A Counterexample-Guided Approach to Symbolic Simulation of Hybrid Systems

Xian Li, Klaus Schneider

Embedded Systems Group
Department of Computer Science
University of Kaiserslautern, Germany

MBMV 2015, Chemnitz, Germany, March 2015
# Table of Contents

1. Introduction

2. The Satisfiability Problem

3. The prototypical Constraint Solver

4. The symbolic Simulation Algorithm

5. Summary
Outline

1. Introduction
2. The Satisfiability Problem
3. The prototypical Constraint Solver
4. The symbolic Simulation Algorithm
5. Summary
Motivation

Hybrid Systems

appropriate mathematical models for embedded control systems

Parametric Analysis of Hybrid Systems

derive constraints on parameters (a fixed subset of the variables) required to meet the specification
Motivation

Hybrid Systems
appropriate mathematical models for embedded control systems

Parametric Analysis of Hybrid Systems
derive constraints on parameters (a fixed subset of the variables) required to meet the specification

Challenge
- the parameterized models supporting reals, integers, booleans, and affine dynamics
Motivation

**Hybrid Systems**
appropriate mathematical models for embedded control systems

**Parametric Analysis of Hybrid Systems**
derive constraints on parameters (a fixed subset of the variables) required to meet the specification

**Challenge**
- the parameterized models supporting reals, integers, booleans, and affine dynamics
- automated analysis of this kind of models leads/corresponds to an *undecidable* satisfiability problem
Contributions

A new prototypical Constraint Solver

Integrate the external tool Bonmin \(^a\) into the BDD package implemented in the Averest system \(^b\).

\(^a\) [BBCC08]
\(^b\) http://www.averest.org/
Contributions

A new prototypical Constraint Solver
Integrate the external tool Bonmin $^a$ into the BDD package implemented in the Averest system $^b$.

\[ [BBCC08] \]

\[ \text{http://www.averest.org/} \]

A Symbolic Simulation Algorithm
Compute \textit{ranges} of the input parameters:
extends each \textit{concrete value} to a \textit{range constraint} until some parameter valuation violates the specification.
Outline

1. Introduction

2. The Satisfiability Problem

3. The prototypical Constraint Solver

4. The symbolic Simulation Algorithm

5. Summary
Syntax

Boolean and numerical expressions:

\[ e_b := x \in \mathcal{V}_B \mid \neg e_b \mid e_b \land e_b \mid e_b \lor e_b \]

\[ e := x \in \mathcal{V}_R \cup \mathcal{V}_Z \mid e + e \mid e - e \mid e \cdot e \mid e/e \]

Constraints:

\[ c := e \circ e' \mid c \land c \text{ where } \circ \in \{\leq, =\} \]

Examples:

\[ \mathcal{V}_R = \{r_1, r_2\}, \, \mathcal{V}_Z = \{z_1, z_2\}, \, \mathcal{V}_B = \{b_1, b_2\} \]

\[ E_0 : b_1 \land b_2 \]

\[ E_1 : r_1 \leq r_2 + z_1 \ast (r_1 - z_2) \]
Syntax

Boolean and numerical expressions:

\[
e_b := x \in \mathcal{V}_B \mid \neg e_b \mid e_b \land e_b \mid e_b \lor e_b \\
e := x \in \mathcal{V}_R \cup \mathcal{V}_Z \mid e + e \mid e - e \mid e \cdot e \mid e/e
\]

Constraints:

\[
c := e \circ e' \mid c \land c \quad \text{where} \quad \circ \in \{\leq, =\}
\]

Examples:

\[
\mathcal{V}_R = \{r_1, r_2\}, \quad \mathcal{V}_Z = \{z_1, z_2\}, \quad \mathcal{V}_B = \{b_1, b_2\} \\
E_0 : b_1 \land b_2 \\
E_1 : r_1 \leq r_2 + z_1 \ast (r_1 - z_2)
\]

Boolean expressions and constraints with \(\exists\)-quantifiers:

\[
(e_b)_Q := (\exists X_B). \ e_b \\
c_Q := (\exists X_R, X_Z). \ c
\]

where \(X_B \subseteq \mathcal{V}_B\), \(X_R \subseteq \mathcal{V}_R\) and \(X_Z \subseteq \mathcal{V}_Z\) are the quantifier sets.

\[
E_2 : (\exists \mathcal{V}_B). \ b_1 \land b_2 \\
E_3 : (\exists \mathcal{V}_R, \mathcal{V}_Z). \ (r_1 \leq r_2 + z_1 \ast (r_1 - z_2))
\]
Syntax

Boolean expressions and constraints with $\exists$-quantifiers:

$$(e_b)_Q := (\exists X_B). \ e_b$$
$$c_Q := (\exists X_R, X_Z). \ c$$

where $X_B \subseteq V_B$, $X_R \subseteq V_R$ and $X_Z \subseteq V_Z$ are the quantifier sets.

$$E_2 : (\exists V_B). \ b_1 \land b_2 \quad E_3 : (\exists V_R, V_Z). \ (r_1 \leq r_2 + z_1 \ast (r_1 - z_2))$$

The satisfiability problem:

$$C_b := \bigvee_{i \in \mathbb{N}} \ ((c_Q)_i \land ((e_b)_Q)_i)$$
**Syntax**

*Boolean expressions and constraints with $\exists$-quantifiers:*

\[
(e_b)_Q := (\exists X_B). e_b \\
c_Q := (\exists X_R, X_Z). c
\]

where $X_B \subseteq V_B$, $X_R \subseteq V_R$ and $X_Z \subseteq V_Z$ are the quantifier sets.

\[
E_2 : (\exists V_B). b_1 \land b_2 \\
E_3 : (\exists V_R, V_Z). (r_1 \leq r_2 + z_1 \ast (r_1 - z_2))
\]

The satisfiability problem:

\[
C_b := \bigvee_{i \in \mathbb{N}} ((c_Q)_i \land ((e_b)_Q)_i)
\]

\[
E_4 : E_3 \land E_2
\]
Decidability and Tools

\[ C_b := \bigvee_{i \in \mathbb{N}} \left( (c_Q)_i \land ((e_b)_Q)_i \right) \]

\((c_Q)_i\) : constraints with \(\exists\)-quantifiers

\(((e_b)_Q)_i\) : boolean expressions with \(\exists\)-quantifiers

An SMT problem: boolean combinations of propositional logic atoms and atoms of non-linear arithmetic theories over integers and reals with \(\exists\)-quantifiers.
Decidability and Tools

\[ C_b := \bigvee_{i \in \mathbb{N}} ((c_Q)_i \land ((e_b)_Q)_i) \]

\((c_Q)_i\) : constraints with \(\exists\)-quantifiers
\(((e_b)_Q)_i\) : boolean expressions with \(\exists\)-quantifiers

An SMT problem: boolean combinations of propositional logic atoms and atoms of non-linear arithmetic theories over integers and reals with \(\exists\)-quantifiers.

Which tool performs best for the satisfiability problem?
Decidability and Tools

\[ C_b := \bigvee_{i \in \mathbb{N}} ((c_Q)_i \land ((e_b)_Q)_i) \]

\((c_Q)_i\) : constraints with \(\exists\)-quantifiers

\(((e_b)_Q)_i\) : boolean expressions with \(\exists\)-quantifiers

An SMT problem: boolean combinations of propositional logic atoms and atoms of non-linear arithmetic theories over integers and reals with \(\exists\)-quantifiers.

Which tool performs best for the satisfiability problem?
Decidability and Tools

\[ C_b' := \bigvee_{i \in \mathbb{N}} (c_Q)_i \]

\((c_Q)_i : \) constraints with \(\exists\)-quantifiers

Is there a solution for a set of Mixed-Integer Nonlinear Program (MINLP) problems, where each MINLP problem corresponds to a subformula \(c_{Q_i}\)?
The general form of a MINLP:

\[
\begin{align*}
\text{minimize} & \quad f(\vec{x}, \vec{y}) \\
\text{subject to} & \quad g(\vec{x}, \vec{y}) \leq 0 \\
& \quad \vec{x}_l \leq \vec{x} \leq \vec{x}_u, \quad x_i \in \mathbb{R} \\
& \quad \vec{y}_l \leq \vec{y} \leq \vec{y}_u, \quad y_i \in \mathbb{Z}
\end{align*}
\]

\( f(\vec{x}, \vec{y}), \ g(\vec{x}, \vec{y}) \): nonlinear functions

e.g. \( g(x) = x^2 + x \)
The general form of a MILP:

\[
\begin{align*}
\text{minimize} & \quad f(\vec{x}, \vec{y}) \\
\text{subject to} & \quad g(\vec{x}, \vec{y}) \leq 0 \\
& \quad \vec{x}_l \leq \vec{x} \leq \vec{x}_u, \quad x_i \in \mathbb{R} \\
& \quad \vec{y}_l \leq \vec{y} \leq \vec{y}_u, \quad y_i \in \mathbb{Z}
\end{align*}
\]

\( f(\vec{x}, \vec{y}), g(\vec{x}, \vec{y}) \): linear functions

e.g. \( g(x) = ax + b \)
The Constraint Problem Perspective

The general form of a **NLP**:

\[
\begin{align*}
\text{minimize} & \quad f(\vec{x}) \\
\text{subject to} & \quad g(\vec{x}) \leq 0 \\
& \quad \vec{x}_l \leq \vec{x} \leq \vec{x}_u \\
& \quad x_i \in \mathbb{R}
\end{align*}
\]

\[f(\vec{x}, \vec{y}), g(\vec{x}, \vec{y}): \text{nonlinear functions}\]

\[\text{e.g. } g(x) = x^2 + x\]
The general form of a MINLP:

minimize \( f(\vec{x}, \vec{y}) \)
subject to \( g(\vec{x}, \vec{y}) \leq 0 \)
\( \vec{x}_l \leq \vec{x} \leq \vec{x}_u \quad x_i \in \mathbb{R} \)
\( \vec{y}_l \leq \vec{y} \leq \vec{y}_u \quad y_i \in \mathbb{Z} \)

\( f(\vec{x}, \vec{y}), g(\vec{x}, \vec{y}) \): nonlinear functions
e.g. \( g(x) = x^2 + x \)
The Constraint Problem Perspective

- LP
  - HySAT, BACH
- Real
  - IMITATOR, KeYmaera, SpaceEx, TReX, MetiTarski, QEPCAD, Redlog, Reduce
- Linear
  - MILP
    - MathSAT5, CVC4, Z3
- Non-linear
  - MINLP
    - iSAT, Bonmin
- Mixed Integer
The Constraint Problem Perspective

Could we use one or some of them as components in the context of other tools for parametric analysis of hybrid systems?

\begin{itemize}
\item \textbf{LP}  
HysAT, BACH
\item \textbf{Real}  
IMITATOR, KeYmaera, SpaceEx, TRex, MetiTarski, QEPCAD, Redlog, Reduce
\item \textbf{Linear}  
\item \textbf{Non-linear}  
\item \textbf{MILP}  
MathSAT5, CVC4, Z3
\item \textbf{MINLP}  
iSAT, Bonmin
\item \textbf{Mixed Integer}
\end{itemize}
Outline

1. Introduction
2. The Satisfiability Problem
3. The prototypical Constraint Solver
4. The symbolic Simulation Algorithm
5. Summary
Integrate the external tool Bonmin into the BDD package implemented in the Averest system

\[ C_b := \bigvee_{i \in \mathbb{N}} ((cQ)_i \land ((e_b)Q)_i) \]

\((cQ)_i\) : constraints with \(\exists\)-quantifiers
\(((e_b)Q)_i\) : boolean expressions with \(\exists\)-quantifiers
The prototypical Constraint Solver

Integrate the external tool Bonmin into the BDD package implemented in the Averest system

\[ C_b := \bigvee_{i \in \mathbb{N}} ((c_Q)_i \land ((e_b)_Q)_i) \]

\((c_Q)_i\) : constraints with \(\exists\)-quantifiers

\(((e_b)_Q)_i\) : boolean expressions with \(\exists\)-quantifiers

Each \((c_Q)_i\) can be processed as a MINLP problem
The prototypical Constraint Solver

Integrate the external tool Bonmin into the BDD package implemented in the Averest system

\[ C_b := \bigvee_{i \in \mathbb{N}} ((c_Q)_i \land ((e_b)_Q)_i) \]

\((c_Q)_i\) : constraints with \(\exists\)-quantifiers

\(((e_b)_Q)_i\) : boolean expressions with \(\exists\)-quantifiers

Each \((c_Q)_i\) can be processed as a MINLP problem
The prototypical Constraint Solver

Integrate the external tool **Bonmin** into the BDD package implemented in the *Averest* system

\[ C_b := \bigvee_{i \in \mathbb{N}} \left( (c_Q)_i \land ((e_b)_Q)_i \right) \]

\((c_Q)_i\) : constraints with \(\exists\)-quantifiers
\(((e_b)_Q)_i\) : boolean expressions with \(\exists\)-quantifiers

**Problem:** Bonmin cannot always find a MINLP problem solution
The prototypical Constraint Solver

Integrate the external tool Bonmin into the BDD package implemented in the Averest system

$$C_b := \bigvee_{i \in \mathbb{N}} ((c_Q)_i \land ((e_b)_Q)_i)$$

$$(c_Q)_i : \text{constraints with } \exists\text{-quantifiers}$$

$$(e_b)_Q)_i : \text{boolean expressions with } \exists\text{-quantifiers}$$

**Problem**: Bonmin cannot always find a MINLP problem solution

**Solution**: Extend the interpretation $\llbracket \cdot \rrbracket_\nu$ by the $\llbracket \cdot \rrbracket_3$
3-Valued Evaluation

\[ [(\exists X_B) \cdot e_b]_3 := [(\exists X_B) \cdot e_b]_v \]

\[ [(\exists X_R, X_Z) \cdot c]_3 := \begin{cases} 
    True, & \text{if Bonmin could obtain a solution of} \\
    & \text{the MINLP problem for } c \\
    Unknown, & \text{Otherwise}
\end{cases} \]

Table 1: Truth Tables for 3-Valued Logic

<table>
<thead>
<tr>
<th>∧₁</th>
<th>T</th>
<th>F</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>U</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>U</td>
<td>U</td>
<td>F</td>
<td>U</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>∨₁</th>
<th>T</th>
<th>F</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>U</td>
</tr>
<tr>
<td>U</td>
<td>T</td>
<td>U</td>
<td>U</td>
</tr>
</tbody>
</table>

Example: \( C = (c_Q)_1 \land ((e_b)_Q)_1 \lor (c_Q)_2 \land ((e_b)_Q)_2 \)

\((c_Q)_1, ((e_b)_Q)_1, (c_Q)_2, \) and \(((e_b)_Q)_2 \) are assigned to \( Unknown, True, Unknown, \) and \( False. \)

\[ [(C]_3 = Unknown \]
Implementation Details

- implemented as F# functions
- the dual-rail representation
- the correctness and the capability
Outline

1. Introduction
2. The Satisfiability Problem
3. The prototypical Constraint Solver
4. The symbolic Simulation Algorithm
5. Summary
The Symbolic Simulation Algorithm

- **Input:**
  - system’s symbolic representation
  - specification
  - finite concrete values of the parameters
  - iteration length

- **Outputs:**
  - range $\Delta_f$: contains parameter valuation(s) violating the specification
  - range $\Delta_u$: no parameter valuation violating the specification has been found yet

- **General idea:**
  widen each concrete value at each iteration step until some parameter valuation violates the specification
Experimental Evaluation

(a) The Ball and Holes Scenario

(b) Coverage for 3 Initial States

Figure 1: Experimental Scenario and Results
Outline

1 Introduction

2 The Satisfiability Problem

3 The prototypical Constraint Solver

4 The symbolic Simulation Algorithm

5 Summary
Summary

A new prototypical Constraint Solver
Integrate the external tool Bonmin into the BDD package implemented in the Averest system.

A Symbolic Simulation Algorithm
It computes ranges of the input parameters: extends each concrete value to a range constraint until some parameter valuation violates the specifications.
Summary

A new prototypical Constraint Solver
Integrate the external tool Bonmin into the BDD package implemented in the Averest system.

A Symbolic Simulation Algorithm
It computes ranges of the input parameters: extends each concrete value to a range constraint until some parameter valuation violates the specifications.

Thank you for your attention!
Implementation Details

- implemented on top of the Averest system for Quartz programs
- benefit from the Averest system

guarded actions $\rightarrow$ symbolic representations
- perform symbolic simulation on different hybrid models