Proceedings of the 2001 IEEE International Symposium on Intelligent Control (ISIC 2001), Mexico City, September 5–7, 2001, p. 150–155
(invited session on "Hybrid Systems in Process Control" organized by N. Rakoto-Ravalontsalama and M.D. Lemmon)

Verification of Hybrid Controlled Processing Systems based on Decomposition and Deduction

Goran Frehse[1], Olaf Stursberg[1], Sebastian Engell[1],
Ralf Huuck[2], Ben Lukoschus[2,3]

[1] Process Control Lab (CT-AST), University of Dortmund,
D-44221 Dortmund (Germany), {g.frehse,o.stursberg,s.engell}@ct.uni-dortmund.de

[2] Chair of Software Technology, Institute of Computer Science and Applied Mathematics,
University of Kiel, D-24098 Kiel (Germany), {rhu,bls}@informatik.uni-kiel.de

[3] Visiting the Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA

Abstract— **While formal verification has been successfully used to analyze several academic examples of controlled hybrid systems, the application to real-world processing systems is largely restricted by the complexity of modeling and computation. This contribution aims at improving the applicability by using decomposition and deduction techniques: A given system is first decomposed into a set of physical and/or functional units and modeled by communicating timed automata or linear hybrid automata. The so-called *Assumption/Commitment* method allows to formulate requirements for the desired behavior of single modules or groups of modules.**

Model-Checking is an appropriate technique to analyze whether the requirements (e.g. the exclusion of critical states) are fulfilled. By combining the analysis results obtained for single modules, properties of composed modules can be deduced. As illustrated for a laboratory plant, properties of the complete system for which direct model-checking is prohibitively expensive can be inferred by the iterative application of analysis and deduction.

Keywords— **Abstraction, Assumption/Commitment, Deductive Analysis, Discrete Controller, Hybrid System, Verification.**

I. Introduction

The safe and economical operation of processing systems essentially depends on the correct design of discrete control algorithms, implemented on Programmable Logic Controllers or Distributed Control Systems. These algorithms must ensure certain sequences of process steps, a safe shutdown in abnormal plant situations, and the exclusion of operator inputs that potentially lead to critical process states. In the last decade, *formal verification* has been explored as a possible means to analyze whether the design of a discrete controller indeed fulfills the requirements for a given plant [1]. In particular, different approaches of *model-checking* [2], [3] have been developed for algorithmic controller analysis, see e.g. [4], [5], [6], [7]. The common principle of these approaches is that the controlled process is represented by a state transition system for which a systematic state-space search reveals whether a formally specified property is true for all possible behaviors of the model.

As effective tools are only available for purely discrete or timed state transition systems, approximation techniques were introduced to extend model-checking to different classes of hybrid systems, e.g. [8], [9], [10]. The hybrid modeling of processing systems is important in order to consider the interaction of the continuous process behavior with the discrete control inputs. The drawback of the current approaches for model-checking of hybrid process models is the computational cost: so far, successful applications are restricted to academic examples or small parts of industrial systems [11].

In order to extend the applicability of verification within the context of process control, we suggest to combine model-checking and deduction techniques. The principle is to decompose the system first into a set of modules which are small enough to be analyzed by model-checking. The so-called *Assumption/Commitment* technique allows to prove properties for single modules by making some assumptions about their environments. By combining the results of such local analyses, these assumptions are checked and revised if necessary. If the local analyses and the combination of their results based on deduction are carried out iteratively, global properties of the complete system can be inferred. The advantage of this procedure is that the costly step of model-checking is applied only to small subsystems.

Numerous approaches related to the proposed Assumption/Commitment method can be found in the literature. Depending on the underlying formalism they are called Rely/Guarantee, Assumption/Commitment or Assume/Guarantee [12], [13], [14]. However, most of them were applied to discrete systems and only recently applications to real time [15], [16], [17] and hybrid systems [18], [19] were reported. The following sections describe the specific formulation of the Assumption/Commitment method that we use to analyze hybrid process models. The procedure is illustrated for a batch production system.

II. The Concept of Modular Analysis of Processing Systems

As illustrated in Fig. 1, our concept of a modular analysis of processing systems consists of four steps: the decomposition, the modeling (and abstraction), the analysis of local

This research was supported by the National Science Foundation under grants CCR-00-82560 and CCR-00-86096.

properties, and the deduction of global properties. In the decomposition step the plant is first divided into a set of modules $\{M_1, \ldots, M_n\}$. The modules represent either single pieces of equipment (as reactors, valves, sensors etc.) or groups of devices which form a functional unit (e.g. a tank including the inlet valves if a filling operation is investigated). The decomposition must consider that the modules should lead to models that are suitable for model-checking with respect to their complexity. The behavior is modeled by *Communicating Linear Hybrid or Timed Automata* (CLHA, see Sec. IV). They allow to introduce separate automata $\{S_1, \ldots, S_n\}$ for each module, and the analysis of reachability properties is known to be (semi-)decidable for this class of systems [20]. The modeling usually comprises abstraction in the sense that the parts of the behavior which are irrelevant for the investigation are omitted. However, for some modules a higher modeling accuracy can be necessary which requires more complex hybrid behavior than that provided by CLHA (e.g. switched non-linear differential equations). In this case, approximation routines must be applied in order to obtain verifiable models such as CLHA (see e.g. [10]).

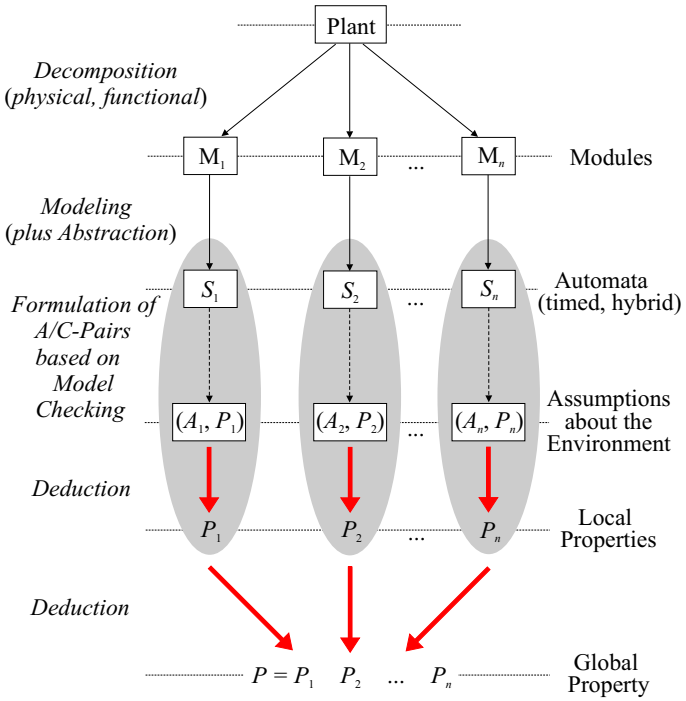

Fig. 1. The Concept

The following step aims at proving that an automaton S_i fulfills a given "local" property P_i. In our case, we predominantly address safety properties, i.e. it must be guaranteed that an undesired state of the automaton S_i is never reached. If the module S_i depends on other modules, P_i might only be true under certain assumptions A_i about these interactions. A so-called Assumption/Commitment pair (A_i, P_i) is formulated expressing that S_i fulfills a commitment corresponding to the property P_i for some assumptions A_i about its environment. The fulfillment of (A_i, P_i) as well as the assumptions about the environment can be proved by model-checking. After it is shown that a set of local properties $P_1, \ldots, P_n$ is true for the corresponding automata $S_1, \ldots, S_n$, the step of deduction derives a global property P. This step usually involves logical operations on the conjunction of the local properties $P_1, \ldots, P_n$ in order to show that a logical formula P is true or false. The deduction can be performed by hand or with the help of a theorem prover. In the next two sections, we explain the formulation of A/C-pairs and the deduction of local and global properties in more detail.

III. Principles of the Assumption/Commitment Method

In general, the Assumption/Commitment method aims at proving that a property P is true for a system S under some assumptions about its environment. A modular analysis can be carried out if S can be decomposed into modules $S_1, \ldots, S_n$ and P can be split up into properties $P_1, \ldots, P_n$ (with S_i corresponding to P_i only). In most cases, $S_i \models P_i$ is rarely fulfilled for all possible behaviors of S_i but rather depends on certain assumptions A_i on the environment of S_i, which is given by $E_i = S_1 || \ldots || S_{i-1} || S_{i+1} || \ldots || S_n$. We write

$$S_i \models (A_i, P_i), \tag{1}$$

if S_i commits itself to P_i under the assumption A_i, and (A_i, P_i) is called an Assumption/Commitment pair. Having shown that $S_i \models (A_i, P_i)$ is valid for all $i \in \{1, \ldots, n\}$, it can be deduced that P is fulfilled for the composition of all modules if additional logical conditions $B(P_1, \ldots, P_n)$ for the local properties are satisfied:

$$\frac{S_1 \models (A_1, P_1) \wedge \ldots \wedge S_n \models (A_n, P_n) \wedge B(P_1, \ldots, P_n)}{(S_1 || S_2 || \ldots || S_n) \models (A, P)}. \tag{2}$$

Here, A denotes the fraction of the assumptions that remains "open", i.e. cannot be confirmed based on the P_i. In order to show that $S \models P$, the P_i must complement the A_i in such a way that $A = True$. To complete the proof of $S \models P_i$, it must be shown that the environments E_i do indeed fulfill the assumptions A_i. Since the environment E_i itself depends on the interaction with S_i, the behavior of the environment must be considered which corresponds to the behavior of S_i under the assumption A_i. We use S_i^P to denote the restriction of S_i to the behavior which fulfills P_i, plus the behavior that immediately leads to states that violate P_i. This extension is necessary in order to be able to analyze the following equations by reachability analysis. The behavior of E_i under the assumption A_i is denoted by E_i^A. We summarize as follows: $S_i \models (A_i, P_i)$ holds if

$$S_i || E_i^A \quad \models \quad P_i. \tag{3}$$

Embedded in the complete system, S_i then fulfills the property P_i if

$$S_i^P || E_i \quad \models \quad A_i. \tag{4}$$

If there is no feedback from the module S_i to its environment E_i, i.e. S_i cannot restrict E_i, (4) simplifies to $E_i \models A_i$.

Model-checking can be used to show (3) and (4), (see Sec. IV-B). If necessary, A_i can be split into parts, each referring only to a corresponding set of modules of E_i. For each of these parts, (3) and (4) must be shown. Note that the proof according to (3), (4) is circular. In order to break the loop additional logical conditions for the property P_i are considered. These conditions, named $B(P_1, \ldots, P_n)$ above, can e.g. refer to the initial state of the system S.

IV. Using the A/C-Method to Verify Controllers for Processing Plants

As stated in [21], a main problem in applying the A/C-method is that the environment E_i includes the entire system except for S_i. This often makes it impossible to show (4) because of complexity reasons. When applying the approach to process control systems, we profit from the special structure of the setting. First, the behavior of a piece of equipment is usually influenced only by a small number of adjacent devices. It often suffices to investigate a small set of devices decoupled from the remainder of the plant. Second, many discrete control functions, especially those which guarantee process safety, act only locally, i.e. they obtain measurements only from a few sensors and also affect only a small number of actuators. This considerably simplifies the application of the A/C-method since it reduces the effort to model and to analyze E_i.

A. Modeling with Communicating Linear Hybrid Automata

An important step is to model the modules and the environment (for a given assumption) such that (i) the communication between S_i and E_i can be represented, and (ii) the model can be analyzed by model-checking. We therefore define *Communicating Linear Hybrid Automata* (CLHA) as an extension of the known linear hybrid automata [22] by input and output variables: A CLHA is a 6-tuple

$$CLHA = (Loc, Var, Lab, Edg, Act, Inv)$$

with:

- a finite set of locations $Loc = \{q_1, \ldots, q_p\}$.
- a finite set of real variables $Var = Var_{in} \cup Var_{int}$ divided into two disjoint sets of input and internal variables. A subset $Var_{out} \subseteq Var_{int}$ defines the output variables. A valuation ν (contained in a valuation set V) is a function which assigns a real value to each variable $x \in Var$, i.e., $\nu(x) \in \mathbb{R}$. A state of $\mathcal{A}$ is a pair (q, ν) of a location and a valuation.
- a finite set $Lab = Lab_{rec} \cup Lab_{send} \cup Lab_{sync}$ that consists of three disjoint sets of symbolic labels, called *receive labels*, *send labels* and *synchronization labels*.
- a finite set Edg of discrete transitions. Each transition $e = (q, l, \rho, q')$ between two locations $q, q' \in Loc$ depends on a label $l \in Lab \cup \{\tau\}$, with τ denoting an internal transition, and an enabling transition relation $\rho \subseteq V \times V$. The transition e is enabled in state (q, ν) if and only if a valuation ν' with $(\nu, \nu') \in \rho$ exists. ν' denotes the evaluation that results from the transition taken in state (q, v). We require $\nu(x) = \nu'(x)$ for all $x \in Var_{in}$, since input variables cannot be changed by discrete transitions. Furthermore, we require that for any location $q \in Loc$ there is a special internal transition $(q, \tau, \{(\nu, \nu) | \nu \in V\}, q) \in Edg$, called "stutter transition".
- A labeling function $Act : Loc \times Var_{int} \to \mathbb{R}$ that denotes the constant rate $Act(q, x) = k$, $k \in \mathbb{R}$ with which the internal variable x changes in location q.
- a labeling function $Inv : Loc \to 2^V$ assigning an invariant $Inv(q) \subseteq V$ to each location $q \in Loc$.

Informally, the behavior of *CLHA* can be understood as follows: Starting from an initial state (q_0, v_0) the automaton remains in the current location until a transition is taken. The continuous evolution within a location is determined by the activities which are assigned to the internal variables. A transition must be taken before the invariant is evaluated to be false. A transition depends on the internal and input variables, but can also be triggered by synchronization with other modules. The parallel composition of two *CLHA* based on synchronization is defined as follows: A transition that depends on an input label $l \in Lab_{rec}$ can occur only when a corresponding output label of another automaton is available. The labels Lab_{sync} are used to model that two automata synchronize on a transition which is labelled by $l \in Lab_{sync}$ in both automata. Finally, a label $l \in Lab_{send}$ refers to the case that the transition does not depend on another automaton but l emits the information about the transition to the environment. The output variables $x \in Var_{out}$ are used to make the valuation of the internal variables available for other modules.

B. Formulation and Model-Checking of A/C-Pairs

CLHA are used to model the systems, environments and assumptions occurring in (3) and (4). When modeling a subsystem S_i, we restrict the behavior to its desired behavior and include an error state that represents the deviation from normal operation. This is sufficient since we want to investigate only the occurrence of a failure, not its subsequent effects. The modeling is correct if the error state is not reachable in the composed system. The reachability is included in the property P_i that we want to check for S_i. Using this approach, the size of the models can be reduced significantly.

The environment model E_i^A must include all plant modules which have an influence on S_i, e.g. the inlet pipes that supply material to a storage tank. Furthermore, we introduce separate automata for the (uncontrolled) plant module S_i and its local controller, i.e. the controller is considered as a part of the environment E_i^A. The assumptions A_i refer to the restrictions that are introduced for the models that form E_i. Since the operation of the plant is known, one can usually specify the restrictions that are likely to enable that S_i fulfills its commitment. The assumptions lead to a reduced representation of the behavior of the environment and therefore to a significantly smaller model. For the composition of the CLHA representing E_i^A and S_i, model-checking reveals if the error states are not reachable, i.e. (3) is true.

To check (4), we need to model a test automaton A_i^T corresponding to A_i. It contains a state "fail" and must be constructed such that

$$S_i^P||E_i||A_i^T \models \neg reach(fail) \quad \Rightarrow \quad S_i^P||E_i \models A_i. \qquad (5)$$

A_i^T models the assumptions and is extended by a failure state which is reachable only if the assumptions are not fulfilled. All three automata communicate with each other by synchronization labels and input/output variables. For the composition $S_i^P||E_i||A_i^T$, model-checking then shows if the state "fail" is reachable. If it is not, the proof according to (3) and (4) is completed, otherwise the controller has to be corrected.

By deduction it is shown that each A_i is fulfilled by $P_1, \ldots, P_n$. This can be carried out by hand or with the aid of software tools for theorem proving. If it is possible to find a set of properties P_j, $j \in J \subseteq \{1, \ldots, n\}$ for each A_i such that $A_i = \bigwedge_{j \in J} P_j$, the deduction becomes trivial and S fulfills the global property P.

V. Example: A Multi-Product Batch Plant

A. Plant Description

In order to illustrate the proposed approach in a realistic setting, a laboratory batch plant is chosen as an example (see Fig. 2). It consists of three tanks to store raw materials (upper level), three reactors to produce two different products (medium level), and two storage tanks on the bottom level. The objective is to meet a constant demand for the products by assigning the vessels to different production tasks. The raw materials are delivered to the upper level tanks according to a defined schedule. The plant operation relies on control programs (implemented on a PLC) which switch the valves in order to direct the material through the production lines. Since the control operations depend on the continuous behavior within the vessels and on the duration of the chemical reactions, the behavior of the process is hybrid.

In the following sections, we focus on the left part of the plant, which comprises the reactor R21, the valves V111, V131 and V211, as well as the tanks B11 and B31. Using the Assumption/Commitment approach we want to investigate whether the controller operates the plant such that the storage tank B31 can never overflow and can never run empty.

B. Modeling and Analysis for a Product Tank

We first focus on the behavior of the controlled vessel B31, and assume that we can abstract from the other plant components. Figure 3 shows a very simple CLHA modeling the liquid level in the tank S_1. The tank has a constant outflow of $r_1 = 1\ cm\ sec^{-1}$ and an inlet valve that fills the tank at a rate of $r_2 = 7/3\ cm\ sec^{-1}$. If the level falls to zero the tank goes into an alarm state, and when it rises above h_{max} the excess liquid is drained by an overflow pipe. The tank is connected to a controller, modelled by S_2, that has to keep the level h between 0 and h_{max}. The controller opens the inlet valve when the level falls below

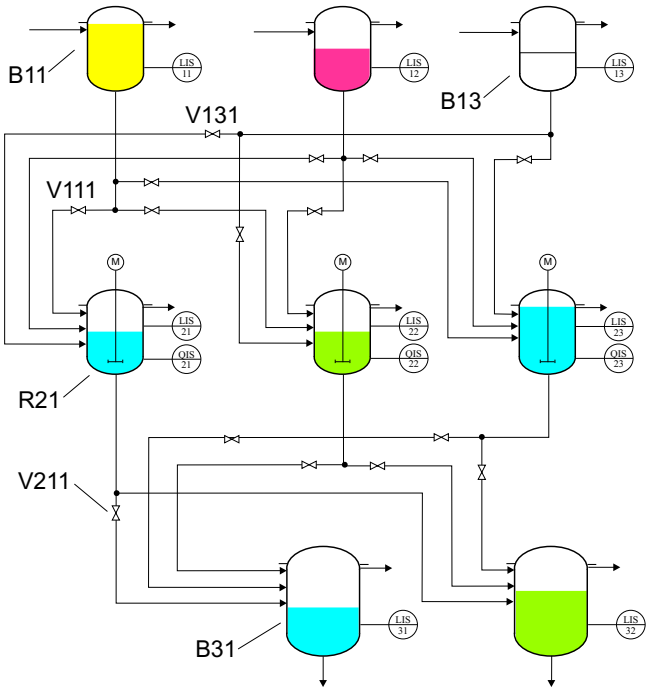

Fig. 2. Multi-Product Batch Plant

h_{fill} and closes it when the level reaches h_{drain}. However, the controller has other tasks as well and is thus busy for durations t_{fill} and t_{drain} after interacting with the tank. The controller communicates with the tank through the labels "drain", "fill" of the type Lab_{send}, therefore marked with "!", and it receives the output variable h of the tank module. The tank has the corresponding labels of the type Lab_{rec}, marked with "?".

The property to be shown is $P = (0 < h < h_{max})$. The Assumption/Commitment method first calls for the analysis of the tank model combined with an abstract model of the environment – the latter is reduced to the controller in this section. We choose the abstract controller model S_2^A shown in Fig. 3, which consists of only two states. It sends the labels "drain" and "fill" just in time so the tank does not overflow or drain fully. The CLHA are translated to hybrid automata and, using the model-checking tool HyTech [23], it is verified that:

$$S_1||S_2^A \models P, \qquad (6)$$

i.e. the error state cannot be reached.

Then it must be checked under which circumstances the controller fulfills its abstraction. To do so, a test automaton S_2^{AT} (see Fig. 3) is constructed that can reach the state "fail" only if the controller violates its abstraction. This analysis includes a tank model S_1^P that is restricted to the behavior that fulfills the property P. We compose the test automaton with the complete controller model and S_1^P and check the equation:

$$S_1^P||S_2||S_2^{AT} \models \neg reach(fail). \qquad (7)$$

The parametric analysis with HyTech yields the result that the state "fail" is not reachable if the following inequalities

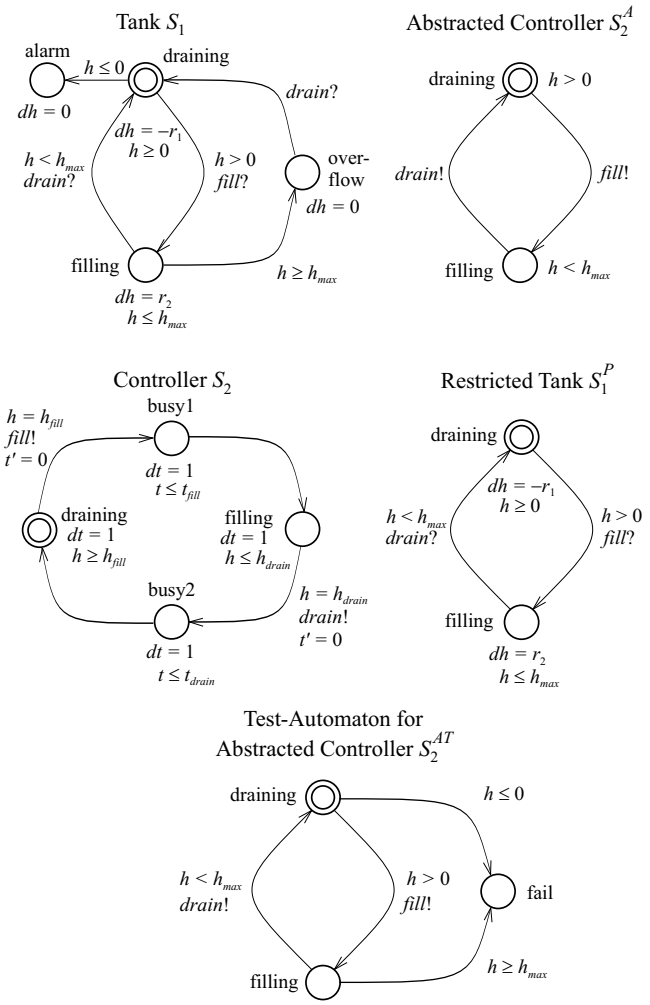

Fig. 3. Automata for the tank B31 (S_1) and its controller

are fulfilled:

$$\begin{aligned} h_{drain} &> h_{fill} \\ h_{drain} &> r_1 t_{drain} \\ h_{fill} &< h_{max} - r_2 t_{fill} \end{aligned} \tag{8}$$

Hence, we get the obviously correct result that the controller must not be busy longer than it takes for the tank to fill up or to empty.

C. Deductive Analysis for a Set of Modules

The plant part in the previous section was chosen in order to illustrate the CLHA modeling and the proof for one A/C-pair. We now enlarge the scope of analysis to other components influencing the controller and the tank B31. Obviously, the supply of liquid from reactor R21 through valve V211, and also the feed into R21 have an effect on the liquid level in B31. For simplicity, we make the assumption that it is sufficient to consider B11, B13, V111, V131, R21, V21 and B31 and the supply of raw material to B11 and B13.

The first step in successively involving further components is to analyze whether B31 does not run empty or overflow (commitment $\hat{B}31$) under the assumption that valve V211 is opened and closed at appropriate points of time (denoted $\hat{V}211$). Next it has to be shown that the switching of the valves is achieved by the controller ($\hat{C}ontr.2$) and under the condition that the reactor R21 can deliver the required liquid (assumption $\hat{R}21.2$). The latter assumption is then verified depending on the behavior of the valves V111, V131 and the corresponding controller function. Following this scheme, the scope of analysis is extended to all components which (indirectly) influence the behavior of tank B31. Table I lists all steps which are involved in this investigation. In each step, the assumptions and commitments (which are briefly explained in Tab. I) are modeled as CLHA, and model-checking proves that the commitments are fulfilled.

Finally the results of these 17 steps of the algorithmic analysis have to be combined by deduction. As for this example, it is often possible to formulate each of the assumptions as a conjunction of the commitments proven in a different step. Hence, these elements can be discharged from the logical expressions that combines the analysis results, and it immediately follows that the commitment $\hat{B}31$ is invariantly true. Note that the parallel composition of automata modeling $Contr.1$, ..., $Contr.10$ yields a controller that keeps the tank level within its limits. This can be exploited for controller synthesis and shall be investigated in the future.

VI. Conclusions

This contribution describes an approach to tackle the problem that hybrid controlled processing systems are in general too complex for an algorithmic analysis. The decomposition into small modules usually leads to pairs of plant components and local properties which have an appropriate size for model-checking. However, it has to be shown that the environment of a module does behave such that the local property is always true. The Assumption/Commitment method provides a suitable means to use abstract models of the environment, and hence to reduce the effort for the analysis. By successively enlarging the environment model by further A/C-pairs (representing relevant plant components) it can be proven that a local property is true for any plant behavior that might occur, and therefore is a global property of the plant. The method can include modules which are modeled by untimed automata, timed automata and simple hybrid systems, as the class of CLHA used here.

The advantage of the approach becomes obvious if we compare the computation times required for the combined algorithmic and deductive approach with the one for a single analysis of the composition of all involved components (which is possible for this rather small example): While the latter takes about 10 minutes and around 70 MB of memory on a standard PC (Celeron, 466 MHz), each of the 17 steps in the combined approach could be finished in less than 10 seconds and with less than 1 MB of memory. However, one must balance this against the effort that is necessary for finding appropriate A/C-pairs. The number of steps largely depends on choosing appropriate assumptions and

TABLE I

STEPS OF THE ALGORITHMIC ANALYSIS

Step	Assumption	Commitment	Description of the requirements
1	$\hat{V}211.1$	$\hat{B}31$	Opening and closing of valve V211 assures that B31 does not overflow or run empty.
2	$\hat{C}ontr.2 \wedge \hat{R}21.2$	$\hat{V}211.1$	Controller acts on V211 in time and R21 has product ready.
3	$\hat{V}111.3 \wedge \hat{V}131.3 \wedge \hat{C}ontr.3$	$\hat{R}21.2$	Valves must drain raw materials to R21 in time.
4	$\hat{C}ontr.4 \wedge \hat{B}11.4 \wedge \hat{R}21.4$	$\hat{V}111.3$	Tank B11 must be ready for draining.
5	$\hat{C}ontr.5 \wedge \hat{B}13.5 \wedge \hat{R}21.5$	$\hat{V}131.3$	Tank B13 must be ready for draining.
6	$\hat{C}ontr.6 \wedge \hat{B}11del.6$	$\hat{B}11.4$	Tank B11 must be ready for the delivery of raw material.
7	$\hat{C}ontr.7 \wedge \hat{B}13del.7$	$\hat{B}13.5$	Tank B13 must be ready for the delivery of raw material.
8	$\hat{C}ontr.8 \wedge \hat{V}131.8 \wedge \hat{V}211.8$	$\hat{R}21.4$	Reactor R21 has to be ready for being filled from B11.
9	$\hat{C}ontr.8 \wedge \hat{V}111.9 \wedge \hat{V}211.8$	$\hat{R}21.5$	Reactor R21 has to be ready for being filled from B13.
10	$Contr \wedge \hat{B}31.10 \wedge \hat{V}111.9 \wedge \hat{V}131.8 \wedge \hat{R}21.2$	$\hat{B}11del.6$	Controller must comply with the delivery of raw materials to B11.
11	$Contr \wedge \hat{B}31.10 \wedge \hat{V}111.9 \wedge \hat{V}131.8 \wedge \hat{R}21.2$	$\hat{B}13del.7$	Controller must comply with the delivery of raw materials to B13.
12	$\hat{C}ontr.4 \wedge \hat{B}11.4 \wedge \hat{R}21.4$	$\hat{V}111.9$	The opening of valve V111 transfers liquid from B11 to R21.
13	$\hat{C}ontr.5 \wedge \hat{B}13.5 \wedge \hat{R}21.5$	$\hat{V}131.8$	The opening of valve V131 transfers liquid from B13 to R21.
14	$\hat{C}ontr.14$	$\hat{V}211.8$	Valve V211 must be ready for opening.
15	$True$	$\hat{C}ontr.2 - 8, 14$	Controller always fulfills the assumptions used above.
16	$V211 \wedge \hat{R}21.2 \wedge \hat{C}ontr.16$	$\hat{B}31.10$	B31 behaves according to controller commands if valve V211 and reactor R21 are ready to supply the product.
17	$True$	$\hat{C}ontr.16$	Controller closes valve V211 12 sec. after opening.

commitments, and often it is not obvious in which sequence the plant components should be considered. The focus of future research is on developing strategies for deriving and combining A/C-pairs in an efficient manner.

Acknowledgement

The authors thank S. Kowalewski and Y. Lakhnech for initiating this research. The financial support from the German Research Foundation (DFG) under grants KO 1430/6-1 and LA 1012/5-1 is gratefully acknowledged.

REFERENCES

[1] S. Kowalewski, P. Herrmann, S. Engell, R. Huuck, H. Krumm, Y. Lakhnech, B. Lukoschus, and H. Treseler, "Approaches to the formal verification of hybrid systems," *at – Automatisierungstechnik*, vol. 49, no. 2, pp. 66–74, 2001.

[2] E.M. Clarke and E.A. Emerson, "Design and synthesis of synchronization skeletons for branching time temporal logic," in *Logics of Programs Workshop*, Dexter Kozen, Ed. 1982, vol. 131 of *LNCS*, pp. 52–71, Springer.

[3] J. P. Queille and J. Sifakis, "Specification and verification of concurrent systems in CESAR," in *Proc. 5^{th} Int. Symp. on Programming*. 1982, vol. 137 of *LNCS*, pp. 337–351, Springer.

[4] A. L. Turk, S. T. Probst, and G. J. Powers, "Verification of real-time chemical processing systems," in *Hybrid and Real-Time Systems*. 1997, vol. 1201 of *LNCS*, pp. 259–272, Springer.

[5] I. Moon, G. J. Powers, J. R. Burch, and E. M. Clarke, "Automatic verification of sequential control systems using temporal logic," *AIChE Journal*, vol. 38, no. 1, pp. 67–75, 1992.

[6] T. Park and P. I. Barton, "Formal verification of sequence controllers," *Comp. and Chemical Engineering*, vol. 23, pp. 1783–1793, 2000.

[7] S. Kowalewski, S. Engell, J. Preussig, and O. Stursberg, "Verification of logic controllers for continuous plants using timed condition/event system models," *Automatica*, vol. 35, no. 3, pp. 505–518, 1999.

[8] A. Chutinan and B. H. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximation," in *Hybrid Systems: Computation and Control*. 1999, vol. 1569 of *LNCS*, pp. 76–90, Springer.

[9] T. Dang and O. Maler, "Reachability analysis via face lifting," in *Hybrid Systems – Computation and Control (HSCC98)*. 1998, vol. 1386 of *LNCS*, pp. 96–109, Springer.

[10] O. Stursberg, "Analysis of switched continuous systems based on discrete approximation," in *Proc. 4^{th} Int. Conf. on Automation of Mixed Processes*, 2000, pp. 73–78.

[11] S. Engell, S. Kowalewski, C. Schulz, and O. Stursberg, "Continuous-discrete interactions in chemical processing plants," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1050–1068, 2000.

[12] J. Misra and K. M. Chandy, "Proofs of networks of processes," *IEEE Transactions on Software Engineering*, vol. 7, no. 7, pp. 417–426, 1981.

[13] A. Pnueli, "In transition for global to modular temporal reasoning about programs," in *Logics and Models of Concurrent Systems*. 1984, vol. 13 of *NATO ASI-F*, Springer.

[14] M. Abadi and L. Lamport, "Conjoining specification," *ACM Transactions on Programming Languages and Systems*, vol. 17, no. 3, pp. 507–534, 1995.

[15] J. Hooman, "Compositional verification of real-time applications," in *Proc. Compositionality - The Significant Difference*. 1997, vol. 1536 of *LNCS*, pp. 276–300, Springer.

[16] E. Chang, Z. Manna, and A. Pnueli, "Compositional verification of real-time systems," in *Proc. 9^{th} IEEE Symp. Logic in Computer Science*, 1994, pp. 458–465.

[17] R. Alur and T. A. Henzinger, "Modularity for timed and hybrid systems," in *Proc. of the 8th Int. Conf. on Concurrency Theory*. 1997, vol. 1243 of *LNCS*, pp. 74–88, Springer.

[18] T. A. Henzinger, S. Qadeer, S. K. Rajamani, and S. Tasiran, "You assume, we guarantee: Methodology and case studies," in *Proc. 10^{th} Int. Conf. on Computer-Aided Verification*. 1998, vol. 1427 of *LNCS*, pp. 440–451, Springer.

[19] R. Alur, L. de Alfaro, T. A. Henzinger, and F. Y. C. Mang, "Automating modular verification," in *Proc. 10^{th} Int. Conf. on Concurrency Theory*. 1999, vol. 1664 of *LNCS*, pp. 82–97, Springer.

[20] G. Lafferriere, G. J. Pappas, and S. Yovine, "Decidable hybrid systems," in *Proc.: School on Computational Aspects and Applications of Hybrid Systems*, Grenoble, 1998.

[21] O. Kupferman and M. Y. Vardi, "On the complexity of modular model checking," in *Proc. 10^{th} IEEE Symp. on Logic in Computer Science*, 1995, pp. 101–111.

[22] 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*, vol. 138, pp. 3–34, 1995.

[23] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," *Software Tools for Technology Transfer*, vol. 1, no. 1/2, pp. 110–122, 1997.