Proceedings of the 8th IEEE Mediterranean Conference
on Control and Automation (MED 2000)
Rio, Patras, GREECE 17-19 July, 2000

ANALYSIS OF CONTROLLED HYBRID PROCESSING SYSTEMS BASED ON APPROXIMATION BY TIMED AUTOMATA USING INTERVAL-ARITHMETIC

OLAF STURSBERG[‡], STEFAN KOWALEWSKI[†]

[‡] Process Control Lab (CT-AST), Dept. of Chemical Eng., University of Dortmund, D-44221 Dortmund (Germany), o.stursberg@ct.uni-dortmund.de

[†] Research and Development (FV/SLD), Robert Bosch GmbH, D-60441 Frankfurt (Germany), stefan.kowalewski@de.bosch.com*

Abstract. This contribution describes an approach to investigate reachability properties for a class of controlled hybrid systems. The continuous dynamics of these so-called *Switched Continuous Systems* (SCS) is selected by the discrete output of a logic controller. While reachability analysis is in general undecidable for this class of systems, the analysis is known to terminate for the class of *Timed Automata* (TA). In order to make reachability analysis amenable to the control structure, we propose an approximating algorithm to convert a SCS into a TA. Different modifications and extensions of the procedure are given and the approach is illustrated by the application to a chemical reaction system.

Key Words. Hybrid systems, model transformation, timed automata, reachability analysis, verification.

1. INTRODUCTION

One topic which has been studied intensively in the context of hybrid systems over the last few years is the verification of safety properties (see e. g. [3, 7, 10]). To proof safe operation of a system with either purely discrete or mixed discrete-continuous dynamics *reachability analysis* is applied: Given an initial set of states S_0, the set of reachable states S_R is computed and it is checked whether a set of forbidden states S_F intersects with S_R ($S_R \cap S_F = \varnothing$?). This method has been investigated in various fields of application as the verification of electronic circuits, the design of autonomous traffic systems and the investigation of processing systems – which is our domain of interest. Specifically, we want to determine whether a given logic/supervisory controller guarantees that devices like chemical reactors or vessels for the storage or separation of substances will never reach states which constitute a danger to the personnel, the equipment or the environment of the processing system. We analyze a hybrid processing system by mapping the original model first into a timed automaton (TA) and then check safety properties of the latter with existing verification algorithms. In contrast to methods which pursue the same objective and which we have previously published (compare e. g. to [5, 6, 8]) the completeness of the model transformation is always guaranteed and by using interval-arithmetic the computationally costly step of simulation is avoided.

While the following section defines the control structure under consideration more formally, Sec. 3. contains a description of the algorithmic procedure which converts the original hybrid system into TA. This section also states some properties and possible extensions of the approach, and Sec. 4. illustrates it by application to a chemical reaction system.

2. THE CONTROL STRUCTURE

The considered structure of a hybrid processing system controlled by a logic controller is shown in Fig. 1: The process model is initially set up as a

*The research presented in this paper was performed while the author was with the University of Dortmund.

Switched Continuous System which we define as follows:

Definition 1
A Switched Continuous Systems (SCS) is given by $P_{SCS} = \{U, \boldsymbol{X}, \boldsymbol{X}_0, \boldsymbol{f}\}$ *with the finite set* $U = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p\}$ *of discrete inputs vectors* $\boldsymbol{u}_k = (u_{k,1}, \ldots, u_{k,m})$, $u_{k,i} \in \mathbb{R}$ *and the continuous state space* $\boldsymbol{X} \subset \mathbb{R}^n$. *We assume that the state space is bounded*[1], *i. e.* $x_j \in [x_{j,min}, x_{j,max}]$, $j \in \{1, \ldots, n\}$, *and* $\boldsymbol{X}_0 \subset \boldsymbol{X}$ *is a compact set of initial states. The continuous dynamics of the SCS is determined by ordinary differential equations* $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$*: For* $\boldsymbol{x}(t_0) \in \boldsymbol{X}_0$, *a trajectory of* P_{SCS} *is the continuous solution* $\boldsymbol{x}(t)$ *of the ODEs on a time interval* $t \in [t_0, t_e]$, *where the discrete input vector* $\boldsymbol{u}_k$ *switches finitely often at distinct points of time* $t_s \in [t_0, t_e]$, *and* $\boldsymbol{u}_k$ *is held constant otherwise.*

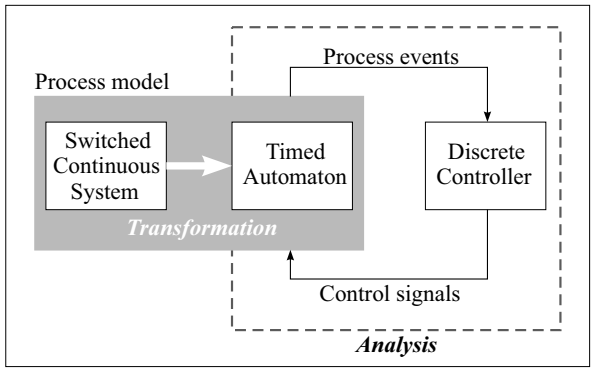

Fig. 1: The control setting.

The discrete inputs $\boldsymbol{u}_k \in U$ are the control signals by which the controller determines the continuous evolution of P_{SCS}. The output of the system, so-called *process events*, are generated if a process variable x_j crosses a specified threshold (see Sec. 3.). These events are reported to a discrete controller for which we assume that its logic can be modeled by a finite state automaton $C = \{PE, Z^C, Z_0^C, U, \phi^C, \gamma^C\}$. The set U is the same as in P_{SCS}, and Z^C denotes the discrete state set with a subset $Z_0^C \subset Z^C$ of initial states. The controller reads a process event $\varepsilon \in PE$, and depending on ε a transition $z_1 \rightarrow z_2$ between two states $z_1, z_2 \in Z^C$ is triggered according to the state transition function $\phi^C : PE \times Z^C \rightarrow Z^C$. When the state z_2 is reached, a new control signal $\boldsymbol{u}_k$ can be emitted by the output function $\gamma^C : Z^C \rightarrow U$.

The target of our model transformation is a timed automaton which we define similarly to [1] by:

[1] If this does not apply, one can obtain a bounded state space e. g. by the transformation $\widetilde{x}_j = \kappa(x_j) = x_j / \sqrt{1 + x_j^2 / \widetilde{x}_{j,b}^2}$ when $\widetilde{x}_j$ denotes the transformed variable which is defined on the interval $[-\widetilde{x}_{j,b}, -\widetilde{x}_{j,b}]$, $\widetilde{x}_{j,b} \in \mathbb{R}$. Additionally, the function $\boldsymbol{f}$ has to be transformed by $\dot{\widetilde{x}} = \frac{d\kappa(\boldsymbol{x})}{d\boldsymbol{x}} \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$.

Definition 2
A Timed Automaton (TA) is an 8-tuple $P_{TA} = \{U, Z^P, Z_0^P, PE, Var, \phi^P, Inv, \gamma^P\}$ *in which* U, PE, Z^P, *and* Z_0^P *again denote the sets of control signals, process events, discrete states, and initial states respectively.* $Var = \{v_1, \ldots, v_p\}$ *is the set of clock variables where each clock* $v \in Var$ *evolves with the rate* $\dot{v} = 1$*; all clock variables are initialized to* $v_j := 0$. *The state transition function* $\phi^P : U \times Z^P \times \delta(v) \rightarrow Z^P$ *determines transitions* $z_1 \rightarrow z_2$ *for two states* $z_1, z_2 \in Z^P$ *depending on a current control signal* $\boldsymbol{u}_k \in U$ *and on so-called guards* $\delta(v)$ *which are conjunctions of expressions* $(\sum_{1 \leq k \leq p} a_k \cdot v_k) \sim c$, $a_k \in \mathbb{Q}$, $\sim \in \{<, \leq, =, \geq, >\}$, $c \in \mathbb{Q}$. *A clock value can be reset to zero* $(v := 0)$ *with a transition. The invariant function* $Inv(z)$ *specifies which conditions must be true for the clock values if* P_{TA} *is in state* z*; the invariants are of the same type as the guards. Finally, the output function* $\gamma^P : Z \times Z \rightarrow PE$ *can generate a process event when a transition occurs. A run of of* P_{TA} *is a sequence* $\stackrel{\tau_0}{\rightarrow} z_0 \stackrel{\tau_1}{\rightarrow} z_1 \stackrel{\tau_2}{\rightarrow} z_2 \stackrel{\tau_3}{\rightarrow} \ldots$ *with* $z_i \in Z^P$, $z_0 \in Z_0^P$. *The values* τ_i $(i = 0, 1, 2, \ldots)$ *with* $\tau_i \leq \tau_{i+1}$ *denote the points of time at which transitions according to* ϕ^P *occur, and they correspond to the evaluations of a clock* $v \in Var$ *which is initialized at* τ_0 *but never reset again.*

Instead of using the input and output signals U and PE to model the communication between the process model P_{TA} and the controller C one could as well use synchronization labels. In both cases, the generation of a process event instantaneously leads to a new controller signal, i. e. there is no delay of the controller response. Also for a process model given as P_{TA}, we assume that the number of process events generated on a bounded time interval is finite. We now propose a transformation algorithm to convert a process model of the type P_{SCS} into the type P_{TA} in order to be able to perform reachability analysis for the controlled system.

3. TRANSFORMATION INTO TIMED AUTOMATA

The transformation scheme is basically split into two steps: The first partitions the continuous state space of the SCS into a finite number of elements and assigns a discrete state of P_{TA} to each element. Transitions among these states are then determined in the second step based on the dynamics of P_{SCS}.

3.1. State Space Partitioning

A first partition of $\boldsymbol{X}$ is obtained from considering those thresholds for each state variable x_j for which the controller has to compute a new control signal. These thresholds and the bounding values for x_j are specified as a so-called *ordered landmark set*:

$$L_j = \{l_{j,1} = x_{j,min}, l_{j,2}, \ldots, l_{j,p_j} = x_{j,max}\} \qquad (1)$$

where $l_{j,k+1} > l_{j,k}$. The set PE (introduced in Sec. 2.) is formed by assigning a process event $\varepsilon_{j,k}(x_j, l_{j,k})$ to a landmark $l_{j,k}$ with $k \in \{1, \ldots, p_j\}$:

$$\varepsilon_{j,k} = \begin{cases} +1 \text{ if } x_j(t < t_e) < l_{j,k} \wedge x_j(t = t_e) = l_{j,k}, \\ -1 \text{ if } x_j(t < t_e) > l_{j,k} \wedge x_j(t = t_e) = l_{j,k}, \\ 0 \text{ else.} \end{cases} \tag{2}$$

The example in Fig. 2 shows a partition which is initially given by $L_1 = \{l_{1,0}, l_{1,2}, l_{1,4}\}$, $L_2 = \{l_{2,0}, l_{2,2}, l_{2,3}\}$. Process events $PE = \{\varepsilon_{1,2}, \varepsilon_{2,2}\}$ are generated if $l_{1,2}$ and $l_{2,2}$ are crossed in appropriate directions. In order to obtain a more regular $\boldsymbol{X}$-partition, additional values can be inserted such that the distance $\lambda_{j,k} = l_{j,k+1} - l_{j,k}$ of adjacent landmarks does not exceed a specified bound. In Fig. 2 the landmarks sets are extended by $l^*_{1,1}$, $l^*_{1,3}$, and $l^*_{2,1}$ respectively (and the indexing is adapted).

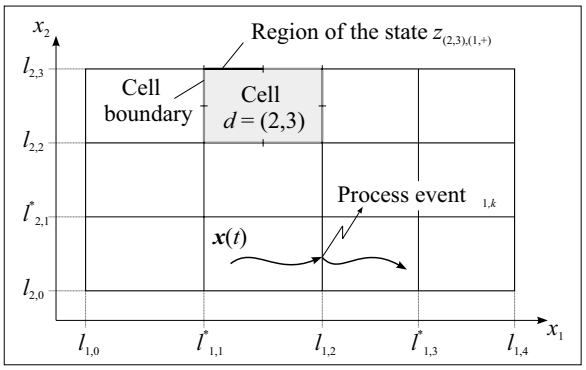

Fig. 2: State space partitioning and event generation.

The extended landmark sets partition $\boldsymbol{X}$ into $\pi = \prod_j p_j$ cells which are of hyper-rectangular geometry. We define a mapping $D_{\boldsymbol{X}} : \{\boldsymbol{X}, L\} \to \Delta$ (with $L = \bigcup_j L_j$) that maps the set of cells into a set $\Delta = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_\pi\}$ of *index vectors* $\boldsymbol{d}$. Each component d_j of an index vector contains the number of the corresponding interval of x_j:

$$d_j = D_{\boldsymbol{X}}(x_j, L_j) = \begin{cases} k \text{ if } x_j \in [l_{j,k-1}, l_{j,k}[, \\ p_j \text{ if } x_j \in [l_{j,p_j-1}, l_{j,p_j}]. \end{cases} \tag{3}$$

It was shown in [9] that a transformation method that is based only on a cell partition requires some additional effort to allow lower time bounds for transitions which are different from a guard "$t \geq 0$" (i. e. the TA must be extended by discrete variables). This can be avoided and the accuracy of representing the continuous state $\boldsymbol{x}$ in P_{TA} can be improved by a further partition: For each of the $(n-1)$-dimensional hyperplanes which constitute the boundary of a cell $\boldsymbol{d}$ a uniform grid is introduced. The granularity of the grid is determined by choosing parameters $\mu_1, \ldots, \mu_n \in \mathbb{N}$, where μ_j specifies the number of elements in which an interval $[l_{j,k}, l_{j,k+1}]$ is partitioned. Figure 2 shows the case that each side of the cell $\boldsymbol{d} = (2,3)$ is divided into two elements ($\mu_1 = \mu_2 = 2$). A discrete state $z_{\boldsymbol{d},\boldsymbol{g}}$ is assigned to each boundary element which is obtained from this partitioning. The index vector $\boldsymbol{g} = (g_1, \ldots, g_n)$ determines the location of the state on the boundary, i. e. a component g_j specifies the number of the element counting from the landmark with the lower k-index. (The invariant coordinate of the boundary element gets a '+' or '-' denoting the lower or upper side of the cell, as for the state $z_{(2,3),(1,+)}$ in Fig. 2). Using this scheme, the set Z^P contains a number of

$$\pi^* = \sum_{j=1}^{n} \left((p_j + 1) \cdot \prod_{k=1, k \neq j}^{n} \mu_k \cdot p_k \right) \tag{4}$$

discrete states. Whereas the internal region of a cell is not explicitly assigned to a discrete state in Z^P, the transformation method described in the next section ensures nevertheless that the behaviour of the SCS is completely captured by P_{TA}: The idea is to represent the behavior of P_{SCS} in the timed automaton by just modeling all intersections of continuous trajectories with the cell boundaries.

3.2. Determination of Transitions based on Interval-Arithmetic

The following algorithm for determining the transitions of P_{TA} (according to ϕ^P) uses interval-arithmetic to evaluate the continuous dynamics of P_{SCS}. This choice accounts for the fact that we have to compute the gradients $\dot{\boldsymbol{x}}$ for regions of the state space instead of single points $\boldsymbol{x}$ only. Let $\mathcal{I}$ be the set of all bounded intervals $I = [i_1, i_2] = \{x \in \mathbb{R} | i_1 \leq x \leq i_2\}$ where $i_1, i_2 \in \mathbb{R}$. Interval-arithmetic (see e. g. [4]) provides rules for the standard binary operations $A \omega B$ with $A, B \in \mathcal{I}$, $\omega \in \{+, -, \cdot, /\}$ and numerous unary operators $\rho(I) = [\min(\rho(x)), \max(\rho(x))]$, $x \in I$ (as e. g. e^I and I^n). If we replace all operations in $\boldsymbol{f}$ by the corresponding interval operations, we can rewrite the continuous dynamics of P_{SCS} to:

$$[\dot{\boldsymbol{x}}] = \widehat{\boldsymbol{f}}([\boldsymbol{x}], \boldsymbol{u}), \tag{5}$$

where $\widehat{\boldsymbol{f}}$ denotes the interval functions and $[\boldsymbol{x}] = ([x_{1,1}, x_{1,2}], \ldots, [x_{n,1}, x_{1,2}])$, $(x_{j,1}, x_{j,2} \in \mathbb{R})$ is the *state interval vector*. The gradients are obtained as the *gradient interval vector* $[\dot{\boldsymbol{x}}] = ([\dot{x}_{1,1}, \dot{x}_{1,2}], \ldots, [\dot{x}_{n,1}, \dot{x}_{1,2}])$.

The algorithm to determine the transitions of P_{SCS} basically consist of three tests to check whether a transition $z_1 \to z_2$ for a pair of states $(z_1, z_2 \in Z^P)$ is possible. In order to reduce the overall computational effort, this structure was chosen such that the computation for $z_1 \to z_2$ stops immediately if a test reveals that the transition does not exist in P_{TA}:

Algorithm 1
Do the following for each pair of a control signal $\boldsymbol{u} \in U$ *and a cell* $\boldsymbol{d} \in \Delta$*:*

1. *Compute the vector* $[\dot{\boldsymbol{x}}]_{\boldsymbol{d}}$ *(which contains all gradients occurring in the cell* $\boldsymbol{d}$*) through Eq. 5 by using the state interval vector* $[\boldsymbol{x}]_{\boldsymbol{d}} = ([l_{1,d_1-1}, l_{1,d_1}], \ldots, [l_{n,d_n-1}, l_{n,d_n}])$.

2. *Choose each state* $z_{\boldsymbol{d},g_1}$ *that is assigned to cell* $\boldsymbol{d}$ *as the source state of a transition, and:*

   (a) *compute the gradient vector* $[\dot{\boldsymbol{x}}]_{\boldsymbol{d},g_1}$ *for the region of the source state* $z_{\boldsymbol{d},g_1}$ *(let* j *be the invariant coordinate of* $[\boldsymbol{x}]_{\boldsymbol{d},g_1}$*).*

   (b) ***Test A****: Check whether the sign of* $[\dot{x}_j]_{\boldsymbol{d},g_1}$ *allows a transition from state* $z_{\boldsymbol{d},g_1}$ *into the cell* $\boldsymbol{d}$ *– this applies for* $max\{[\dot{x}_j]_{\boldsymbol{d},g_1}\} \geq 0$ *if* $z_{\boldsymbol{d},g_1}$ *lies on the lower cell border (in coordinate* j*), respectively for* $min\{[\dot{x}_j]_{\boldsymbol{d},g_1}\} \leq 0$ *if* $[\boldsymbol{x}]_{\boldsymbol{d},g_1}$ *is part of the upper cell border. If this test fails the procedure continues with the next state in step (2.a).*

   (c) *If each component* $[\dot{x}_j]_{\boldsymbol{d},g_1}$ $(j \in \{1, \ldots, n\})$ *contains the value zero, a self-loop transition* $z_{\boldsymbol{d},g_1} \rightarrow z_{\boldsymbol{d},g_2}$ *is possible.*

   (d) *Choose each state of cell* $\boldsymbol{d}$ *(except of* $z_{\boldsymbol{d},g_1}$*) as possible target state* $z_{\boldsymbol{d},g_2}$ *of a transition, and:*

      i. ***Test B****: Based on the signs of the vector* $[\dot{\boldsymbol{x}}]_{\boldsymbol{d}}$*, check whether the transition* $z_{\boldsymbol{d},g_1} \rightarrow z_{\boldsymbol{d},g_2}$ *is possible – this applies if for* ***each*** *coordinate* $j \in \{1, \ldots, n\}$ *a positive (negative) value is contained in* $[\dot{x}_j]_{\boldsymbol{d}}$ *while the distance* s_j *between the regions* $[\boldsymbol{x}]_{\boldsymbol{d},g_2}$ *and* $[\boldsymbol{x}]_{\boldsymbol{d},g_1}$ *is positive (negative). If not, the procedure continues with the next target state (step 2.d).*

      ii. *For each coordinate, compute a time interval:*

$$\triangle t_j = \left[\frac{s_{j,min}}{max\{[\dot{x}_j]_{\boldsymbol{d}}\}}, \frac{s_{j,max}}{min\{[\dot{x}_j]_{\boldsymbol{d}}^{>0}\}} \right] \quad (6)$$

if the distance s_j *is positive, or a time interval:*

$$\triangle t_j = \left[\left| \frac{s_{j,min}}{min\{[\dot{x}_j]_{\boldsymbol{d}}\}} \right|, \left| \frac{s_{j,max}}{max\{[\dot{x}_j]_{\boldsymbol{d}}^{<0}\}} \right| \right] \quad (7)$$

if s_j *is negative. The parameters* $s_{j,min}$ *and* $s_{j,max}$ *denote the distance which must be covered at least, or at most respectively in coordinate* j*. The indices* $^{<0}$ *and* $^{>0}$ *mark the negative and positive part of an interval.*

      iii. ***Test C****: Check whether the intersection of the time intervals of all coordinates is empty:*

$$\triangle t_{\boldsymbol{d},g_1 \rightarrow g_2} = \bigcap_{j=1,\ldots,n} \triangle t_j = \varnothing ? \quad (8)$$

Then, the transition $z_{\boldsymbol{d},g_1 \rightarrow g_2}$ *does not exist and the procedure continues for the next target state in step 2.d.*

      iv. *For* $\triangle t_{\boldsymbol{d},g_1 \rightarrow g_2} \neq \varnothing$ *the transition* $z_{\boldsymbol{d},g_1} \rightarrow z_{\boldsymbol{d},g_2}$ *is possible provided the guard* $v \in \triangle t_{\boldsymbol{d},g_1 \rightarrow g_2}$ *is true for the given* $\boldsymbol{u} \in U$*.*

The procedure is illustrated in Fig. 3: If the first component of $[\dot{\boldsymbol{x}}]_{\boldsymbol{d},(-,3)}$ contains a value ≥ 0, a continuous trajectory exists through which the cell $\boldsymbol{d}$ is not left immediately, i. e. a transition within cell $\boldsymbol{d}$ is possible (test A passed). For the combination of $z_{\boldsymbol{d},(-,3)}$ and a target state $z_{\boldsymbol{d},(4,-)}$, test B checks whether the values in $[\dot{\boldsymbol{x}}]_{\boldsymbol{d}}$ have appropriate signs to allow the transition $z_{\boldsymbol{d},(-,3)} \rightarrow z_{\boldsymbol{d},(4,-)}$; this is the case for $\max\{[\dot{x}_1]_{\boldsymbol{d}}\} > 0$, $\min\{[\dot{x}_2]_{\boldsymbol{d}}\} < 0$. Finally, the time intervals $\triangle t_1$ and $\triangle t_2$ are computed, and if these intervals overlap the transition $z_{\boldsymbol{d},(-,3)} \rightarrow z_{\boldsymbol{d},(4,-)}$ is assessed to be possible (test C).

If carried out for all cells and all control signals, this algorithm maps all continuous trajectories of P_{SCS} which connect different parts of a cell boundary into discrete transitions. The trajectories which end inside of a cell region $[\boldsymbol{x}]_{\boldsymbol{d}}$ are captured as self-loop transitions of those states $z_{\boldsymbol{d},g}$ that pass test A. Furthermore, the trajectories

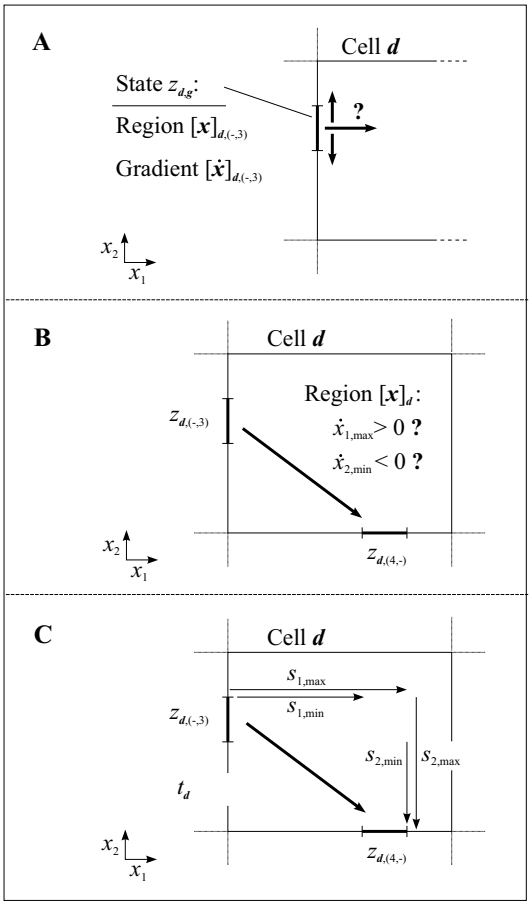

Fig. 3: Determination of transitions.

which start inside of a cell $\boldsymbol{d}$ have to be considered (this is only relevant for the cells which include the initial region of P_{SCS}): In this case, an additional state $z_{\boldsymbol{d}}$ is assigned to cell $\boldsymbol{d}$ and transitions from $z_{\boldsymbol{d}}$ to all states $z_{\boldsymbol{d},\boldsymbol{g}}$ are introduced with an guard which includes the transition times $\triangle t_{\boldsymbol{d},\boldsymbol{g}_1\rightarrow\boldsymbol{g}_2}$ of all transitions which have been computed for $\boldsymbol{d}$ with the current $\boldsymbol{u}$.

Eventually, P_{TA} results from the following assignments: the sets U of P_{SCS} and P_{TA} are equivalent; Z^P follows from the partitioning in Sec. 3.1.; Z_0^P is the set of states which are assigned to the cells that contain $\boldsymbol{X}_0$; PE is the set of events that are obtained from the initial cell partition; Var contains just one clock variable v; the function ϕ^P specifies the transitions which are determined from the procedure described above, and the clock v is reset with each transition; the transition guards are the conjunctions of a control signal $\boldsymbol{u}$ and a time constraint $v \in \triangle t_{\boldsymbol{d},\boldsymbol{g}_1\rightarrow\boldsymbol{g}_2}$; $Inv(z_{\boldsymbol{d},\boldsymbol{g}})$ is given by $0 \leq v \leq t_{\boldsymbol{d},\boldsymbol{g},max}$ where $t_{\boldsymbol{d},\boldsymbol{g},max}$ denotes the largest upper time bound of all transitions through which $z_{\boldsymbol{d},\boldsymbol{g}}$ is left; the function γ^P generates an event $\varepsilon_{j,k} \in PE$ when a transition into a state $z_{\boldsymbol{d},\boldsymbol{g}}$ occurs which is assigned to a landmark value $l_{j,k}$ of the initial cell partition.

3.3. Modifications and Extensions

To take into account that the parameters occurring in models of processing systems are often not known precisely, the method can easily be extended to the case of uncertain parameters. If a parameter in $\boldsymbol{f}$ is given by $q \in I = [q_{min}, q_{max}]$ rather than by a constant $q \in \mathbb{R}$, the computation according to Eq. 5 is straightforward: To obtain the gradient interval-vector $[\dot{\boldsymbol{x}}]$ by interval-arithmetic, a parameter interval $[q_{min}, q_{max}]$ is dealt with in the same way as with the interval argument $[\boldsymbol{x}]$.

It shall be noted that due to the nature of interval-arithmetic the computation of $[\dot{\boldsymbol{x}}]$ for an interval vector $[\boldsymbol{x}]^*$ according to Eq. 5 can lead to large over-approximations for some functions $\boldsymbol{f}$. This means that for the diameter $\langle[\dot{x}_j]\rangle := \dot{x}_{j,2} - \dot{x}_{j,1}$ of a component of $[\dot{\boldsymbol{x}}]$ applies: $\kappa = \langle[\dot{x}_j]\rangle / \langle[\dot{x}_j]^{real}\rangle \gg 1$ (while a conservative approximation is assured). $\langle[\dot{x}_j]^{real}\rangle$ denotes the diameter of the gradient interval $[\dot{x}_j]^{real}$ which would be obtained if $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ was evaluated for all $\boldsymbol{x} \in [\boldsymbol{x}]^*$. In these cases it might be advantageous to retrieve $[\dot{\boldsymbol{x}}]$ from constrained optimization instead (for a fixed $\boldsymbol{u}$):

$$[\dot{x}_j] = \left[\min_{\boldsymbol{x}\in[\boldsymbol{x}]^*} \left(f_j(\boldsymbol{x})\right), -\min_{\boldsymbol{x}\in[\boldsymbol{x}]^*} \left(-f_j(\boldsymbol{x})\right) \right], \qquad (9)$$

with $j \in \{1, \ldots, n\}$. While this modification produces smaller values of the ratio $\langle[\dot{x}_j]\rangle / \langle[\dot{x}_j]^{real}\rangle$ for many functions f_j, a conservative approximation ($[\dot{x}_j] \supseteq [\dot{x}_j]^{real}$) is only guaranteed for convex functions f_j. The computational effort is usually much smaller if interval-arithmetic is used.

Another extension worth to be mentioned is the case of switched continuous systems for which the state space $\boldsymbol{X}$ is divided into a set of r disjunct regions $\boldsymbol{X}_i$ with $\bigcup_{i=1}^{r} \boldsymbol{X}_i = \boldsymbol{X}$ and $\boldsymbol{X}_i \cap \boldsymbol{X}_j = \emptyset$ ($i \neq j$, $j \in \{1, \ldots, r\}$), and where a specific ODE-system $\dot{\boldsymbol{x}} = \boldsymbol{f}_i(\boldsymbol{x}, \boldsymbol{u})$ is assigned to each $\boldsymbol{X}_i$. The method described above can be applied straightforwardly if the boundary between each pair of adjacent regions is an $(n-1)$-dimensional hyperplane that is orthogonal to one coordinate ($x_j = l_{j,k}$). Then the boundary is inserted into the landmark set L_j and the computation of $[\dot{\boldsymbol{x}}]_{\boldsymbol{d}}$ is carried out under consideration which $\boldsymbol{f}_i$ is valid in the cell $\boldsymbol{d}$. In the case that a region boundary is given by a hyperplane that is not orthogonal to any coordinate, the method can be applied if one can find a similarity transformation which results in an orthogonal cell partition of $\boldsymbol{X}$.

4. AN APPLICATION EXAMPLE

The application to a chemical reaction system (see also [6]) is used to illustrate the approach from Sec. 3.1./3.2.: The dynamical behavior of the chemical reactor is modeled by the SCS:

$$\begin{aligned}
\dot{x}_1 &= s_1 k_1 (k_2 - x_1)/x_3 - k_3 x_1^2 \exp(-k_4/x_2), \\
\dot{x}_2 &= (s_1 k_1 k_5 + s_2 k_6 k_7 - x_2(s_1 k_1 + s_2 k_6))/x_3 \\
&\quad - k_8 x_1^2 \exp(-k_4/x_2), \\
\dot{x}_3 &= s_1 k_1, \qquad (10)
\end{aligned}$$

in which the state variables are the concentration of the reacting substance (x_1), the reactor temperature (x_2), and the liquid level (x_3). The constants k_1 to k_8 are the model parameters and the discrete input vector $\boldsymbol{u} = (s_1, s_2)$ contains the switching variables s_1, s_2. These variables model the valve settings for the inflow of reactants (s_1) respectively the reactor cooling (s_2), where $s_i = 1$ refers to an opened and $s_i = 0$ to a closed valve. Using the input set $U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3\} = \{(1,0), (0,0), (0,1)\}$, a discrete controller realizes the following three phases of the process: The filling of the reactor is started by setting the input to $\boldsymbol{u}_1$, and a reaction timer is started. (This timer can either be included in the SCS as additional state variable or in a controller model of the TA-type.) If an upper limit x_3^u of x_3 is reached, the controller switches the input to $\boldsymbol{u}_2$, i. e. the reaction proceeds without cooling. Since the reaction is exothermal the temperature x_2 increases. In order to avoid that the reactor content starts to boil, the cooling is switched on ($\boldsymbol{u}_3$) at a specified temperature threshold x_2^s (but x_2 shows an overshooting behavior for a given efficiency of the cooling device). It is important that the cooling is not switched on at a too low temperature, because then the product yield will be low. For the controlled system, we want to investigate whether the threshold x_2^s was chosen such that a boiling temperature x_2^b is never reached **and** that the product yield (determined via x_1) exceeds a de-

sired value when a specified reaction time is elapsed (measured by the reaction timer).

To solve the problem the switching rules sketched above are transformed into the controller model first, i. e. depending on the measured signals (the process events x_3^u, x_2^s) the discrete inputs $\boldsymbol{u}$ are set appropriately. The state space $\boldsymbol{X}$ of SCS is partitioned into cells by 4 landmarks for x_1, 4 (respectively 5) landmarks for x_2 and 3 landmarks in L_3. The values 2 or 3 are chosen for the partitioning parameters μ_j. Depending on these values the state space of the P_{TA}-model comprises 300 to 882 discrete states. The transitions including guards and the invariants were determined by applying a MATLAB implementation of the algorithm in Sec. 3.2. Requiring a computation time of 7 to 36 minutes (on a Pentium 200 PC) models of the P_{TA}-type with 4547 to 17896 transitions were generated.

In order to analyze the composition of the models P_{TA} and C with the model-checker HYTECH [2], both models are transformed in the corresponding input language. We supplied the requirement specification that none of the states assigned to the high temperature level (x_2^b) must be reached and that an x_1-state corresponding to a high product yield has to be reached within the reaction time. For different configurations (i. e. different values for k_1 to k_8 and for the partitioning parameters) we could investigate which threshold value x_2^s is suitable such that both requirements are fulfilled. While the analysis needed a computation time of only 1 to 4 minutes, the memory requirement of HYTECH for the largest models exceeded 150 MB. Hence, the size of this example roughly marks the complexity of systems which can be analyzed with HYTECH.

5. CONCLUSIONS

The scheme to transform P_{SCS} into P_{TA} allows to perform reachability analysis for the class of switched continuous systems algorithmically. By applying model checking tools for timed automata to the transformed model (in conjunction with a behavior specification such as a set of forbidden states) we gain information about safety properties of the original model. Obviously, the model conversion is an approximation that has an impact on the verification result: On the one hand, the approximation is conservative since the use of the gradient intervals $[\dot{\boldsymbol{x}}]_d$ ensures that all continuous trajectories of the SCS are represented by discrete transitions in P_{TA} – hence, if a forbidden state is found not to be reachable in P_{TA} the same applies for the original system. On the other hand, the model P_{TA} contains behavior without a correspondence in P_{SCS} for two reasons: First, the use of interval-arithmetic operations leads in general to gradient intervals $[\dot{\boldsymbol{x}}]$ that are larger ($\kappa > 1$) than those which would be obtained if one evaluated the function $\boldsymbol{f}$ for all points in $[\boldsymbol{x}]_d$. Secondly, the algorithm permits that each value in $[\dot{\boldsymbol{x}}]_d$ can be valid in each point within the cell $\boldsymbol{d}$, i. e. it is abstracted from the change of gradients along a continuous trajectory leading to an over-approximation of the transition guards. Both points promote the situation that a forbidden state is found to be reachable in verifying P_{TA} whereas this result does not hold for P_{SCS}. Thus, a repetition of the procedure with a refined partition is helpful to reveal whether one can trust a negative verification result.

Acknowledgements: Thanks go to S. Engell for helpful comments and to S. Panek for his part in implementing the transformation algorithm.

6. REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[2] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1/2):110–122, 1997.

[3] T. A. Henzinger and S. Sastry, editors. *Hybrid Systems – Computation and Control (HSCC'98)*, volume 1386 of *Lecture Notes in Computer Science*. Springer, 1998.

[4] R. B. Kearfott. Interval computations: Introduction, uses, and resources. *Euromath Bulletin*, 2(1):95–112, 1996.

[5] S. Kowalewski, S. Engell, J. Preussig, and O. Stursberg. Verification of logic controllers for continuous plants using timed condition/event system models. *Automatica – Special Issue on Hybrid Systems*, 35(3):505–518, 1999.

[6] S. Kowalewski, S. Engell, and O. Stursberg. Verification of logic controllers for continuous plants. In P. Frank, editor, *Advances in Control (Highlights of ECC'99)*, pages 345–389. Springer, 1999.

[7] O. Maler, editor. *Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *Lecture Notes in Computer Science*. Springer, 1997.

[8] O. Stursberg, S. Engell, and S. Kowalewski. Timed approximations of hybrid processes for controller verification. In *Proc. 14th IFAC World Congress*, pages 73–78, 1999.

[9] O. Stursberg, S. Kowalewski, and S. Engell. On the generation of timed discrete approximations for continuous systems. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):51–70, 2000.

[10] F. W. Vaandrager and J. H. van Schuppen, editors. *Hybrid Systems: Computation and Control (HSCC'99)*, volume 1569 of *Lecture Notes in Computer Science*. Springer, 1999.