With the rule 'LNecR' we can infer the formula 'L f' from the formula 'f'.

3. An Example of RTOSL and the Deductive System Taken from the 'Package-Router'-Problem

In this section we give a specification of the aspect sort `package` from the 'Package-Router'-problem and reason about it with the deductive system. The behavior of a package can be visualized as follows (using the graphical description mechanism OBLOG ([SernadasA *et al.* 91], [Costa *et al.*])).

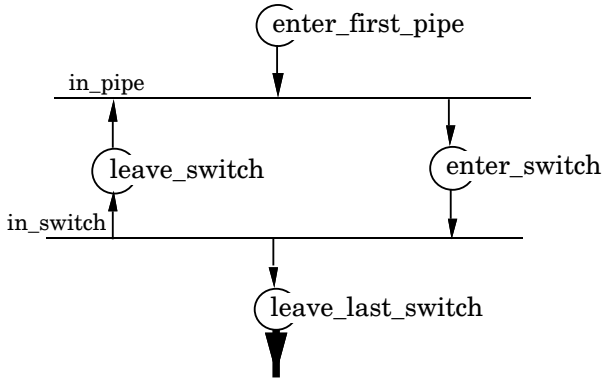

diagram 3.1

The circles are actions. This so-called behavior diagram merely illustrates that a package is alternately in a switch or in a pipe. Obviously enter_switch and leave_switch alternate except for the

first and the last action (enter_first_pipe and leave_last_switch, respectively). For the sake of simplicity we make the following assumptions in this section:

- All actions are instantaneous.
- Actions do not have parameters.
- There can only be one package in the PDS at a time.
- There is only one switch to be passed.

Some of these points are in contrast to the 'Package-Router'-problem. However, we want this example to be as simple as possible. (Even trivial proofs tend to be relatively long.) Finally we postulate that the time a package remains in the switch is between four and five seconds, the time a package remains in a pipe is between two and three seconds.

The aspect sort `package` with the restrictions given so far can be described with the following specification:
The signature contains the observation symbols destination with parameter `destination_type` (which might be `integer` or `string`, this is not important) and situation with a parameter of sort '`enum: in_switch, in_pipe`'. It contains the action symbols enter_first_pipe, enter_switch, leave_switch and leave_last_switch. (They are as in the behavior diagram).

The proper axioms of aspect sort `package` are as follows:

(p1) $p.\nabla_p enter_first_pipe_1 \Rightarrow X\ p.\Diamond situation(in_pipe)$

(p2) $p.\nabla_p enter_switch_n \Rightarrow p.\Diamond situation(in_pipe)$

(p3) $p.\nabla_p enter_switch_n \Rightarrow X\ p.\Diamond situation(in_switch)$

(p4) $p.\nabla_p leave_switch_n \Rightarrow p.\Diamond situation(in_switch)$

(p6) $p.\nabla_p leave_switch_n \Rightarrow X\ p.\Diamond situation(in_pipe)$

(p7) $p.\nabla_p leave_last_switch_n \Rightarrow p.\Diamond situation(in_switch)$

(p8) $p.\nabla_p leave_switch_n{:}t_d \wedge p.\nabla_p enter_switch_{n'}{:}t_a \Rightarrow \neg(0<t_d\text{-}t_a<2)$

(p9) $p.\nabla_p leave_last_switch_n{:}t_d \wedge p.\nabla_p enter_switch_{n'}{:}t_a \Rightarrow \neg(0<t_d\text{-}t_a<2)$

(p10) $p.\nabla_p enter_switch_n{:}t_a \Rightarrow \exists n', t_d: ((p.\nabla_p leave_switch_{n'}{:}t_d \vee p.\nabla_p\ leave_last_switch_{n'}{:}t_d) \wedge 0<t_d\text{-}t_a\leq 3)$

(p11) $p.\nabla_p leave_switch_n{:}t_d \Rightarrow \exists n', t_a: (p.\nabla_p enter_switch_{n'}{:}t_a \wedge 0<t_d\text{-}t_a\leq 3)$

(p12) $p.\nabla_p leave_last_switch_n{:}t_d \Rightarrow \exists n', t_a: (p.\nabla_p enter_switch_{n'}{:}t_a \wedge 0<t_d\text{-}t_a\leq 3)$

(p13) $p.\nabla_p enter_switch_n{:}t_a \wedge p.\nabla_p leave_switch_{n'}{:}t_d \Rightarrow \neg(0<t_a\text{-}t_d<4)$

(p14) $p.\nabla_p enter_switch_n{:}t_a \wedge p.\nabla_p enter_first_pipe_{n'}{:}t_b \Rightarrow \neg(0<t_a\text{-}t_b<4)$

(p15) $p.\nabla_p enter_switch_n{:}t_a \Rightarrow \exists n', t_d: ((p.\nabla_p leave_switch_{n'}{:}t_d \vee p.\nabla_p enter_first_pipe_{n'}{:}t_d) \wedge 0<t_a\text{-}t_d\leq 5)$

(p16) $p.\nabla_p leave_switch_n{:}t_d \Rightarrow \exists n', t_a: (p.\nabla_p enter_switch_{n'}{:}t_a \wedge 0<t_a\text{-}t_d\leq 5)$

(p17) $p.\nabla_p enter_first_pipe_n{:}t_b \Rightarrow \exists n', t_a: (p.\nabla_p enter_switch_{n'}{:}t_a \wedge\ 0<t_a\text{-}t_b\leq 5)$

(p18) $p.\Diamond e(\underline{x}) \Rightarrow Y\ P\ p.\nabla_p enter_first_pipe_1$ for each event symbol e

(p19) $p.\Diamond e(\underline{x}) \Rightarrow KP\ p.\nabla_p leave_last_switch_1$ for each event symbol e.

(p20) $\neg\ M\ p.\nabla_p enter_switch_2$

With these axioms and the axioms of the proof system we prove that no package is in the PDS longer than 8s. More precisely, we prove the following formula:

$$p.\nabla_p enter_first_pipe_1{:}t_b \Rightarrow \exists n'', t_d: (p.\nabla_p leave_last_switch_{n''}{:}t_d \wedge t_d\text{-}t_b\leq 8)$$

(Being in the PDS begins with enter_first_pipe and ends with leave_last_switch.)

Lemma A: M $p.\nabla_p$enter_switch$_{n'} \Rightarrow n'=1$

Proof of lemma A:

1	M $p.\nabla_p$enter_switch$_{n'}$	;Hypothesis 1
2	$n'>1$	;Hypothesis 2 to be disproven
3	L $(n'>1)$	;2, LNecR
4	$(L\ \alpha \wedge M\ (\beta \wedge \gamma)) \Rightarrow M\ (\alpha \wedge \beta \wedge \gamma)$	;theorem of temporal logic
5	M $(p.\nabla_p$enter_switch$_{n'} \wedge n'>1)$	1, 3, 4, CongA, DetR
6	$(n'>1) \Rightarrow ((n'=2) \vee (n'>2))$	;tautology
7	L $((n'>1) \Rightarrow ((n'=2) \vee (n'>2)))$	;6, LNecR
8	M $(p.\nabla_p$enter_switch$_2 \vee (p.\nabla_p$enter_switch$_{n'} \wedge n'>2))$	;4, 5, 7, CongA, DetR
9	$M\ (\alpha \vee \beta) \Leftrightarrow M\ \alpha \vee M\ \beta$	;theorem of temporal logic
10	M $p.\nabla_p$enter_switch$_2 \vee$ M $(p.\nabla_p$enter_switch$_{n'} \wedge n'>2)$	;8, 9, CongA, DetR
11	L $((p.\nabla_p$enter_switch$_{n'} \wedge n'>2) \Rightarrow Y\ P\ p.\nabla_p$enter_switch$_2)$	;ContA, LNecR
12	M $p.\nabla_p$enter_switch$_2 \vee$ M $(((p.\nabla_p$enter_switch$_{n'} \wedge n'>2)$ $\Rightarrow Y\ P\ p.\nabla_p$enter_switch$_2) \wedge (p.\nabla_p$enter_switch$_{n'} \wedge n'>2))$	;4, 10, 11, CongA, DetR
13	$M\ (\alpha \wedge (\alpha \Rightarrow \beta)) \Rightarrow M\ \beta$	;theorem of temporal logic
14	M $p.\nabla_p$enter_switch$_2 \vee$ M Y P $p.\nabla_p$enter_switch$_2$	;12, 13, CongA, DetR
15	$M\ Y\ P\ \alpha \Rightarrow M\ \alpha$	;theorem of temporal logic
16	M $p.\nabla_p$enter_switch$_2$	;14, 15, CongA, DetR
17	M $p.\nabla_p$enter_switch$_2 \wedge \neg$ M $p.\nabla_p$enter_switch$_2$	;16, p20
18	n'>;/1	;2, 17 {hypothesis 2 is now disproven}
19	$n'<1$	;Hypothesis 3 to be disproven
20	L $(n'<1)$	;19, LNecR
21	M $(p.\nabla_p$enter_switch$_{n'} \wedge n'<1)$	:1, 4, 20, CongA, DetR
22	L $(p.\nabla_p$enter_switch$_{n'} \Rightarrow n' \geq 1)$	;PosIndA, DetR
23	$(M\ (\alpha \wedge \beta) \wedge L\ (\alpha \Rightarrow \gamma)) \Rightarrow M\ (\beta \wedge \gamma)$	;theorem of temporal logic
24	M $(n'<1 \wedge n' \geq 1)$	;21, 22, 23, CongA, DetR
25	n'<;/1	19, 24 {hypothesis 3 is now disproven}
26	$n'=1$	;18, 25
27	M $p.\nabla_p$enter_switch$_{n'} \Rightarrow n'=1$	;1, 26

- end of proof of lemma A.

Lemma B: $\neg$ M $p.\nabla_p$leave_switch$_1$

Proof of lemma B:

1	M $p.\nabla_p$leave_switch$_1$	;Hypothesis to be disproven
2	$\exists t$: $(p.\nabla_p$leave_switch$_1$:t)	;1, DefA, DetR
3	$\exists n', t'$: $(p.\nabla_p$enter_switch$_{n'}$:t' $\wedge$ 0<t-t')	;2, p11, DetR
4	M $p.\nabla_p$enter_switch$_{n'}$	;3, DefA, DetR
5	$n'=1$	;4, lemma A, DetR
6	$\exists t''$: $(p.\nabla_p$enter_switch$_1$:t' $\wedge$ 0<t-t')	;3, 5, CongA, DetR
7	$\exists n'', t''$: $(p.\nabla_p$enter_switch$_{n'}$:t'' $\wedge$ t''-t>0)	;2, p16, DetR
8	$\exists t''$: $(p.\nabla_p$enter_switch$_1$:t'' $\wedge$ t''-t>0)	;"analogous to '3→6"
9	t'=t''	;6, 8, UITA, DetR

10 $0<t-t'$;6

11 $t'-t>0$;8, 9, CongA, DetR

12 $t>t' \wedge t<t'$;10, 11

13 $\neg$ M $p.\nabla_p leave_switch_1$;12 {hypothesis is now disproven}

- end of proof of lemma B.

Lemma C: $\neg\ p.\nabla_p leave_switch_n{:}t$

Proof of lemma C:

1 $\exists n, t: (p.\nabla_p leave_switch_n{:}t)$;Hypothesis to be disproven

2 M $p.\nabla_p leave_switch_n$;1, DefA, DetR

3 $p.\nabla_p leave_switch_n \Rightarrow (p.\nabla_p leave_switch_1 \vee (p.\nabla_p leave_switch_n \wedge n>1))$;tautology

4 $p.\nabla_p leave_switch_n \Rightarrow (p.\nabla_p leave_switch_1 \vee$ Y P $p.\nabla_p leave_switch_1)$;3, ContA, DetR

5 $\alpha \Rightarrow$ P α, Y P $\alpha \Rightarrow$ P α ;theorem of temporal logic

6 $p.\nabla_p leave_switch_n \Rightarrow$ P $p.\nabla_p leave_switch_1$;4, 5, CongA, DetR

7 $(\alpha \Rightarrow$ P $\beta) \Rightarrow$ (M $\alpha \Rightarrow$ M $\beta)$;theorem of temporal logic

8 M $p.\nabla_p leave_switch_1$;2, 6, 7, CongA, DetR

9 M $(p.\nabla_p leave_switch_1) \wedge \neg$ M $(p.\nabla_p leave_switch_1)$;lemma B, 8

10 $\neg\ p.\nabla_p leave_switch_n{:}t$;9 {hypothesis is now disproven}

- end of proof of lemma C.

Main Proof: $p.\nabla_p enter_first_pipe_1{:}t_b \Rightarrow \exists n'', t_d: (p.\nabla_p leave_last_switch_{n''}{:}t_d \wedge t_d-t_b \leq 8)$

1 $p.\nabla_p enter_first_pipe_1{:}t_b$;Hypothesis

2 $\exists n', t_a: (p.\nabla_p enter_switch_n{:}t_a \wedge\ t_a-t_b \leq 5)$;1, p17, DetR

3 M $p.\nabla_p enter_switch_n$;3, DefA, DetR

4 $n=1$;3, lemma A, DetR

5 $\exists t_a: (p.\nabla_p enter_switch_1{:}t_a \wedge\ t_a-t_b \leq 5)$;2, 4, CongA, DetR

6 $(\neg\ p.\nabla_p leave_switch_{n'}{:}t_d \wedge 0<t_d-t_a \leq 3) \Rightarrow$ $\exists n'', t_d: (p.\nabla_p leave_last_switch_{n''}{:}t_d \wedge t_d-t_a \leq 3)$;5, p10, DetR

7 $(\neg\ (\alpha \wedge \beta) \Rightarrow\ \gamma) \Rightarrow\ (\neg\ \alpha \Rightarrow\ \gamma)$;tautology

8 $\neg\ p.\nabla_p leave_switch_{n'}{:}t_d \Rightarrow \exists n'', t_d: (p.\nabla_p leave_last_switch_{n''}{:}t_d \wedge t_d-t_a \leq 3)$;6, 7, CongA, DetR

9 $\exists n'', t_d: (p.\nabla_p leave_last_switch_{n''}{:}t_d \wedge t_d-t_a \leq 3)$;lemma C, 8, DetR

10 M $p.\nabla_p leave_last_switch_{n''}$;9, DefA, DetR

11 $n''=1$;10, "analogous to 'lemma A'", DetR

12 $\exists t_d: (p.\nabla_p leave_last_switch_1{:}t_d \wedge t_d-t_a \leq 3)$;7, 11, CongA, DetR

13 $\exists t_d: (p.\nabla_p leave_last_switch_1{:}t_d \wedge t_d-t_b \leq 8)$;5, 12

14 $p.\nabla_p enter_first_pipe_1{:}t_b \Rightarrow\ \exists n'', t_d: (p.\nabla_p leave_last_switch_{n''}{:}t_d \wedge t_d-t_b \leq 8)$;1, 13

- end of main proof.

4. Conclusions

We have presented and illustrated a real-time object specification logic (RTOSL). It is based on the object specification logic OSL introduced in [SernadasA *et al.* 92b]. More comprehensively, the relation between OSL and RTOSL is as follows:

- In both OSL and RTOSL there is a distinction between observation symbols and action symbols (on the signature level).
- In both OSL and RTOSL it is distinguished between event enabling and event occurrence. (The operators are '◊' and '∇', respectively.) In RTOSL, however, the conception of 'action occurrence' is further refined:
  - Let c(x) be an action. The OSL-notion 'c(x) happens.' is replaced either with 'c(x) begins.', 'c(x) ends.' or 'c(x) occurs instantaneously.'.
  - Whereas in OSL actions happen, in RTOSL action instances happen. (Action instances are unique.)
  - Instantaneous occurrence of an action instance, its begin or end can be pinned to an absolute point of time. This induces a type of formula that is not in OSL. It reads 'Action instance $c_n(x)$ begins/ends/occurs instantaneously at instant t.'. (This is an application of the notion of 'explicit-clock-variable' from real-time temporal logic.)

  Temporal operators and quantors are part of both formalisms.
- There is a notion of 'life-cycle' both in OSL and RTOSL. In OSL a life-cycle is as follows:
  - Every sort symbol is mapped to a domain (i.e. the set of possible instances).
  - The "underlying time structure" is a sequence of points of time. Each of them is mapped to the set of events that are enabled and the ones that occur at that position.

  This principally holds for RTOSL, too. Moreover, there is the following extension:
  - Every position is mapped to a point of real time.
- A large number of the axioms of OSL's deductive system are also in RTOSL's. A lot of the new axioms of the proof system of RTOSL reflect the fact that there are two equivalent ways to express temporal ordering: First, using temporal operators and, second, by comparing references to absolute time.

We see our contribution in providing a logic-based specification mechanism combining both object-orientation and real-time.

It is a given that the expressiveness of a logic-based formalism is very high. This cannot be said of data models. (In [Albano *et al.* 91] bounds of conventional object-oriented data models are identified.) We do not see limits of RTOSL's expressiveness. For example, data consistency constraints can be formulated in both OSL and RTOSL with ease. (A sample constraint might be 'The salesperson with the highest average sales per month has the highest salary.' `salesperson` would be an aspect sort with observations average_monthly_sales and salary.)

RTOSL is first-order, linear-time, point-based and with real-numbered time. It is a (syntactic) variant of the 'explicit-clock-variable'-approach. The operators for the past have been included originally to formalize general axioms EBA and ContA in the language. However they also turned out to be very useful for proper specifications. The modal operator M can be applied in proper specifications. (This can be seen in section 3.)

In this paper only hard real-time constraints have been taken into account. For soft real-time constraints the notion of a system fulfilling a specification is not as trivial as for the hard real-time case. (We doubt that there is just one notion.) To extend OSL such that soft real-time constraints can be taken into account a two-valued logic might not be sufficient. We reckon that

the modifications in detail necessary to take soft real-time constraints into account would be major.

We concede that specifications with RTOSL are not easy to survey. In practice RTOSL is not a tool for an early stage of the specification process. We have been working on a graphical specification method called OBLOG*time* for object real-time behavior [Böhm and Sernadas 93b]. It does not have that weakness. On the other hand OBLOG*time* is not fit to reason about objects and is less expressive than RTOSL. Instead of developping a model structure for OBLOG*time*, its constructs are introduced as abbreviations of RTOSL-constructs. Then in the specification process the advantages of both RTOSL and OBLOG*time* - expressiveness and the option to reason about the specification on one hand, easy viewing and quick outlining (at an early stage) on the other hand - can be exploited.

Acknowledgements

We are indebted to Profª Cristina Sernadas. Her review of this paper has been very gainful. Thanks are also due to the group of Prof. A. Buchmann from Technische Hochschule Darmstadt, Germany, for lasting interest and helpful comments.

References

[Albano *et al.* 91]

Antonio Albano, Giorgio Ghelli, Renzo Orsini "A Relationship Mechanism for a Strongly Typed Object-Oriented Database Programming Language", in G.Lohmann, A.Sernadas, R.Camps (eds) *Proceedings of the 17th International Conference on Very Large Data Bases*, 1991, 565-575

[Alur and Henzinger 92]

Rajeev Alur, Thomas A.Henzinger, "Logics and Models of Real Time: A Survey", in *Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, June 1991, Proceedings*, Lecture Notes in Computer Science 600, Springer-Verlag, 1992, 74-106

[Böhm and Sernadas 93a]

Klemens Böhm, Amílcar Sernadas, "An Institution of Real-Time Object Behavior", to be published

[Böhm 93]

Klemens Böhm, "Real-Time Object Specification Logic", diploma thesis

[Böhm and Sernadas 93b]

Klemens Böhm, Amílcar Sernadas, "OBLOG*time* - a Graphical Specification Method for Objects with Real-Time Features", to be published

[Costa *et al.*]

José Félix Costa, Amílcar Sernadas, Cristina Sernadas, *Formal Specifications*, Department of Mathematics - IST (Lisbon Institute of Technology), Course Material

[Emerson 90]

E.A.Emerson, "Temporal and Modal Logic", in J.van Leeuwen (ed), *Handbook of Theoretical Computer Science*, vol. B, Elsevier Science Publishers (North-Holland), 1990, 995-1072

[Goguen and Burstall 84]

J.A.Goguen, R.M.Burstall, "Introducing Institutions", in E.Clarke (ed) *Logics of Programming Workshop*, Springer-Verlag, 1984, 221-256

[SernadasA and Ehrich 90]

A.Sernadas and H.-D.Ehrich, "What is an Object, After All", in R.Meersman, W.Kent and S.Khosla (eds), *Object-oriented Databases: Analysis, Design and Construction*, North-Holland, 1991, 39-69

[SernadasA *et al.* 91]

Amílcar Sernadas, Cristina Sernadas, Paula Gouveia, Pedro Resende, João Gouveia, *OBLOG An Informal Introduction*, Computer Science Group INESC/Department of Mathematics - IST (Lisbon Institute of Technology), Course Material, 1991

[SernadasA *et al.* 92a]

Amílcar Sernadas, José Félix Costa, Cristina Sernadas, "An Institution of Object Behavior", ADT/COMPASS'92, Caldes de Malavella, Oct. 26-30, 1992

[SernadasA *et al.* 92b]

Amílcar Sernadas, Cristina Sernadas, José Félix Costa, "Object Specification Logic", INESC Research Report 1992, submitted for publication