Reachability Analysis for Hybrid Dynamic Systems*

Olaf Stursberg

Faculty of Electrical Engineering and Information Technology
Technische Universität München

* Thanks to: Matthias Althoff, Edmund M. Clarke, Ansgar Fehnker, Bruce H.Krogh, Sven Lohmann, Tina Paschedag, Michael Theobald
Contents

- Hybrid Dynamic Systems: Motivation and Definition
- Principles of Reachability Analysis
- Abstractions in Computing Reachable Sets
- Verification based on Reachability Analysis
- Extension to Uncertain Hybrid Systems
- Reachable Sets in Controller Design
- Optimization using Abstractions
- Conclusions
Hybrid Dynamic Systems (HDS):

„Discrete event system equipped with continuous-valued dynamics“

∨ „Continuous dynamics enriched by discontinuities (switching, jumps)“

Examples:

(a) Walking humanoid robots

- $x_i$: joint positions, velocities, …
- $z_i$: ground contact situation

(b) Autonomously driving cars

- $x_i$: distances, velocities, …
- $z_i$: driving modes (gears; accelerating, braking, …)

$\mathbf{x}_i \in \mathbb{R}, \; \mathbf{z}_i \in \mathbb{N}$

[Wünsche et al., UniBW]

[Ulbrich et al., TUM]
Motivation for Modeling by Hybrid Dynamic Systems

Examples (ctd.):

(c) Manufacturing plants

- $x_i$: work piece positions, robot arm control, …
- $z_i$: processing status, resource conditions, …

(d) Chemical processing systems

- $x_i$: temperature, levels, concentrations, …
- $z_i$: production phase, actuator state, …

$\forall i \in \mathbb{N}$: $x_i \in \mathbb{R}$, $z_i \in \mathbb{N}$

hierarchical, distributed heterogenous control

[Zäh et al., TUM] [BASF]
Motivation for Modeling by Hybrid Dynamic Systems

Examples (ctd.):

(e) Air conflict resolution

- $x_i$: speed, orientation, ...
- $z_i$: flight mode, (cruise, conflict resolution)

(f) Multi robot systems

- $x_i$: position, speed, ...
- $z_i$: communication topology, formation, ...

$x_i \in IR, \ z_i \in IN$

[Tomlin et al., Stanford]

[LSR, TUM]
Hybrid Automaton: \( HA = (X, U, V, Z, \text{inv}, \Theta, g, r, f) \)

- Continuous states: \( x \in X \subseteq \mathbb{R}^{n_x} \)
- Continuous inputs: \( u \in U = [u_1^-, u_1^+] \times \ldots \times [u_{nu}^-, u_{nu}^+] \)
- Discrete inputs: \( v \in V = \{v_1, \ldots, v_{nd}\}, \quad v_j \in \mathbb{R}^{n_v} \)
- Discrete states (locations): \( Z = \{z_1, \ldots, z_{n_z}\} \)
- Invariants: \( \text{inv} : Z \rightarrow 2^X \)
- Transitions: \( (z_1, z_2) \in \Theta \subseteq Z \times Z \)
- Transition guards: \( g : \Theta \rightarrow 2^X \)
- Reset functions: \( r : \Theta \times X \rightarrow X \)
- Continuous dynamics: \( f : Z \times X \times U \times V \rightarrow \mathbb{R}^{n_x}, \quad \text{such that: } \dot{x} = f(z, x, u, v) \)
Hybrid Dynamic Systems: Semantics

Set of event times: \( T = \{t_0, t_1, t_2, \ldots \} \)

Input trajectories:
\[ \phi_u = (u_0, u_1, \ldots) \in \Phi_u, \]
\[ \phi_v = (v_0, v_1, \ldots) \in \Phi_v \]
with inputs \( u_k, v_k \) on \( t \in [t_k, t_{k+1}[ \)

Hybrid States:
\( s_k \in (z_k, x_k) \in S \) with:
\[ x_k = x(t_k), z_k = z(t_k) \]

Feasible execution for given \( s_0, \phi_u, \phi_v \):
\( \phi_s = (s_0, s_1, s_2, \ldots) \) with \( s_k \) from:

(i) contin. evolution: \( \chi(0) = x_k \) and \( \chi(t) \) is unique solution of ODEs for \( t \in [0, \tau] \); \( \chi(t) \in inv(z_k) \), (optional: \( \chi(t) \not\in g((z_k, \bullet)) \) für \( t < \tau \))

(ii) transition: \( (z_k, z_{k+1}) \in \Theta, \chi(\tau) \in g((z_k, z_{k+1})) \) and
\[ x_{k+1} = r((z_k, z_{k+1}), \chi(\tau)) \in inv(z_{k+1}) \]
Investigations for Hybrid Dynamic Systems

- Simulation (including sliding modes)
- Abstraction and Refinement
- Reachability Analysis and Verification
- Optimal Control
- Scheduling for TA
- Controller Synthesis
- Networked Control Systems
- Design of SFC controllers with subsequent testing / verification

O. Stursberg: Reachability Analysis for Hybrid Dynamic Systems
Design Tasks for HDS using Reachability Computation

(a) Verification:
given:  
• plant $P$ of type $HA$
• controller $C$ of type $HA$
• specification $\gamma$
  (e.g. safety: $AG \rightarrow S_{unsafe}$)  
show that: $P \parallel C \models \gamma$
if false, redesign $C$

(b) Controller Synthesis:
given:  
• plant $P$ of type $HA$
• specification $\gamma$
  (goal attainment: $AF S_{target}$)  
generate $C$ such that:
$P \parallel C \models \gamma$

(c) Optimal Control:
given:  
• plant $P$ of type $HA$
• goal and safety spec. $\gamma$
• performance measure $\psi$
compute $C$ such that:
$\min_{\phi_u, \phi_v} \psi$
subject to: $P \parallel C \models \gamma$
Reachable Set Computation for HDS

Def.: Reachable Set of \( HA \) given:

- initialization \( S_0 \subset S \),
- sets of input trajectories \( \Phi_u, \Phi_v \)

\[
R := \left\{ s \in S \mid \exists s_0 \in S_0, \phi_u \in \Phi_u, \phi_v \in \Phi_v : s \text{ is reached along any } \phi_s \in \Phi_s \right\}
\]

Assumption: \( M := P \| C \), i.e. \( M \) is autonomous (\( \Phi_u, \Phi_v \) restricted by \( C \))

Standard algorithm for computing \( R \):

\[
\begin{align*}
S_0 & := \{z_0\} \times X_0, \ k := 0 \\
D & := S_0, \ R := \emptyset \\
\text{WHILE } (D \neq \emptyset) & \quad \text{Termination?} \\
& \quad \text{step } k: \\
& \quad \quad - \text{transition} \\
& \quad \quad - \text{time step} \\
& \quad \quad \text{ } \\
& \quad \quad k := k + 1 \\
& \quad \quad R := R \cup D \\
& \quad \quad S_k := \text{Reach}(D) \\
& \quad \quad D := S_k \setminus R \\
\text{END}
\end{align*}
\]
The Challenge

Problem: scales badly in most respects!
- infinitely many executions of $M$ must be analyzed
- reachable sets have to be represented efficiently
- set intersection, subtraction, and union must be computable

In contrast:

For finite state automata $A = (Z, z_0, \Theta)$, the reachable set:

$$R := \{z_k \in Z \mid \exists z_0 \in Z_0 : (z_0, z_1) \in \Theta, \ldots, (z_{k-1}, z_k) \in \Theta\}$$

is efficiently computable.

[Verification of $A \models \gamma$ successfully reported for systems with $|Z| \approx 10^{20}$.]

Approach:

- Use of abstractions $A$ of $HA$
- Consider the specification of analysis, synthesis, or optimization for computing $R$ → reduce use of $Reach, \cup, /$
Abstraction-Based Reachability Analysis: Principle

**Objective:** identify evolutions of $M$ that potentially violate $\gamma$ based on abstractions $A$ and evaluate $\text{Reach}(D)$ only for these evolutions!

**Principle:**
- Generate a **discrete abstraction** $A$ of the hybrid model $M$ ($A$: state transition system)
  - Determine **counterexamples** (CEs) for $A$ as evolutions of $M$ that potentially violate the specification
    (CE: run of $A$ that connects the initial and critical state)
  - (In-)**Validate** CE for $M$
  - if CE is invalid, add details to $A$ (refinement)

**Assumption:** let $\gamma$ denote a safety specification:

$$\text{given } S_{\text{unsafe}} \subset S : \neg \exists (s_0 \in S_0, \phi_s \in \Phi_s) : (z, \chi(\tau)) \in S_{\text{unsafe}}$$
Abstraction-Based Reachability Analysis: Principle

- **Generate an Initial Abstraction**
  - $M, \gamma$
  - $A$

- **Model Checking**
  - $A \not= \gamma$

- **Generate a Counterexample CE**
  - $CE$

- **Counterexample Validation**
  - $CE$ is spurious
  - $CE$ is not spurious

- **Refinement of A**

- **no CE exists**
  - $M \models \gamma$

[for discrete automata: Clarke et al., 2000]
Initial Abstraction

Abstract away the continuous part of $M$, retain the discrete dynamics:

- a state in $A$ represents a location in $M$
  (exception: initial location)
- one transition in $A$ for each transition in $M$

$A$ is a simple state transition system: $A = (S^A, s^A_0, \Theta^A)$

$A$ is an abstraction: it contains all evolutions of $M$
- can contain additional behaviors
Abstraction-Based Reachability Analysis: Principle

1. Generate an Initial Abstraction $M, \gamma$
2. Model Checking $A$
   - If $A \not\models \gamma$, Generate a Counterexample $CE$
     - CE Validation: If $CE$ is spurious, Stop
     - If $CE$ is not spurious, Refinement of $A$
   - If no $CE$ exists, Stop

3. Continue with step 2 until all conditions are met.
- Standard model checking for FSA can be applied (breadth-first search for $S^A$ starting from $s^A_0$)
- if $\gamma$ is violated, i.e., a critical state $s^A_f$ is reachable:

  counterexample (CE): $(s^A_0, s^A_1, ..., s^A_f)$

Question: Does a corresponding evolution exist for $M$?

$\Rightarrow$ validation along the counterexample
Abstraction-Based Reachability Analysis: Principle

1. Generate an Initial Abstraction

2. Model Checking

3. Generate a Counterexample CE

4. Refinement of A

5. CE Validation

- If $M, \gamma$ is not CE, then stop with $M, \gamma$.
- If $M, \gamma$ is CE, then stop with $M, \gamma$.
- If $M, \gamma$ is CE is not spurious, then stop with $M, \gamma$.
- If $M, \gamma$ is CE is spurious, then refine $A$ and go back to Model Checking.
- If no CE exists, then stop with $M, \gamma$.

O. Stursberg: Reachability Analysis for Hybrid Dynamic Systems
Validation of Counterexamples (1)

**VM1: Intersection Check**
transition of $A$ is invalid if:
$$r_j(x) \not\in \text{inv}(z_2) \ \forall \ x \in g_j$$

**VM2: Gradient Check**
determine gradient on the guard boundaries
transition of $A$ is invalid if:
$$\min (c^T \cdot f(x)) > 0 \ \forall \ x \in \partial g$$

**VM3: Connectivity Check**
transition of $A$ is invalid if:
$$\min \{d\} > 0 \ \forall \ x_0 \in E$$
\[\text{s.t. dynamics of } HA \text{ in } z_1\]
Validation of Counterexamples (2)

VM4: Flowpipe Approximation

computation of $S_k = \text{Reach}(D)$:

for each segment:

(1) simulate vertices for a timestep

(2) determine an oriented hyper-rectangle
    (orientation from sample covariance matrix)

(3) enlarge hull (nonlinear optimization with embedded simulation)
Validation of Counterexamples (3)

Four methods to refute the existence of counterexamples:

VM1 VM2 VM3 VM4

accuracy (over-approximation)

computation cost

⇒ refute with the least effort possible

Application mode:

sequential:

alternating:

terminate process as soon as one transition is refuted!
Abstraction-Based Reachability Analysis: Principle

\[ M, \gamma \]

Generate an Initial Abstraction

\[ A \]

Model Checking

\[ A \not\models \gamma \]

Generate a Counterexample CE

\[ CE \]

Counterexample Validation

\[ CE \text{ is not spurious} \]

\[ M \models \gamma \]

Stop

\[ M \not\models \gamma \]

Refinement of \( A \)

\[ \text{CE is spurious} \]
Refinement of $A$

Refinement based on the flowpipe approximation:
If $x(\tau)$ exists, i.e. the transition $s_1^A \rightarrow s_2^A$ is validated, the automaton $A$ is refined by splitting $s_2^A$:

Purging of $A$:
If $x(\tau)$ does not exist, the corresponding transition is removed from $A$, and the method proceeds with a new CE.
Example: Verification of a Cruise Controller (1)

Control objectives:
- **Mode A**: constant speed
- **Mode B**: distance control

Safety specification $\gamma$: $-(r < r_{\text{critical}})$

Discrete Dynamics:

<table>
<thead>
<tr>
<th>Mode B</th>
</tr>
</thead>
<tbody>
<tr>
<td>4th Gear</td>
</tr>
<tr>
<td>Mode A</td>
</tr>
<tr>
<td>4th Gear</td>
</tr>
<tr>
<td>r &gt; r_d + h</td>
</tr>
<tr>
<td>r &lt; r_d - h</td>
</tr>
<tr>
<td>v &gt; 29.8</td>
</tr>
<tr>
<td>v &lt; 29.8</td>
</tr>
<tr>
<td>r &gt; r_d + h</td>
</tr>
<tr>
<td>r &lt; r_d - h</td>
</tr>
<tr>
<td>v &gt; 29.8</td>
</tr>
<tr>
<td>v &lt; 29.8</td>
</tr>
<tr>
<td>v &gt; r_d + h</td>
</tr>
<tr>
<td>v &lt; r_d - h</td>
</tr>
<tr>
<td>v &gt; 14.2</td>
</tr>
<tr>
<td>v &lt; 14.2</td>
</tr>
<tr>
<td>r &gt; r_d + h</td>
</tr>
<tr>
<td>r &lt; r_d - h</td>
</tr>
<tr>
<td>v &gt; 14.2</td>
</tr>
<tr>
<td>v &lt; 14.2</td>
</tr>
<tr>
<td>v &gt; 6.7</td>
</tr>
<tr>
<td>v &lt; 6.7</td>
</tr>
<tr>
<td>r &gt; r_d + h</td>
</tr>
<tr>
<td>r &lt; r_d - h</td>
</tr>
<tr>
<td>v &gt; 6.7</td>
</tr>
<tr>
<td>v &lt; 6.7</td>
</tr>
</tbody>
</table>

Continuous Dynamics:
- in "Collision": $\dot{r} = 0$, $\dot{v} = 0$
- else:
  \[ \dot{r} = v_l - v \]
  \[ \dot{v} = \min\left( \max\left( 0, \frac{a_d + 3.5}{a_{par} + 3.5} \right), a_{par} + 3.5 \right) - 3.5 \]
  \[ a_d = f(Mode, v, v_d, v_l, r), \quad a_{par} = f(gear) \]
Example: Verification of a Cruise Controller (2)

Verification for given parametrization:
- 10 counterexamples
- VM4 (reachable set computation) only applied once
- final abstract model A: 11 states
- computation time: < 10 seconds on a standard PC

Result: \( \neg(r < r_{\text{critical}}) \) does hold

Reachable sets:
- abstraction-based
- complete \( R \)-computation
Extension to Stochastic Verification

Context:
iterative online verification of driving strategies for autonomous cars

Method:
- hybrid model with uncertain dynamics: \( \dot{x} = [H]z_i \cdot x + [V]z_i \)
- reachable set computation based on zonotopes
- abstraction into Markov chains
- computation of collision probabilities

Computation time: 0.88 seconds for 3.2 seconds in real time using Matlab on a notebook processor (1.66 GHz).
Synthesis of Supervisory Controllers

**Modification of HA:**
- no continuous inputs $u$
- discrete input $v$ changes only when a transition is taken
- if $x(t)$ enters $g$, the transition must be taken before $g$ is left

**Given sets** (one $z_i$, compact in $X$)
- initial set: $S_0$
- forbidden sets: $S_{F,i}$
- goal set: $S_G$

**Synthesis Problem:**
compute $\phi_v = (v_0, v_1, v_2, \ldots)$ such that any $(z_0, x_0) \in S_0$ is driven into $S_G$ by a feasible run of $HA$ that never encounters any $s \in S_F = \bigcup_i S_{F,i}$
Abstraction-Based Synthesis: Principle

Principle:

- **rewrite** $HA$ into closed system $Hac$ by considering any $v \in V_z, z \in Z$
- **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

$\rightarrow$ a validated $CP$ represents a proper control strategy
Abstraction and Candidate Paths

Abstract Model: \( A^{(0)} = (S^A, s^A_0, \Theta^A) \)
represents the discrete dynamics of \( HA^c \) (as in verification)

Candidate Path: \( CP = (s^A_0, s^A_1, ..., s^A_p) \) with \( s^A_0 \in S^A_0, s^A_p \in S^A_G \) and \( s^A_k \notin S^A_F \)
for all \( k \in \{0, ..., p\} \)
search for \( CP \): standard forward breadth-first algorithm
\[ \rightarrow \text{returns one of the shortest candidate paths existing for } A^{(i)} \]
Validation (1)

Check for any pair $\left(s_k^A, s_{k+1}^A\right)$ of CP whether it realizes a feasible control action for $HA^c$:

$I \subset \text{inv}(z_k)$ - set of continuous states represented by $S_k^A$

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

\textbf{stricter condition, higher computational effort}
Validation (2)

(1.) **Intersection Check:** control action is invalid, if no \( x \in g(z^c_k, z^c_{k+1}) \) is mapped into \( inv(z^c_{k+1}) \) by \( r(z^c_k, z^c_{k+1}, x) \).

(2.) **Search for invalidate trajectories:**

target set \( T \): subset of \( g(z^c_k, z^c_{k+1}) \) that is mapped into \( inv(z^c_{k+1}) \) by the reset. control action is invalid, if any \( x(t_f) \notin T \) is found during solving:

\[
\max_{x_0 \in I} \| x(t_f) - x_{cent,g} \|_2
\]

with the terminal state \( x(t_f) \) determined by numeric simulation as:

(a) \( x(t_f) \in T \)

(b) \( x(t_f) \in g(z^c_k, z^c_q) \) with \( z^c_q \neq z^c_{k+1} \)

(c) \( x(t_f) \in F \)

(d) \( x(t_f) \notin inv(z^c_k) \) and \( x(t_f^c) = inv(z^c_k) \)

(e) \( x(t) \in inv(z^c_k) \) and \( t_{max} < t < t^*_f \)

invalidating cases
(3.) Flowpipe enclosure:
over-approximation of the continuous set reachable inside of \( \text{inv}(z^c_k) \) starting from \( I \)

\[ \to \text{series of oriented hyper-rectangles [Krogh et al., 2003]}: \]
each hyper-rectangle computed by numeric simulation embedded into optimization

control action is invalid, if the flowpipe does not completely lead into the set \( g((z^c_k, z^c_{k+1})) \)

\[ \to \text{if a control action is invalid, refute CP!} \]
\[ \to \text{if a control action is valid: continue with the next step of CP.} \]
Refinement of $A$

$A^{(i)}$ is refined to $A^{(i+1)}$ 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 $\left( S_k^A, S_{k+1}^A \right)$ is removed from $E$.

(2) if the other two validation methods show that $\left( S_k^A, S_{k+1}^A \right)$ is invalid, it can not be removed from $\Theta^A$ 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 $\text{inv}(z_{k+1}^c)$
- transition set $\Theta^A$ modified according to the reachability result
- can be advantageous to (in-)validate a CP computed later
Application: Chemical Reactor

Continuous chemical liquid-phase reactor:
- exothermic reaction of two components
- 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 values
- hybrid automaton:
  - 12 locations (32 for HA$^c$), 22 transitions
  - dynamics for $x_1 < 0.8$:

  for $x_1 \geq 0.8$:
  \[
  \begin{align*}
  \dot{x}_1 &= k_1 + F_2 + F_3, \\
  \dot{x}_2 &= \frac{k_2(k_3 - x_2) + F_2(k_4 - x_2)}{x_1} + k_5K(k_6 - x_2)\left(\frac{k_7}{x_1} + k_8\right), \\
  \dot{x}_3 &= \frac{k_9 - (k_{10} + F_2)x_3}{x_1}
  \end{align*}
  \]

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

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

O. Stursberg: Reachability Analysis for Hybrid Dynamic Systems
Task: find $\phi_v$ for start-up from $S_0$ into nominal operation $S_G$  
($S_0$: reactor empty and cold; $S_G$: high level, temperature, and yield)

Control synthesis:
- 17$^{th}$ CP: feasible strategy
- six phases $p_1$ to $p_6$:
  
  $\begin{bmatrix} v_{p1} \\ v_{p6} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

- 16 CP invalidated by the 2$^{nd}$ validation method
- obtained in approx. 4 minutes on a standard PC (P4-1.5GHz)
Model: \( HA \) as defined initially, but transitions are taken deterministically

Sets: 
- initial states: \( s_0 \in S_0 \) with \( z_0 \in Z, x_0 \in \text{inv}(z_0) \)
- goal states: \( s_G \in S_G \) with \( z_G \in Z, x_G \in \text{inv}(z_G) \)
- unsafe states: \( S_F = \{S_{F,1},...,S_{F,p}\} \)

Performance criterion \( \psi \) with the number of event times \( n_e = |\phi_s| \):

\[
J(z,x,u,v) = \sum_{k=1}^{n_e-1} t_k \int L(x,u,v)dt + \sum_{k=1}^{n_e-2} c_{disc}(z_{k-1},z_k)
\]

Hybrid optimal control problem:

\[
\min_{\phi_u,\phi_v} J(z,x,u,v)
\]

subject to: \( \phi_u, \phi_v \) lead to a feasible run of \( HA \)

\[
s_0 \in S_0, s(t_f) \in S_G, s(t) \notin S_F \quad \forall t \in [t_0, t_f]
\]
Scheme of Abstraction-Based Optimal Control

**Idea:**
- simplify hybrid optimal control problem by abstraction
- use reachability analysis for updating the abstracted model and cost function

**Steps:**
1. define abstraction maps for $HA$ & $J_{HA}$
2. solve abstracted optimization problem
3. refine to trajectories of $HA$ by reduced optimization problem
4. evaluate trajectory in terms of original costs
5. update abstract model and cost criterion iteratively
Conclusions and Future Work

- Modeling with HA useful for a wide range of applications
- Computation of $R$ is the core of many design techniques for HA
- Reachability analysis is computationally costly: use of abstractions can reduce the load
- Choice of abstraction (preserved property, degree of detail) is crucial
  [use of $A$ such that it encodes discrete dynamics of HA is not always a suitable choice]

Future work:
- Extend verification to other specifications $\gamma$ than safety
- Improve efficiency of computation with respect to $n$
- Use hierarchies of abstractions
- Include model uncertainties and robustness