1st HYCON PhD School on Hybrid Systems

www.unisi.it

www.ist-hycon.org

Hybrid Systems in Industrial Process Control

Olaf Stursberg

University of Dortmund, Germany

o.stursberg@ct.uni-dortmund.de

Hybrid systems combine continuous dynamics (differential or difference equations) typical of physical plants and discrete dynamics (automata and logical conditions) typical of control logic. By combining disciplines of computer science and systems and control theory, research on hybrid systems provide a solid theory and computational tools for the analysis, simulation, verification, and control design of "embedded systems", and are used in a large variety of applications (automotive systems, air traffic management, biological systems, process industries, and many others).

HYSCOM
IEEE CSS Technical Committee on Hybrid Systems

Information Society Technologies

Siena, July 19-22, 2005 - Rectorate of the University of Siena

Hybrid Systems in Industrial Process Control

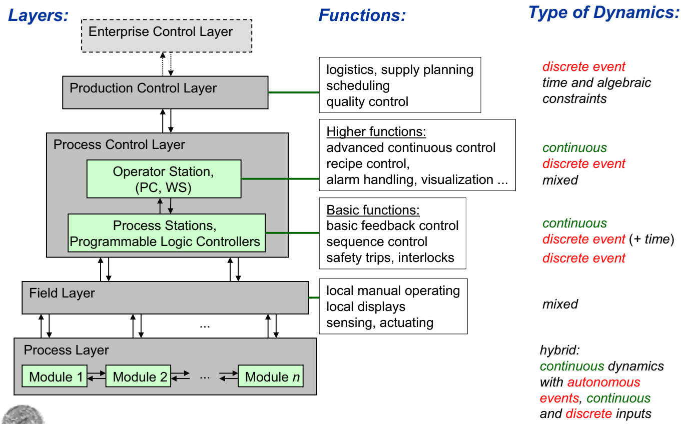
Olaf Stursberg

Process Control Laboratory
University of Dortmund, Germany

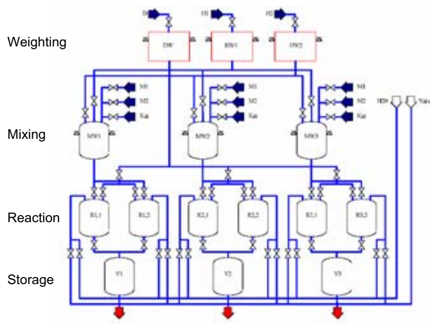
Outline

- Control Architecture of Production Systems
- Design Tasks
- Modeling with Hybrid Automata
- Optimal Control of Transition Procedures
- Synthesis of Supervisory Controllers
- Controller Verification using Reachability Analysis
- Conclusions and Open Problems

Control Structure of Production Systems



Example: Polymer Production Plant



Controller Functions:

(1) Coordination control:
- follow recipes
- assign task to resources

[mostly DCS, operator]

(2) Group control:
supervise the sequence of control actions in one unit
[PLC + Industrial PC]

(3) Basic Control:
supervise / control one (or a few) dependent process quantities
[mostly PLC, hard-wired; feedback loops]

Design Task

Control design for production plants ...

... is challenging:
- heterogeneity: the various control functions require different models and design techniques
- interdependency: functions on different layers affect each other; *how to guarantee consistency?*
- complexity:
  - due to dynamic type (nonlinearity, non-convexity)
  - due to size (large number of state variables, manipulated variables, operating modes, etc.)
  - hardware requirements
- modularity: *suitable communication paradigms?*

... includes the following tasks:
- optimization of transition procedures → **optimal control**
- algorithmic generation of supervisory controllers → **synthesis**
- a-posteriori analysis of the control design → **verification**

What can hybrid systems contribute?

Modeling with Hybrid Automata – Syntax

Hybrid automaton: $HA = (X, U, V, Z, inv, \Theta, g, r, f)$

- continuous states: $x \in X \subseteq R^{n_x}$
- continuous inputs: $u \in U = [u_1^-, u_1^+] \times \ldots \times [u_{n_u}^-, u_{n_u}^+]$
- finite set of discrete inputs: $v \in V = \{v_1, \ldots, v_{n_d}\},\ v_j \in R^{n_v}$
- finite set of locations: $Z = \{z_1, \ldots, z_{n_z}\}$
- invariants: $inv : Z \to 2^X$, polyhedral for all z
- transitions: $(z_1, z_2) \in \Theta \subseteq Z \times Z$
- guards: $g : \Theta \to 2^X$, polyhedral
- resets: $r : \Theta \times X \to X$
- flow functions: $f : Z \times X \times U \times V \to R^{n_x}$

  s.t. $\dot{x} = f(z, x, u, v)$ defines a continuous vector field

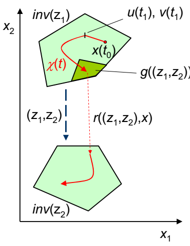
[for simplicity: no synchronization]

Modeling with Hybrid Automata – Semantics

Set of event times: $T = \{t_0, t_1, t_2, ...\}$

Input trajectories: $\phi_u = (u_0, u_1, ...) \in \Phi_u$,
$\phi_v = (v_0, v_1, ...) \in \Phi_v$
with u_k, v_k constant for $t \in [t_k, t_{k+1}[$

Hybrid state: $\sigma_k \in (z_k, x_k) \in \Sigma$ with
$x_k = x(t_k)$, $z_k = z(t_k)$



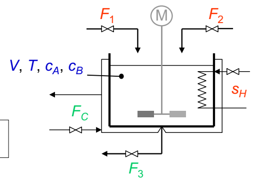
Feasible run of *HA* for given σ_0, ϕ_u and ϕ_v :
$\phi_\sigma = (\sigma_0, \sigma_1, \sigma_2, ...)$ with σ_k from:

(i) continuous evolution: $\chi(0) = x_k$ and $\chi(t)$ is the unique solution of the flow function for $t \in [0, \tau]$; $\chi(t) \in inv(z_k)$ but $\chi(t) \notin g((z_k, \bullet))$ for $t < \tau$

(ii) transition: $(z_k, z_{k+1}) \in \Theta$, $\chi(\tau) \in g((z_k,z_{k+1}))$, and
$x_{k+1} = r((z_k,z_{k+1}), \chi(\tau)) \in inv(z_{k+1})$

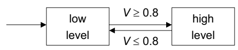
Modeling with Hybrid Automata – Example

Reactor with liquid-phase chemical reaction:



Variables:
- discrete inputs: F_1, F_2, s_H
- continuous inputs: F_C, F_3
- state variables: V, T, c_A, c_B

Discrete dynamics: (no resets)



Continuous dynamics:

$$\frac{dV}{dt} = F_1 + F_2 - F_3$$

$$\frac{dT}{dt} = \frac{F_1(k_1 - T) + F_2(k_2 - T)}{V} + F_c k_3 (k_4 - T) f_{v,1} + k_5 f_r + \underbrace{s_H k_6 (k_7 - T) f_{v,2}}_{\text{only for "high level"}}$$

$$\frac{dc_A}{dt} = \frac{F_1(k_8 - c_A) - F_2 c_A}{V} + k_9 f_r$$

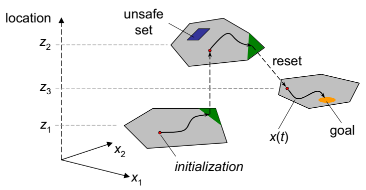
$$\frac{dc_B}{dt} = \frac{F_1(k_{10} - c_B) - F_2 c_B}{V} + k_{11} f_r$$

with : $f_{v,1} = k_{12} + \frac{k_{13}}{V}$, $\quad f_{v,2} = k_{14} - \frac{k_{15}}{V}$, $\quad f_r = c_A c_B^2 \exp\left(\frac{-k_{16}}{T}\right)$

Task 1: Optimal Control of Transition Procedures

given:
- hybrid automaton
- specifications: → transfer from initial state to goal set
  → safety restriction (exclusion of unsafe states)
  → maximized performance / minimized costs

[industrial relevance: start-up, shut-down, or change-over of processing systems]



Objective:
Determine input trajectories such that the specs are met!

Literature:
different approaches suggested; most based on piecewise affine approximations

Problem Statement

Target region: $(z_{tar}, x_{tar}) \in \Sigma_{tar} \subset \Sigma$, with one $z_{tar} \in Z$, $x_{tar} \in inv(z_{tar})$

Forbidden sets: $F = \{F_1, \dots F_{n_j}\}$ with $F_j \subset \Sigma$, polyhedral continuous sets

Assume: time set $T = \{t_0, t_1, ..., t_f\}$ is finite

Optimal control task:

determine ϕ_u^*, ϕ_v^* such that ϕ_σ^* is the solution to:

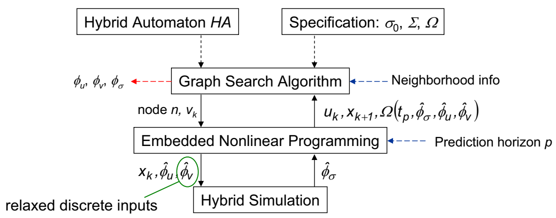
$$\min_{\phi_u \in \Phi_u, \phi_v \in \Phi_v} \Omega(t_f, \phi_u, \phi_v, \phi_\sigma)$$

subject to:
- $\phi_\sigma \in \Phi_\sigma$ (set of feasible runs)
- $\sigma_0 = (z_0, x_0)$, $\sigma_f \in \Sigma_{tar}$, $\sigma \notin F$

Chosen cost function Ω :
t_f in combination with weighted distances of σ_k to Σ_{tar}

Decomposition Approach

Principle:
- separate the optimization of continuous and discrete degrees of freedom:
  (i) high level: search tree encoding the discrete DOF → $v(t)$
  (ii) low level: embedded NLP for the continuous DOF → $u(t)$
- branch&bound and heuristics to prune the search tree efficiently
- cost function evaluated by hybrid simulation



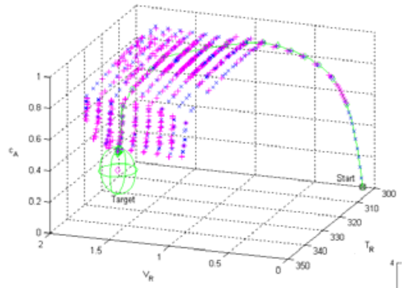
Results for the Example (1)

Objectives:
- reach nominal reaction (target) from an initially empty reactor
- time optimality
- avoid overflow and critical temperatures

Configurations:
- best-first search (throughout)
- pruning based on an adjacency criterion (chooses a locally best v_k)
- prediction horizon: $p = 2$

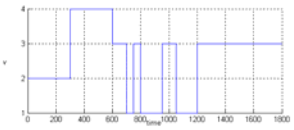
Results:
- termination after 959 nodes,
- 721 nodes fathomed due to adjacency, the remainder due to costs [theoretical number of nodes for the encountered path length: $3 \cdot 10^{14}$]
- computation time: 484 CPU-sec (P4-1.5 GHz)

Results for the Example (2)

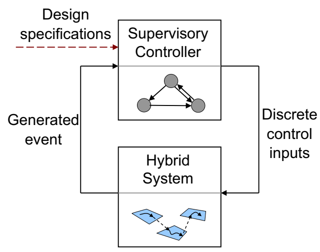


Projection into the (V_R, T_R, c_A) space:
– red: fathomed
– blue: not fathomed
– green: target, best found solution

Corresponding trajectory ϕ_V



Task 2: Synthesis of Supervisory Controllers



given:

(1) hybrid automaton *HA*:
- no continuous inputs
- discrete inputs: degree of freedom
- continuous controllers (if any) are embedded in the continuous dynamics
- generates events if transitions occur

(2) design specifications:
- goal attainment: reach target set from initial set
- safety: always avoid critical sets

Objective: *determine a supervisory controller as finite automaton*
- selects discrete control inputs upon receiving events
- here: no delay of the controller response

Synthesis Problem

Modified semantics of *HA*:
- if $x(t)$ enters a guard g, the transition must be taken before g is left
- the discrete input v can be changed only when a transition is taken

Given Sets:
- initial set of hybrid states: $(z_0, x_0) \in \Sigma_0$ with $z_0 \in Z$, $x_0 \in inv(z_0)$
- forbidden sets of hybrid states: $\Sigma_F = \{\Sigma_{F1}, \ldots, \Sigma_{Fp}\}$
- hybrid goal set: $(z_G, x_G) \in \Sigma_G$ with $z_G \in Z$, $x_G \in inv(z_G)$

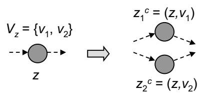
Synthesis Problem:

compute an input sequence $(v_0, v_1, v_2, \ldots)$ such that any $(z_0, x_0) \in \Sigma_0$ is driven into Σ_G by a feasible run that never encounters any hybrid state in Σ_F.

First Step:

rewrite *HA* into a closed system HA^c:

consider that any $v \in V_z$ can be applied in $z \in Z$

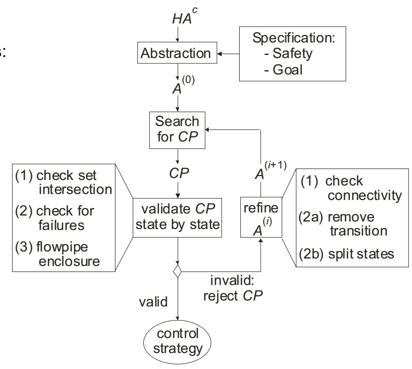


Solution based on Abstraction Refinement

Principle:
- use an abstract model to identify promising evolutions: *candidate paths CP*
- validate *CP* for the original model with lowest possible computational effort
- if necessary: refine the abstract model for the next iteration

→ a validated *CP* represents a proper control strategy

[similar to counterexample-guided verification of HA^c, see below]



Abstraction and Candidate Paths

Abstract Model: represents the discrete dynamics of HA^c

Finite state automaton: $A^{(0)} = (\hat{S}, \hat{S}_0, \hat{E})$

Abstraction function: $\alpha : Z^c \times X \to \hat{S}$ such that:

- states: – one state $\hat{s}$ for any location z^c
  – separate states for the sets: $\hat{S}_0, \hat{S}_G, \hat{S}_F$
- transitions: $\hat{E}$ according to the set Θ^c

Run of $A^{(j)}$: $(\hat{s}_0, \hat{s}_1, \hat{s}_2, \ldots)$

Candidate Path: $CP = (\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_p)$ with $\hat{s}_0 \in \hat{S}_0$, $\hat{s}_p \in \hat{S}_G$ and $\hat{s}_k \notin \hat{S}_F$ for all $k \in \{0, \ldots, p\}$

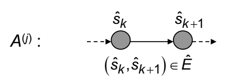
Search for *CP*: standard forward breadth-first algorithm

→ returns one of the shortest candidate paths existing for $A^{(j)}$

Validation

Check for any pair $(\hat{s}_k, \hat{s}_{k+1})$ of CP whether it realizes a feasible control action for HA^c:

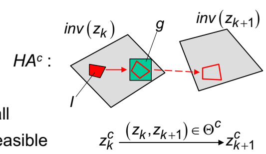
$I \subset inv(z_k^c)$ - set of continuous states represented by $\hat{s}_k$



⇒ any state $x \in I$ must be transferred into $inv(z_{k+1}^c)$ by:

(i.) continuous evolution

(ii.) transition and reset



Validation procedure: determine with an as small effort as possible that the control action is not feasible

(1) intersection check

(2) search for invalidating trajectories

(3) flowpipe enclosure

stricter condition, higher computational effort

[details: Stursberg et al. 2005]

Refinement

$A^{(j)}$ is refined to $A^{(j+1)}$ in the following cases:

(1) if the intersection check shows that $\left(z_k^c, z_{k+1}^c\right) \in \Theta^c$ can never be taken, the corresponding transition $\left(\hat{s}_k, \hat{s}_{k+1}\right)$ is removed from $\hat{E}$.

(2) if the other two validation methods show that $\left(\hat{s}_k, \hat{s}_{k+1}\right)$ is invalid, it cannot be removed from $\hat{E}$ immediately.

[Krogh et al., 2003: optimization-based method to show that $\left(z_k^c, z_{k+1}^c\right) \in \Theta^c$ cannot occur]

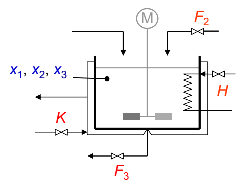
(3) if flowpipe approximation validates a control action, state splitting can be used optionally:

- new abstract state for the reachable subset of $inv\left(z_{k+1}^c\right)$
- transition set $\hat{E}$ modified according to the reachability result
- can be advantageous to (in-)validate a *CP* computed later

Application: Reactor Example (1)

Continuous chemical liquid-phase reactor:

modified setting:



- state variables: x_1 (level), x_2 (temperature), x_3 (concentration)
- inputs: F_2 (flow), F_3 (flow), K (cooling), H (heating)
  16 possible combinations of discrete values
- hybrid automaton:
  - 12 locations (32 for HA^c), 22 transitions
  - dynamics for $x_1 < 0.8$:

$$\dot{x}_1 = k_1 + F_2 + F_3,\ \dot{x}_2 = \frac{k_2(k_3 - x_2) + F_2(k_4 - x_2)}{x_1} + k_5 K(k_6 - x_2)\left(\frac{k_7}{x_1} + k_8\right),\ \dot{x}_3 = \left(k_9 - (k_{10} + F_2)x_3\right)/x_1$$

for $x_1 \geq 0.8$:

$$\dot{x}'_2 = \dot{x}_2 + k_{11}(k_{12} - x_2)(k_{13} - k_{14}/x_1)H$$

for $(0, k_{15}, k_{16}) \cdot x \geq k_{17}$:

$$\dot{x}''_2 = \dot{x}'_2 + x_3\left(k_{18} + k_{19}x_2^2\right),\ \dot{x}'_3 = \dot{x}_3 + k_{20} \cdot \exp(k_{21}/x_2)$$

Application: Reactor Example (2)

Task: find a control strategy for start-up into nominal operation
initially: reactor empty, goal state: high level, temperature, and yield

Control synthesis:

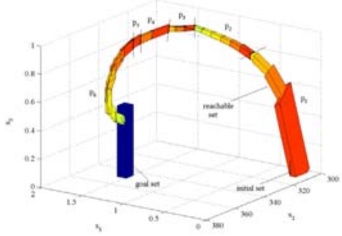
- 17th *CP*: feasible strategy
- six phases p_1 to p_6:

v_{p1} … v_{p6}

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

- 16 *CP* invalidated by the 2nd validation method
- obtained in approx. 4 minutes on a standard PC (P4-1.5GHz)

reachable set for the control strategy:



Task 3: Verification of Logic Controllers

Controlled System

Dependability: availability, reliability, maintainability, integrity, confidentiality, ..., functional safety

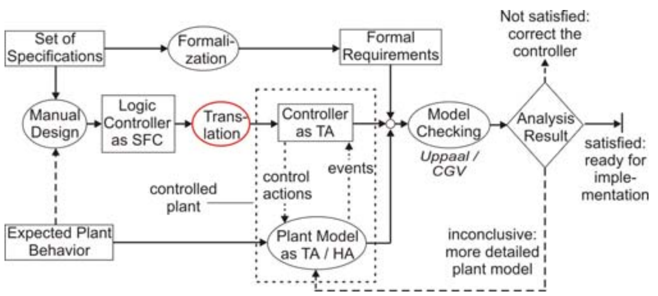
IEC 61508 – Functional Safety:
execution of safety-related functions on electric or electronic programmable systems such that significant hazards (harming the personnel, equipment, or environment) are prevented.

How to check that the PLC code implements functional safety?

hybrid model of a production system ⇔ Programmable Logic Controller (PLC)

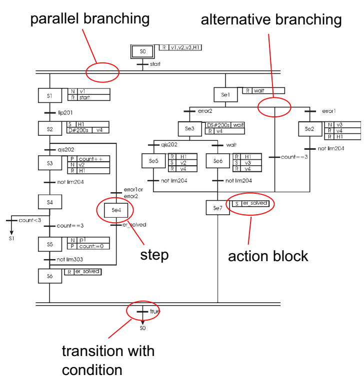
Design specifications:
- goal attainment
- safety related controls
- non-functional requirements

Verification Approach



SFC: Sequential Function Charts (standardized language for logic controllers)
TA: Timed Automata
CGV: Matlab tool for counterexample-guided verification

Sequential Function Charts



$SFC = (ST, s_0, X, G, T, A, \alpha, C)$

with the sets of

- steps: ST, $s_0 \in ST$
- variables: $X = X_{in} \cup X_{internal} \cup X_{out}$
- transition conditions: G
- transitions: $T \subseteq (2^{ST} \setminus \{\emptyset\}) \times (2^{ST} \setminus \{\emptyset\})$
- actions: $a = (q, \tau, o, f) \in A$
  q - action qualifier, τ - time quantifier
  o - operand, f - execution flag
- clocks: C

and:

- the function: $\alpha: ST \rightarrow B$
  with action blocks $b \in B$

Execution of SFC on PLC

Cyclic update of the SFC configuration γ_k:

1.) read inputs (X_{in}^k)

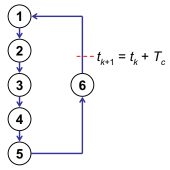
2.) execute active actions (X^k)

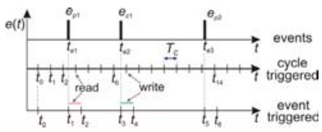
3.) execute enabled transitions (ST^k)

4.) determine new set of active actions (A^k)

5.) write outputs X_{out}^k

6.) wait until cycle completed: $T_c \in [T_c^{min}, T_c^{max}]$

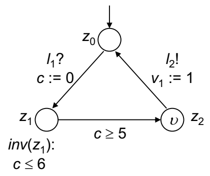




Event-triggered transformation of SFC into Timed Automata:

- event $e(t_e)$: change of any variable evaluation at time t_e
- $e(t_e)$ triggers the controller model at: $t \in [t_e, t_e + T^c_{max}]$

Timed Automata



→ states:
can be marked as urgent (υ)

→ synchronization:
- strictly pairwise
- broadcasting
- urgent

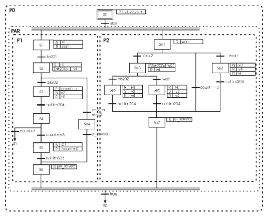
$TA = (Z, z_0, V, Lab, C, E, inv)$ [as in *Uppaal*]

with the finite sets of

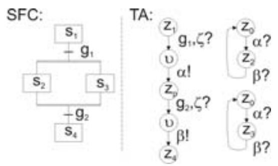
- states: Z, $z_0 \in Z$
- variables: $V = V_{in} \cup V_{internal} \cup V_{out}$
- synchronization labels Lab
- clocks: C
- transitions: $e = (z, z', l, g, \chi) \in E$
  z - source states, z' - target state, $l \in Lab$
  g - transition guard, χ - value assignment for all clocks and variables
- and the function $inv(z)$ assigning clock conditions to $z \in Z$

Transformation of SFC into TA

1. decompose *SFC* into units according to the reduction rules of a graph grammar
   [Bauer et al. 2004]
2. introduce the *TA* for event-triggering
3. introduce one *TA* per unit of the decomposition:
   - one state per step
   - each transition of *SFC* mapped into one $e \in E$
   - special structure for parallel branching:



units: *P0*, *PAR*, *P1*, *P2*



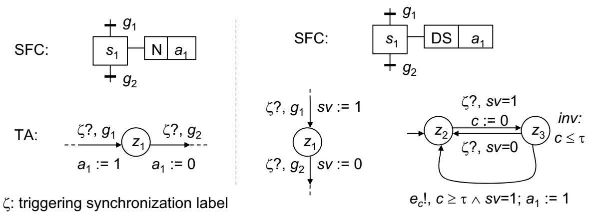
the states z_0 represent inactivity of the parallel branching

ζ: triggering synchronization label

Transformation of SFC into TA

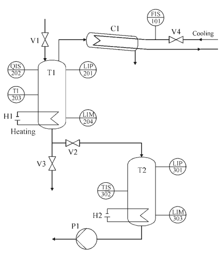
4. map clocks and variables identically from *SFC* to *TA*
5. add transition guards using the corresponding logical conditions
6. modeling of actions:
   - *untimed actions* (N, R, S, P) and *simple timed actions* (L, D): embedded into the unit automaton
   - *complex timed actions* (SD, DS, SL): introduce separate automata

Examples:



ζ: triggering synchronization label

Application: Evaporator Example

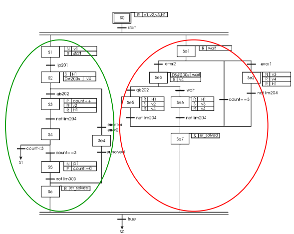


Batch operation:

- tank T1 filled through V1, content of T1 heated up
- evaporation until a dissolved substance has reached a desired concentration
- vapor condensation in C1
- three batches of product (from T1) are collected in T2, then emptying through P1

Error handling:

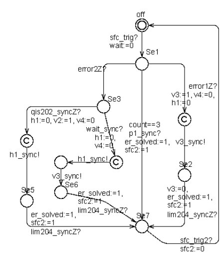
- failure of H1 in T1: content disposed through V3
- failure of C1: continue heating to avoid crystallization (but also over-pressure)
- resume nominal operation after repair



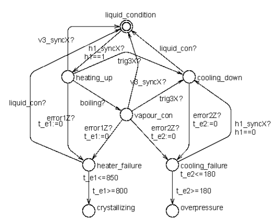
Application: Transformation and Modeling

- **Controller transformation:** 6 automata (3 for the units, one each for the trigger, the evaluation check, and the action DS)
- **Plant model:** 5 timed automata (two levels, heating, condenser, state of aggregation); transition times measured for the real system

Controller unit P2: error handling



Plant as TA: e.g. state of aggregation



Application: Verification with *Uppaal*

- **Parallel composition** of all controller and plant automata; two additional automata to model malfunctions of the heater and the cooling
- **Formal requirements:**
  - (1) E <> tank2.emptying (production goal can be reached)
  - (2) A [ ] not liquid.crystallizing (if heater malfunction, temp. in T1 not too low)
  - (3) A [ ] not liquid.overpressure (if cooling malfunction, temp. in T1 not too high)
- **Verification:**
  - model checking with *Uppaal* [Larsen et al., 2000]
  - standard PC (Pentium 1.5 GHz)
- **Results:**
  - → no deadlock, state 'tank2.emptying' eventually always reached (nominal production realized)
  - → 2nd and 3rd property satisfied ⇒ functional safety achieved
  - → computation time: below 1 second in all cases

Application: Verification with *CGV* (1)

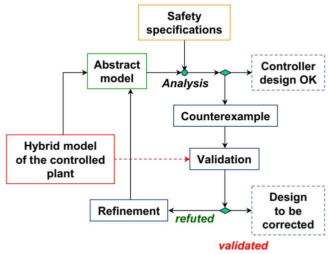
Hybrid Model:

- State variables: x_1 (temperature in T1), x_2 (level in T1), x_3 (level in T2)
- Nonlinear ODEs for the combinations of the following cases:
  - levels x_2 and x_3: *filling*, *emptying*, *constant;* for x_2 in addition: *evaporating*
  - temperature x_1: *heating*, *heat loss to environment*

Verification with *CGV*:

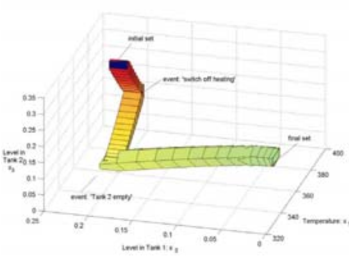
- Abstractions used to identify potentially unsafe behaviors
- Hierarchy of different validation methods

⇒ computation of reachable sets only for a 'small' part of the hybrid state space



Application: Verification with *CGV* (2)

computed reachable hybrid set:



- Scenario:
  - → during evaporation (x in initial set): malfunction of the condenser
- Task: verify whether x_1 > 394K and x_1 < 339 K is avoided
- Computational time: around 1 minute (P4-1.8 GHz)
- Result:

  critical temperatures not reached (no overpressure, final set: T1 emptied before crystallization at x_1 < 339K)

  → the SFC-controller fulfills the requirements!

Conclusions and Open Problems

Status Quo:

- Hybrid models are appropriate to formulate the different dynamics occurring in industrial plants.
- First tools for optimization, control synthesis, and verification exist.
- Size of problems tractable so far: only relatively small parts of the plant, or only simple specifications.
- Robustness / model uncertainties not yet considered to a sufficient extent.

Open Problems:

- Find clever methods to handle complexity (decomposition, abstraction, ...).
- Use of stochastic models and corresponding techniques.
- Connect hybrid models to the languages used in industry.
- Make practitioners aware of design techniques based on hybrid systems. (... *modeling must be easy* ...)

References

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*, Vol. 279, 2002, 337-352.

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

O. Stursberg: Synthesis of Supervisory Controllers for Hybrid Systems using Abstraction Refinement. *16th IFAC World Congress*, 2005, ID: We-M12-TO/2.

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. *Intern. Journal Foundations of Computer Science*, Vol. 14 (4), 2003, 583-604.

O. Stursberg, S. Lohmann, S. Engell: Improving Dependability of Logic Controllers by Algorithmic Verification. *16th IFAC World Congress*, 2005, ID: Mo-E17.TO/6.

O. Stursberg, B.H. Krogh: Efficient Representation and Computation of Reachable Sets for Hybrid Systems. In: *Hybrid Systems – Computation and Control*, Springer, LNCS, Vol. 2623, 2003, 482-497.

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 Specification Techniques for Applications in Engineering*, Springer, LNCS, Vol. 3147, 2004, 517-540.