Modeling and Analysis of Switched Buffer Networks using Hybrid Automata

Goran Frehse and Oded Maler

Verimag, {goran.frehse,oded.maler}@imag.fr,
WWW home page: http://www-verimag.imag.fr/~{frehse,maler}

Abstract. In this work we develop a methodology for modeling and analyzing switched buffer networks. Within our modeling framework one can represent the topology of such a network along with bounds on capacities of buffers and communication channels, and controllers that decide when to open and close these channels. This high-level description is then transformed into a product of hybrid automata which can accomodate for disturbances in the flow rates between buffers, while remaining faithful to conservation laws. The hybrid automaton model can be used to verify properties of the controlled system (as we demonstrate in the paper using the PHAVer tool) and eventually for automatic synthesis of controllers that satisfy requirements or optimize some performance measures.

1 Introduction

Many situation in various application domains can be formalized as *switched buffer networks*, that is, networks of buffers (containers) in which quantities of some substances (liquids, materials, bits) are stored, transformed from one form to another and transported at various rates to other buffers. Among the phenomena that can thus be modeled we mention chemical plants, whole enterprises with their logistics (value chains), communication networks and stream-processing applications (audio, video). A *mode* of such a system is defined by the channels that are active at a given time, which determine the rates of change in the quantities of the substances in all the buffers. Switching occurs while opening or closing channels, starting or stopping a reaction, thus causing the system to move from one mode to another. Hybrid automata provide a natural modeling formalism for such systems, a model on which one can verify properties (lack of overflow or deadlock, arrival of products to certain buffers at pre-specified times and quantities) and even automatically synthesize switching controllers that achieve such goals in an efficient manner. Such verification and synthesis techniques can complement traditional analytic techniques that are harder to apply as the switching aspects become more dominant. Looking from the other side of the spectrum, reachability-based methods can be seen adding more rigor and coverage to simulation-based methodologies.

As a first step toward a computer-aided methodology for designing such systems, we present a simple class of such networks, motivated by chemical engineering applications. For this class of networks we define an automatic and

compositional translation into "linear" hybrid automata (LHA), that is, automata where in each mode the derivatives of all continuous state variable are constant or bounded by some linear relation over constants.

The modeling of switched buffer networks in a manner that can express non determinism in flow rates while obeying conservation laws, requires a syntactic departure from the standard LHA model by allowing continuous variables that are shared by several processes. Fortunately, this variation does not change the expressive power. The automatic translation preserves the semantics and avoids many modeling errors that are likely to occur in a manual process. Once a translation is established, we use the tool PHAVer for verifying the correct functioning of the system [1].

The rest of the paper is organized as follow: In Section 2 we define the class of networks that we deal with. Section 3 describes the automatic translation to hybrid automata. Section 4 illustrates the modeling and translation using a case-study from the VHS project, inspired by chemical engineering applications. We finish with conclusions and plans for future work.

2 Switched Buffer Networks

In this paper we focus on a simple class of networks where substances are only *transported* between buffers without being subject to "reactions" that change their type. Most of the modeling problems and solutions are demonstrated already by this type of networks and their generalization is a topic of ongoing work. The basic entities in our system are

- **Buffers** (nodes, containers): These correspond to physical locations where material can be stored, and between which material can be transferred. Buffers are characterized by their storage capacities.
- **Channels**: (transfers, pipes): These can transport substances between buffers. In this work we focus on continuous channels that reduce at some rate the quantity at a source buffer and augment the quantity with the same rate on the target buffer. Channles are characterized by their rates of transfer.

Let $\mathbb{I}^+$ be the set of positive closed intervals over $\mathbb{R}$.

Definition 1 (Switched Buffer Network). *A switched buffer network* $\mathcal{B} = (\mathcal{S}, \mathcal{C}, \sigma, \gamma)$ *consists of:*

- *A set of buffers* $\mathcal{S} = \{s_1, \ldots, s_n\}$,
- *a set of channels* $\mathcal{C} \subseteq S \times S$,
- *a storage capacity over* $\mathcal{S}$ *given by a function* $\sigma : \mathcal{S} \to \mathbb{R}_+$, *stating an upper bound on the quantity of material at each buffer, and*
- *a channel capacity over* $\mathcal{C}$ *given by a function* $\gamma : \mathcal{C} \to \mathbb{I}^+$, *determining the rates at which material can be transported through an active channel.*

The interval $\gamma(c) = [\underline{\gamma}(c), \overline{\gamma}(c)]$ reflects outside influences (disturbances) on the flow rate in an active channel. The state of a switched buffer network has two components: The *discrete global state* (mode) of the system is a function $p : \mathcal{C} \to \{0, 1\}$ with $p(c) = 1$ indicating that channel c is *active*. The *continuous global state* of the system is a function $x : \mathcal{S} \to \mathbb{R}_+$ indicating the quantities of substances at buffers. We use $P \times X$ to denote the set of *global states* of the form (p, x).

The decisions whether to open or close channels is done by a controller which observes the state of the system. In this paper we restrict ourselves to memoryless controllers, that is, controllers that do not have a state of their own and can be expressed as a function of the form $u : P \times X \to P$ meaning that when the network is in state (p, x), it will switch immediately to $(p', x) = (u(p, x), x)$. Note that this restriction is just for notational convenience and our framework can accomodate for any controller specified as a LHA. The long-term goal of our work is to synthesize such switching controllers automatically from specifications, and as a first step we attack a slightly-easier problem, the verification of a *given* controller.

2.1 Stationary Behavior

The dynamics of the network in a given mode is rather straightforward where each active channel pulls substance from it source buffer and transfers it to its target buffer. There are however two major subtleties in the modeling. The first is related to the proper handling of rate nondeterminism, and the second to situations of *starvation* and *saturation* where buffers, respectively, become empty or reach their maximal capacity. We first describe the *stationary* dynamics of buffers which are not starved nor saturated.

The continuous state x evolves in a discrete state p according to a derivative $\dot{x}$ that depends on the active channels. Due to the nondeterminism in the channel capacities, this derivative is only known up to an interval. However, we wish to impose conservation of material, that is, the material leaving a buffer via some channel must appear in the exact same quantity in the target buffer of the channel. So while the change is only known up to some bounds, there must be a pairwise match across channels that we describe with a function $v : \mathcal{C} \to \mathbb{R}^+$ called *throughput* and satisfying

$$v(s, s') \quad \begin{cases} \in \gamma(s, s') & \text{if } p(s, s') = 1 \\ = 0 & \text{otherwise} \end{cases} \tag{1}$$

The *inflow* and *outflow* of a buffer s are defined as

$$v_{in}(s) = \sum_{s'} v(s', s) \text{ and } v_{out}(s) = \sum_{s'} v(s, s')$$

for any throughput v satisfying (1), and thus the derivative of each buffer satisfies

$$\dot{x}(s) = v_{in}(s) - v_{out}(s). \tag{2}$$

Note that due to the non deterministic choice in (1) this is a differential *inclusion*.

2.2 Starvation and Saturation

When a buffer becomes empty in a discrete global state in which one or more of its outgoing channels is active, we need to fix the throughput of this channel.[1] One solution would be to set the throughput to zero but then, if the buffer admits also an active incoming channel, its quantity will become immediately positive and will lead to a Zeno behavior (infinite number of switchings in the channel throughput). One solution to this problem would be to use *hysteresis*, that is, using a condition like $x(s) = 0$ to deactivate the channel and condition of the form $x(s) > d > 0$ to reactivate it. Another modeling approach would be to declare situations where a buffer overflows or where an empty buffer has an active outgoing channel as *error* states ("never pump from an empty tank"). The form of the appropriate modeling solution may depend on the application domain, and each solution has its cost in terms of adding auxiliary variables or making the dynamics more complex.

For the purpose of this paper we use the following modeling approach. When a buffer s is empty we relax the lower bounds on the throughput of its outgoing channels, allowing them to be as low as zero. The system can stay for a non-zero duration at a state where $x(s) = 0$ only if $v_{in}(s) = v_{out}(s)$, otherwise it moves immediately to a state where $x(s) > 0$. Likewise, when a buffer is full we relax the lower bounds on the rates of incoming channels.

Example 1. An example shall illustrate saturation under the presence of nondeterminism. Consider the buffer network shown in Fig. 1(a). Figure 1(b) shows how the levels can change over time, starting from $x(S_1) = 1000$, $x(S_2) = 850$, and $x(S_3) = 0$. If S_2 drains fast enough, it is empty by the time S_3 reaches saturation. Otherwise, material is left in both S_1 and S_2. While at the end the possible levels of S_1,S_2,S_3 each cover an interval, it is important that the correlation between the possible levels satisfies the material balance $x_1 + x_2 + x_3 = const.$

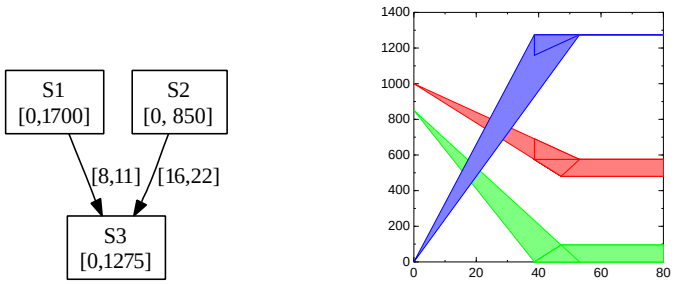
We defer the discussion of changes in the discrete state (corresponding to opening or closing of channels) to the hybrid automaton model.

3 Modeling with Hybrid Automata

We will now show how we transform buffer networks into products of hybrid automata whose set of runs correspond to the possible evolutions of the network. At a first glance, the class of hybrid automata that we use seems richer than LHA, because the derivative of $x(s)$ is a function of the throughput variables of the forms $v(s, s')$ and $v(s', s)$ but since these variables have no "state" of their own (their future value does not depend on their current value), they can be projected away and the obtained automaton, as we shall see, is a LHA.

For the sake of readability we do do not give formal definitions but rather write down the two building blocks that we use, the automata for buffers and for channels.

[1] The case when a buffer becomes *full* when it has an active *incoming* channel is symmetric.



(a) Network graph (b) Levels over time

Fig. 1. Saturation and nondeterminism

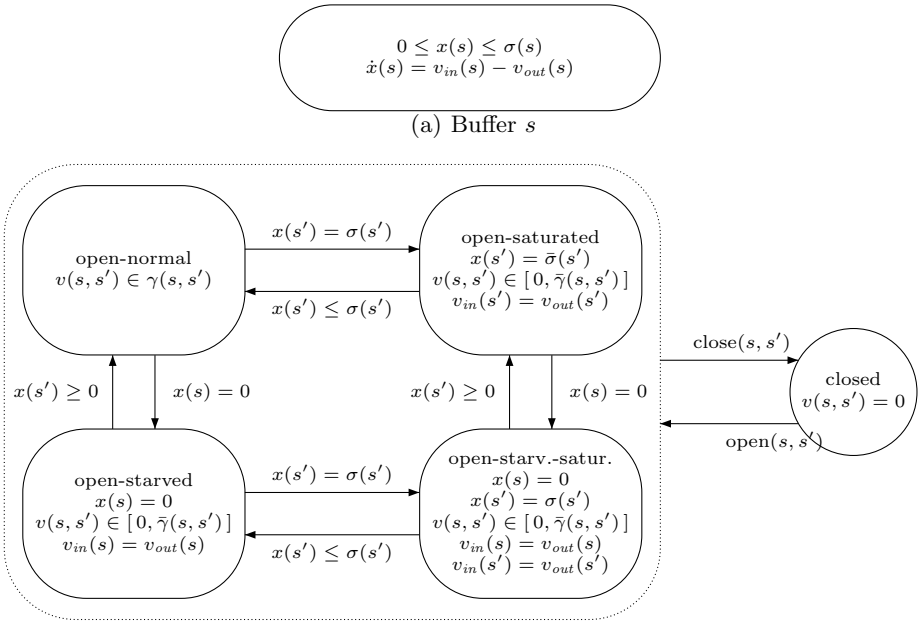
Modeling Buffers and Channels We model each buffer s with the one-state automaton shown in Fig. 2(a). This automaton just realizes the basic balance equation (2) letting $x(s)$ evolve according to the value of the throughput variables that it observes but does not control.

The major ingredient of our modeling approach is the automaton for a channel (s, s') depicted in Fig. 2(b). This automaton has 5 states, one representing an inactive state (closed) and 4 states that refine the active (open) state according to whether the source buffer s is empty and whether target buffer s' is full. The automaton thus observes the values of $x(s)$ and $x(s')$ which determine its discrete state. In each of these states the automaton may choose non deterministically the throughput $v(s, s')$ while satisfying the respective constraints as described in the previous section. The transitions between the *open* states and *closed* state are initiated by the controller. It is not hard to see that these automata represent faithfully the semantics of the buffer network.

Not surprisingly, the number of states in the product automaton grows exponentially with the number of channels: 5^m states for m channels. A common partial remedy to this problem is to generate the states on the fly from an initial set and thus restrict the analysis to reachable states. The exponent can be reduced from 5^m to 2^m if we assume that starvation or saturation of any buffer is considered an error.

Modeling Control Strategies We consider control strategies in which the target state p' differs from the source state p only one channel.[2] Given the network as a hybrid automaton N, the controlled network can be obtained by modifying the transitions as follows: For a location l in the automaton, let $l{\downarrow_P}$ be the discrete global state in location l. For each transition from location l to l' and a controllable label (either open or close, not τ), intersect the guard with $\{x | u(p, x) = p'\}$. Make the transition *urgent* if $p \neq p'$, i.e, the transition is taken immediately when the guard is entered.

[2] Every implementable strategy can be brought to this form by splitting transitions.



(a) Buffer s

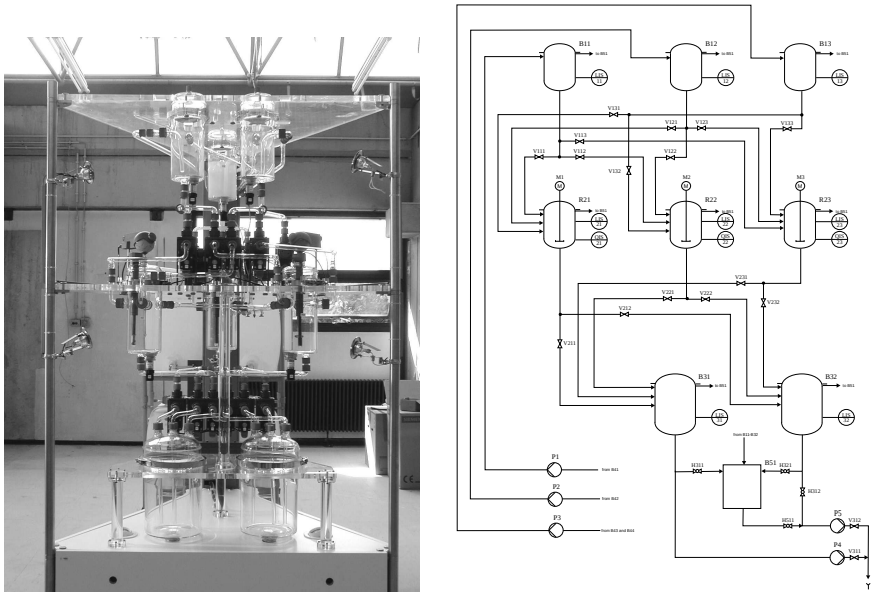
(b) Channel (s, s')

Fig. 2. Hybrid automata models

A simple controller automaton C can be constructed if the initial state of the network is known, say $p = p_0$: Let the locations of C be all possible p, and let there be transitions from all p to all p' when $p \neq p'$, with guard $\{x|u(p,x) = p'\}$. The transitions do not change the continuous state of the automaton, and all transitions of C are urgent.

Transformation into LHA The composed hybrid automaton model has a linear continuous dynamics as $\dot{x}$ is a function of the throughput variables v. However, since the values of these variables are selected by the channel automata without reference to their previous values, and since we do not really care about their values, we can apply quantifier elimination on v and change the dynamics from the form $\dot{x} = f(v)$ to the form $\dot{x} \in F$ where $F = \{y : \exists v f(v) = y\}$. Since polyhedral sets are closed under projection we obtain a LHA.

This concludes the description of our modeling framework which leads to an automatic translation of arbitrary buffer networks with uncertainty in flow rates into linear hybrid automata. In the next section we demonstrate the advantage of having such models by analyzing the behavior of a scheduling policy (controller) for an 8-buffer batch plant.



(a) Physical plant (b) Piping diagram

Fig. 3. Multi-product batch plant

4 Multi-Product Batch Plant

We demonstrate the approach by modeling and analyzing the multi-product batch plant from [2], shown in Fig. 3. The plant has three levels: On the top level, three buffer tanks B11 to B13 contain the raw materials *yellow*, *red* and *white*. On the second level, there are three reactors R21 to R23 that can be filled from B11, B12, B13. Mixing Yellow and White in a reactor results in the product *Blue*, while Red and White become *Green*. From the reactors, the product is drained into either of two buffer tanks B31 and B32 on the third level, from which it is extracted by the consumer. We verify for a given strategy that it never results in overflow of the buffers and reactors (exceed capacity by more than 100ml), and that the buffers $B31$ and $B32$ are never empty, i.e., the consumer demand is always met.

4.1 Plant Model

The plant is readily modeled as a switched buffer network. Figure 4 shows the network graph for the plant with deterministic rates. The network includes the site *delivery* for the source of the raw materials, and a site representing the *consumer*. We examine different scenarios, like variations in consumer demand, for which we replace the deterministic rates by intervals.

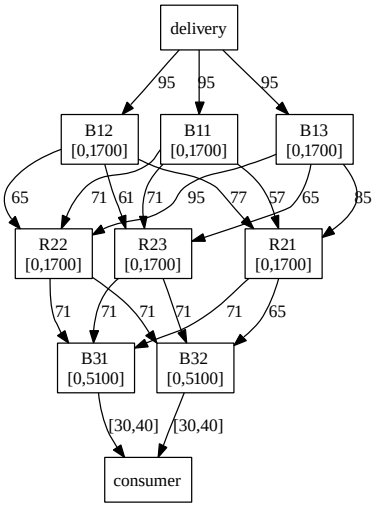


Fig. 4. Network graph

In total, our model has 8 buffers and 20 channels. Since saturation (overflow) is considered a failure case, we can omit the two locations in each channel automaton that correspond to saturation. Each channel then has only 3 locations, and our network model 3^{20}. Since the level of buffers representing the delivery and consumer is not of interest, we replace the differential equation in their automata models with a constraint that fixes the level to be constant – neither in saturation nor starvation.

The automata, together with an automaton modeling the controller, are composed in PHAVer and then the throughput variables as well as the level of source and sink are projected away (composition and projection can be alternated to improve speed).

4.2 Verifying Control Strategies

We verify a control strategy that uses R21 solely to produce Blue, and R22 to produce Green, while R23 is alternatingly used to produce either one, with a timing ratio that slightly in favor of Blue. It is constrained by the specifications that, apart from the product buffers, a buffer should not have more than one active incoming channel, and that White must always enter a reactor after Yellow or Red. The strategy is represented in Table 1 as a sequence of transfers of one or two batches from one buffer to another. Each row represents the transfer of one batch (850ml) of material. The source is indicated by the column and the target by the row. A circle indicated that the buffer or reactor of that column is being filled. After the last row is finished, the process goes back to the first row. The transfers from a reactor to a buffer comprise two batches of material, and so span two rows. Their beginning is indicated with ↑ and their end with

Table 1. Control strategy as sequence of batch transfers (column: from, rows: to)

row	delivery*	B11	B12	B13	R21	R22	R23
1	B11,B13$_2$	∘	–	∘	–	B32↓	B32↑
2	–	–	R22	R21$_1^*$	∘	∘	B32↓
3	B12	R23	∘	R22$_0$	B31↑	∘	∘
4	B11,B13$_2$	∘	–	∘	B31↓	B32↑	–
5	–	R21	–	R23$_1^*$	∘	B32↓	∘
6	B11	∘	R22	R21$_0$	∘	∘	B31↑
7	B12,B13$_2$	–	∘	∘	B31↑	–	B31↓
8	–	–	R23	R22$_1^*$	B31↓	∘	∘
9	B12	R21	∘	R23$_0$	∘	B32↑	∘

* time critical; $_{2,1,0}$ fill/drain to level $x_{B13} = 1700, 850, 0$

↓. When a specific quantity must be transferred and the level at the end of the transfer is larger than zero, the channel must be closed in order to keep more material from passing through. We call such transfers *time critical*, and they are indicated in the table with *.

The strategy is modeled with a hybrid automaton as discussed in Sect. 3. The initial state of the network is fixed (all channels closed), and the controller automaton is simply a sequence of transitions to open and close channels, processing the strategy as indicated in Table 1 row by row. E.g., the transfer from $B12$ to $R22$ on the second row is modeled by a transition with label $open(B12, R22)$ and guard $B12 \geq 850$, followed by a transition with label $close(B12, R22)$ and guard $B12 \leq 0$. The controller initializes the network by delivering raw material to B11,B12,B13, and then starts with at first row. The strategy has a parameter x_{Space}, which determines how much space must be in a product buffer before the reactor is emptied into it. In rows 1 and 7 of Table 1 two reactors drain simultaneously into the same buffer. If not enough space is available, the product buffer will overflow. The controller automaton could be a simple cycle without branching were it not for time critical transfers. When several time critical transfers are running in parallel, it must be taken account in which sequence they could possibly finish. In the controller automaton, whose discrete structure is shown in Fig. 5, this is reflected in the classic diamond shape of interleaving transitions known from concurrent processes. The figure also shows the discrete structure of the controlled process, which shows the superposition of the controller with the nondeterministic transitions between stationary and starved locations in the network model.

We verify the following scenarios:

BP8.1: Deterministic Case The consumer draws at a fixed rate of 1 batch/30sec. from both product buffers B31 and B32. Figure 6(a) shows a plot of the B31 and B32 over time. B32 drops considerably below B31 because of the asymmetry in

Table 2. PHAVer performance*

					Automaton		Reachable Set	
Instance	Time [s]	Mem. [MB]	Depth[a]	Checks[b]	Loc.	Trans.	Loc.	Poly.
BP8.1	120	267	173	279	266	823	130	279
BP8.2	139	267	173	422	266	823	131	450
BP8.3	845	622	302	2669	266	823	143	2737
BP8.4	1243	622	1071	4727	266	823	147	4772

* on Xeon 3.20 GHz, 4GB RAM running Linux; [a] lower bound on depth in breadth-first search; [b] number of applications of post-operator

the strategy, but the specification holds. PHAVer is able to show this within a couple of minutes, see Table 2 for performance measurements. Note that about 90sec of that time are spent composing and projecting the automaton, which is also responsible for the memory consumption.

BP8.2: Varying Initial States We allow the top level buffers to contain anywhere between zero and one batch, and the product buffers to be between 5 and 6 batches. The specification still holds, as PHAVer shows in about the same time, although it finds about 50% more polyhedra. The corresponding plot is shown in Fig. 6(b).

BP8.3: Varying Consumer Demand The initial state is deterministic, as in BP8.1, but the consumer demand varies by ±1sec/batch. PHAVer detects the violation after 845s, and a set of error traces is shown in Fig. 6(c).

BP8.4: Varying but Slow Demand Similar to BP8.3, but with the consumer drawing only 1 batch/100sec. Nonetheless, the strategy fails. PHAVer detects the violation after 1243s, and a set of error traces is shown in Fig. 6(d). In fact, the strategy fails if either one or the other product is consumed quicker. Lowering the demand on B32 leads to starvation of B31. In row 7 the strategy waits for B31 to empty until it may be too late for B32, compare Fig. 6(d).

5 Related and Future Work

The idea of modeling switched buffer systems using hybrid automata is quite natural and has been explored, to a certain extent, in the early days of hybrid systems research, see, for example, [3] or [4] or modeling approaches based on continuous Petri Nets [5]. Dynamic properties, such as stability, of switched buffer network has been the object of study of many papers, e.g., [6, 7], but as in other domains, methods based on hybrid automata are not restricted to the steady-state behavior of the network but can handle also transients. The contribution of our approach is in the combination of a general rigorous translation

combined with the availability of a powerful tool like PHAVer that can handle automata derived from non trivial networks and find subtle bugs in their controllers.

In industrial applications, controller for buffer networks are usually implemented on a Programmable Logic Controller (PLC), which operates in a cycle of input acquisition, computation and update of outputs, with a cycle time in the tens or hundreds of milliseconds. Its control functions are usually specified in high level languages such as Sequential Function Charts (SFCs) that include some degree of concurrency. In the low level implementation, the controller is purely sequential as well as deterministic. A generic approach to translate SFCs into timed automata was proposed in [8], but it models the cycles of the PLC explicitly and introduces the cycle time explicitly into the model. The disparity between the dynamics of the network and the relatively small cycle time dramatically increase the verification costs. For this reason, we prefer the macroscopic view of a controller as a control strategy, which abstracts the cycles to instantaneous (urgent) transitions of the controller.

Timed automata suggest an alternative modeling formalism which can be analyzed more efficiently, but their lack of expressive power will lead to overly-conservative verification results. Using timed automata, buffer levels should be discretized into a finite range from empty to full, and timing bounds assigned to transitions between these states. While the assignment of timing bounds can be made exact in the case where each buffer admits only one active channel at any time, more general situation will lead to large over approximation.

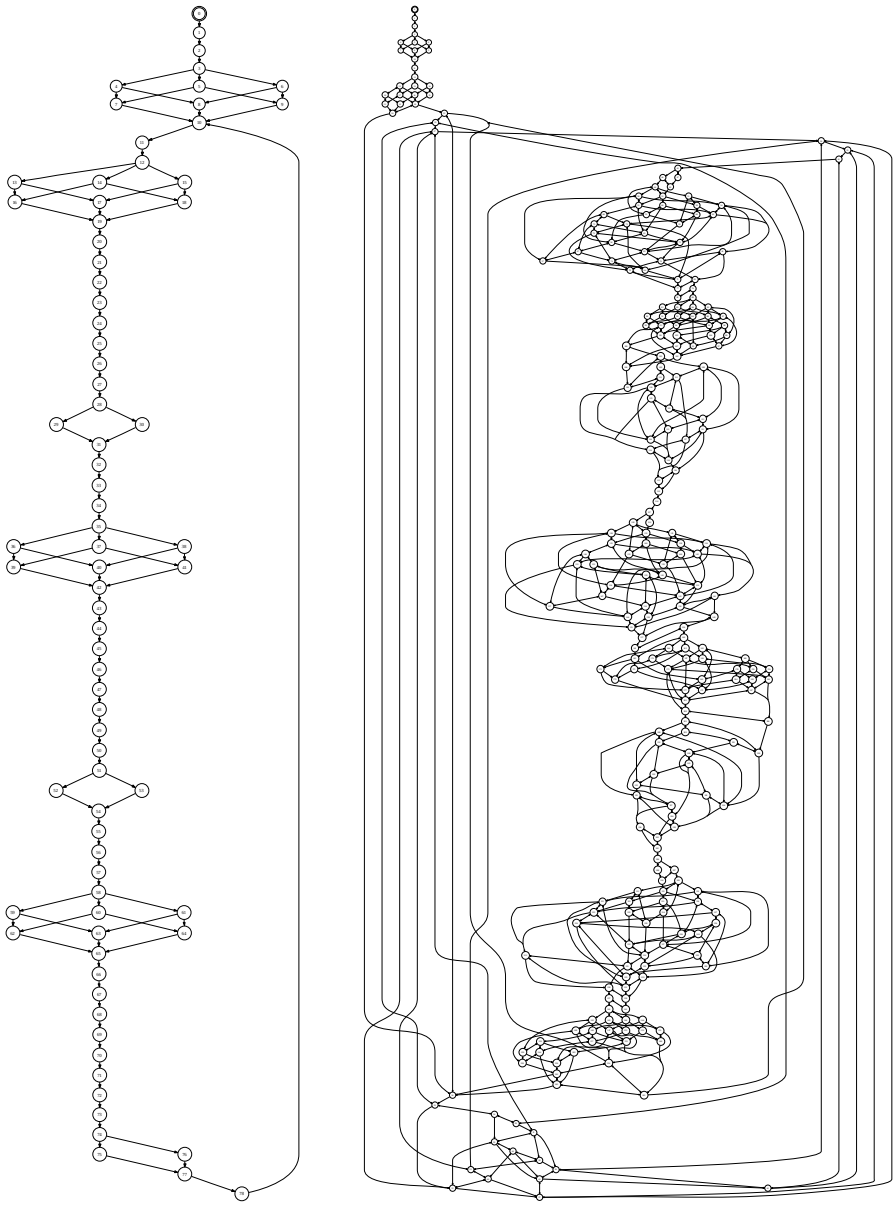
For future work we consider three major directions:

1. *Scaling up*: like any model of interacting components, ours suffers from the state-explosion problem that will not allow us to analyze networks with more than a dozen channels. In order to scale up, an accompanying methodology for abstracting sub-networks (in the spirit of the technique of [9] for timed automata) and compositional reasoning in general [10, 11] should be developed.
2. *Expressivity*: We work on enriching the model to capture more complex situations such as reactions that change substance types, channels that may have several speeds or discrete transportation (quantities are removed from the source buffer and put at the target after some delay).
3. *Synthesis*: We work on adapting the synthesis algorithms described in [12–14] to derive controllers automatically from specifications.

We believe to that the effective analysis of switched buffer networks is a topic of both theoretical and practical interset whose importance will increase in the future, especially for communication and computation application. Our work lays the foundation for a class of promising modeling, analysis and design techniques.

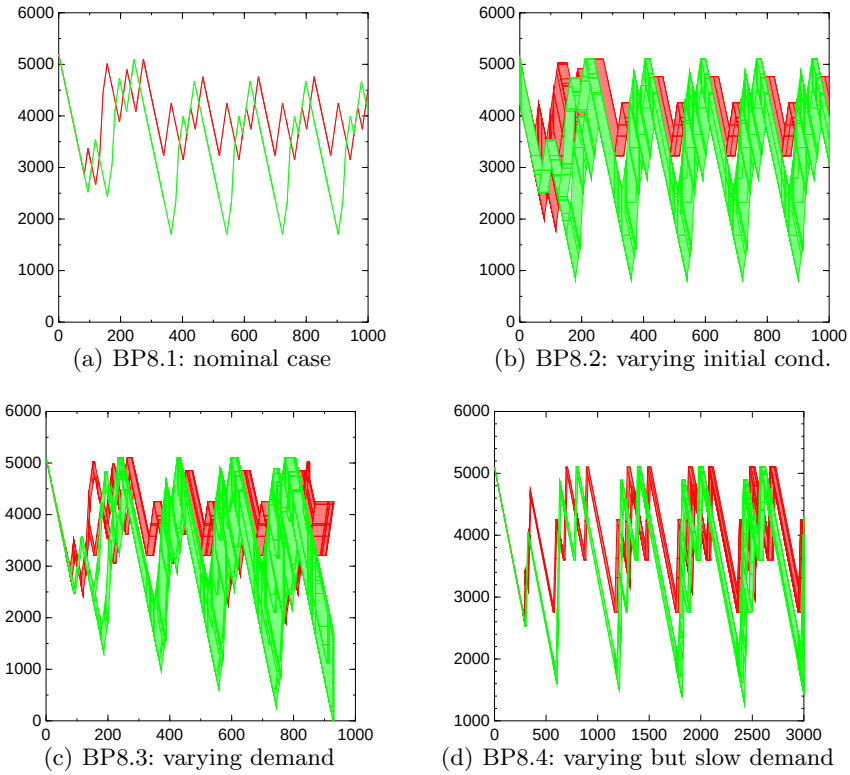
References

1. Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In Manfred Morari and Lothar Thiele, editors, *Hybrid Systems: Computation and Control (HSCC'05), Mar. 9–11, 2005, Zürich, CH*, volume 2289 of *LNCS*. Springer, 2005. PHAVer is available at `http://www.cs.ru.nl/~goranf/`.
2. Nanette Bauer, Stefan Kowalewski, Guido Sand, and Thomas Löhl. A case study: Multi product batch plant for the demonstration of control and scheduling problems. In Sebastian Engell, Stefan Kowalewski, and Janan Zaytoon, editors, *4th Int. Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems (ADPM 2000)*, Berichte aus der Automatisierungstechnik, pages 383–388, Aachen, Germany, 2000. Shaker Verlag.
3. Michael Tittus. *Control Synthesis for Batch Processes*. PhD thesis, Chalmers University of Technology, 1995.
4. Stefan Kowalewski, Sebastian Engell, Jörg Preußig, and Olaf Stursberg. Verification of logic controllers for continuous plants using timed condition/event-system models. *Automatica*, 35(3):505–518, mar 1999. Special Issue on Hybrid Systems.
5. René David and Hassane Alla. *Discrete, Continuous and Hybrid Petri Nets*. Springer, 2005.
6. C. Horn and P.J. Ramadge. Dynamics of switched arrival systems with thresholds. In *IEEE Conf. Decision and Control*, volume 1, pages 288–293, 1993.
7. P.R. Kumar and S.P. Meyn. Stability of queueing networks and scheduling policies. *IEEE Transactions on Automatic Control*, 40(2):251–260, February 1995.
8. Nanette Bauer, Ralf Huuck, Ben Lukoschus, and Sebastian Engell. A unifying semantics for sequential function charts. In Hartmut Ehrig, Werner Damm, Jörg Desel, Martin Große-Rhode, Wolfgang Reif, Eckehard Schnieder, and Engelbert Westkämper, editors, *SoftSpez Final Report*, volume 3147 of *LNCS*, pages 400–418. Springer, 2004.
9. R. Ben Salah, M. Bozga, and O. Maler. Automatic abstraction of timed components. unpublished manuscript.
10. Goran Frehse, Zhi Han, and Bruce H. Krogh. Assume-guarantee reasoning for hybrid i/o-automata by over-approximation of continuous interaction. In *Proc. IEEE Conf. Decision & Control (CDC'04), Dec. 14–17, 2004, Atl., Bahamas*, 2004.
11. Goran Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. PhD thesis, Radboud Universiteit Nijmegen, October 2005.
12. Eugene Asarin, Oded Maler, and Amir Pnueli. Symbolic controller synthesis for discrete and timed systems. In Panos J. Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, *Hybrid Systems*, volume 999 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 1994.
13. H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *IEEE Conf. Decision and Control*, pages 4607–4612, 1997.
14. E. Asarin, O. Bournez, T. Dang, A. Pnueli, and O. Maler. Effective synthesis of switching controllers for linear systems. *Proc. of the IEEE*, 88(7):1011–1025, 2000.



(a) Controller (b) Controlled network

Fig. 5. Discrete structure of hybrid automata models



(a) BP8.1: nominal case

(b) BP8.2: varying initial cond.

(c) BP8.3: varying demand

(d) BP8.4: varying but slow demand

Fig. 6. Levels of product buffers B31 and B32 [ml] over time [s]