Counterexample Guided Abstraction Refinement for Hybrid Systems Diagnosability Verification

Hadi Zaatiti (CEA and Université Paris-Sud)
Lina Ye (CentraleSupélec and LRI)
Philippe Dague (LRI)
Jean-Pierre Gallois (CEA)
Outline

• Scientific context and challenges

• Hybrid Automaton definition

• Timeless and timed abstractions

• Abstraction Refinement

• Counter Example Guided Abstraction Refinement
Scientific Context and Challenges 1/3

Formal methods for the verification of complex systems

- Hybrid Systems are dynamical systems with:
  - **Continuous-time** dynamics (ODE)
  - **Discrete-time** dynamics (Automata, Discrete Event Systems)

- Hybrid Automata: Powerful, intuitive and general formal representation:
  - Aeronautics, railway, vehicles, robotics
  - Cyber-Physical System: analog electronics + physical environment
Scientific Context and Challenges 2/3

Qualitative vs Quantitative

- Qualitative Simulation at modeling stage
  - Qualitative “grouping” of set of executions
  - Global verification by over approximation of the system

- Challenges:
  - No standardized specification language
    - Tools with many languages
  - Hybrid Systems must be studied as a whole:
    - Two continuous systems are stable separately
    - Combined together are not necessarily stable

- Numerical Simulation approaches
  - Executions one by one
  - Local verification
• **Property Verification** is undecidable except for simple hybrid systems:

  • **Reachability**: System reaches only certain discrete/continuous states?
  
  • **Liveness**: system reaches all modes infinitely often

  • **Diagnosability**: Given a set of observations, can all modeled faults be detected and isolated?

• Hybrid Systems **Verification** Technics:

  • **Abstraction** technics
  
  • **Invariant** Computation
• **Property Verification** is undecidable except for simple hybrid systems:

  • **Reachability**: System reaches only certain discrete/continuous states?
  
  • **Liveness**: System reaches all modes infinitely often
  
  • **Diagnosability**: Given a set of observations, can all modeled faults be detected and isolated?

• Hybrid Systems **Verification** Technics:

  • **Abstraction** technics
  
  • **Invariant** Computation
Hybrid Automata Definition

- **Hybrid System automaton model:**
  - **Mode**: finite set of locations
  - **X**: set of variables
  - **F**: differential equations/inclusions
  - **Init**: Initial set where all executions start from
  - **Inv**: constraint must be satisfied while in a mode
  - **G**: Guard set of valuations for enabling mode change
  - **R**: Reset new valuations of variable after mode change

**Particular Case:**
**Timed Automata**, all variables are clocks i.e. \( \dot{x} = 1 \)
Timeless and Timed Abstractions

• **Abstractions** of hybrid systems as **discrete event systems** and **timed automata**

• **Discretizations** based on qualitative reasoning: **partition the hybrid state space into set of regions** $p$

Hybrid Automaton ➔ Timeless Abstraction Function ➔ Discrete Event System ➔ Timeless Abstraction Function ➔ Timed Automaton

- Models reachability between regions
  - Do I reach $(Mode 2, p)$ from $(Mode 1, p')$?

- Models time constraints to reach regions
  - Do I reach $(Mode 2, p)$ from $(Mode 1, p')$ in less than $t$ time units?
Timeless and Timed Abstractions

Discrete Event System

Reachability algorithm using safe over-approximations

Flow-Pipe construction

Regions are polyhedra, zonotopes, star sets

Timed Automaton

Do I reach \((\text{Mode } 2, p)\) from \((\text{Mode } 1, p')\) in less than \(t\) time units?

Compute maximum and minimum sojourn time in \((\text{Mode } 1, p')\): \(t_{\text{min}}\) and \(t_{\text{max}}\)

\((\text{Mode } 1, p')\) \[\text{Inv: } t < t_{\text{max}}\]

\(G: t > t_{\text{min}}\)

\((\text{Mode } 2, p)\)
Abstraction Refinement

- **Refining** the abstraction

- **State splitting**: e.g. region cut in half

- **Time Constraints** tightening

- Refinement operation recomputes only local transitions
Abstraction Refinement

- **Refining** the abstraction

- **State splitting**: e.g. region cut in half

- **Time Constraints** tightening

- Refinement operation recomputes only local transitions
Abstraction Refinement

- **Refining** the abstraction
- **State splitting**: e.g., region cut in half
- **Time Constraints** tightening
- Refinement operation recomputes only local transitions
Abstraction Refinement

- **Refining** the abstraction

- **State splitting**: e.g. region cut in half

- **Time Constraints** tightening

- Refinement operation recomputes only local transitions
Abstraction Refinement: refined constraints

Discrete Event System: State Split

\[ p = p_1 \cup p_2 \]

Timed Automaton: Time Constraint tightening

\[ I_p = [t_{min}, t_{max}] \]

\[ I_{p_1} = [t_{min1}, t_{max1}] \quad t_{min1,2} \geq t_{min} \]

\[ I_{p_2} = [t_{min2}, t_{max2}] \quad t_{max1,2} \leq t_{max} \]
Counterexample Guided Abstraction Refinement Loop*


Counterexample Guided Abstraction Refinement Loop*


Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model $H$

Property $P$

$A := A^0$

$A$ verifies $P$?

True, no counter-examples

\textit{return $H$ verifies $P$}


Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model

\( H \)

Property \( P \)

\[ A := A^0 \]

\( A \) verifies \( P \)?

True, no counter-examples

\( \text{return } H \text{ verifies } P \)

---


Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model $H$

Property $P$

$A := A^0$

$A$ verifies $P$?

True, no counter-examples

return $H$ verifies $P$

Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model $H$

Property $P$

$A := A^0$

$A$ verifies $P$?

True, no counter-examples

$\text{return } H \text{ verifies } P$


Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model $H$
Property $P$

$A := A^0$

$A$ verifies $P$?

True, no counter-examples

False, Counter-example found $C.E$

Validate $C.E$

return $H$ verifies $P$

Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model $H$

Property $P$

$A := A^0$

$A$ verifies $P$?

True, no counter-examples

return $H$ verifies $P$

False, Counter-example found $C.E$

Validate $C.E$

True, Counter-Example is Valid

return $H$ does not verify $P$


Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model $H$

Property $P$

$A := A^0$

A verifies $P$?

False, Counter-example found $C.E$

Validate $C.E$

True, no counter-examples

return $H$ verifies $P$

True, Counter-Example is Valid

return $H$ does not verify $P$


Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model $H$

Property $P$

$A := A^0$

A verifies $P$?

True, no counter-examples

return $H$ verifies $P$

False, Counter-example found $C.E$

Validate $C.E$

True, Counter-Example is Valid

return $H$ does not verify $P$


Counterexample Guided Abstraction Refinement Loop

Hybrid Automaton Model \( H \)
Property \( P \)

\[ A := A^0 \]

True, no counter-examples
\[ \text{return } H \text{ verifies } P \]

False, Counter-example found \( C.E \)

\[ A \text{ verifies } P? \]

True, Counter-Example is Valid
\[ \text{return } H \text{ does not verify } P \]

Verify \( C.E \)

\[ A = \text{refine}(A,H,C.E) \]

False, \( C.E \) invalid in \( H \)


Hybrid Automaton Model $H$

Property $P$

$A := A^0$

A verifies $P$?

True, no counter-examples

return $H$ verifies $P$

True, Counter-Example is Valid

return $H$ does not verify $P$

False, Counter-example found $C.E$

Validate $C.E$

validate

False, $C.E$ invalid in $H$

$A = refine(A, H, C.E)\]


Counterexample Guided Abstraction Refinement Loop*

Hybrid Automaton Model $H$

Property $P$

$A := A^0$

$A$ verifies $P$?

True, no counter-examples

return $H$ verifies $P$

True, Counter-Example is Valid

Validate $C.E$

False, Counter-example found $C.E$

False, $C.E$ invalid in $H$

$A = \text{refine}(A,H,C.E)$


CEGAR for diagnosability: counterexample generation

- **Twin Plant Method for Discrete Event Systems and Timed Automata**

  - Compose the system with itself and synchronize observations
  - For discrete event systems: PTIME, for Timed automata: PSPACE
  - Find a critical Path in the Twin Plant

\[
\begin{align*}
(S_0, S_0) & \xrightarrow{a} (S_1, S_1) \xrightarrow{\epsilon} (S_2, S_3) \xrightarrow{b} (S_4, S_5)\\
S_0 & \xrightarrow{a} S_1 \quad S_2 & \xrightarrow{b} S_4 \quad S_3 & \xrightarrow{b} S_5
\end{align*}
\]

CEGAR for diagnosability: counterexample generation

- **Twin Plant Method for Discrete Event Systems and Timed Automata**
  - Compose the system with itself and synchronize observations
  - For discrete event systems: PTIME, for Timed automata: PSPACE *
  - Find a critical Path in the Twin Plant

CEGAR for diagnosability: counterexample generation

- **Twin Plant Method for Discrete Event Systems and Timed Automata**
  - Compose the system with itself and synchronize observations
  - For discrete event systems: PTIME, for Timed automata: PSPACE *
  - Find a critical Path in the Twin Plant

\[ (S_0, S_0) \xrightarrow{a} (S_1, S_1) \xrightarrow{\epsilon} (S_2, S_3) \xrightarrow{b} (S_4, S_5)^* b \]

CEGAR for diagnosability: counterexample generation

- **Twin Plant Method for Discrete Event Systems and Timed Automata**
  - Compose the system with itself and synchronize observations
  - For discrete event systems: PTIME, for Timed automata: PSPACE
  - Find a critical Path in the Twin Plant

**CEGAR for diagnosability: counterexample generation**

- **Twin Plant Method for Discrete Event Systems and Timed Automata**
  - Compose the system with itself and synchronize observations
  - For discrete event systems: PTIME, for Timed automata: PSPACE *
  - Find a critical Path in the Twin Plant

\[
\begin{align*}
S_0 & \xrightarrow{a} S_1 \\
S_1 & \xrightarrow{t \geq 3} S_2 \\
S_2 & \xrightarrow{b} S_4 \\
S_4 & \xrightarrow{b} S_5 \\
S_5 & \xrightarrow{b} S_5
\end{align*}
\]

\[
\begin{align*}
(S_0, S_0) & \xrightarrow{a} (S_1, S_1) \xrightarrow{\epsilon} (S_2, S_3) \xrightarrow{b} (S_4, S_5)^* b \\
(S_0, S_0) & \xrightarrow{a, 0} (S_1, S_1) \xrightarrow{\epsilon} (S_2, S_3) \xrightarrow{b, 3} (S_4, S_5)^* b
\end{align*}
\]

• **Safety:** an abstract trajectory reaching an unsafe state
  
  \[(Mode 1, region 1) \rightarrow (Mode 2, region 3) \rightarrow (Mode 1, region 2)\] and \[(Mode 1, region 2) \cap Unsafes \neq \emptyset\]

• **Diagnosability:** two abstract trajectories, one faulty, one nominal having same observations

• Fault Modeling:
  - We addressed parametric change

• Partially Observable system
  - Mode change
  - Continuous variables

• Diagnosability Property:
  - For all modeled faults and under all executions \(\Rightarrow\) do observations allow fault detection?
CEGAR for diagnosability: validation or refusal of the $C.E$

- **Validating a counterexample** $C.E = (h, h^F)$:
  - Given a critical pair at the abstract level, find a concretization of it
  - Flow-Pipe construction tracing the counterexample*

- **Refusal of $C.E$: partition to avoid the future occurrence of the counter-example**
  - Spurious reachability
  - Spurious timed constraints
  - Spurious observability

CEGAR for diagnosability: validation or refusal of the $C.E$

- **Validating a counterexample $C.E = (h, h^F)$:**
  - Given a critical pair at the abstract level, find a concretization of it
  - Flow-Pipe construction tracing the counterexample*

- **Refusal of $C.E$: partition to avoid the future occurrence of the counter-example**
  - Spurious reachability
  - Spurious timed constraints
  - Spurious observability

CEGAR for diagnosability: validation or refusal of the $C.E$

- **Validating a counterexample $C.E = (h, h^F)$:**
  - Given a critical pair at the abstract level, find a concretization of it
  - Flow-Pipe construction tracing the counterexample*

- **Refusal of $C.E$: partition to avoid the future occurrence of the counter-example**
  - Spurious reachability
  - Spurious timed constraints
  - Spurious observability

CONCLUSION

• **Abstraction method** using **discrete event systems** based on qualitative reasoning

• **Timed abstractions** of hybrid systems using **timed automata**

• Counter-example guided abstraction refinement loop for diagnosability verification
The way forward

- Investigate $t_{\text{min}}, t_{\text{max}}$ during mode jumps

Assume diagnosability is proved and requires knowing:

- Predictability: know the occurrence of an inevitable fault before it happens
- Extend to polynomial hybrid automata
- (Semi)-Automating the refinement process (in some situations)
APPENDIX

• Citation article:
  • “Abstractions for Hybrid Systems” A. Tiwari
  • “Theory of hybrid automata” T. Henzinger
  • “Qualitative Theory of Hybrid Dynamical Systems” S. Matveev

• Tools:

<table>
<thead>
<tr>
<th>Automated theorem prover</th>
<th>KeyMaera</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support function based flowpipe</td>
<td>SpaceEx</td>
</tr>
<tr>
<td>Conservative abstraction - polyhedral</td>
<td>HyTech - PhaVer</td>
</tr>
<tr>
<td>Convex polyhedron</td>
<td>CheckMate</td>
</tr>
<tr>
<td>Interval verified integration</td>
<td>HySat/iSAT</td>
</tr>
</tbody>
</table>
APPENDIX

Qualitative and Geometric Approaches

\[ \dot{x} = 1 - (b + 1)x + ax^2 y \]
\[ \dot{y} = bx - ax^2 y \]

Saddle point: (1; 2.5)
APPENDIX

Qualitative and Geometric Approaches

\[ \dot{x} = 1 - (b + 1)x + ax^2y \]
\[ \dot{y} = bx - ax^2y \]

Calculate x and y Null clines

\[ \dot{x} = 0 \]
\[ \dot{y} = 0 \]

Saddle point: (1; 2.5)
APPENDIX

Qualitative and Geometric Approaches

\[ \dot{x} = 1 - (b + 1)x + ax^2y \]
\[ \dot{y} = bx - ax^2y \]

Calculate x and y Null clines

\[ \dot{x} = 0 \]
\[ \dot{y} = 0 \]

Saddle point: \((1; 2.5)\)

\[ y = \frac{(b + 1)x - 1}{ax^2} \]
Qualitative and Geometric Approaches

\[
\dot{x} = 1 - (b + 1)x + ax^2y \\
\dot{y} = bx - ax^2y
\]

Calculate x and y Null clines

\[
\dot{x} = 0 \\
\dot{y} = 0
\]

Saddle point: \((1; 2.5)\)
APPENDIX

Qualitative and Geometric Approaches

\[ \dot{x} = 1 - (b + 1)x + ax^2y \]
\[ \dot{y} = bx - ax^2y \]

Calculate x and y Null clines

\[ \dot{x} = 0 \]
\[ \dot{y} = 0 \]

\[ y = \frac{(b + 1)x - 1}{ax^2} \]
\[ x = 0 \quad \text{or} \quad y = \frac{b}{ax} \]

Saddle point: (1; 2.5)
APPENDIX

Qualitative and Geometric Approaches

\[ \dot{x} = 1 - (b + 1)x + ax^2y \]
\[ \dot{y} = bx - ax^2y \]

Calculate x and y Null clines

\[ \dot{x} = 0 \quad \text{or} \quad y = \frac{b}{ax} \]
\[ \dot{y} = 0 \quad \text{or} \quad y = \frac{(b+1)x - 1}{ax^2} \]

Saddle point: (1; 2.5)
APPENDIX

Qualitative and Geometric Approaches

\[ \dot{x} = 1 - (b + 1)x + ax^2y \]
\[ \dot{y} = bx - ax^2y \]

Calculate \( x \) and \( y \) Null clines

\[ \dot{x} = 0 \]
\[ \dot{y} = 0 \]

Qualitative regions can now be determined

Saddle point: (1; 2.5)
PERSPECTIVES

- Perspectives: Development of the method using:
  - SAT modulo ODE (or SMT) solvers
  - QepCAD for polynomial constraint solving
  - Automatic generation of qualitative abstraction model