A process algebraic approach to hybrid systems

A Process Algebraic Approach to Hybrid Systems

Extended Abstract

Tim A.C. Willemse
Eindhoven University of Technology
Department of Mathematics and Computing Science, Formal Methods Group
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
timw@win.tue.nl

Abstract— **Many hybrid systems are *safety critical* systems, i.e. the incorrect functioning of the system can have severe consequences. Being able to model and analyse such systems prior to their implementation is vital. Using the process algebra μCRL$_t$ a case study on a conveyor belt system has been conducted (see [12]). The great complexity allowed for a thorough identification of gaps in the formalism for applications in the area of hybrid systems. The models in this paper are slightly simplified versions of the models in [12].**

Keywords— **Process Algebra, Hybrid Systems, embedded systems**

I. Introduction

Hybrid systems are systems in which the essence of the system cannot be captured by describing merely the continuous components or the discrete components; the combination of both is essential for a correct representation of the system. The interest in such systems has grown enormously in the last decades. Application areas for hybrid systems are abundant, ranging from process control to avionics. Often, serious safety requirements are imposed on such systems, demanding no calamities happen. Even for elementary (hybrid) systems it is often hard to prove requirements are met, be it safety or liveness requirements.

In order to guarantee, with some level of confidence, that a system adheres to its requirements, models describing in great detail the essence of the system are made. These models often are subject to analysis, validation and verification.

Formalisms that allow for modelling hybrid systems have been proposed in the past (e.g. [9], [2], [1]), some of which have become quite popular. On the one hand, theory in these formalisms is investigated and on the other hand, the theory is applied on case studies.

In this paper, a process algebraic approach is investigated. Process algebras come in several variants (e.g. CSP [10], CCS [11], Lotos [4] , ACP [3]). Application of process algebras in the domain of hybrid systems is not unprecedented (e.g. [8], [5]). In this paper, the use of various techniques, originally developed for the untimed and timed variants of μCRL [7] are shown in the context of hybrid systems. Most notably, the use of invariants for proving conformance to design requirements of a system, is sketched. Some parts are illustrated by means of a simplified version of the case study that appeared in [12], consisting of a conveyor belt system.

This paper is outlined as follows: the next section describes the formalism μCRL$_t$ [6]. Section III discusses parts of the case study and section IV describes the possibilities for analysis and verification. Section V concludes with some remarks, discusses ongoing research and hints at future research.

II. Timed μCRL

The formalism μCRL$_t$ [6] is a time extended variant of the process algebra μCRL [7], which in turn is strongly rooted in ACP [3]. Unlike in ACP, in μCRL and μCRL$_t$ data is an integral part of the formalism. Much research has already been conducted in the area of untimed process algebras and nowadays, there also is a lot of information available on timed process algebras. The combination of time and data, however, has been studied less intensively.

The basic elements in μCRL$_t$ are elements of a set of actions A. Actions are abstract representations of events in the real world that is described. In μCRL$_t$, an action consists of a name possibly followed by one or more data terms within brackets. An action followed by one or more data terms within brackets is called an action term.

The two elementary operators for constructing processes are the sequential composition operator, written as $p \cdot q$ and the alternative composition operator $p + q$, for process terms p and q. The process $p \cdot q$

first performs the actions of p until p terminates, and then continues with actions from q. The process $p+q$ behaves like p or q, depending on which of the two performs the first action. The operators $+$ and $\cdot$ are associative and the operator $+$ is also commutative. A constant δ denotes the inaction, or *deadlock*.

The $\_ \lhd \_ \rhd \_$ operator is the conditional operator. The process $p \lhd b \rhd q$ behaves like p if b is equal to $\mathbf{t}$ and like q if b is equal to $\mathbf{f}$. The sum operator $\sum_{d:D} P(d)$ behaves like $P(d_1) + P(d_2) + \cdots$, i.e. , as the choice between $P(d_i)$ for any data term d_i taken from D. No finiteness is required for the sort D. Data terms are considered modulo α-*conversion*.

The novelty of μCRL$_t$ over μCRL is the addition of time. To this end, a data sort $\mathbb{T}$ is assumed, modelling the time domain in μCRL$_t$. No requirements are imposed on the time domain except for the existence of a total ordering $\leq$ and the existence of a minimal element $\mathbf{0}$. This allows for choosing between e.g. discrete, dense or circular time. An additional operator for μCRL$_t$ is the *at* operator $p \lhd t$, expressing at which time actions must take place. Process $p \lhd t$ behaves like process p, with the restriction that the first action in p must take place at time t. Processes $\delta \lhd t$ model *timed deadlocks*, i.e. , processes that can idle up to time t, but no further. The neutral element for alternative composition is $\delta \lhd \mathbf{0}$. This means that for open process expressions, the following equality is required: $x + \delta \lhd \mathbf{0} = x$.

In order to deal with more complex systems, mechanisms for composing systems from a collection of components are needed. The natural operator for describing this in μCRL$_t$ is by using the *merge* operator $\|$. The parallel execution of p and q is denoted by $p \| q$. This process can start with an action from p, an action from q or with a communication (*synchronisation*) between p and q. Hence, the merge operator describes concurrency based on an interleaving semantics. Communication is defined by a partial binary function γ, defined on *action names*. Communication between two actions only occurs if γ defines their communication and if their data parameters match. By *encapsulating* actions, actions are screened from interaction with the outside world. Encapsulation is described by the operator ∂_H for $H \subseteq A$. The process $\partial_H(p)$ filters all actions in the set H.

III. A Conveyor Belt System

The case study presented in this section deals with a conveyor belt system, consisting of a single belt of length l ($l > 0$) which is controlled by a dedicated controller. Its main purpose is the transportation of trays of a fixed length n ($0 < n < \frac{1}{2}l$) from the front end of the belt to the back end of the belt. The environment is assumed to be always willing to accept trays from the belt, and regularly delivers new trays at the front end of the belt. Design requirements can be divided in three categories: safety requirements, liveness requirements and efficiency requirements. The conveyor belt system must adhere to the following requirements:
1. (safety) No collision of trays on the belt is allowed
2. (safety) The speed of the belt is at all times between 0 and v_{max}, where $0 < v_{max}$,
3. (liveness) Trays arriving at the front end of the belt are eventually delivered at the back end of the belt,

The belt is physically equipped with three sensors, positioned at the front end of the belt (position 0), half-way the belt (position $\frac{1}{2}l$) and at the back end of the belt (position l). Moreover, a motor is present, exerting a force on the belt, causing it to accelerate or decelerate. The assumption is that the torque has a one to one correspondence with the acceleration or deceleration of the belt. Setting the torque will thus change the speed of the belt.

For simplicity's sake, the following assumptions have been made in modelling the conveyor belt system:
1. The torque of the belt can be set to any value in the set $T = \{-T_m, 0, T_m\}$ ($0 < T_m$). Here, T_m denotes the maximal torque that can be applied to the motor, yielding maximal acceleration/deceleration. A torque 0 leaves the belt running at a constant speed,
2. All trays, together with their load have the same fixed mass,
3. The belt can carry up to two trays,
4. The position of a tray is determined by the position of its *fixed reference point*, which is the same for every tray.

Describing a controller, guaranteeing the above requirements, is inherently complicated by the combination of continuous and discrete dynamic behaviour.

A. Continuous Dynamic Model

Using the well established technique of *Differential Algebraic Equations* (DAEs), the continuous dynamic behaviour of the conveyor belt system can be captured. Such a model should include both the positions of the trays on the belt and the speed of the belt. Obviously, these two variables are linked. The distance covered by a tray on the belt can be determined by the speed of the belt. This leads to the following DAEs:
1. In case there is no tray on the belt: $\dot{x_0} = bf$,

2. When there is one tray on the belt: $\dot{x_1} = Ax_1 + Bf$,
3. If there are two trays on the belt: $\dot{x_2} = Cx_2 + Df$.

In these equations, the constants A, B, C and D are of appropriate dimensions, defined as:

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ b \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \text{ and } D = \begin{pmatrix} 0 \\ b \\ 0 \end{pmatrix}$$

The constant b is introduced to account for a number of physical phenomena like friction and weight, and f is the torque exerted by the motor over time. Essentially, x_0 describes merely the speed of the belt over time, whereas the variable x_n ($n \in \{1, 2\}$) describes the speed of the belt (second component), together with the position of the tray(s). The control of the conveyor belt is governed by choosing appropriate values for f over time.

B. Discrete Dynamic Model

The discrete dynamic behaviour of the conveyor belt is rather elementary. Four events are important: the arrival of a tray at either the front end of the belt or half way the belt, the departure of a tray from the belt and the setting of the torque. A μCRL$_t$ model, projecting on the discrete events is given by the following set of processes:

$$\begin{aligned}
&\textbf{proc} \quad B_0 = a_b \cdot B_1^0 + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_0 \\
&\textbf{proc} \quad B_1^0 = a_b \cdot B_2^0 + m_b \cdot B_1^1 + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_1^0 \\
&\textbf{proc} \quad B_1^1 = a_b \cdot B_2^1 + d_b \cdot B_0 + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_1^1 \\
&\textbf{proc} \quad B_2^0 = a_b \cdot \delta + m_b \cdot B_2^1 + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_2^0 \\
&\textbf{proc} \quad B_2^1 = a_b \cdot \delta + m_b \cdot B_2^2 + d_b \cdot B_1^0 + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_2^1 \\
&\textbf{proc} \quad B_2^2 = a_b \cdot \delta + d_b \cdot B_1^1 + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_2^2
\end{aligned}$$

The set of actions is $\{a_b, d_b, m_b, F_b(\beta) | \beta \in T\}$, where event a represents the arrival of a tray, event m represents the arrival of a tray at the mid-point of the belt and event d represents the departure of a tray from the belt. The events $F(\beta)$ represent the adjusting of the torque exerted by the motor. Intuitively, for $0 < i,\ j \leq i$, process B_i^j represents the belt carrying i trays of which already j trays have passed the mid-point of the belt.

C. Hybrid Dynamic Model

Clearly, both the continuous dynamic model and the discrete dynamic model are complementary. On many occasions one may either be interested in merely the continuous dynamic behaviour or the discrete dynamic behaviour of the system, but not both. Yet, not all requirements can always be proved by focusing only on the discrete or continuous behaviours. This is where the hybrid models come into play. Essentially a hybrid dynamic model is the combination of the discrete dynamic model and the continuous dynamic model. The discrete model can serve as a framework, connecting various *control modes*. Each control mode is characterised by its own set of DAEs, describing the evolution of the continuous variables of the models in a specific state.

Under the assumption that one can specify arbitrarily complex functions in the data part of μCRL$_t$, together with operations such as differentiation and integration, one can describe hybrid dynamic models in μCRL$_t$.

As an example, a hybrid dynamic model for the conveyor belt is shown below.

$$\begin{aligned}
&\textbf{proc} \quad B_0(t:\mathbb{T}, x_s:\mathbb{R}, f:T) = \\
&\textstyle\sum_{u:\mathbb{T}, x_0:\theta_0} [\quad a_b \cdot B_1^0(u, x_0(u)), f) \\
&\quad + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_0(u, x_0(u), \beta) \quad]\triangleleft u \\
&\quad \triangleleft(\forall t' : \dot{x}_0(t') = bf) \wedge x_s = x_0(t) \triangleright \delta \triangleleft \mathbf{0}
\end{aligned}$$

$$\begin{aligned}
&\textbf{proc} \quad B_1^0(t:\mathbb{T}, x_s:\mathbb{R}^2, f:T) = \\
&\textstyle\sum_{u:\mathbb{T}, x_1:\theta_1} [\quad a_b \cdot B_2^0(u, x_1(u), f) \\
&\quad + m_b \cdot B_1^1(u, x_1(u), f) \triangleleft \pi_1(x_1(u)) = \tfrac{1}{2} l \triangleright \delta \\
&\quad + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_1^0(u, x_1(u), \beta) \quad]\triangleleft u \\
&\quad \triangleleft(\forall t' : \dot{x}_1(t') = Ax_1(t') + Bf) \wedge x_s = x_1(t) \triangleright \delta \triangleleft \mathbf{0}
\end{aligned}$$

$$\begin{aligned}
&\textbf{proc} \quad B_1^1(t:\mathbb{T}, x_s:\mathbb{R}^2, f:T) = \\
&\textstyle\sum_{u:\mathbb{T}, x_1:\theta_1} [\quad a_b \cdot B_2^1(u, (x_1(u), 0), f) \\
&\quad + d_b \cdot B_0(u, f) \triangleleft \pi_1(x_1(u)) = l \triangleright \delta \\
&\quad + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_1^1(u, x_1(u), \beta) \quad]\triangleleft u \\
&\quad \triangleleft(\forall t' : \dot{x}_1(t') = Ax_1(t') + Bf) \wedge x_s = x_1(t) \triangleright \delta \triangleleft \mathbf{0}
\end{aligned}$$

$$\begin{aligned}
&\textbf{proc} \quad B_2^0(t:\mathbb{T}, x_s:\mathbb{R}^3, f:T) = \\
&\textstyle\sum_{u:\mathbb{T}, x_2:\theta_2} [\quad a_b \cdot \delta \\
&\quad + m_b \cdot B_2^1(u, x_2(u), f) \triangleleft \pi_1(x_2(u)) = \tfrac{1}{2} l \triangleright \delta \\
&\quad + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_2^0(u, x_2(u), \beta) \quad]\triangleleft u \\
&\quad \triangleleft(\forall t' : \dot{x}_2(t') = Cx_2(t') + Df) \wedge x_s = x_2(t) \triangleright \delta \triangleleft \mathbf{0}
\end{aligned}$$

$$\begin{aligned}
&\textbf{proc} \quad B_2^1(t:\mathbb{T}, x_s:\mathbb{R}^3, f:T) = \\
&\textstyle\sum_{u:\mathbb{T}, x_2:\theta_2} [\quad a_b \cdot \delta \\
&\quad + m_b \cdot B_2^2(u, x_2(u), f) \triangleleft \pi_1(x_2(u)) = \tfrac{1}{2} l \triangleright \delta \\
&\quad + d_b \cdot B_1^0(u, \pi_{(3,2)}(x_2(u)), f) \triangleleft \pi_3(x_2(u)) = l \triangleright \delta \\
&\quad + \textstyle\sum_{\beta:T} F_b(\beta) \cdot B_2^1(u, x_2(u), \beta) \quad]\triangleleft u
\end{aligned}$$

$\triangleleft(\forall t' : \dot{x}_2(t') = Cx_2(t') + Df) \wedge x_s = x_2(t) \triangleright \delta \triangleleft \mathbf{0}$

proc $B_2^2(t : \mathbb{T}, x_s : \mathbb{R}^3, f : T) =$
$\sum_{u:\mathbb{T},x_2:\theta_2} [\quad a_b \cdot \delta$
$+d_b \cdot B_1^1(u, \pi_{(3,2)}(x_2(u)), f) \triangleleft \pi_1(x_2(u)) = l \triangleright \delta$
$+\sum_{\beta:T} F_b(\beta) \cdot B_2^2(u, x_2(u), \beta) \quad] \triangleleft u$
$\triangleleft(\forall t' : \dot{x}_2(t') = Cx_2(t') + Df) \wedge x_s = x_2(t) \triangleright \delta \triangleleft \mathbf{0}$

The type θ_i is a short-hand notation for the set of functions with signature $\mathbb{T} \to \mathbb{R}^i$. The function π_i is the projection function, displaying the i^{th} coordinate of a tuple. The function $\pi_{(i,j)}$ is the projection function, yielding a pair, consisting of the i^{th} and j^{th} element of a tuple.

The structure of the hybrid dynamic model bears close resemblance with the structure of the purely discrete dynamic model of the conveyor belt. However, the occurrences of events have been made more precise by adding appropriate constraints, governed by the continuous dynamic model.

D. Control for the Conveyor Belt

The controller for the conveyor belt can again be specified in three ways: purely discrete, purely continuous and in a mixed, hybrid fashion. The main task for the controller is to ensure the conveyor belt system does not violate any requirement. Here, only a hybrid model is shown.

proc $C_0(t : \mathbb{T}, v_s : \mathbb{R}, f : T) =$
$\sum_{u:\mathbb{T},v:\theta_0} [\quad a_c \cdot C_1^a(u, v(u), f)$
$+F_c(0) \cdot C_0(u, 0, 0) \triangleleft f \neq 0 \wedge v(u) = 0 \triangleright \delta$
$+F_c(-T_m) \cdot C_0(u, v_s, -T_m) \triangleleft f \neq -T_m \wedge v_s \neq 0 \triangleright \delta] \triangleleft u$
$\triangleleft(\forall t' : \dot{v}(t') = bf) \wedge v_s = v(t) \wedge 0 \leq v(u) \leq v_{max} \triangleright \delta \triangleleft \mathbf{0}$

proc $C_1(t : \mathbb{T}, v_s : \mathbb{R}, f : T) =$
$[F_c(0) \cdot C_1^{na}(t, v_s, 0) \triangleleft v_s = v_{max} \wedge f \neq 0 \triangleright \delta$
$+F_c(T_m) \cdot C_1^{na}(t, v_s, T_m) \triangleleft v_s < v_{max} \wedge f \neq T_m \triangleright \delta$
$] \triangleleft t$

proc $C_1^{na}(t : \mathbb{T}, v_s : \mathbb{R}, f : T) =$
$\sum_{u:\mathbb{T},v:\theta_0} [\quad a_c \cdot C_1^a(u, v(u), f)$
$+F_c(0) \cdot C_1^{na}(u, 0, 0) \triangleleft f \neq 0 \wedge v(u) = 0 \triangleright \delta] \triangleleft u$
$\triangleleft(\forall t' : \dot{v}(t') = bf) \wedge v_s = v(t) \wedge 0 \leq v(u) \leq v_{max} \triangleright \delta \triangleleft \mathbf{0}$

proc $C_1^a(t : \mathbb{T}, v_s : \mathbb{R}, f : T) =$
$\sum_{u:\mathbb{T},v:\theta_0} [\quad a_c \cdot C_2(u, v(u), f)$
$+d_c \cdot C_0(u, v(u), f)$
$+F_c(0) \cdot C_1^a(u, 0, 0) \triangleleft f \neq 0 \wedge v(u) = 0 \triangleright \delta] \triangleleft u$
$\triangleleft(\forall t' : \dot{v}(t') = bf) \wedge v_s = v(t) \wedge 0 \leq v(u) \leq v_{max} \triangleright \delta \triangleleft \mathbf{0}$

proc $C_2(t : \mathbb{T}, v_s : \mathbb{R}, f : T) =$
$\sum_{u:\mathbb{T},v:\theta_0} [\quad d_c \cdot C_1^{na}(u, v(u), f)$
$+F_c(0) \cdot C_2(u, 0, 0) \triangleleft f \neq 0 \wedge v(u) = 0 \triangleright \delta] \triangleleft u$
$\triangleleft(\forall t' : \dot{v}(t') = bf) \wedge v_s = v(t) \wedge 0 \leq v(u) \leq v_{max} \triangleright \delta \triangleleft \mathbf{0}$

Obviously, the main tasks of the controller are adjusting the velocity of the belt, by setting the torque, such that it does not exceed its limits and ensuring the right distance between two successive trays on the belt. By adding the *invariant* $0 \leq v(u) \leq v_{max}$ to the conditions guarding the actions, the model explicitly expresses that actions may only happen when the requirements on the continuous variables are met.

E. A Model for the Conveyor Belt System

Using the parallel constructs of the language μCRL$_t$, a model for the conveyor belt system can be described as the parallel composition of both the model for the conveyor belt and the controller. The communication function is defined as follows:

$\gamma(a_b, a_c) = a$
$\gamma(d_b, d_c) = d$
$\gamma(m_b, m_c) = m$
$\gamma(F_b, F_c) = F$

The model for the conveyor belt system is then given by the process CBS, where

proc $CBS(t_b, t_c : \mathbb{T}, x_s, v_s : \mathbb{R}, f_b, f_c : T) =$
$\partial_H(B_0(t_b, x_s, f_b) \| C_0(t_c, v_s, f_c))$,

where $H = \{a_b, a_c, d_b, d_c, m_b, m_c, F_b, F_c\}$.

IV. Analysis and Verification of Hybrid Systems

Section III showed that the formalism μCRL$_t$ in essence can be used for writing models of hybrid systems. However, writing models without any means to verify or analyse them is less useful.

Analysis and verification often consists of proving absence of deadlock and/or showing a suitable relation between a specification and its implementation. However, in the area of hybrid systems, it is just as important to prove that a system adheres to its design requirements. In all these cases, the notions of a *Linear Process Operator* and of *invariants* are especially useful.

For a detailed account on the analysis and verifications of the conveyor belt system, please refer to [12].

A. Linear Process Operators

To prove that a system meets its requirements, or has certain desired properties can be an elaborate task, especially when parallelism is involved. The problem with parallelism is that this construct does not allow for performing proofs on directly, hence, the parallelism should first be eliminated from the models. Elimination of parallelism is eased by first rewriting the models into a *Linear Process Operator* (LPO). This process of rewriting is often refered to as *linearising*.

Basically, the LPO format encodes the process structure in the data structure, thereby significantly reducing the complexity of the process structure, but increasing the complexity of the data structure. The basic structure of an LPO Φ over data type D is as follows:

$\lambda pd. \sum_{i,\overline{e}_i:E_i,t} \; a_i(f_i(d,e_i)) \cdot t \cdot p(g_i(d,e_i)) \lhd b_i(d,e_i) \rhd \delta \cdot \mathbf{0}$

for data types E_i, D_i and functions $f_i : D \to E_i \to D_i$, $g_i : D \to E_i \to D$, $b_i : D \to E_i \to \mathbb{B}$. The bound variable p ranges over processes parameterised with a datum of sort D.

Rewriting an ordinary specification into an LPO can be very tedious in some cases. Therefore, in the end, tool support will be necessary. For (untimed) μCRL, already an automated tool exists for rewriting an arbitrary μCRL model into an LPO. For μCRL_t, however, currently no tool support is available for the rewriting.

B. Invariants

In the area of model checking, proving that a system adheres to various (design) requirements is rather elementary. However, proving that a system meets its design requirements using syntactic manipulation only is far from standard.

Requirements can often be stated as predicates over the data structure of a given model. Non-violation of a requirement means that the requirement should hold invariably, hence, the notion of invariants is sufficient for proving conformance of a model to its (design) requirements. Formally, an invariant of an LPO is defined as a function $\mathcal{I} : D \to \mathbb{B}$, such that for all $i \in I$ and for all $d : D, e_i : E_i$

$b_i(d,e_i) \wedge \mathcal{I}(d) \;\Rightarrow\; \mathcal{I}(g_i(d,e_i))$

Intuitively, this boils down to proving that the execution of an action, will not violate the invariant. By definition, deadlocks cannot violate an invariant.

When dealing with hybrid systems, actual design requirements can often be stated as predicates over the continuous variables occuring in a model. By proving these predicates are invariant in the model, it is automatically proved that the model adheres to its requirements, provided the system is deadlock-free.

V. Conclusion

In this paper, several of the possibilities for applying the process algebra μCRL_t to the area of hybrid systems are shown by means of a case study. Some investigations into the theory of differential equations in combination with abstract datatypes is needed to see whether the theory of μCRL_t needs extension to deal with DAEs. Furthermore, already several formalisms exist that are tailor-made to deal with hybrid systems. Investigation into the relation between these formalisms and μCRL_t are needed to find out the suitability of μCRL_t for this area. Also, the concepts of linear process operators and invariants in a timed setting require some more attention.

References

[1] R. Alur, T.A. Henzinger, and E.D. Sontag, editors. *Hybrid Systems III*, Lecture Notes in Computer Science 1066. Springer-Verlag, 1996.

[2] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems II*, Lecture Notes in Computer Science 999. Springer-Verlag, 1995.

[3] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1990.

[4] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. In *Computer Networks and ISDN Systems*, 1987.

[5] D. A. van Beek, J. E. Rooda, and S. H. F. Gordijn. Hybrid modelling in discrete-event control system design. In *Proceedings of the 1996 IEEE/IMACS Conference on Computational Engineering in Systems Applications*, 1996.

[6] J.F. Groote. The syntax and semantics of timed μCRL. Technical Report SEN-R9709, CWI, June 1997.

[7] J.F. Groote and A. Ponse. The syntax and semantics of μCRL. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Algebra of Communicating Processes '94*, Workshops in Computing Series, pages 26–62. Springer Verlag, 1995.

[8] J.F. Groote and J.J. van Wamel. Analysis of three hybrid systems in timed μCRL. Technical Report SEN-R9815, CWI, September 1998.

[9] R.L. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors. *Hybrid Systems*, Lecture Notes in Computer Science 736. Springer-Verlag, 1993.

[10] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[11] R. Milner. *Communication and Concurrency*. Prentice Hall International, 1989.

[12] T.A.C. Willemse. The analysis of a conveyor belt system: a case study in hybrid systems and timed μCRL. Computer Science Reports CSR 99-10, Eindhoven University of Technology, 1999.