Discrete Event Dyn Syst (2008) 18:241–262
DOI 10.1007/s10626-007-0023-2

Timed Discrete Event Control of Parallel Production Lines with Continuous Outputs

Stephanie Geist · Dmitry Gromov · Jörg Raisch

Received: 25 October 2006 / Accepted: 26 July 2007 /
Published online: 15 August 2007

Abstract In this contribution we present an approach to formulate and solve certain scheduling tasks for hybrid systems using timed discrete event control methods. To demonstrate our approach, we consider a cyclically operated plant with parallel reactors using common resources and a continuous output. For this class of systems, we show how to pose the control problem within a discrete event framework by modelling system components as multirate timed automata. We propose a supervisory control strategy incorporating off-line optimisation to assure safety and nonconflicting use of resources. These properties have to be achieved in the presence of a class of bounded errors/disturbances and can be verified by applying formal methods.

Keywords Multirate timed automata · Scheduling · Parallel production lines · Hybrid systems · Verification · Discrete event control

1 Introduction

Coordinating the interaction of components is an essential task in the control of chemical production processes, particularly with regard to parallelised processes.

Work partially done in the framework of the HYCON Network of Excellence, contract number FP6-IST-511368.

S. Geist (✉) · D. Gromov · J. Raisch
Fachgebiet Regelungssysteme, Technische Universität Berlin, Berlin, Germany
e-mail: geist@control.tu-berlin.de

D. Gromov
e-mail: gromov@control.tu-berlin.de

J. Raisch
Systems and Control Theory Group, Max-Planck-Institut für Dynamik komplexer technischer Systeme, Magdeburg, Germany
e-mail: raisch@control.tu-berlin.de

In this context, scheduling problems aiming at the non-conflicting use of limited resources are of crucial importance. In chemical industry, batch processes are often connected to continuous processes by material supply and frequently the tasks of continuous flow control and the nonconflicting use of resources cannot be decomposed. In these cases the hybrid character of the plant cannot be neglected.

We present an approach to a scheduling problem guaranteeing the nonconflicting use of resources and safety despite disturbances using discrete event control methods. The approach combines standard off-line scheduling methods for batch processing and discrete event supervisory control which results in a flexible scheduling strategy for a class of cyclically operated plants.

In this contribution, we propose a control approach for a *"parallelised" production line with resource constraints and continuous output*. The system consists of an arbitrary number of parallel batch reactors sharing an arbitrary number of resources, e.g., reactants, hot steam or cool water. The reactors are discharged into a shared storage tank or another continuously processed production unit that has a continuous outflow which must not be interrupted. For this hybrid system, the goal is to assure nonconflicting work of the reactors and to prevent over- and underfilling of the tank in the presence of disturbances. Such a plant has been proposed as a case study within the EU Network of Excellence HYCON (Simeonova et al. 2005).

Solving the described problem in a monolithic way by the use of standard optimisation-based scheduling methods (Méndez et al. 2006; Schilling and Pantelides 1999) is hardly tractable for various reasons. Some constraints on the operation sequence require a continuous time formulation of events and most approaches cannot deal with this; an exception is Wu and Ierapetritou (2004). Cyclically operated plants may be very sensitive to disturbances, thus, frequent computationally expensive rescheduling is necessary in a standard framework. The major limiting factors in the applicability of standard scheduling methods are the constraints on the storage tank and its continuous outflow. Hence, material flow has to be considered additionally, and this makes the problem a hybrid one. Taking into account all these constraints, the described scheduling task results in a highly complex optimisation problem. To overcome these difficulties, we propose a method that combines off-line scheduling and discrete event supervisory control in a hierarchical way. The feedback structure increases robustness under uncertainties, and hierarchical decomposition can remarkably reduce the complexity of the problem.

The potential of discrete event methods for scheduling problems has been demonstrated in Abdeddaïm et al. (2006), Abdeddaïm and Maler (2001), Panek et al. (2006), Niebert and Yovine (2000), where timed automata have been used for modelling and solving job-shop scheduling problems by verification-based methods. Scheduling strategies in the presence of uncertainties have been proposed in Abdeddaïm et al. (2006). In contrast to these publications, we consider cyclically operated plants and additionally take continuous flows between the plant components into account. For the modelling of scheduling tasks in combination with continuous output flow control we use multirate timed automata.

To combine off-line optimisation and discrete event control, we formulate the requirement of nonconflicting use of resources by timed automata, where free parameters can be optimised off-line with respect to a given cost function. The resulting automata can be composed with other supervisor automata to enforce

the overall specifications. In this way, we can exploit feedback capabilities to avoid undesired behaviours and can use efficient off-line scheduling methods.

Describing scheduling problems in a timed automata framework is very intuitive. All system components including the resources can be considered as subsystems which can be easily described by timed automata and subsequently composed to form the overall problem. A major advantage of using timed automata methods for modelling is that controllers can be represented in the same formal framework. This facilitates the use of formal methods, which, in contrast to heuristic methods, can guarantee desired properties. Standard tools and methods for verification of timed automata are available and can be modified for multirate timed automata.

This contribution is an extended version of the conference paper (Gromov et al. 2006) and is structured as follows: in Section 2, we give a formal description of the overall problem. Multirate timed automata are introduced in Section 3. In Section 4, the modelling of the system components is described. In Section 5, we propose control strategies to ensure safety and a non-conflicting performance. In Section 6, we discuss verification techniques for problems described by multirate timed automata. Finally, in Section 7, we apply our approach to the HYCON benchmark described in Simeonova et al. (2005).

2 Problem statement

Figure 1 presents a schematic view of the chemical plant considered in the sequel. The system consists of n parallel batch reactors with k common resources, e.g., reac-

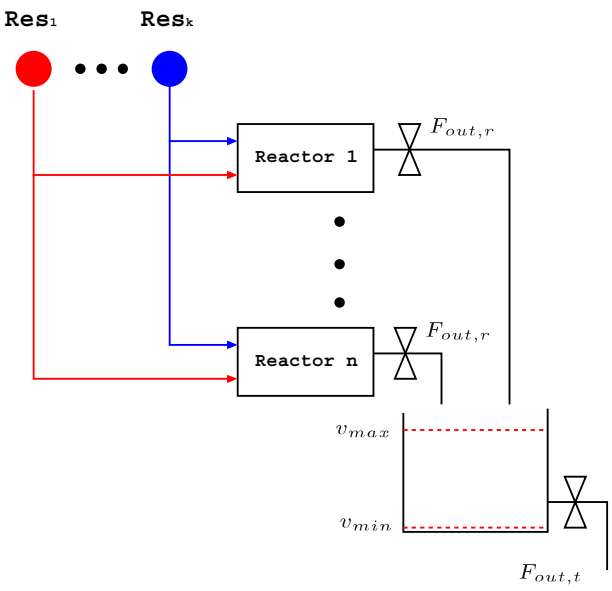

Fig. 1 A parallelised production line with resource constraints

tants, cold/hot water supplies and pumps. The reactors of volume V are discharged into one tank that acts as an output buffer and has the continuous output flow $F_{\text{out},t}$. Only one reactor can be discharged at any instant of time. The volumetric flow $F_{\text{out},r}$ during the discharging of a reactor is fixed, the output flow of the tank $F_{\text{out},t}$ can be adjusted within a given range. In each reactor the same process is performed. The goal is to assure the nonconflicting use of resources and to keep the level of the tank volume between given values $v_{\max}$ and $v_{\min}$. We assume that $v_{\max} > V$. Furthermore, due to technological restrictions, the tank outflow once started may not be interrupted.

A production cycle in the j-th reactor consists of a set of operations: $O_j = \{o_{ij}\},\ i = 1 \ldots m$, e.g., "heating," "cooling," "reaction," "discharging" and so on. The temporal ordering of these operations is fixed and given by the index i. We assume that there is efficient control of each operation, e.g. temperature control during the operation "heating." Thus, operations can be characterised by their processing times d_{ij}. We will also consider varying processing times of operations for which an upper and lower bound is known: $d_{ij} \in [d_i^* - \underline{d}_i; d_i^* + \overline{d}_i]$. These deviations in processing times may be caused by disturbances.

There are a set of resources R and sets of "resource-sensitive" operations $O'_j \subset O_j,\ j = 1, \ldots, n$. A map $r_j : O'_j \to R$ associates a resource to each operation $o_{ij} \in O'_j$ for reactor j. Here we assume that these maps are bijective, i.e. resources are used only once within the production cycle of a reactor.

Due to technological or safety requirements adjacent (in the temporal order) operations must sometimes be processed without delay. These operations are grouped into tasks $K_j^l = \{o_{\mu j}^l\},\ l = 1, 2 \ldots;\ p\ \mu = 1, 2, \ldots$, where different tasks are disjoint, i.e. $K_j^{l_1} \cap K_j^{l_2} = \emptyset$, for all $l_1 \neq l_2$. By requiring that each operation belongs to a task, we have $\bigcup_l K_j^l = O_j$. We further require that each task contains at least one resource-sensitive operation. This implies that an "isolated" operation also forms a task if it is resource-sensitive. Otherwise, it can be joined with the neighbour task. The index μ describes the temporal ordering of operations within the task. Note that there is a fixed relation between the temporal position of an operation within the task and within the overall sequence of operations in the reactor. Each operation within a task corresponds to an operation in the reactor cycle, $o_{\mu j}^l \mapsto o_{ij}$ where $i = \sum_{q=1}^{l-1} |K_j^q| + \mu$.

The goal of the control system to be designed is to assure cyclicity of the entire plant, i.e. safety requirements and the nonconflicting use of resources have to be guaranteed to avoid a shut-down of the plant. This has to be achieved in the presence of disturbances characterised by varying operation durations. Furthermore, in the nominal case of fixed durations, the plant output has to be maximised, i.e. the control system has to minimise the cycle duration. In the presence of disturbances, the cycle duration has to be minimised for a worst case scenario. Note that the worst case is a priori unknown.

The above assumptions can be relaxed without affecting our approach, but this would further complicate notation. For example, the approach can be generalised and adapted to other plant structures, e.g. plants with several tanks with continuous outflows, reactors processing different sequences of operations, and multiple use of resources within a reactor cycle.

3 Multirate timed automata

Timed automata (Alur and Dill 1994) are finite automata augmented with a finite set of continuous clocks whose values grow uniformly. Clocks can be reset independently of each other at certain transitions. Transitions and locations can be equipped with *clock constraints* representing continuous time information. To model our problem adequately we need an extended class of timed automata, namely *multirate timed automata* (Alur et al. 2000). In contrast to timed automata, clock rates are not fixed, but may change when transitions occur.

Multirate timed automata are a special case of piece-wise linear hybrid systems, where the continuous dynamics are modelled by affine differential equations in each location, e.g. Sontag (1996). In the context of our scheduling problem, the approximation of material flow between plant components by integrators with switched integration constants is sufficient. Therefore, multirate timed automata are the simplest adequate class of models where formal methods can be applied and analytical solutions can be obtained.

Multirate timed automata are formalised in the following definition.

Definition 1 A multirate timed automaton is a tuple $\mathcal{A} = (L, l_0, \Sigma, X, x_0, I, E, c, \lambda)$, where

- L is a finite set of locations,
- $l_0 \in L$ is the initial location,
- Σ is a finite set of events,
- X is a finite set of clocks. A clock valuation for the set X is a real vector $x \in \mathbb{R}^{|X|}$ where x_i is the value of the i-th element of X.
- x_0 is the initial clock value.
- I is a map that associates a clock constraint in $\Phi(x)$ to each location, i.e. $I : L \to \Phi(x)$, $I(l)$ is called an invariant of l.
- $E \subseteq L \times \Sigma \times \Phi(x) \times L$ is a set of transitions, where transition $e = (l, \sigma, \phi, l')$ is a directed arc between locations l and l' characterised by an event σ and a guard ϕ.
- $c : L \to \mathbb{Q}^{|X|}$ is a function that defines the rates of change of all clocks in each location. Thus, the dynamics of the clock variable x_i in location l can be described by a simple differential equation $\dot{x}_i = c_i(l) = const$. If $c_i(l)$ is equal to 1 for all indices l and i, we recover the case of pure timed automata.
- $\lambda : E \to 2^X$ associates to each transition a set of clocks to be reset to zero. $[\lambda(e) \mapsto 0]x$ denotes the vector of the clock values after the reset related to transition e, i.e. clock x_i is reset if and only if the respective clock belongs to $\lambda(e)$.

Clock values are used to check whether a clock constraint is satisfied. Clock constraints $\Phi(x)$ are defined over x with $\phi \in \Phi$ expressed in the following way:

$$\phi(x) = \phi_1(x_1) \wedge \phi_2(x_2) \wedge \ldots$$

$$\phi_i(x_i) : = k_{1i} \geq x_i \ \vee \ x_i \geq k_{2i} \ \vee \ k_{1i} \leq x_i \leq k_{2i}$$

$k_{1i}, k_{2i} \in \mathbb{Q} \cup \{-\infty, \infty\}$ are constants. This is to be interpreted as follows: the value of x_i is required to be either $\leq k_{1i}$ or $\geq k_{2i}$ or between k_{1i} and k_{2i}. This means that each clock constraint can be represented as a union of inequalities. Sometimes it is more convenient to consider a symbolic clock constraint as a set of clock values that satisfies some constraint ϕ. In this case we write $\phi(X) \subset \mathbb{R}^{|X|}$.

Note that a transition may be equipped with clock constraints which are interpreted as enabling or guard conditions. Clock constraints attached to locations can be interpreted as invariants.

We assume throughout this paper that the multirate timed automata are well posed regarding the reset function of clocks. This means that the set of clock values after each transition must agree with the invariant of the successor location:

Definition 2 A multirate timed automaton is said to be well posed if for each transition $e = (l, \sigma, \phi, l') \in E$ the vector of clock values after the transition satisfies $I(l')$:

$$[\lambda(e) \mapsto 0]x \in I(l')(X).$$

A state of a multirate timed automaton is a pair (l, x), where l and x are the current location and the clock value, respectively. To describe the dynamics of a multirate timed automaton, transition rules which form a so called transition system are introduced. We have to distinguish two possible scenarios: the evolution of time while staying in a location and the switching from one location to another.

Definition 3 A transition system of a well posed multirate timed automaton consists of two kinds of transitions:

1. Continuous transitions

$$(l, x) \xrightarrow{\tau} (l, x + \tau c(l)),\ \tau \in \mathbb{R}_+ \quad \text{if } x \in I(l)(X) \text{ and } x + \tau c(l) \in I(l)(X);$$

2. Discrete transitions

$$(l, x) \xrightarrow{e} (l', x'),\ e = (l, \sigma, \phi, l') \in E \text{ if } x \text{ satisfies } \phi, \text{ and } x' = [\lambda(e) \mapsto 0]x.$$

In the present paper, we describe the behaviour of the overall system in a modular way by several automata. To express the overall system behaviour by parallel composition, we use the standard definition of the product of timed automata (Alur and Dill 1994) extended to multirate timed automata.

Definition 4 Let $\mathcal{A}_1 = (L_1, l_{01}, \Sigma_1, X_1, x_{01}, I_1, E_1, c_1, \lambda_1)$ and $\mathcal{A}_2 = (L_2, l_{02}, \Sigma_2, X_2, x_{01}, I_2, E_2, c_2, \lambda_2)$ be two timed automata. Assume that the clock sets X_1 and X_2 and the location sets L_1 and L_2 are disjoint. Then, the product, denoted by $\mathcal{A}_1||\mathcal{A}_2$, is the timed automaton $(L, l_0, \Sigma, X, x_0, I, E, c, \lambda)$, where $L = L_1 \times L_2$, $l_0 = (l_{01}, l_{02})$, $\Sigma = \Sigma_1 \cup \Sigma_2$, $X = X_1 \cup X_2$ and $x_0 = (x_{01}, x_{02})$. The functions I, c, λ and the transition structure E are defined as follows:

1. $I(l_1, l_2) = I_1(l_1) \wedge I_2(l_2)$
2. $c : L_1 \times L_2 \to \mathbb{Q}^{|X_1|+|X_2|}$ with $c(l_1, l_2) = \begin{pmatrix} c_1(l_1) \\ c_2(l_2) \end{pmatrix}$

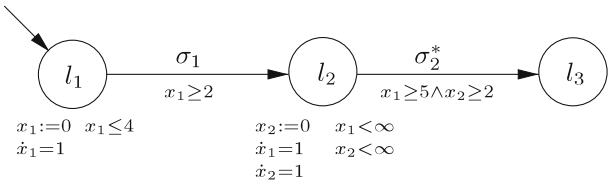

Fig. 2 Example automaton. The transition from location l_2 to l_3 must take place at the first possible time instant. Switching between location l_1 and l_2 can take place for $2 \leq x_1 \leq 4$

3. a. $\sigma \in \Sigma_1 \cap \Sigma_2$.
   $e = ((l_1, l_2), \sigma, \phi, (l_1', l_2')) \in E \Longleftrightarrow (l_1, \sigma, \phi_1, l_1') \in E_1$ and $(l_2, \sigma, \phi_2, l_2') \in E_2$, $\phi = \phi_1 \wedge \phi_2$.
   Then $\lambda(e) = \lambda_1(e_1) \cup \lambda_2(e_2)$.
   b. $\sigma \in \Sigma_1 \setminus \Sigma_2$.
   $e = ((l_1, l_2), \sigma, \phi, (l_1', l_2)) \in E \Longleftrightarrow (l_1, \sigma, \phi_1, l_1') \in E_1, \phi = \phi_1$.
   Then $\lambda(e) = \lambda_1(e_1)$.
   c. $\sigma \in \Sigma_2 \setminus \Sigma_1$.
   $e = ((l_1, l_2), \sigma, \phi, (l_1, l_2')) \in E \Longleftrightarrow (l_2, \sigma, \phi_2, l_2') \in E_2, \phi = \phi_2$.
   Then $\lambda(e) = \lambda_2(e_2)$.

We further introduce a set $\Sigma^* \subset \Sigma$ of forced events. Transitions labelled by forced events are called forced transitions, and they must occur at the first possible time instant, i.e. as soon as the clocks satisfy the guard ϕ. This time instant is not known in all locations due to varying clock values at the previous switching time.

The example automaton in Fig. 2 illustrates the use of forced transitions. In our modelling process, we want to switch from location l_2 to location l_3 at the first possible time instant. The switching time from location l_1 to l_2 can vary in the interval [2; 4] and not all clocks are reset to zero. In Fig. 3 the trajectories of clock values x_1 and x_2 are shown. The guard set can be reached at different time instants. By assigning $\sigma_2^* \in \Sigma^*$, we force the transition to switch at the earliest possible time.

4 Timed automaton model of the plant

In this section we use multirate timed automata for the modelling of the chemical plant by assigning one automaton to each plant component. The modelling is

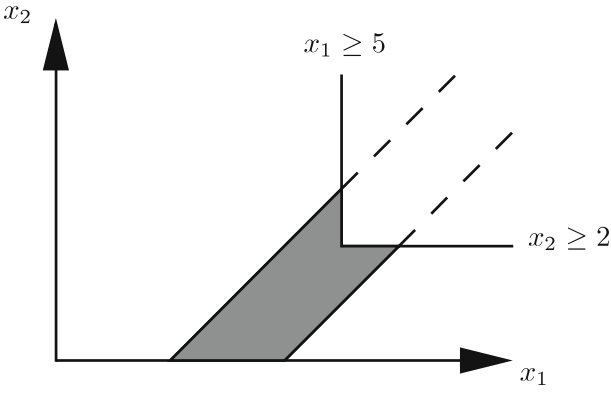

Fig. 3 Possible trajectories in location l_2 of the example automaton. The guard set can be reached at different time instants

straightforward and common practice. Nevertheless, the modelling procedure is briefly described for the sake of completeness, especially as we also apply forced transitions to reflect our problem properly. The plant consists of reactors, resources and an output tank. Reactors can be modelled by pure timed automata, i.e. clock rates do not change. In the output tank model the clock value represents the liquid level in the tank. Its rate can change depending on the in- and outflow, and a multirate timed automaton is used for modelling. Resources can be in use or not and are described by simple finite automata. Note that both standard untimed and timed automata can be interpreted as special cases of multirate timed automata. Hence, the product is well defined, and the behaviour of the entire plant can be modelled by the product of all automata resulting in a multirate timed automaton. In the following we explain the models of all types of components in detail.

4.1 Reactors

The first step is the modelling of the reactors using timed automata. Since the operation sequence is identical in each reactor, they can be described in a uniform way, as shown in Fig. 4.

The operations processed in each reactor are represented by the locations wo_{ij} and o_{ij}, which mean "wait before i-th operation starts in reactor j" and "i-th operation is active in reactor j." The events Sto_{ij} and Eo_{ij} denote start and end of the i-th operation in the j-th reactor, respectively. Fig. 4 is to be interpreted as follows: in location o_{ij}, the progress of time is measured by a clock modelled by $\dot{x}_j = 1$, and the clock is reset to zero when the transition from wo_{ij} to o_{ij} takes place, i.e. when event Sto_{ij} occurs. We are only allowed to stay in the location o_{ij} if $x_j \leq (d_i^* + \overline{d}_i)$ holds (invariant). The event Eo_{ij} may only occur if $x_j \geq (d_i^* - \underline{d}_i)$ holds (guard). Hence, the transition between location o_{ij} to location $wo_{(i+1)j}$ has to happen when $(d_i^* - \underline{d}_i) \leq x_j \leq (d_i^* + \overline{d}_i)$. The switching can take place at an arbitrary time instant within this interval, so that all possible disturbances are included in the model. In location wo_{ij}, there are two possibilities: either invariant and guard of the outgoing transition enforce an immediate switch to the next location (an example for this case is location wo_{2j} in Fig. 4), or invariant and guard allow an arbitrarily long stay within the location (an example for this case is location wo_{3j} in Fig. 4). If a wo_{ij}-location is of the for-

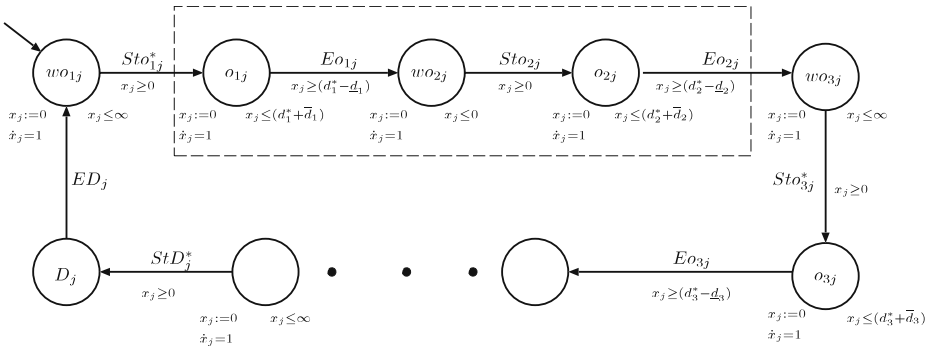

Fig. 4 A timed automaton model of reactor j

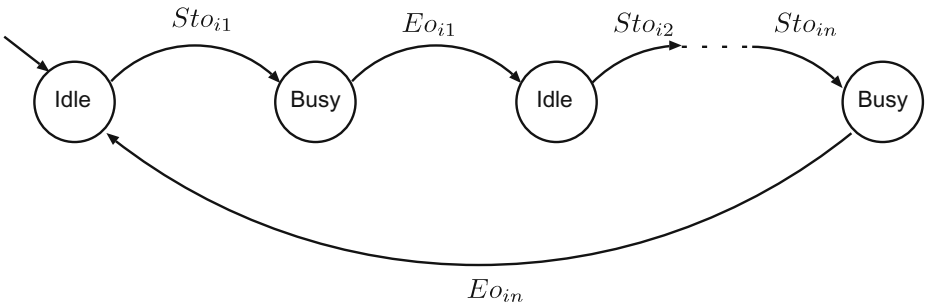

Fig. 5 A finite automaton modelling the availability of resource i

mer type, $o_{(i-1)j}$ and o_{ij} must be processed without any delay in between and belong to the same task (this is illustrated by the dashed box in Fig. 4). For the latter type of location we want transitions to take place at the first possible time instant. Hence, transitions are interpreted as forced in sense of Section 3 and are denoted by *. Forced events always start a task.

The last operation, denoted by D_j, is the discharging of reactor j. Note that the operation "discharging" always represents a task since the output tank can be interpreted as an external resource.

4.2 Resources

The next step is to model the restrictions on the availability of resources. The simplest way is to build a finite automaton for each resource $R_i, i = 1, \ldots k$, and the corresponding resource sensitive operations $r_j^{-1}(R_i) \in O'_j,\ \ j = 1, \ldots, n$ as shown in Fig. 5. The depicted timed automaton represents a simple rule: a resource sensitive operation can be simultaneously carried out in one reactor only. Whether event Sto_i. is forced or not depends on the reactor model.

To enforce uniqueness of the solution, a sequence of reactors is predetermined, and the temporal order corresponds to the indices of the reactors. As all reactors are equal, this does not restrict generality.

4.3 Output tank

Another element of the plant is the output tank. Its timed automaton model is presented in Fig. 6. The transitions StD^* and ED denote "start discharging" and "discharging is finished." The clock variable v models the amount of liquid in the tank. We assume that the initial value v_0 is greater than $v_{\min}$. Here, v has two different rates, $a = F_{\text{out},r} - F_{\text{out},t} > 0$ when the output valve is open and one reactor is being discharged and $b = -F_{\text{out},t}$ when no reactor is being discharged while the output valve is still open, where $F_{\text{out},r}$ is the volumetric rate of the flow from any reactor j to the tank during discharging and $F_{\text{out},t}$ is the volumetric rate of the output flow of the tank. The outlet valve of the tank is only opened when the first discharging operation starts and, as part of requirements, must not be closed thereafter. If v becomes too small or too large, the automaton goes to one of the locations modelling a forbidden situation, *overfilling* or *underfilling*.

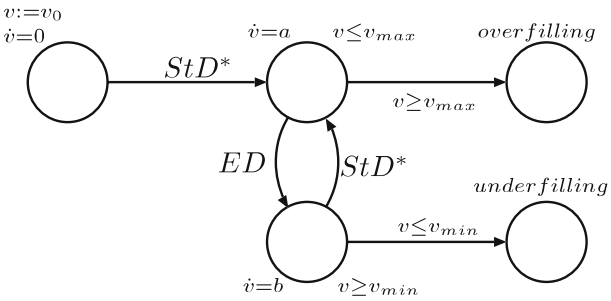

Fig. 6 A multirate timed automaton modelling the output tank

5 Control synthesis

In the controlled system, resources must be allocated in a nonconflicting way. In addition, we have safety specifications, the locations *overfilling* and *underfilling* must be rendered unreachable.

In the following we present control strategies to enforce the specifications. The approach combines feedforward scheduling strategies with hierarchical discrete event feedback methods, which makes the overall scheduling problem less complex and more robust. Note that by adding a formal verification step, we can provide a formal guarantee for the safety specifications to hold, even if the actual design process contains some heuristics (for the safety aspect).

The modular modelling framework allows us to apply a two-layer controller structure (Fig. 7). We first synthesise a controller which, by appropriate scheduling of reactor operations, guarantees nonconflicting resource allocation. This design step incorporates off-line optimisation in on-line discrete event control. Other on-line strategies can be added to solve further subproblems. E.g., in a second step, we design a controller for the output tank to guarantee safety.

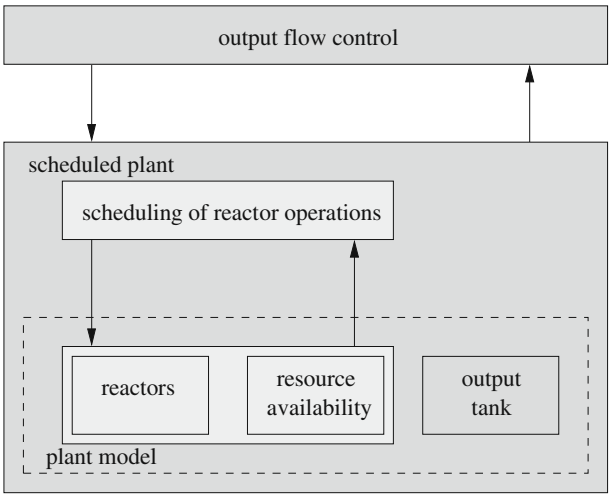

Fig. 7 Two-layer controller structure

5.1 Scheduling of reactor operations

The goal of the scheduling subproblem is to generate a nonblocking interaction of reactor models (Fig. 4) and resource availability models (Fig. 5) for all possible variations in operation durations. We further want to minimise the duration t_c between two discharging operations in the same reactor. For the case of varying operation durations, t_c has to be minimised for the worst case, i.e. we want to minimise the maximal duration $t_{c,\max}$.

It is obvious that, in the uncontrolled case, the synchronous product of the n reactor models (Fig. 4) and the k resource availability models (Fig. 5) may give rise to blocking. This can happen in the following way: a resource, say R_i, is being allocated by an operation o_{ij} in reactor j. At the same time, an operation $o_{(i-1)(j-1)}$ is finished in reactor $(j-1)$ and the succeeding operation $o_{i(j-1)}$ belonging to the same task as $o_{(i-1)(j-1)}$ attempts to allocate R_i. In this situation, $o_{i(j-1)}$ must start immediately after $o_{(i-1)(j-1)}$ has finished. This is clearly not possible as the corresponding resource is still being used by reactor j.

It is intuitively possible to prevent blocking by appropriately delaying the starting times of tasks in the individual reactors. To determine the minimal necessary delays, we solve the corresponding scheduling problem in an off-line fashion and represent the result as timed automata which, when composed with reactor and resource automata, prevent blocking.

First, we give an algebraic formulation of the scheduling problem. We formulate all constraints to achieve a non-conflicting use of resources and meet the process requirements. Operations have to be processed until completion which gives a relation between start times s_{ij} and finish times f_{ij} of operations o_{ij}:

$$f_{ij}^{(\rho)} = s_{ij}^{(\rho)} + d_{ij}^{(\rho)},\ i = 1, \ldots, m,\ \ j = 1, \ldots, n,\ \forall \rho \in \mathbb{N}, \tag{1}$$

where index ρ denotes the cycle number. Operations must be performed in a given sequence denoted by index i. The i-th operation must be completed before the $(i+1)$-th operation in the same reactor can start:

$$s_{(i+1)j}^{(\rho)} \geq f_{ij}^{(\rho)},\ i = 1, \ldots, m-1,\ \ j = 1, \ldots, n, \tag{2}$$

$$s_{1j}^{(\rho+1)} \geq f_{mj}^{(\rho)},\ j = 1, \ldots, n,\ \forall \rho \in \mathbb{N}. \tag{3}$$

We further must take task constraints into account, namely time delays between operations within a task must not occur:

$$s_{(\mu+1)j}^{l(\rho)} = f_{\mu j}^{l(\rho)},\ l = 1, \ldots, p,\ \ j = 1, \ldots, n,\ \mu = 1, \ldots |K_j^l| - 1,\ \forall \rho \in \mathbb{N}, \tag{4}$$

where the index l denotes the task and μ the temporal position of an operation within a task. Without loss of generality, we fix a reactor sequence to reduce degrees of freedom:

$$s_{1(j+1)}^{l(\rho)} \geq s_{1j}^{l(\rho)},\ \ j = 1, \ldots, n-1,\ l = 1, \ldots p,\ \forall \rho \in \mathbb{N}, \tag{5}$$

$$s_{11}^{l(\rho+1)} \geq s_{1n}^{l(\rho)},\ l = 1, \ldots, p,\ \forall \rho \in \mathbb{N}. \tag{6}$$

The following resource constraints exclude overlapping of two operations in different reactors using the same resource R_i:

$$s_{i(j+1)}^{(\rho)} \geq f_{ij}^{(\rho)},\ j = 1, \ldots, n-1,\ i = 1, \ldots, m,\ \forall o_{ij} \in O'_j, o_{i(j+1)} \in O'_{j+1}, \tag{7}$$

$$s_{i1}^{(\rho+1)} \geq f_{in}^{(\rho)},\ i = 1, \ldots, m,\ \forall o_{i1} \in O'_1, o_{in} \in O'_n,\ \forall \rho \in \mathbb{N}. \tag{8}$$

This algebraic problem description corresponds to the timed automata representation set up previously. In particular, constraints 1 to 4 represent the "reactor automata" (Fig. 4) and constraints 5 to 8 the "resource automata" (Fig. 5). The equivalence can be shown by listing the temporal restrictions implied by the logical structure of the automata and their invariants and guards.

The inequality system 1 to 8 has to be solved for all possible variations of operation durations while minimising $t_{c,\max}$. In this inequality system, the start times s^l_{1j} of tasks l are the only free parameters. We express the relation between these start times in different reactors by parameters w^l: $s^l_{1j} = s^l_{1(j-1)} + w^l,\ j = 2, \ldots, n$.

For fixed processing times, the duration between two discharging operations in the same reactor can be calculated by:

$$t_c = \begin{cases} n \max\limits_l w^l, & \text{for } \sum\limits_{i=1}^{m} d_i^* < n \max\limits_l w^l \\ \sum\limits_{i=1}^{m} d_i^*, & \text{otherwise.} \end{cases} \tag{9}$$

Note that the minimum of t_c is bounded from below by $\sum\limits_{i=1}^{m} d_i^*$. If all parameters w^l are minimal, t_c is the minimal solution of the scheduling problem. Parameters w^l have to be minimised such that constraints 1 to 8 are satisfied. From the Gantt charts in Fig. 8, it can be seen that minimal w^l are achieved if for each $l \in \{1, \ldots, p\}$

$$\min_{\mu : o^l_{\mu j} \in O'_j} \left(s^l_{\mu(j+1)} - f^l_{\mu j} \right) = 0 \tag{10}$$

holds, where $s^l_{\mu 1} = s^l_{11} + \sum\limits_{k=1}^{\mu-1} d^l_{k1},\ f^l_{\mu 1} = s^l_{\mu 1} + d^l_{\mu 1},\ s^l_{\mu j} = s^l_{1(j-1)} + w^l + \sum\limits_{k=1}^{\mu-1} d^l_{kj},\ j > 1$. $s^l_{\mu j}$ and $f^l_{\mu j}$ denote the start and end times of operation $o^l_{\mu j}$; its duration is $d^l_{\mu j}$.

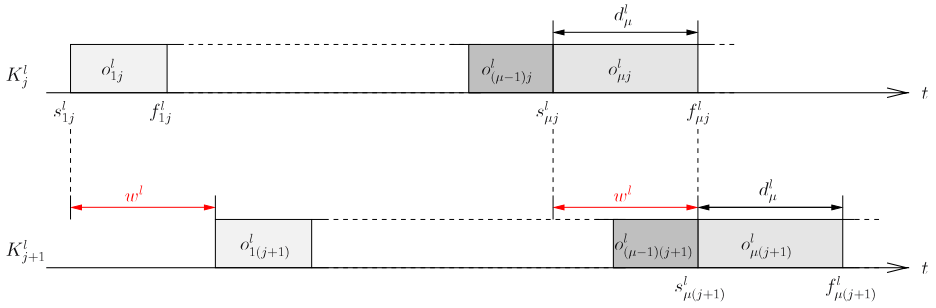

Fig. 8 Gantt chart of task K^l_j in reactor j and task $K^l_{(j+1)}$ in reactor $j+1$ for fixed processing durations

In the nominal case, the duration of each operation is known and fixed for all reactors: $d^l_{\mu j} = d^l_\mu$. Thus, the solution of the optimisation problem is $w^l = \max_\mu d^l_\mu$. This is illustrated by the Gantt charts in Fig. 8. There, operation $o^l_{\mu j}$ is a resource sensitive operation which satisfies condition 10. The start $s^l_{\mu(j+1)}$ of the same task in the subsequently used reactor is delayed by w^l such that condition 10 holds. Hence, the first operation of a task is not allowed to start until the resource availability for all resource sensitive operations in the same task can be guaranteed. It can be seen that the resulting w^l are indeed minimal subject to constraints 1 to 8.

The situation becomes more complicated if the processing durations are only known imprecisely, i.e., $d^l_{\mu j} \in [d^*_i - \underline{d}_i; d^*_i + \overline{d}_i]$, where the relation between i and μ is explained in Section 2. Condition 10 then takes the form

$$\min_{\substack{\mu: o^l_{\mu j} \in O'_j \\ d^l_{\mu j}, d^l_{\mu(j+1)} \in [d^*_i - \underline{d}_i; d^*_i + \overline{d}_i]}} \left(s^l_{\mu(j+1)} - f^l_{\mu j} \right) = 0. \tag{11}$$

Thus, w^l is calculated in a worst-case fashion and is therefore conservative. In Fig. 9 the Gantt charts are depicted for this worst case situation. In reactor j, the processing durations are smaller than in the nominal case while in reactor $(j+1)$ operations last longer. This results in a larger w^l than in the nominal case.

Hence, the solution of the scheduling problem is the computation of delays w^l. Delaying task l by w^l guarantees nonblocking while, in the nominal case, ensuring time-optimal solutions. The insertion of delays can be easily translated into the timed automata modelling framework. This is done by defining p timed automata, one for each task l, $l = 1, \ldots, p$, as shown in Fig. 10. Note that events Sto^{l*}_{1j} in Fig. 10 correspond to events Sto^*_{ij} in the "reactor automata" if $i = \sum_{q=1}^{l-1} |K^q_j| + 1$ (compare Section 2). When forming the product between these automata and the plant model, i.e. the "reactor automata" and the "resource automata," the start of tasks in subsequently used reactors is suitably delayed.

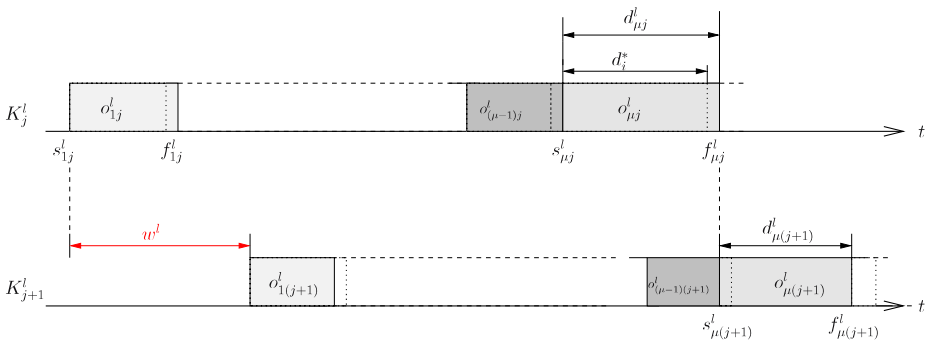

Fig. 9 Gantt charts of task K^l_j in reactor j and task $K^l_{(j+1)}$ in reactor $(j+1)$ for imprecisely known processing durations

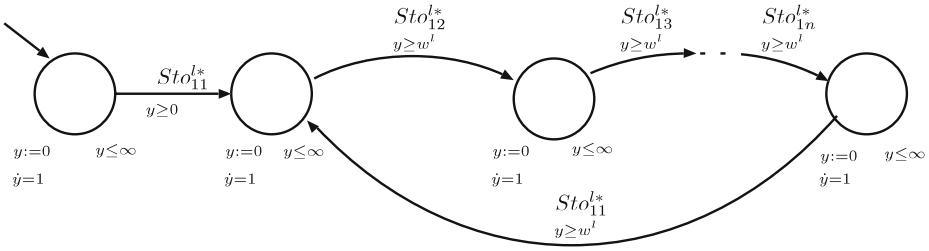

Fig. 10 A timed automaton modelling the control of starting times of task l in reactors 1 to n

5.2 Output flow control

The process under low-level control (denoted as "scheduled plant" in Fig. 7) is represented by the synchronous product of the reactor models (Fig. 4), the resource availability models (Fig. 5), the task control automata (Fig. 10) and the tank model (Fig. 6). As we have previously determined suitable delays w^l, in this step only nonconflicting resource allocation schemes are possible. The remaining problem is to find an outputflow rate $F_{\text{out},t}$ for the storage tank that ensures that the locations *underfilling* and *overfilling* are not reached. Recall that the outflow from the tank, once started, may not be interrupted.

For the case where all processing times are known we set

$$F_{\text{out},t} = \frac{nV}{t_c},$$

where V is the volume of one reactor and t_c is the cycle time (see Section 5.1). Clearly, this is the maximal possible outflow rate and any larger value would cause underfilling at some instant of time. This choice of output flow may cause overfilling in some cases. To prevent this, one needs an additional degree of freedom. One possibility is to introduce additional waiting times for the discharging of the reactors. We force reactors to delay discharging if the amount of liquid v in the output tank is above a certain threshold v^* ($v_{\min} < v^* < v_{\max}$) to prevent overfilling. If the liquid level is below this threshold, the start of discharging of reactors is enabled. If the threshold is set to the maximal possible level, $v^* = v_{\max} - V$, the added waiting times do not effect the cycle time t_c in the case of fixed processing times and under the assumption of $v_{\max} - V > v_{\min}$.

This results in a modified multirate timed automaton modelling the controlled output tank (Fig. 11). Using the verification algorithm in Section 6, it may be shown that underfilling and overfilling is avoided.

For the case of unknown operation durations, over- and underfilling of the output tank can be avoided in a similar way. To avoid *overfilling*, a waiting location can be introduced as described above (Fig. 11). The outflow of the output tank can be initially set to

$$F_{\text{out},t} = \frac{nV}{t_{c,\max}},$$

where $t_{c,\max}$ is the maximal cycle duration, i.e. the maximal time between two discharging operations by the same reactor. $t_{c,\max}$ can be obtained by applying verification procedures to the product of reactor models (Fig. 4), resource availability

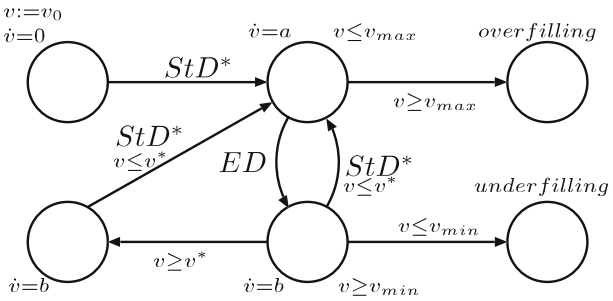

Fig. 11 A modified timed automaton modelling the output tank when overfilling is prevented by an additional waiting location

models (Fig. 5) and task control automata (Fig. 10). Applying the verification algorithm presented in Section 6 we check whether *underfilling* can be reached. If yes, $F_{\text{out},t}$ can be reduced iteratively. For highly uncertain processing times the simultaneous avoidance of under- and overfilling may not be possible with a constant outflow rate.

An alternative is to discretise the allowed range of the tank outflow $F_{\text{out},t}$ and switch the output rate on-line between several values $F_{\text{out},ti}$, $F_{\text{out},ti} < F_{\text{out},t(i+1)}$, $i = 1, 2, \ldots q$, where the outflow rate $F_{\text{out},ti}$ is applied if the liquid level v is between thresholds v_{i-1} and v_i ($v_0 < v_1 < \ldots < v_{q-1} < v_q$), with $v_0 = v_{\min}$ and $v_q = v_{\max}$. As in the method above, verification is required. If verification fails outflow rates or thresholds need to be adapted iteratively.

A modified output tank model with two different outflow rates is presented in Fig. 12.

We have proposed strategies for the output tank control. Other algorithms e.g. the combination of waiting locations with switched outflow rates, can be modelled in a similar way as multirate timed automata. We have seen that even the rather simple

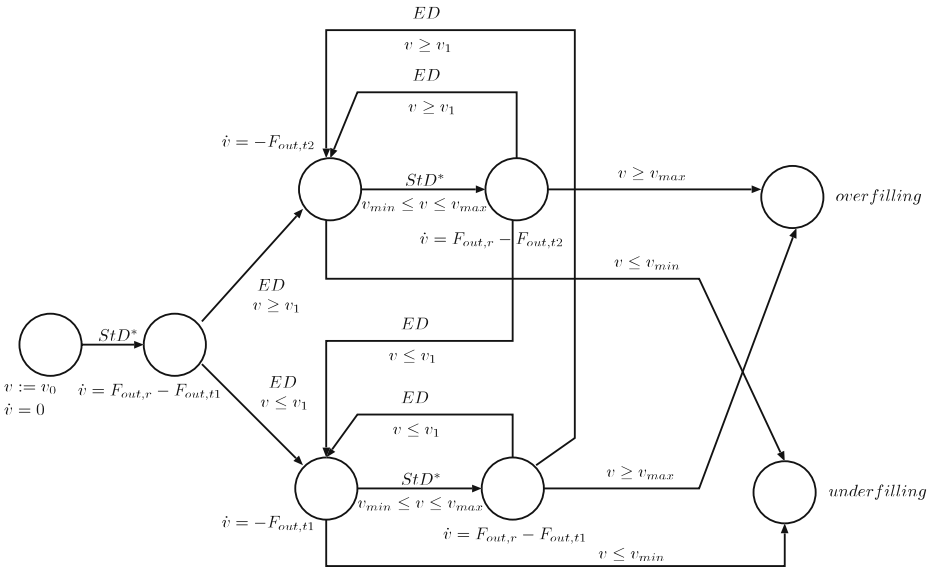

Fig. 12 A timed automaton modelling the controlled output tank with two different outflow rates

methods presented in this section require verification results. Hence, we focus on verification in the next section.

6 Verification

Our control approach in Section 5.2 contains some heuristics. Therefore, in particular for large systems, it is essential to verify safety of the overall control system behaviour. For more complex system structures than considered in the present paper, it might be necessary to integrate verification into an iterative controller synthesis procedure. In this section, we present an algorithm for safety verification of multirate timed automata. We show how to extend symbolic methods for "pure" timed automata to multirate timed automata. We also address some computational issues to achieve a tractable problem and improve efficiency.

All possible temporal evolutions of (l, x) must meet a given safety requirement. The safety requirement is connected to a reachability problem, i.e. forbidden states must not be reached. In our system the locations *underfilling* and *overfilling* of the output tank automaton are forbidden.

One of the most important questions in the analysis of timed automata (and, in general, of all hybrid systems) is the reachability of a given state or a set of states. We give a formal definition of reachability, introduce some symbolic operations and propose an algorithm for reachability verification. The following definition is adapted from Alur et al. (1995):

Definition 5 For a (multirate) timed automaton $\mathcal{A}$ with initial state (l_0, x_0), the state (l_f, x_f) is reachable if there exists a sequence of discrete transitions $e_1, e_2, \ldots$ and durations $\tau_1, \tau_2 \ldots$ such that

$$(l_0, x_0) \xrightarrow{\tau_1} (l_0, \tilde{x}_0) \xrightarrow{e_1} (l_1, x_1) \xrightarrow{\tau_2} \ldots \longrightarrow (l_f, x_f). \tag{12}$$

Moreover, given a constraint $\phi \in \Phi(x)$, we say that the configuration (l_f, ϕ) is reachable if there exists a sequence of transitions and durations such that Eq. 12 holds for some x_f satisfying ϕ.

Numerical methods for reachability verification of timed automata are described in Pettersson (1999), Bengtsson and Yi (2004), Bozga et al. (1998) and symbolic methods are presented, for instance, in Asarin et al. (1995). Symbolic methods are based on the partitioning of the state space of a timed automaton into symbolic states which are represented by clock zones or regions. In the following we show how to extend the symbolic framework to multirate timed automata and propose an algorithm for solving the reachability problem. Particular attention is paid to forced transitions.

A *zone* is a generalisation of a state of a (multirate) timed automaton. A zone D is defined as a pair (l_D, S_D), where S_D is a set of clock values, such that each $x \in S_D$ satisfies the constraint $I(l_D)$. Zone D is said to be *regular* if S_D is bounded and can be represented as a set of clock constraints

$$S_D = \{x | Ax \leq b\,,\, A \in \mathbb{Q}^{n \times k}, b \in \mathbb{Q}^n, k = |X|\}.$$

A lifting operation is introduced as a generalisation of a continuous transition (see Def. 3).

Definition 6 A lifting operation $S_D^{\uparrow}(\tau)$ is defined as

$$S_D^{\uparrow}(\tau) = \{\tilde{x} \mid \tilde{x} = x + \tau c(l),\ (l, x) \in D\} \cap I(l)(X).$$

Furthermore, it can be generalised to an untimed lifting operation $S_D^{\uparrow} = \bigcup_{\tau=0}^{\infty} S_D^{\uparrow}(\tau)$.

Based on this, one can define a generalised (symbolic) transition relation Bengtsson and Yi (2004).

Definition 7 The symbolic transition relation, denoted by $\rightsquigarrow$, is defined by the following rule:

$$(l, S) \rightsquigarrow (l', [\lambda(e) \mapsto 0](S^{\uparrow} \cap \phi(X))),\ \text{if } \exists e = (l, \sigma, \phi, l') \in E \setminus E^*.$$

In our case, the above transition relation must be extended by a new transition rule for forced transitions $e \in E^*$. A forced transition must take place as soon as clocks satisfy the guard ϕ. Thus, the corresponding transition rule can be defined as

$$(l, S) \overset{*}{\rightsquigarrow} (l', [\lambda(e) \mapsto 0](S^{\uparrow} \cap \underline{\phi}(X))),\ \text{if } \exists e = (l, \sigma, \phi, l') \in E^*,$$

where $\underline{\phi}(X)$ is a part of the boundary $\phi(X)$ characterised by $\underline{\phi}(X) = \{x \in \phi(X) | x_i \leq \tilde{x}_i \forall \tilde{x} \in \phi(X)$ for at least one $i\}$.

Inspired by Bengtsson and Yi (2004), the algorithm below checks whether the given automaton can reach zone (l_f, S_f) starting from (l_0, S_0).

Algorithm Reachability analysis.

```
New := {l0, S0}, Checked := ∅
While New ≠ ∅
   Next := ∅
   For each (l, S) ∈ New
      For each l' ∈ Post(l)
         If (l, σ, φ, l') ∈ E\E*
            (l', S') = (l', [λ(e) ↦ 0](S↑ ∩ φ(X)))
         Else
            (l', S') = (l', [λ(e) ↦ 0](S↑ ∩ φ̲(X)))
         End If
         Next := Next ⋃ (l', S')
      End For
   End For
   If Next ∩ (lf, Sf) ≠ ∅
      Type ("zone (lf, Sf) is reachable")
      Stop
   End If
   Checked := Checked ⋃ (Next \ New)
   New := Next \ Checked
End While
Type ("zone (lf, Sf) is not reachable")
```

The computational details are out of the scope of this paper. Nevertheless, a few short remarks seem appropriate.

There are only few operations that need to be performed cyclically during the reachability computation. First, we need to compute the result of the untimed lifting operation $S_D^{\uparrow}$ and find - if it exists - its intersection with the guard $\phi(X)$. This can be done using the quantifier elimination algorithm (see Anai and Weispfennig 2001 and references within). As a result of quantifier elimination one gets a set of equations that describes all possible clock values that can be reached via the respective transition. Obviously, an empty set means that this transition cannot take place and, therefore, the successor location cannot be reached via this transition.

For a discussion of implementation issues of the remaining set-theoretic operations we refer to Avis et al. (2002), Halbwachs et al. (1994), Halbwachs et al. (2006), Goodman and O'Rourke (1997).

Obviously, multirate timed automata form a subclass of piecewise linear hybrid systems. Therefore, in principle, methods and tools for this class of systems could be applied. However, regarding computational efficiency, it seems wise to exploit the additional structure embodied in multirate timed automata.

7 Example

We now consider a specific example described in detail in Simeonova et al. (2005). The plant consists of two reactors. In each reactor the following sequence of operations is performed: filling ($d_1^* = 0.17\,\text{h}$), heating ($d_2^* = 0.45\,\text{h}$), maintaining temperature ($d_3^* = 3.44\,\text{h}$), cooling ($d_4^* = 0.92\,\text{h}$) and discharging ($d_5^* = 0.17\,\text{h}$). The operations filling, heating and cooling are resource-sensitive. The set of operations has been partitioned into three tasks: K_j^1={filling}, K_j^2={heating, maintaining temperature, cooling} and K_j^3={discharging}, $j = 1, 2$. The time for heating is only known imprecisely: $d_2 \in [d_2^* - \underline{d}_2, d_2^* + \overline{d}_2]$ where $\underline{d}_2 = \overline{d}_2 = 0.13\,\text{h}$. The volumes of both reactors are $27\,\text{m}^3$. The minimal and maximal volume of liquid in the tank is $v_{\min} = 0$ and $v_{\max} = 50\,\text{m}^3$, respectively.

We applied the method presented in the previous sections to obtain a solution that allocates resources in a nonconflicting way, performs a worst-case minimisation of cycle duration and guarantees safety for all possible variations of parameters. Using the first method from Section 5.2, the maximal admissible constant tank outflow is

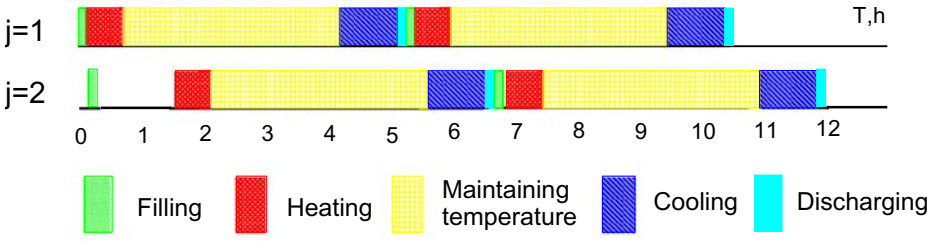

Fig. 13 Resulting schedule for $d_{2j} = d_2^* + \overline{d}_2, \quad j = 1, 2$

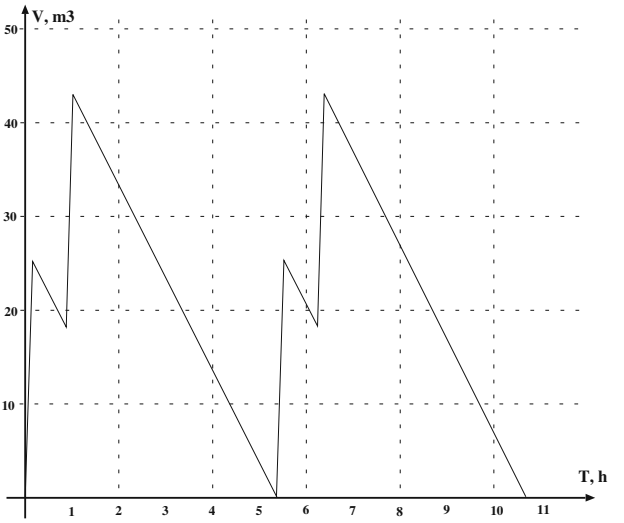

Fig. 14 Liquid volume in the tank

$F_{\text{out},t} = 10.23\,\text{m}^3/\text{h}$. The resulting schedule for the worst case $d_2 = d_2^* + \overline{d}_2$ is shown in Fig. 13, the resulting change of the liquid volume in the tank is shown in Fig. 14.

8 Conclusions

In this contribution, we have investigated the use of multirate timed automata for the scheduling and control of a class of parallel production lines taking material flow into account. The aim is to allocate resources in a nonconflicting way while minimising cycle durations, and to guarantee safety. We have addressed the case when uncertainties regarding certain operating times are present. Although the described approach contains heuristic elements in the design procedure, we can guarantee safety. This is assured by using a standard verification procedure which has been adapted to the case of multirate timed automata. We have applied this procedure to a specific process which has been suggested as a benchmark problem within the EU Network of excellence HYCON.

Our approach combines "classical" optimisation-based scheduling methods and discrete-event methods. Since our approach was motivated by an application, it is currently restricted to a particular type of problems. Future research may investigate how this idea can be extended to more general problems using different types of scheduling methods.

References

Abdeddaïm Y, Asarin E, Maler O (2006) Scheduling with timed automata. Theor Comp Sci 354(2):272–300

Abdeddaïm Y, Maler O (2001) Job-shop scheduling using timed automata. In: Computer aided verification, LNCS 2102, Springer, pp 478–492

Alur R, Coucoubetis C, Halbwachs N, Henzinger TA, Ho P-H, Nicollin X, Olivero A, Sifakis J, Yovine S (1995) The algorithmic analysis of hybrid systems. Theor Comp Sci 138:3–34

Alur R, Dill DL (1994) A theory of timed automata. Theor Comp Sci 126:183–235
Alur R, Henzinger TA, Lafferriere G, Pappas GJ (2000) Discrete abstractions of hybrid systems. Proc IEEE 88(7):971–984
Anai H, Weispfennig V (2001) Reach set computations using real quantifier elimination. In: Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control, number 2034 in LNCS, Springer, pp 63–76
Asarin E, Maler O, Pnueli A (1995) Symbolic controller synthesis for discrete and timed systems. In: Hybrid Systems II, LNCS 999, Springer, pp 1–20
Avis D, Fukuda K, Picozzi S (2002) On canonical representations of convex polyhedra. Mathematical Software, World Scientific, pp 351–360
Bengtsson J, Yi W (2004) Timed automata: Semantics, algorithms and tools. In: Reisig W, Rozenberg G (eds) Lecture Notes on Concurrency and Petri Nets, LNCS 3098. Springer
Bozga M, Daws C, Maler O, Olivero A, Tripakis S, Yovine S (1998) Kronos: a model-checking tool for real-time systems. In: Hu AJ, Vardi MY (eds) Computer Aided Verification, CAV '98, Vancouver, Canada, LNCS 1427, Springer, pp 546–550
Goodman JE, O'Rourke J (eds) (1997) Handbook of discrete and computational geometry. CRC Press
Gromov D, Geist S, Raisch J (2006) Timed discrete control of a parallel production line with continuous output. In: Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems, Alghero, Italy, pp 205–210
Halbwachs N, Merchat D, Gonnord L (2006) Some ways to reduce the space dimension in polyhedra computations. Form Methods Syst Des 29(1):79–95
Halbwachs N, Proy Y-E, Raymond P (1994) Verification of linear hybrid systems by means of convex approximations. In: International Symposium on Static Analysis, SAS'94
Méndez CA, Cerdá J, Grossmann IE, Harjunkkoski I, Fahl M (2006) State-of-the-art review of optimization methods for short-term scheduling of batch processes. Comput Chem Eng 30: 913–946
Niebert P, Yovine S (2000) Computing optimal operation schemes for chemical plants in multi-batch mode. In: HSCC, pp 338–351
Panek S, Stursberg O, Engell S (2006) Efficient synthesis of production schedules by optimization of timed automata. Control Eng Pract (14):1183–1197
Pettersson P (1999) Modelling and verification of real-time systems using timed Automata: theory and Practice. PhD thesis, Uppsala University
Schilling G, Pantelides CC (1999) Optimal periodic scheduling of multipurpose plants. Comput Chem Eng 23:635–655
Simeonova I, Warichet F, Bastin G, Dochain D, Pochet Y (2005) On-line scheduling of chemical plants with parallel production lines and shared resources: a feedback implementation. In: Proceedings IMACS World Congress, Paris
Sontag ED (1996) Interconnected automata and linear systems: a theoretical framework in discrete time. In: Alur R, Henzinger TA, Sontag ED (eds) Hybrid systems III: verification and control (Lecture notes in computer science) Springer, pp 436–448
Wu D, Ierapetritou M (2004) Cyclic short-term scheduling of multiproduct batch plants using continuous-time representation. Comput Chem Eng (28):2271–2286

Stephanie Geist is currently a Ph.D. student in the Control Systems group at Technische Universität Berlin, Germany. She received her Diploma in automatic control from Otto-von-Guericke University in Magdeburg, Germany, in 2004. Her work is in the area of hybrid control systems and hierarchical control. She is participating in the International Curriculum Option of Doctoral Studies in Hybrid Control for Complex, Distributed and Heterogeneous Embedded Systems and is an active member of the European Network of Excellence project HYCON.

Dmitry Gromov received his Diploma in automatic control from the Belorusian State University of Informatics and Radioelectronics, Minsk, Belarus, in 1996. Currently, he is working towards his PhD degree at Technische Universität Berlin. His research interests include supervisory and optimal control of hybrid systems as well as hierarchical control and process control applications. He is participating in the International Curriculum Option of Doctoral Studies in Hybrid Control for Complex, Distributed and Heterogeneous Embedded Systems and the European Network of Excellence HYCON.

Jörg Raisch is a professor at Technische Universität Berlin, where he heads the Control Systems Group within the Department of Electrical Engineering and Computer Science. He is also head of the Systems and Control Theory Group at the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg, Germany. He studied Engineering Cybernetics and Control Systems at Stuttgart University and UMIST, Manchester. He got his Ph.D and "Habilitation", both from Stuttgart University, in 1991 and 1998, respectively. From 1991–1993 he was a postdoc in the Systems Control Group at the University of Toronto. From 2000–2006 he was a professor at the Otto-von-Guericke University Magdeburg, where he headed the automatic control lab. His research interests are in hybrid systems and hierarchical control and include biomedical control and chemical process control applications.