Timed Condition/Event Systems: A Framework for Modular Discrete Models of Chemical Plants and Verification of Their Real-Time Discrete Control

Stefan Kowalewski* and Jörg Preußig*

ABSTRACT This paper describes the use of timed Condition/Event (C/E) systems, a real-time extension of the C/E system framework introduced by Sreenivas and Krogh, for building models of chemical plants in a modular fashion and as a basis for the model-based analysis of their discrete control. The approach is illustrated by applying it to the safety control logic of a laboratory batch process.

1 Introduction

Most plants in the chemical industry are controlled by discrete controllers, realized by distributed control systems (DCS), programmable logic controllers (PLC), PC's or dedicated hardware. These controllers have to ensure correct startup and shutdown, realize sequence control, and guarantee safe and reliable process operation. Although there are successful research activities in the field of controller verification (see for example [2], [6], and [7]), up to now the correctness of discrete controllers is only tested manually and no model-based analytical verification methods are applied in practice. The main reason is that the development of a formal model of the uncontrolled plant behavior including all disturbances and operator failures is hardly feasible due to the complexity of such systems. To overcome this situation, modeling procedures have to be developed which make it possible to handle complexity by appropriate structuring strategies. One approach in this direction is modular modeling, i. e. the building of complex system descriptions by setting up small, local models independently

*Lehrstuhl für Anlagensteuerungstechnik (Process Control Group) Fachbereich Chemietechnik (Chemical Engineering Dept.) Universität Dortmund, D-44221 Dortmund, Germany, email: stefan@ast.chemietechnik.uni-dortmund.de

from each other and representing the interaction by signals connecting the subsystems. A modeling framework which is particularly well suited for modular modeling are the Condition/Event systems (C/E systems) introduced by Sreenivas and Krogh [9]. C/E systems provide the possibility to couple interconnected discrete event systems by real-time signals in a block diagram and signal flow fashion.

In this paper we will describe the use of the C/E system framework to build discrete models of chemical processes and to analyse discrete controllers for such processes. Since the latter task makes it necessary to treat timing functions in the control programs whereas the original C/E model is untimed, we introduce a real-time extension for C/E systems and describe reachability analysis for timed C/E systems.

The paper is organized as follows. In the next section the basic concept of a C/E system is reviewed and illustrated with the help of a small example. Section 3 presents the real-time extension of C/E systems which is realized by introducing C/E timers and the corresponding analysis. The modeling and analysis approach is then applied to a process control example in Sec. 4. Finally, in Sec. 5, we draw some conclusions, look at the relation of our approach to related work and discuss possible directions for future research.

2 Condition/Event Systems: Basic Concepts

Condition/Event systems were introduced by Sreenivas and Krogh in [9]. The authors were motivated by the observation that in existing DES models the interaction between systems is either based on synchronization of events or conditioning of event occurence depending on state information. Sometimes, both options are used for different purposes as for instance in the supervisory control theory of Ramadge and Wonham [8]. To incorporate both concepts into a single, unified representation of interconnected DES, Sreenivas and Krogh define two classes of signals which can both be input and output signals of one system: *condition signals* and *event signals*. A condition signal $c(\bullet)$ is a right continuous function $c : [t_0, \infty) \rightarrow C$ with limits from the left, with C being a nonempty, finite, and countable set of conditions. Therefore, condition signals are time-dependent signals which are piecewise constant. An event signal $e(\bullet)$ is a function $e : [t_0, \infty) \rightarrow E_0 = E \cup 0$ for which $e(t_0) = 0$ (the *null* or *zero event*) and $t \in [t_1, t_2] | e(t) \neq 0$ is finite for all finite intervals $[t_1, t_2] \in [t_0, \infty)$ and E is a nonempty, finite, and countable set of events. We may say that event signals are only pointwise "nonzero". The set of all condition or all event signals on $[t_0, \infty)$ is written as $\mathbf{C}(C, t_0)$ or $\mathbf{E}(E, t_0)$, respectively.

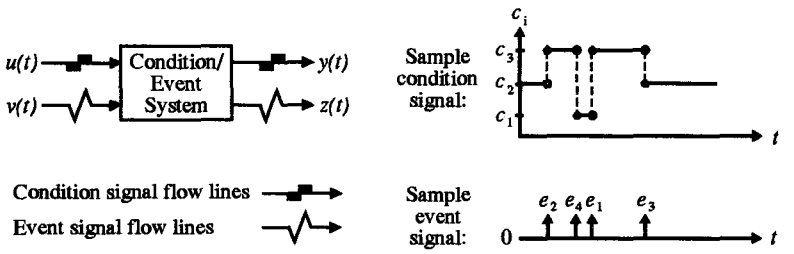

FIGURE 1. Condition/Event signals and systems.

In general, a *C/E system* has a *conditon input signal* $u(t)$, an *event input signal* $v(t)$, a *condition output signal* $y(t)$ and an *event output signal* $z(t)$, as it is illustrated in Fig. 1. The corresponding sets of conditions or events are U, V, Y, and Z, respectively. A C/E system is then defined as a mapping Σ which gives the set of admissable output signal trajectories for each possible input signal trajectory: $\Sigma : \mathbf{C}(U, t_0) \times \mathbf{E}(V, t_0) \rightarrow 2^{\mathbf{C}(Y,t_0)\times\mathbf{E}(Z,t_0)}$, such that for any input $(u(\bullet), v(\bullet)) \in \mathbf{C}(U, t_0) \times \mathbf{E}(V, t_0)$ there exists at least one output $(y(\bullet), z(\bullet)) \in \mathbf{C}(Y, t_0) \times \mathbf{E}(Z, t_0)$ fulfilling $(y(\bullet), z(\bullet)) \in \Sigma(u(\bullet), v(\bullet))$.

In the definition above, a C/E system is characterized by its input/output behaviour. There is no specification or restriction on the model of the internal dynamics of a C/E system. In [9], any formal representation which describes an appropriate input/output relation is called a *C/E model.* One example of a C/E model, based on Petri nets, is presented in [10]. Here, we will consider two different C/E models which realize C/E systems: First, we recall discrete state C/E systems which are used to model untimed DES (Def 1). Then, in Sec. 3, we introduce *C/E timers* (Def. 2) to incorporate quantitative timing information into C/E systems.

Definition 1 (*Discrete State C/E System*): A *discrete state C/E system* is a 9-Tupel $S = (U, V, X, Y, Z, f, g, h, x_0)$, where U, V, X, Y and Z are finite, countable mutually disjoint sets representing the *input conditions*, the *input events* (not including the null event 0, we write V_0 for $V \cup \{0\}$), the *states*, the *output conditions* and the *output events* (again not including the null event, $Z_0 = Z \cup \{0\}$), respectively. f, g and h are functions defined as: $f : X \times U \times V_0 \rightarrow 2^X - \emptyset$, the *state transition function* satisfying $\forall x \in X, u \in U : x \in f(x, u, 0)$, and $g : X \times U \rightarrow Y$, the *condition output function*, and $h : X \times X \times V_0 \rightarrow Z_0$, the *event output function* satisfying $\forall x \in X : 0 = h(x, x, 0)$; and $x_0 \in X$ is the *initial state*. Given a C/E system as defined above and input signals $u(\bullet)$ and $v(\bullet)$, the set of valid state trajectories $x(\bullet)$ and output signals $y(\bullet)$ and $z(\bullet)$ is represented by the following three equations.

$$x(t) \quad \in \quad f(x(t^-), u(t^-), v(t)), \tag{1}$$

$$y(t) = g(x(t), u(t)), \quad (2)$$
$$z(t) = h(x(t^-), x(t), v(t)), \quad (3)$$

where $x(t^-)$ is an abbreviation of $\lim_{\Delta \to 0} x(t - \Delta)$ (the same applies to $u(t^-)$). ■

A C/E system can be regarded as an untimed finite state machine which is embedded in a time-dependent signals space formed by the condition and event input and output signals. The condition input signal constitutes conditions for changes of the state of the system (hence can *disable* or *enable* certain transitions) whereas the event input signal can *force* transitions. Event outputs can be generated if and only if the state of the system changes, condition outputs provide information about the actual state of the system. Transitions can be forced (by event signals) or occur spontaneously, and can be nondeterministic. The two properties $\forall x \in X, u \in U : x \in f(x, u, 0)$ and $\forall x \in X : 0 = h(x, x, 0)$ stated in Def. 1 guarantee that transitions and output events cannot be forced by condition input changes.

To provide a flavour of how technological systems are modeled by means of C/E systems we present a small example. Figure 2 shows a tank which can be filled via an inlet valve and illustrates the corresponding C/E model for the tank level and its influences by its block diagram representation. Condition signal flow lines are characterized by two black rectangles whereas event signal flow lines can be recognized by a lightning symbol. The model is divided into three subsystems, "*valve*", "*feed pressure*" and "*level*", which are connected by condition signals and event signals. The dynamic behaviour of each system is represented as a state graph. For a detailed description of the behaviours and interactions between these blocks, the possible successor states and output signal values have to be specified for all combinations of states and input signal values for all three systems. This information is represented for each system by the three functions f, g and h which were introduced in Def. 1. For this example, we will not discuss the complete model but only some aspects of it in order to illustrate the important concepts of forcing and enabling/disabling.

Consider the situation when the tank is empty, the valve is closed and the pressure is sufficient. A transition from "*open*" to "*closed*" in the system "*valve*" will then be indicated to the system "*level*" by the event signal value "*valve opens*" and it will force the transition from "*empty*" to "*medium & rising*". This is possible because the feed pressure is sufficient which is indicated to the system "*level*" by the condition signal value "*pressure is up*". So, the transition is enabled by the condition input signal. If the system "*feed pressure*" would be in state "*not sufficient*", the condition input signal would have the value "*pressure is down*" and therefore disable

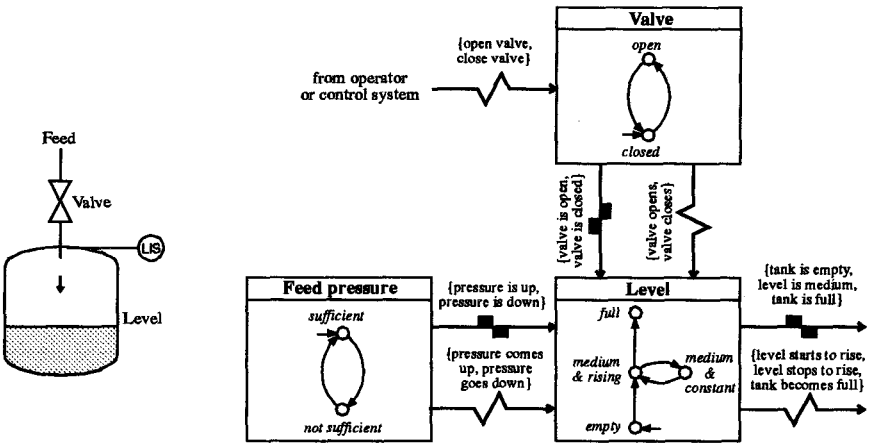

FIGURE 2. Example of a C/E system model.

the transition from "*empty*" to "*medium & rising*".

An example of a spontaneous event is given by the transition from "*medium & rising*" to "*full*". It has to be enabled by the condition input "*valve is open*" and "*pressure is sufficient*", but the actual occurence is spontaneous and cannot be forced by any external input.

The relevant part of the state transition function of the system "*level*" for the behavioural aspects described above is the following (note that $u(t^-)$ and $v(t)$ are pairs of signals because they are determined by the cross product of all input signals coming from different systems):

f(*empty, (pressure is up, valve is closed*), (0,*valve opens*))
= {*medium & rising*},

f(*empty, (pressure is down, valve is closed*), (0,*valve opens*))
= {*empty*},

f(*medium & rising, (pressure is up, valve is open*), (0, 0))
= {*medium & rising, full*}.

3 Timed C/E Systems

3.1 C/E Timers

The C/E system paradigm as described above constitutes an untimed discrete event system model. In technological systems however, transitions usually require a certain amount of time or have to occur within certain time intervals. Consequentially, the qualitative behaviour is strongly influ-

enced by timing constraints. Effective modeling paradigms therefore must allow one to include timing constraints in a natural manner. Since the communication between C/E systems is already defined in terms of continuous time signals, the C/E system paradigm is very well suited to incorporate time without giving up the advantages mentioned above. For this purpose, a special class of C/E models called *C/E timers* are introduced (see Def. 2). The C/E timers are coupled to the untimed logical system part to represent the overall timed behaviour of the system. Thus the timed model remains completely within the original conceptual framework. The advantage is obvious: Describing the timing is part of the modular concept. The untimed dynamics and the timing conditions are seperated and can be modeled independendly. Timing information can be added to an already existing model without changing any block of the C/E block diagram by simply adding the necessary timer blocks and connecting them to the appropriate discrete blocks. We will call such an configuration *timed C/E system.*

Definition 2 (*C/E timer*): Given an initial time $t_0 \in \Re$ and a *threshold time* $T_\theta \in \Re^+ - \{0\}$, a *C/E timer* θ on $[t_0, \infty)$ is a mapping $\theta : \mathbf{E}(V_\theta, t_0) \rightarrow \mathbf{C}(Y_\theta, t_0) \times \mathbf{E}(Z_\theta, t_0)$, with $V_\theta = \{\text{“}t_\theta := 0\text{”}\}, Y_\theta = \{\text{“}t_\theta < T_\theta\text{”}, \text{“}t_\theta > T_\theta\text{”}\}$, and $Z_\theta = \{\text{“}t_\theta = T_\theta\text{”}\}$, such that for any event input signal $v_\theta(t) \in E(V_\theta, t_\theta)$, the output $(y_\theta(t), z_\theta(t))$ is determined by

$$y_\theta(t) = \begin{cases} \text{“}t_\theta < T_\theta\text{”} & if \quad \tau(t) < T_\theta, \\ \text{“}t_\theta > T_\theta\text{”} & if \quad \tau(t) > T_\theta, \end{cases} \tag{4}$$

and

$$z_\theta(t) = \begin{cases} \text{“}t_\theta = T_\theta\text{”} & if \quad \tau(t) = T_\theta, \\ 0 & else \end{cases} \tag{5}$$

in which $\tau : [t_0, \infty) \rightarrow [t_0, \infty)$ is the *clock function* given by

$$\tau(t_0) = T_\theta + \epsilon, \epsilon > 0, \tag{6}$$

and for all $t \in [t_0, \infty]$:

$$\dot{\tau}(t) = \begin{cases} 1 & if \quad v_\theta(t) = 0, \\ undefined & else \end{cases} \tag{7}$$

and $\tau(t) = 0$ if $v_\theta(t) \neq 0$. ■

A C/E timer can be regarded as an alarm clock which is reset and started by the input event “$t_\theta := 0$” and which indicates that it has reached its threshold time T_θ by sending out an event “$t_\theta = T_\theta$”. The condition outputs “$t_\theta < T_\theta$” and “$t_\theta > T_\theta$” indicate that the threshold has not yet been reached or has been crossed, respectively.

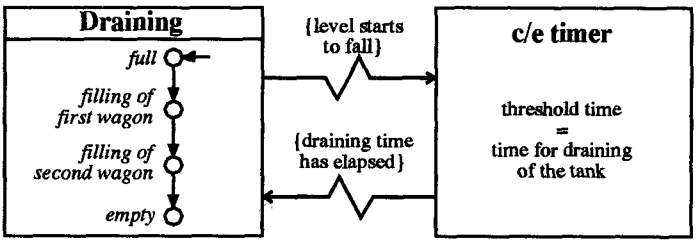

FIGURE 3. Adding time to a C/E system.

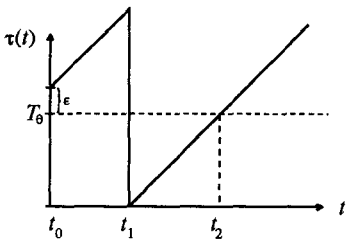

FIGURE 4. Clock function $\tau(t)$.

Figure 3 illustrates this concept for a small example. The left block represents an untimed C/E model for the emptying of a tank filled with liquid, say the tank from the example in Fig. 2. The tank content is drained into two tank wagons which are placed under the outlet valve of the tank one after another, which leads to the four states shown in the state graph. In this model, all transitions are spontaneous. Usually, the time needed to empty the full tank completely can easily be determined. So this information should be incorporated into the model. In Fig. 3, this is done by simply adding a C/E timer to the discrete "*draining*" block which is started by the event "*level starts to fall*" and generates the event "*draining time has elapsed*" after the appropriate time. The transition from "*filling of second wagon*" to "*empty*" is now no longer spontaneous but forced by the event "*draining time has elapsed*".

Figure 4 shows a sample trajectory of the clock function $\tau(t)$. It is easy to see that T_θ is the only free parameter for the C/E timer behavior, while it is not affected by the value of the initial offset ε. The internal behaviour of a C/E timer is comparable to an integrator with constant input which is reset by an event input (here by "*level starts to fall*" at time t_1) and sends out an event when a threshold is reached (here "*draining time has elapsed*" at time t_2).

3.2 Reachability Analysis

To be useful in the context of controller verification, timed C/E systems must be eligible for reachability analysis. In the following, we propose an algorithm to solve this problem. Since spontaneous transitions are still allowed and time is continuous, the problem arises that the number of possible timed state trajectories becomes infinitely large. However, it is still possible to represent the available information of the past of the system in finite expressions and to determine an upper bound of the possible timed system behaviors. For this purpose, the set of (uncountable many) possible current clock function values of all C/E timers under consideration is represented by a so-called *distance matrix*. A distance matrix M is an n-dimensional matrix which is assigned to an n-tupel $I_M = (\theta_1, \theta_2, \ldots, \theta_{n-1}, now)$ in which each θ_i represents a C/E timer and *now* stands for the current point of time for which M describes the timing status of the system. For the reachability algorithm, *now* will be the time when the last state of the currently considered path has been reached. The matrix entries of M have the following meaning:

- For $i < j$, m_{θ_i,θ_j} is interpreted as an upper bound for the distance between the last reset of the i-th and the j-th timer (resp. *now*) in I_M. If there is no such upper bound m_{θ_i,θ_j} is '∞'. Otherwise, the entries are natural numbers and represent time units.
- For $i > j$, m_{θ_i,θ_j} is interpreted as a lower bound for the distance between the last reset of the i-th and the j-th timer (resp. *now*) in I_M. These entries are always natural numbers giving a lower bound in time units.
- The entries on the main diagnal are not used and always set to zero.

The reachability analysis is realized by a depth first search algorithm that is given in pseudo code in Figure 5. The algorithm starts in the initial state with no timing information (i.e. the empty distance matrix) and moves along any admissible path of the state graph of a C/E system. In every step of the recursion it determines the possible successor states from logical and time dependent desription of the system, i.e. the state transition function f and the current distance matrix. Depending on the result, it updates the distance matrix and moves to an admissible successor state. The recursion terminates when the current state is visited with the same distance matrix a second time or when it has no more admissible successor states.

For the termination of the algorithm, it is crucial to determine whether a discrete state has been visited already with the same timing information. This is possible because distance matrices provide a canonical representation of this information. A distance matrix M of dimension n is said to be

```
PROCEDURE Main
{
  READ input_system;
  Goto_State(initial_state , empty_distance_matrix);
  WRITE set_of_reachable_states;
}
PROCEDURE Goto_State(state , distance_matrix)
{
  ADD state TO set_of_reachable_states;
  {
    IF distance_matrix NOT_IN set_of_distance_matrices(state)
    {
       ADD distance_matrix TO set_of_distance_matrices(state);
       FIND list_of_admissible_succ_states;
       FOR_ALL succ_states IN list_of_admissible_succ_states;
       {
         FIND succ_distance_matrix;
         Goto_State(succ_state , succ_distance_matrix);
       }
    }
  }
}
```

FIGURE 5. Algorithm in pseudo code.

in *normal form* if two constraints hold for M:

$$\forall i \in \{1, 2, \ldots, n-1\} \quad : \quad m_{\theta_i, now} \leq T_{\theta_i} \vee m_{\theta_i, now} = \text{“}\infty\text{”} \tag{8}$$

$$\forall i \in \{1, 2, \ldots, n-1\} \quad : \quad m_{now, \theta_i} \leq T_{\theta_i} \tag{9}$$

The first constraint postulates that the upper bound $m_{\theta_i, now}$ for the distance between the last reset time of timer θ_i and the current time *now* is always less than or equal the threshold of θ_i or otherwise it is the symbol '∞'. To normalize the distance matrix if this constraint is broken, the symbol '∞' is entered for $m_{\theta_i, now}$. This means that in a normalized distance matrix the irrelevant information for how long a clock might be over its threshold is generalized to the information that this clock simply might be over its threshold.

The second constraint postulates that the lower bound m_{now, θ_i} for the distance between the last reset of a timer θ_i and *now* is always less than or equal the threshold of θ_i. If this constraint is broken, the distance matrix is normalized by deleting the symbol θ_i from I_M and all matrix entries related to clock θ_i from M. Consequentially, a distance matrix in normal form contains no symbols of timers which are definitely over their threshold. Two normalized distance matrices M_1 and M_2 are said to be equal if all entries in the matrices and the tupel I_{M_1} and I_{M_2} are equal. If normalized distance matrices are used to determine whether a state has been visited with the same timing information before, the algorithm in Fig. 5 will terminate

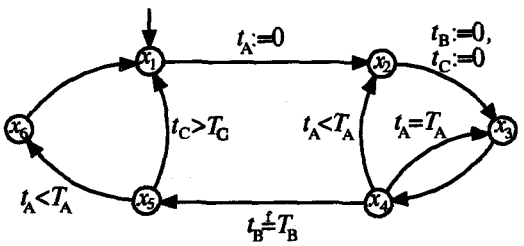

FIGURE 6. Example of a state graph of a C/E system with timers.

because the number of normalized distance matrices for a given timed C/E system is finite.

3.3 Example

We illustrate the basic idea of the reachability algorithm based on distance matrices with the help of a small example. Fig. 6 shows the state graph of a C/E system with three C/E timers, A, B, and C. The threshold times are $T_A = 3$ time units, $T_B = 5$, and $T_C = 6$. The state transitions are labeled according to their temporal conditions. For example, $t_A < T_A$ indicates that this transition may only occur when the clock function value of timer T_A is below its threshold. A transition forced by a timer output is labeled by $t_\theta = T_\theta$. If this case does not include the possibility to stay in the current state, a small "f" is written above the "=". If no condition is assigned to an arc, the transition is not depending on any timer.

Consider the case that the reachability algorithm has followed the path x_1, x_2, x_3, x_4, x_5. The available information about the temporal past at state x_5 is the following: First, timer A was started. Then the system remained in x_2 for an unknown period of time, because the transition to x_3 is purely spontaneous. With this transition, timer B and C were started. The residence times in x_3 and x_4 again are unknown. The transition from x_4 to x_5 is forced by timer B. This means, we know that timer B has just reached its threshold when the system moves to x_5 and that the distance between the instant when the transition from x_2 to x_3 took place and the instant when the transition from x_4 to x_5 occured is exactly 5 time units. So, the last reset of timer A at the transition from x_1 to x_2 is at least 5 time units ago. The corresponding distance matrix is:

$$M(A, C, now) = \begin{pmatrix} 0 & \infty & \infty \\ 0 & 0 & 0 \\ 5 & 5 & 5 \end{pmatrix}$$

Obviously, $M(A, C, now)$ does not fulfill constraint (7) because $m_{A,now} = 5$

and $T_A = 3$. This means that timer A must have reached its threshold, too. For this reason, $M(A, C, now)$ has to be normalized in the following manner.

$$M(C, now) = \begin{pmatrix} 0 & 5 \\ 5 & 0 \end{pmatrix}$$

Based on the information represented by $M(C, now)$, the reachability algorithm will now discover that when the system has moved along the path x_1, x_2, x_3, x_4, x_5, it will not go to x_6 in the next transition because the condition "$t_A < T_A$" is not fulfilled. In fact, x_6 is not reachable.

4 Application

We illustrate the application of the described analysis method to the verification of discrete controllers with the help of a process control example. Consider the flowchart in Fig. 7. It shows a part of a laboratory batch plant at the Chemical Engineering Dept. in Dortmund. In this part of the plant, the following production sequence takes place: Salt solution is filled into tank T1 and then evaporated until a desired concentration is reached. During evaporation, the condenser C1 is in operation and captures the steam coming from T1. When the desired concentration is reached, the material is drained from T1 into T2 as soon as T2 becomes empty. A postprocessing step takes place in T2, before the material can be pumped out of T2 to a subsequent part of the plant. We suppose that in the undisturbed case the operation sequence described above is ensured by the controller and direct our interest to the problem of appropriate control reaction on disturbances. In particular, we look at the consequences of a cooling breakdown in the condenser. This failure can lead to a dangerously high pressure in the condenser tube, if the evaporating is continued. When the heating in T1 is switched off, the pressure in C1 will not rise any more. To make the problem more demanding, we assume that the material in T1 gets solid after a certain time when it cools down and cannot be drained into T2. As a consequence, it is possible that the controller should not switch off the heating immediately after cooling breakdown, because this may lead to solid material in T1. If T2 is still full, it has to wait some time to ensure that T1 can be drained before the material gets solid. However, a meaningful controller will start draining T2 as soon as a cooling breakdown occures, so that no time is lost. The described behavior and more details are represented by the timed C/E system shown in Fig. 8.

The names of the states are based on the following convention: The first character describes the state of T1: e = empty, p = processing (evaporat-

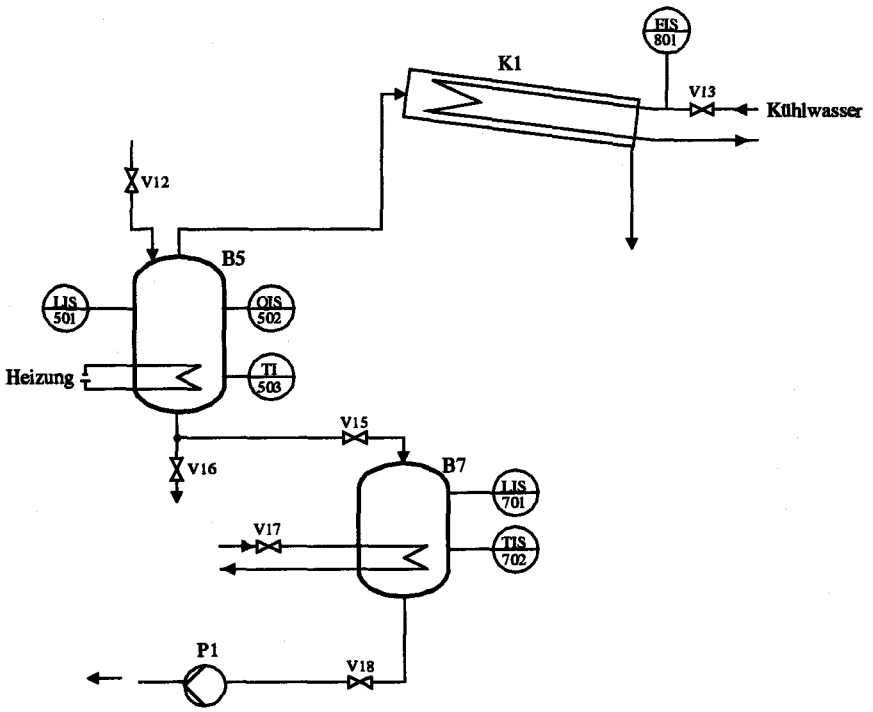

FIGURE 7. Flowchart of the example process.

ing), r = ready (and waiting for T2), d = being drained, i = interrupt (heating switched off before desired concentration was reached), and s = solid; the second character represents T2: e = empty, p = (post-)processing, f = being filled, d = being drained; and the third character symbolizes the state of the condenser: n= not in operation, c = cooling, b = breakdown, and a = pressure alarm (too high). For all devices, an x stands for undetermined. Table 1 and 2 explains the states and timers in more detail.

The verification problem now is to analyze whether a given controller with a waiting time $T_W = 5$ (which means that it waits 5 minutes after cooling breakdown before it will switch off the heating) will prevent the system from reaching the forbidden states **pxa** and **sxx**. Applying reachability analysis shows that for such a controller, the state **sxx** is reachable for the system while it prevents the system from reaching state **pxa**. So, in case of a cooling breakdown, the material in T1 will get solid. Two recursion steps of the algorithm will now be presented in detail.

After 5 steps the algorithm may reaches the state **pdp** if it is currently following a path consisting of the sequence of states: **een, pec, dfn, epn, ppc, pdb**. Starting with the empty distance matrix $M_0(now)$ in the initial

States	
epn	T1 empty, T2 processing, C1 not in operation.
edn	T1 empty, T2 is being drained, C1 not in operation.
een	T1 empty, T2 empty, C1 not in operation.
ppc	T1 evaporating, T2 processing, C1 cooling.
pdc	T1 evaporating, T2 is being drained, C1 cooling.
pec	T1 evaporating, T2 empty, C1 cooling.
pxa	Alarm: pressure in C1 too high.
pdb	T1 evaporating, T2 is being drained, cooling breakdown in C1.
peb	T1 evaporating, T2 empty, C1: cooling breakdown.
rpn	T1 ready and waiting for T2, T2 processing, C1 not in operation.
rdn	T1 ready and waiting for T2, T2 is being drained, C1 not in operation.
dfn	Draining from T1 into T2.
sxx	Material in T1 has become solid.
idb	Interrupt in T1 (heating switched off), T2 is being drained, cooling breakdown.
dfb	Emergency draining of T1.

TABLE 1. States from Fig. 8

Timers		
A	Time between cooling breakdown and pressure alarm if heating is continued.	$T_A = 8$ min
D	Time for draining of T2.	$T_D = 10$ min
E	Minimal time for evaporation in T1.	$T_E = 60$ min
P	Time for processing the material in T2.	$T_P = 40$ min
S	Time between heating switched off and material becoming solid.	$T_S = 4$ min
W	Time which the controller waits after cooling breakdown before it switches off the heating.	$T_W = 5$ min

TABLE 2. Timers from Fig. 8

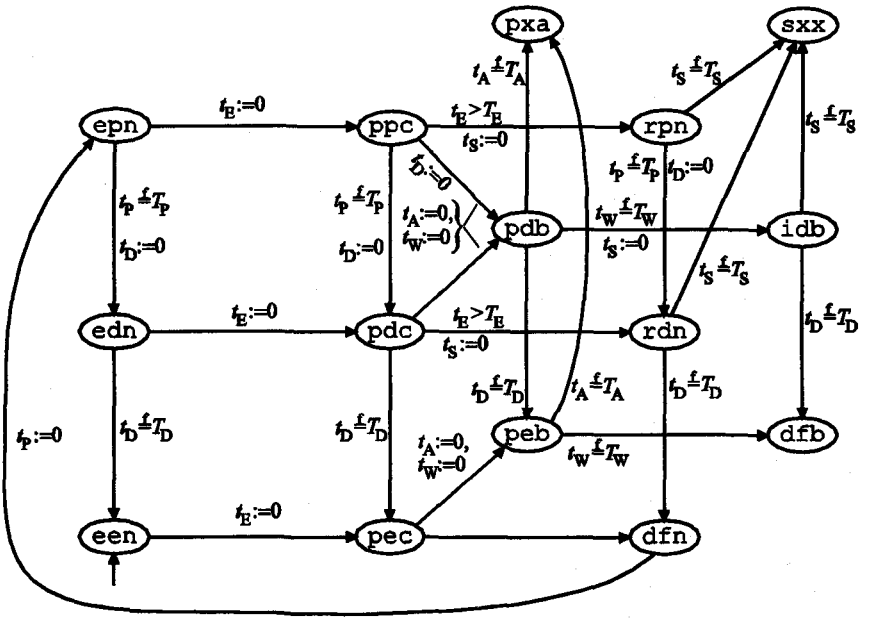

FIGURE 8. State graph for the example.

state **een**, the algorithm then is at state **pdb** with the distance matrix:

$$M_5(P,E,D,A,W,now) = \begin{pmatrix} 0 & 40 & 40 & 40 & 40 & 40 \\ 0 & 0 & 40 & 40 & 40 & 40 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

With this distance matrix the algorithm can only choose the transition to **idb** in its next step, because clock W will reach its threshold before the clocks A and D. Taking the transition from **pdb** to **idb** as the next transition in its 6th step the algorithm calculates the following distance matrix:

$$M_6(P,E,D,A,S,now) = \begin{pmatrix} 0 & 40 & 40 & 40 & 45 & 45 \\ 0 & 0 & 40 & 40 & 45 & 45 \\ 0 & 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 0 & 5 & 5 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \end{pmatrix}$$

After normalisation:

$$M_6(P,E,D,A,S,now) = \begin{pmatrix} 0 & 40 & 40 & 40 & 45 & \infty \\ 0 & 0 & 40 & 40 & 45 & 45 \\ 0 & 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 0 & 5 & 5 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \end{pmatrix}$$

Analysing the distance matrix $M_6(P,E,D,A,S,now)$ reveals that the state **sxx** is reachable, because timer S will reach its threshold before timer D does. So, the given controller is incorrect and in case of a cooling breakdown, the material in T1 will get solid.

5 Discussion

We have reported on the use of C/E systems for a model-based verification of discrete controllers for chemical plants. To capture critical timing constraints, the original framework was extended by C/E timers which make it possible to model quantitative time in C/E systems without changing the conceptual framework. A reachabilty algorithm has been sketched. Finally, we described the application of the approach to a laboratory batch plant and parts of its safety control.

The timed C/E system model has many similarities to the timed automata approach by Alur and Dill [1]. In fact, the creation and updating of the distance matrix during the reachability algorithm can be regarded as a method to build the region graph used by Alur and Dill to analyze reachability. We are currently investigating further relations between the two approaches. The reason why timed automata were not applied to model the laboratory plant is twofold. First, they do not provide special support for modular modeling in form of intuitive interaction concepts as C/E systems do. Second, discrete state C/E models were already available for the laboratory batch plant ([4], [3]) and it was necessary to add the timing information without changing the basic framework.

There are several directions of further research. In the described model, transitions can depend at most on the standings of a single timer. We are currently extending the framework to be able to use Boolean connectives on timer queries and compare different timer standings. Since the described analysis method is based on a comprehensive search, we are also interested in symbolic and compositional analysis.

Acknowledgements. Prof. Heiko Krumm and Prof. Sebastian Engell contributed to the presented work by helpful and instructive discussions. We are also grateful to the anonymous referees who helped to improve the presentation.

6 REFERENCES

[1] R. Alur, D. Dill: *The Theory of Timed Automata.* Theoretical Computer Science, 126, 1994: 183-235.

[2] V. D. Dimitriadis, N. Shah und C. C. Pantelides: *Modeling and Safety Verification of Discrete/Continuous Processing Systems Using Discrete Time Domain Models.* Proceedings Workshop Analysis and Design of Event-Driven Operations in Process Systems, Imperial College, London, 10.-11. April 1995.

[3] S. Kowalewski, S. Engell, M. Fritz, R. Gesthuisen, G. Regner, M. Stobbe: *Modular discrete modeling of batch processes by means of condition/event systems.* Workshop Analysis and Design of Event-Driven Operations in Process Systems, Imperial College, London, 10.-11. April 1995.

[4] S. Kowalewski, R. Gesthuisen and V. Romann: *Model-based verification of batch process control software.* Proc. IEEE Conf. on Systems, Man, and Cybernetics, San Antonio, 331-336, 1994.

[5] B. H. Krogh: *Condition/Event Signal Interfaces for Block Diagram Modeling and Analysis of Hybrid Systems.* Proc. 8th Int. Symp. on Intell. Cont. Sys., June 1993.

[6] I. Moon, G. J. Powers, J. R. Burch und E. M. Clarke: *Automatic Verification of Sequential Control Systems Using Temporal Logic.* AIChE Journal 38, 1992.

[7] G. J. Powers und S. T. Probst: *Safety and Operability Analysis of Chemical Process Designs Using Symbolic Model Verification.* AIChE Annual Meeting, San Francisco, USA, 1994.

[8] P. J. Ramadge and W. M. Wonham: *Supervisory control of a class of discrete event processes.* SIAM J. Control Optim. 25(1987): 206-230.

[9] R.S. Sreenivas and B.H. Krogh: *On Condition/Event Systems with Discrete State Realizations.* Discrete Event Dynamic Systems 1(1991): 209-236.

[10] R.S. Sreenivas and B.H. Krogh: *Petri Net Based Models for Condition/Event Systems.* Proc. American Control Conf. 1991, Boston, USA.