Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Hybrid Control Techniques for the Design of Industrial Controllers

Sebastian Engell, Olaf Stursberg
Process Control Laboratory (BCI-AST)
University of Dortmund, 44221 Dortmund, Germany.
Email: {s.engell | o.stursberg}@bci.uni-dortmund.de

Abstract— **This tutorial paper provides an overview of where techniques based on hybrid dynamic models are suitable or promising for designing controllers of industrial plants, in particular chemical processing systems. After summarizing the typical control tasks prevalent in the hierarchical automation structure of industrial plants, the paper focusses on two techniques employing hybrid models that recently have gained much attention by the research community: the algorithmic verification of safety-related discrete controls, and the optimal control of large transitions, like startup, shutdown, or product switch-over.**

Index Terms— **Automation, Hybrid Dynamics, Optimal Control, Safety, Supervisory Control, Verification.**

I. INTRODUCTION

While continuous or quasi-continuous sampled data control has been the main topic of control education and research for decades, in industrial practice discrete-event or logic control is at least as important for the correct and efficient functioning of production processes than continuous control. A badly chosen or ill-tuned continuous controller only leads to a degradation of performance and quality as long as the loop remains stable, but a wrong discrete input (e.g. switching on a motor that drives a mass against a hard constraint or opening a valve at the wrong time) will most likely cause severe damage to the production equipment or even to the people on the shop floor, and to the environment. In addition, discrete and logic functions constitute the dominant part of the control software and are responsible for most of the effort spent on the engineering of control systems of industrial processes.

Generally, several layers of industrial control systems can be distinguished (see Fig. 1). The first and lowest layer of the hierarchy realizes safety and protection related discrete controls. This layer is responsible for the prevention of damage to the production site including the personnel. For example, a robot is shut down if someone enters its workspace, or the fuel flow to a burner is switched off if no flame is detected within a short period after its start. Most of the safety-related control logic is consciously kept simple in order to enable inspection and testing of the correct function of the interlocks and safety-trips. This has the drawback that a part of the plant may be shut down if one or two of the sensors associated with an interlock system indicate a potentially critical situation while a consideration of the information provided by a larger set of sensors would have led to the conclusion that there was in fact no critical situation. As shutdowns cause significant losses of production, there is a tendency to install more sophisticated interlock systems which can no longer be verified by simply looking at the code or performing simple tests. In the sequel, we do not distinguish between strictly safety-related and emergency-shutdown systems (which have to be presented to and checked by the authorities outside the plant), and more general protection systems which prevent damage or degradation of the equipment or unwanted situations causing large additional costs or the loss of valuable products – from a design and verification point of view, there is no difference between the two. Clearly, the correct function of safety and protection related controls depends on the interaction of the discrete controller with the continuous and possibly complex plant dynamics.

The second layer of the control system is constituted by continuous regulation loops, e.g. for temperatures, pressures, and the speeds of drives. These loops receive their set-points or trajectories from the third layer which is responsible for the sequence of operations required to process a part or a batch of material. On this layer, mostly discrete switchings between different modes of operation are controlled, but also continuous variables may be computed and passed to the lower-level continuous control loops. If these sequences are performed repeatedly in the same manner, they are usually realized by computer control. If there are a large variations of the sequence of operations or of the way in which the steps are performed, as in some chemical or biochemical batch processes, sequence control is mostly performed by the operators. The same is true for the start-up of production processes or for large transitions between operating regimes, which usually do not occur too often. On a fourth layer of the control hierarchy, the various production units are coordinated and scheduled to optimize the material flow.

A major part of the control code (or of the task of the operators) on the sequential control layer is the handling of exceptions from the expected evolution of the production process: drills break, parts are not grasped correctly, controlled or supervised variables do not converge to their set-points, valves do not open or close, etc. While there usually is only one correct sequence, a possibly different recovery sequence must be implemented for each possible fault. Exception handling in fact also is responsible for a large fraction of the code in continuous controllers.

Safety and protection related discrete controls and sequen-

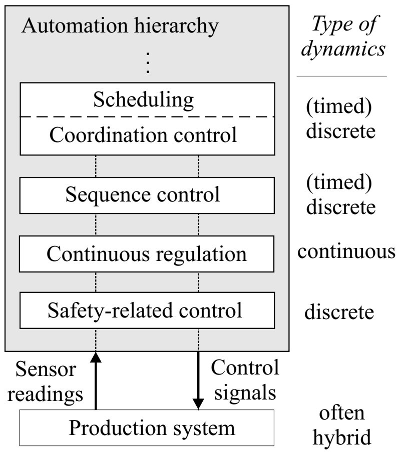

Fig. 1. Typical control hierarchy of production systems.

tial discrete or mixed continuous-discrete controls are of key importance for the safe and profitable operation of present-day production processes. Their correctness and efficiency cannot be assessed by testing the logic independently as they are determined by their interaction with the (mostly) continuous dynamics of the physical system. This calls for systematic, model-based design and verification procedures that take the hybrid nature of the problem into account. In practice, however, discrete control logic is usually developed at best in a semi-formal manner. Starting from partial and partly vague specifications, code is developed, modified after discussions with the plant experts, simulated using a very crude plant model or with the programmer acting as the plant model, and then tested, debugged and modified during start-up of the plant. The main reason that this approach does not lead to complete failure is that for the most part logic control software from other projects is re-used and only small modifications and extensions are added. However, taking into account the low-level programming languages used and the lack of formal documentation, such software systems may become harder and harder to maintain.

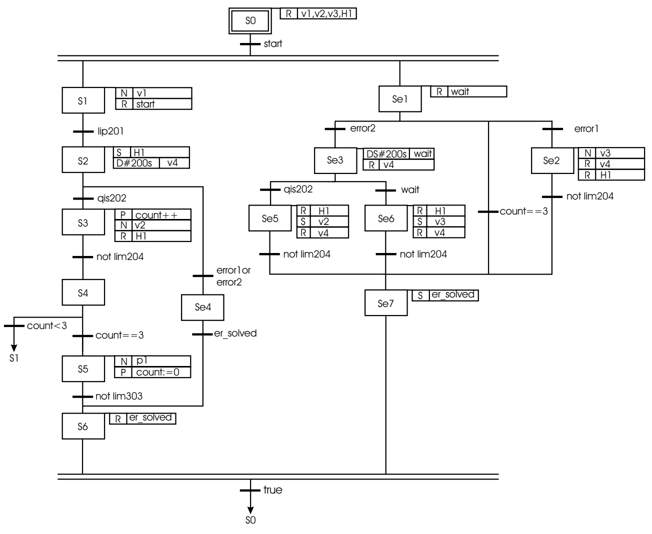

Fig. 2. Supervisory controller as SFC.

In the remainder of this paper, we try to highlight the potential of the application of hybrid systems and control techniques in the area of industrial controls. We focus on the two layers on which the hybrid nature of the controlled plant is most relevant, safety and protection related controls, and sequence control. In the latter area, we describe some recent work on one of the most interesting problems, the control of large transitions in processing plants. This topic is most challenging because it requires taking continuous dynamics of considerable complexity into account as well as a large number of discrete and continuous variables over long horizons, rendering brute-force approaches not very promising.

II. VERIFICATION OF SAFETY-RELATED LOGIC CONTROLLERS

In order to be accepted by practitioners, verification procedures for safety and protection related industrial controllers must be able to handle the control logic as it is implemented on the control hardware, usually a programmable logic controller (PLC) or a distributed control system (DCS). For the implementation of logic controls, the standard IEC-61131-3 [1] defines several standard formats. Among these, sequential function charts (SFC) are best suited to represent sequential behaviors and the parallel (simultaneous) or alternative execution of program steps, and to structure logic control programs. Control code written in other IEC-61131-3 languages (Ladder Diagrams, Instruction List, Structured Text, or Function Block Diagrams) can be embedded in SFC. According to [1], SFC consist of alternating sequences of steps and transitions, where actions are associated with steps and conditions with transitions. As an example, Fig. 2 shows the graphical representation of SFC, in which rectangles denote the steps (with actions blocks attached to the right), bold horizontal lines the transitions (including conditions), and vertical lines the flow of execution (from top to bottom). Action blocks contain a list of actions which are either simple manipulations of logical variables (most importantly the outputs to the plant), or activities that are limited to a specified period of time (or start after a given delay), or the activation of other SFC. The transition conditions may involve Boolean expressions of sensor readings and internal program variables.

The goal of the verification of this type of logic controllers is to guarantee that the controller prevents the plant from reaching unwanted or dangerous states and/or ultimately steers it to the desired terminal state. Therefore, the plant dynamics must be described formally by an (untimed, timed or hybrid) automaton model, and a formal specification must be provided in a temporal logic framework (see e.g. [2]). Before model checking can be applied, the control logic (e.g. an SFC) must be represented as a state transition system. For logic control programs that contain timers or delayed actions, timed automata (TA) are the most suitable format. After composition of the plant model and the controller model, the

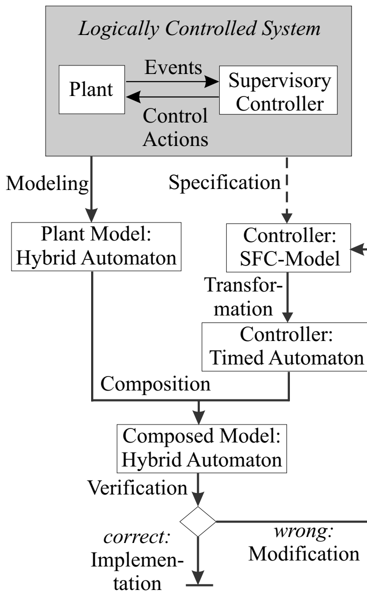

Fig. 3. Control design scheme.

overall model can be checked against the formal specification using one of the available tools, e.g. SMV for purely discrete models, UPPAAL for timed automata models, or the tools sketched in [3] and [4] for hybrid models. The scheme of the overall procedure is shown in Fig. 3. In the sequel, we discuss the steps of the procedure in more detail for a specific approach that implements this general idea.

A. Transformation of SFC into TA

As proposed in [5], the transformation of a controller given as SFC into a set of timed automata can be accomplished by a procedure that first uses a graph grammar to partition the SFC into syntactical units. Such a unit is either a sequence of steps and transitions including alternative branches or a block representing parallel branches of the SFC. By scanning the SFC controller in a top-down manner, a structure of these two types of units is obtained such that a modular timed automaton model can be generated in a straightforward manner: each of the units is mapped into a single timed automaton, and the activation of the automata according to the execution of the SFC is established by synchronization labels. The state-transition structure of the automata follows directly from the step-transition sequences of the SFC. The transition conditions, which involve either inputs from the plant or internal variables of the SFC, are expressed by synchronization labels as well. Finally, the actions associated with the steps are modelled by separate automata, which can include clocks for the case of time-dependent action qualifiers. For modeling the actions, the procedure proposed in [5] uses a scheme that explicitly accounts for the cyclic scanning mode in which SFCs are executed on programmable logic controllers.

B. Model Composition and Verification

In order to consider the plant behavior, the part of the plant which is affected by the safety-related controller should be identified, and the behavior of this part is represented by a suitable model. If the verification aims at analyzing that the controller drives the plant into particular sets of continuous states (or just prevents the plant from reaching them) a hybrid dynamic model, like hybrid automata [6], is an appropriate choice. The communication between the controller and the plant model can be realized by synchronization of transitions, or by shared variables between both models. If the verification is carried out by the approach of abstraction-based and counterexample-guided model checking [4], the modular model is next transformed into a single composed hybrid automaton. The principle of abstraction-based and counterexample-guided model checking method for verifying safety properties can be summarized as follows: An initial abstract model, given as a finite automaton, follows from abstracting away the continuous dynamics of the composed hybrid automaton. Applying model checking to the abstract model identifies behaviors (the *counterexamples*) for which safety property is violated. In a validation step, it is analyzed whether for these particular behaviors counterexamples exist also for the hybrid automaton. If this applies, the procedure terminates with the result that the hybrid automaton does not fulfil the safety requirement. If none of the counterexamples for the abstract model can be validated for the hybrid automaton, the safety of the latter is proved. The validation step involves the evaluation of the continuous dynamics of the hybrid automaton, i.e. sets of reachable hybrid states are determined for locations encountered along the potential counterexample. Each time a counterexample of the abstract model is invalidated, the information about enabled or disabled transitions (according to the reachable hybrid states in the respective locations) is used to refine the abstract model.

If the verification reveals that the composed hybrid automaton satisfies all relevant requirements, the original SFC-model of the controller represents an implementable supervisory controller. Otherwise, the counterexample corresponding to the requirement violation must be examined in order to identify in which respect the SFC controller has to be modified.

C. Application to an Evaporation System

In order to illustrate the verification procedure, it is applied to the case study of a batch evaporation system [7], [8]. As shown in Fig. 4, the system consists of two tanks (T1, T2) with heating devices, a condenser with cooling (C1), connecting pipes with valves (V1, V2, V3) and a pump (P1), as well as different sensors for liquid levels (LIS), temperatures (TI), and concentration (QIS). The intended operation is to evaporate the liquid from a mixture in T1 until a desired concentration is reached, to collect three batches of the product in T2, and to empty the latter afterwards through P1. Figure 2 shows a possible SFC-controller which not only realizes the desired procedure (left branch) but also includes exception routines (right branch) for the cases of evaporator breakdown (error1) and malfunction of the heating device of T1 (error2).

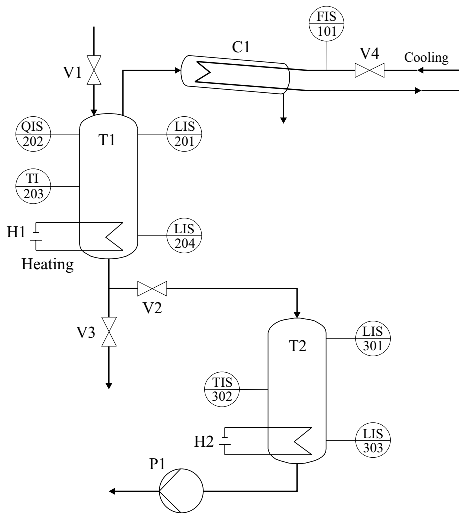

Fig. 4. Flowchart of the evaporation system.

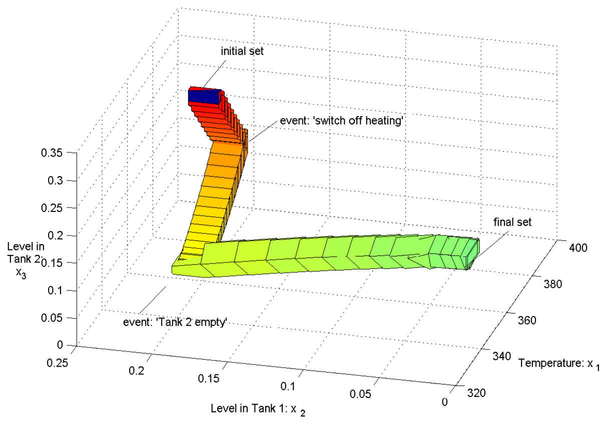

Fig. 6. Reachable continuous set (the final set shows that a critically low temperature ($x_1 = 338K$) is not reached before Tank 1 is empty ($x_2 \leq 0.01m$)) .

Since the SFC-controller contains two time-dependent actions (marked by 'D#200s' and 'DS#200s'), it is transformed into a set of timed automata following the procedure sketched in Sec. II-A. Figure 5 shows the automata that represent the SFC structure. The complete TA model additionally contains automata that model the actions.

One possible verification objective is to check whether the controller avoids safety-critical states, which are a critically high and a critically low temperature of the mixture in T1, for the two failure cases. Assuming that a condenser malfunction occurs while the evaporation in T1 runs and T2 is partly filled, the relevant plant behavior can be restricted to three phases: P1 - heating in T1 while T2 is drained, P2 - draining of T2 without heating in T1, P3 - transferring the content of T1 into T2. The corresponding hybrid automaton contains nonlinear differential equations for the temperature of the liquid in T1, as well as the liquid levels in T1 and T2. The verification procedure described above was applied to the composition of all automata. As the set of reachable continuous states in Fig. 6 shows, a critically low temperature of 338K is not reached before T1 is emptied, i.e., it can be concluded that the SFC-controller works as desired for this configuration. This result was obtained within a computation time of around one minute on a PC with a 1.8 GHz P4-CPU.

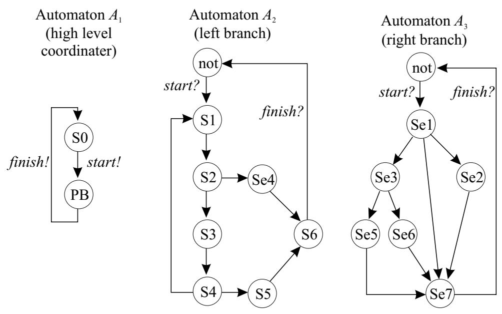
Automaton A_1 (high level coordinater)
Automaton A_2 (left branch)
Automaton A_3 (right branch)
S0
PB
finish!
start!
not
start?
finish?
S1
S2
S3
S4
S5
S6
Se4
Se1
Se2
Se3
Se5
Se6
Se7

Fig. 5. Separate automata to model the SFC structure (inputs / outputs and time conditions are omitted).

III. OPTIMAL STARTUP AND SHUTDOWN OF INDUSTRIAL PLANTS

While most processing systems are operated by a combination of continuous and discrete controls (see Sec. I), both types of controllers are usually designed separately – however, transition procedures like start-up, shutdown, or product change-over, require the simultaneous consideration of both types of controls to avoid opposing effects. This section addresses the task of designing continuous and discrete controls in an integrated fashion. In particular, we consider the aspects of modeling the process dynamics by hybrid automata, formulating the transition procedure as an optimization problem, and computing the (optimal) control inputs efficiently.

Different approaches to the optimization of hybrid systems have been published in recent years, ranging from rather generic formulations to specific methods for certain subtypes of hybrid systems, see e.g. [9], [10], [11], [12]. One branch of methods follows the idea of transforming the hybrid dynamics into a set of algebraic (in-)equalities that serve as constraints for a mixed-integer program [13], [14]. If all constraints are written in linear form, mixed-integer linear (or quadratic) programming can be used for the solution, i.e., standard solvers that employ branch-and-bound strategies, where bounds are obtained from linear relaxations, can be used. In [15], it has been shown exemplarily for the approach in [14] that a drawback of this approach is the limited applicability for larger systems. As an alternative, the following section sketches a method with the following characteristics [16], [17]:

(a) the discrete degrees of freedom are determined by a graph search algorithm with problem specific heuristics to determine the optimal discrete control sequence with low effort,
(b) the continuous degrees of freedom are obtained from solving embedded nonlinear programming problems

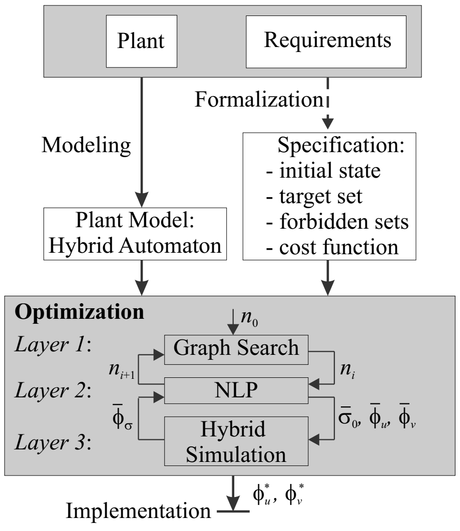

Fig. 7. Scheme for the optimization approach.

(NLP),

(c) the cost function is evaluated by hybrid simulation which takes care of the state-dependent structural changes of the model.

A. Graph Search with Embedded NLP

Figure 7 provides an overview of the method: The starting point are the given plant dynamics and an informal listing of the requirements for the controlled behavior of the plant. The dynamics is represented by a deterministic hybrid automaton as introduced in [16], i.e. characterized by continuous and discrete input variables, autonomous switching between different continuous models, and possible resets associated with transitions. For given trajectories of the continuous (ϕ_u) and discrete (ϕ_v) inputs, the formal definition of the automaton defines feasible state trajectories ϕ_σ as a series of hybrid states $\sigma = (z(t), x(t))$, which consist of a discrete location $z(t)$ and a continuous state $x(t)$.

The requirements of the transition procedure are formalized by specifying the initialization of the hybrid model $\sigma_0 = (z_0, x_0)$, a set of hybrid target states Σ_{tar} (in which the plant has to be driven by the controller), a set of hybrid forbidden states $F = \{F_1, F_2, \ldots\}$ (that must never be encountered), and a cost criterion Ω. The latter specifies a performance measure, such as the startup time or the resource consumption during startup, which has to be minimized. If t_f denotes the final time and $\sigma_f := (z(t_f), x(t_f))$ the final hybrid state, the following optimal control problem is posed:

$$\min_{\phi_u \in \Phi_u, \phi_v \in \Phi_v} \Omega(t_f, \phi_\sigma, \phi_u, \phi_v) \qquad (1)$$

$$\begin{aligned} s.t.\quad & \phi_\sigma = (\sigma_0, \ldots, \sigma_f) \text{ with: } \sigma_f \in \Sigma_{tar}, \text{ and for} \\ & \phi_\sigma \text{ applies in each phase of cont. evolution:} \\ & \sigma \notin F_j \ \forall\ F_j \in F. \end{aligned}$$

The solution of the optimization problem returns the input trajectories ϕ_u^*, ϕ_v^* that lead to a feasible run ϕ_σ^* which minimizes Ω.

The key idea in solving the optimization approach is to separate the optimization of the continuous and of the discrete degrees of freedom in the following sense: The discrete choices (i. e., the input trajectories ϕ_v) are determined by a graph search algorithm resembling the well-known principle of shortest-path search. For each node n_i contained in the search graph, an embedded optimization for the continuous degrees of freedom (and optionally for relaxed discrete degrees of freedom for future steps) is carried out. Within this embedded nonlinear programming, numerical simulation is employed to evaluate the hybrid dynamics of the hybrid automaton, leading to a cost value for the corresponding evolution of the system. These costs are used in the graph search to apply a branch-and-bound strategy, i.e., upper (and lower) bounds on the optimal costs for the transition procedure are iteratively computed to prune branches of the search tree as early as possible.

B. Application to a Chemical Reactor

The method is illustrated for the start-up of a continuous stirred tank reactor (CSTR), as described in [14]. The system consists of a tank equipped with two inlets, a heating coil, a cooling jacket, a stirrer, and one outlet (see Fig. 8). The inlets feed the reactor with two dissolved substances A and B which react exothermically to form a product D. The inlet flows F_1 and F_2 (with temperatures T_1 and T_2) can be switched discretely between two values each. The outlet flow F_3 is controlled continuously. In order to heat up the reaction mixture to a desired temperature range with a high reaction rate, the heating can be switched on (denoted by a discrete variable $s_H \in \{0, 1\}$). The continuously controlled cooling flow F_C serves as a means to remove an excess of heat once the reaction has started. The objective for this system is to determine the input trajectories that drive the initially empty reactor into a desired operation in which the liquid volume V_R, the temperature T_R, and the concentrations c_A and c_B have reached nominal ranges. Additionally, the regions of the state space where $T_R \geq 360$ or $V_R \geq 1.6$ are forbidden.

To model the system, the state vector is defined as $x := (V_R, T_R, c_A, c_B)^{\mathrm{T}}$, the continuous input vector as $u := (F_3, F_C)^{\mathrm{T}}$, and the discrete input vector as $v :=$

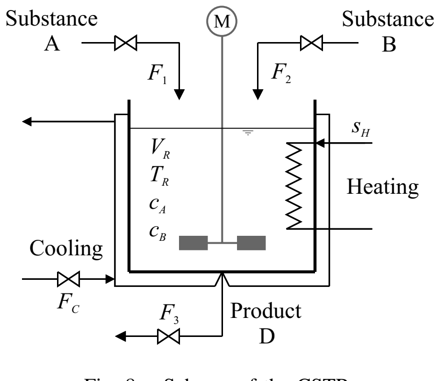

Fig. 8. Scheme of the CSTR.

$(F_1, F_2, s_H)^{\mathrm{T}}$. Depending on the continuous state, the system dynamics can be written as $\dot{x} = f(z, x, u, v)$ where:

- for z_1 with $V_R \in \ [0.1, 0.8]$:

$$f^I = \begin{pmatrix} F_1 + F_2 - F_3 \\ (F_1(T_1 - T_R) + F_2(T_2 - T_R))/V_R \\ \quad + F_C k_1 (T_C - T_R)(k_2/V_R + k_3) - k_4 q \\ (F_1 c_{A,1} - c_A(F_1 + F_2))/V_R + k_9 q \\ (F_2 c_{B,2} - c_B(F_1 + F_2))/V_R + k_{10} q \end{pmatrix}$$

- for z_2 with $V_R \in \ ]0.8, 2.2]$:

$$f^{II} = \left(f_1^I,\ f_2^I + s_H k_6 (T_H - T_R)(k_7 - \frac{k_8}{V_R}),\ f_3^I,\ f_4^I \right)^{\mathrm{T}},$$

and $q = c_A c_B^2 \exp(-k_5/T_R)$. The separation into two V_R-regions (and thus two locations and transitions in both directions between them) accounts for the fact that the heating is only effective above $V_R = 0.8$. The initial state is $x_0 = (0.1, 300, 0, 0)^{\mathrm{T}}$ and the target is given by z_2 and a hyper-ball with radius 0.1 around the continuous state $x_{tar} = (1.5, 345, 0.4, 0.2)^{\mathrm{T}}$. The optimization was run with the cost criterion that the transition time for the startup procedure is minimized. The strategy chosen is that depth-first search is used until a first solution is found, then a breadth-first strategy is applied. Figure 9 shows the state trajectory representing the best solution obtained for a search comprising 400 nodes. This result has been obtained within 2 minutes of computation time on a PC with a 2.0 GHz Pentium processor.

IV. CONCLUSIONS

The tasks of verifying properties like safety or goal attainment for industrial plants and of computing optimal control trajectories for procedures like startup or shutdown are two examples where the design procedure can be suitably supported by the use of hybrid models. At the time being, a number of successful applications of such techniques have been reported in literature – however, most of these applications refer to relatively small parts of industrial plants, or systems on a laboratory scale. The following two aspects seem most important to achieve that industrial control engineers include hybrid control techniques into their toolboxes: (a) the awareness of existing hybrid modeling techniques has to be increased, (b) the efficiency of methods for the analysis, design, and optimization of hybrid systems must be further improved to enhance the applicability to industrial-size problems. These are two main objectives of the Network of Excellence *Hybrid Control* (funded by the European Union), which includes one area of activities that explicitly aims at further developing hybrid control techniques based on case-studies provided by (mainly) the processing industries.

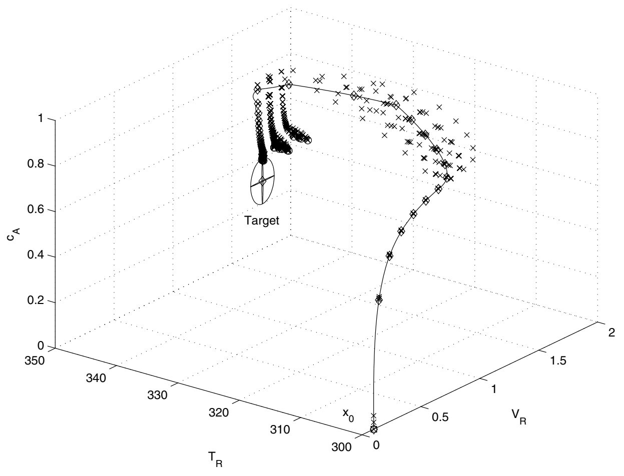

Fig. 9. CSTR: The optimal x-trajectory (solid line) projected in the (V_R, T_R, c_a)-space. Explored nodes are marked by crosses.

REFERENCES

[1] Intern. Electrotechnical Commission (Committee No. 65): "Programmable Controllers – Programming Languages, IEC 61131-3", Ed. 2.0, 2003.

[2] E.M. Clarke, O. Grumberg, D.A. Peled: "Model Checking", MIT Press, 1999.

[3] B.I. Silva, O. Stursberg, B.H. Krogh, S. Engell: "An Assessment of the Current Status of Algorithmic Approaches to the Verification of Hybrid Systems", Proc. IEEE Conf. on Decision and Control, 2001, 2867-2874.

[4] E. Clarke, A. Fehnker, Z. Han, B.H. Krogh, J. Ouaknine, O. Stursberg, M. Theobald: "Abstraction and Counterexample-Guided Refinement in Model Checking of Hybrid Systems", Int. Journal Foundations of Computer Science, Vol. 14(4), 2003, 583-604.

[5] N. Bauer, S. Engell, R. Huuck, S. Lohmann, B. Lukoschus, M.P. Remelhe, O. Stursberg: "Verification of PLC Programs given as Sequential Function Charts", In: Integration of Software Specif. Techn. for Applic. in Eng., Spinger Series LNCS, Vol. 3147, 2004, 517-540.

[6] T.A. Henzinger: "The Theory of Hybrid Automata", Proc. 11^{th} IEEE Symp. on Logic in Comp. Science, 1996, 278-292.

[7] S. Kowalewski, O. Stursberg, N. Bauer: "An Experimental Batch Plant as a Case Example for the Verification of Hybrid Systems", European Journal of Control, Vol. 7, 2001, 366-381.

[8] S. Engell, S. Lohmann, O. Stursberg: "Verification of Embedded Supervisory Controllers Considering Hybrid Dynamics", Int. Journal of Software Eng. and Knowledge Eng., Vol. 15/2, 2005, 307-312.

[9] M.S. Branicky, V.S. Borkar, S.K. Mitter: "A Unified Framework for Hybrid Control: Model and Optimal Control Theory", IEEE Trans. Automatic Control, 1998, 43(1), 31-45.

[10] M. Broucke, M.D. Di Benedetto, S. Di Gennaro, A. Sangiovanni-Vincentelli: "Theory of Optimal Control Using Bisimulations", In: Hybrid Systems: Comp. and Control, Springer, LNCS 1790, 2000, 89-102.

[11] M. Buss, O. von Stryk, R. Bulirsch, G. Schmidt: "Towards Hybrid Optimal Control", Automatisierungstechnik, 9(2000), 448-459.

[12] J. Oldenburg, W. Marquardt, D. Heinz, D. Leineweber: "Mixed Logic Dynamic Optimization Applied to Batch Distillation Process Design", AICHE Journal, 49(2003), 2900-2917.

[13] A. Bemporad, M. Morari: "Control of Systems Integrating Logic, Dynamics, and Constraints", Automatica, 35(3), 407-427.

[14] O. Stursberg, S. Panek, J. Till, S. Engell: "Generation of Optimal Control Policies for Systems with Switched Hybrid Dynamics", In: Modelling, Analysis, and Design of Hybrid Systems, Springer, LNCIS 279, 2002, 337-352.

[15] J. Till, S. Engell, S. Panek, O. Stursberg: "Applied Hybrid System Optimization - An Empirical Investigation of Complexity", Control Engineering Practice, Vol. 12/10, 2004, 1291-1303.

[16] O. Stursberg: "Dynamic Optimization of Processing Systems with Mixed Degrees of Freedom", In: Proc. 7th Int. Symposium on Dynamics and Control of Process Systems, 2004, ID: 164.

[17] O. Stursberg: "A Graph-Search Algorithm for Optimal Control of Hybrid Systems", In: 43rd IEEE Conf. on Decision and Control, 2004, 1412-1417.