Runtime Verification of Hybrid Systems with Affine Arithmetic Decision Diagrams

Hagen Heermann and Christoph Grimm

Technical University of Kaiserslautern (RPTU)

March 2023

Table of Contents

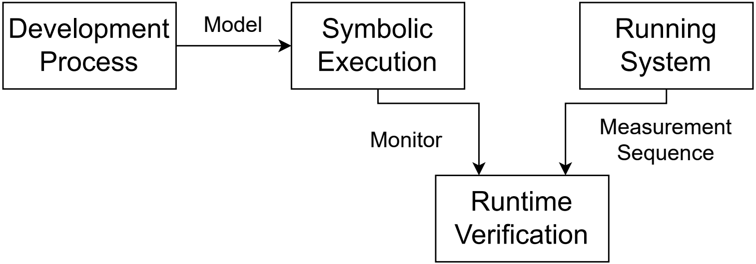
Goals and Motivation

- **Problem:** Complexity of hybrid systems lead to many **unforeseen errors**!
- **Approach:** Checking against the **expected modeled behaviour** instead of an potential **incomplete** list of **failure modes**.
- **State of the Art**: **STL properties** used to define **monitors** to check for **failure modes**
- **State of the Art**: ModelPlex generates monitors using **Hybrid Programs** and **Theorem Prover** (KeYmaera)

Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals.

Stefan Mitsch and André Platzer. Modelplex: Verified runtime validation of verified cyber-physical system models.

General Structure



Solution Sketch

- **Symbolic execution** of the **model** with **Affine Arithmetic Decision Diagrams** results in a compact over-approximation of the **possible trajectories**.

- We use the information represented in **Affine Arithmetic Decision Diagrams** to show that the **measured trajectories** are contained in the results of the **symbolic execution**.

- This is achieved by **translating** the **Affine Arithmetic Decision Diagrams** together with the **measured trajectories** into a **system of linear inequalities**.

Affine Forms

Definition (Affine Form)

$$\tilde{x} = c + \sum_i a_i \epsilon_i$$

- $c \in \mathbb{R}$ being called the **center value.**
- $a_i \in \mathbb{R}$ are called the **partial deviations.**
- Unknown real variables $\epsilon_i \in [-1, 1]$ called **noise symbols.**

Marcus Vinicius Alvim Andrade, Joao Luiz Dihl Comba, and Jorge Stolfi. Affine arithmetic.

Affine Arithmetic Decision Diagrams

Definition (AADD)

An AADD $\hat{x}$ is a DAG with internal nodes Q, leaves T, edges E, conditions $\mathbb{X}$, and it holds:

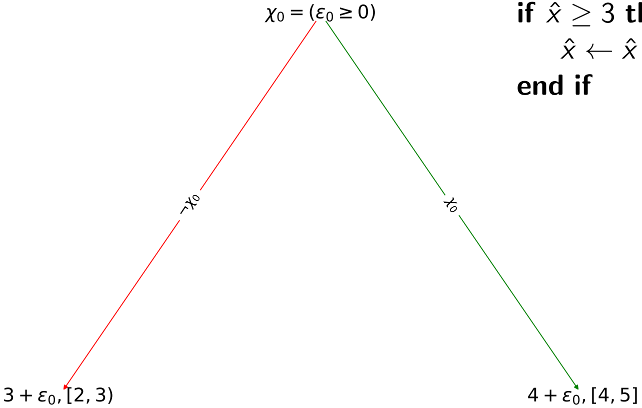
- Internal nodes $v \in Q$ have two leaving edges $e_0, e_1 \in E$ that lead to child nodes $0(v), 1(v) \in T \cup Q$ and are labeled with $\text{index}(v) \in \mathbb{N}$.
- AADD are ordered: For $(v_i, v_j) \in E$ from v_i to v_j: $\text{index}(v_i) < \text{index}(v_j)$.
- Leaves $v \in T$ are labeled with an affine form $\tilde{v}$.
- Conditions $\chi_i \in \mathbb{X}$ are of type $\tilde{x} \geq 0$, where $\tilde{x}$ is an affine form. Each $\text{index}(v) = i, v \in Q$ refers to a unique condition $\chi_i \in \mathbb{X}$ with the same index. The conditions $\mathbb{X}$ are the same in all AADD.

Carna Zivkovic et al., Hierarchical verification of AMS systems with Affine Arithmetic Decision Diagrams.

Affine Arithmetic Decision Diagrams

Example

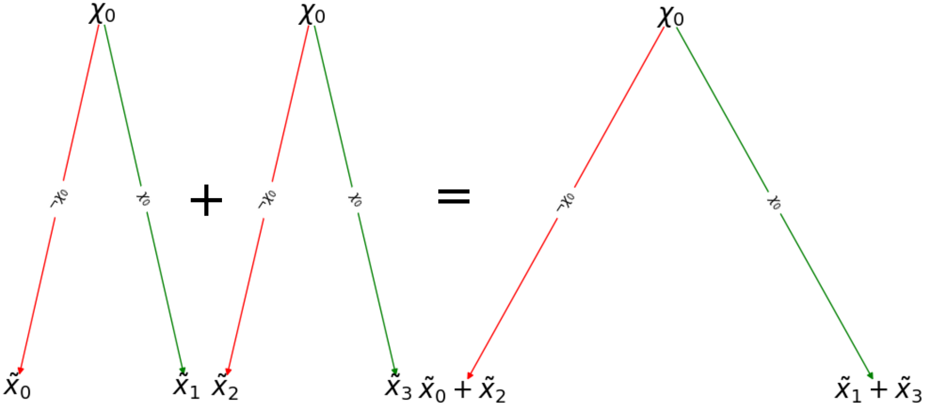
AADDs are created during the **control flow** execution of programs.



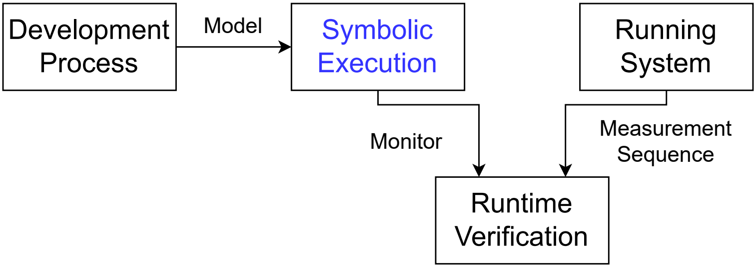
$\hat{x} \leftarrow 3 + \epsilon_0$
if $\hat{x} \geq 3$ **then**
  $\hat{x} \leftarrow \hat{x} + 1$
end if

Affine Arithmetic Decision Diagrams

We implemented arithmetic operations over AADDs that also take the control flow into consideration (**analogous to the Apply Operation for BDD**).



Symbolic Execution



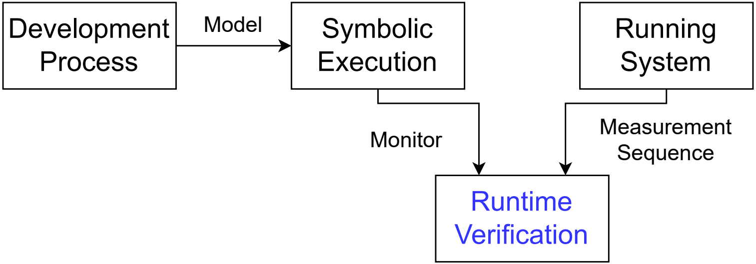
Symbolic Execution

- **Input:**
  - Parameters $\vec{p}$ modeled by affine forms, **unknown parameters**.
  - Variables $\vec{x}$ modeled initially as AADD leafs, **unknown starting states**.
- Run Symbolic Execution for the desired simulated time.
- **Output:**
  - For every variable in $\vec{x}$ we get a Signal Set $S_{i,T} = \langle \hat{s}_0, \hat{s}_1, \hat{s}_2, \ldots \rangle, i \in 1 \ldots n$.
  - All $\hat{s}_j$ from $S_{i,T}$ are AADDs.

Symbolic Execution

- We use the Signal Sets $S_{i,T} = \langle \hat{s_0}, \hat{s_1}, \hat{s_2}, \ldots \rangle$, $i \in 1 \ldots n$ in our runtime verification approach as a **monitor** due to the following properties:
  - **The leafs of the AADDs model the potential value ranges of the corresponding variable at the specific point in time**.
  - **The constraints of the internal nodes are modelling the control flow that is required to reach the specific leaf**.

Verification Algorithm



Verification Algorithm

Input for Verification Algorithm

1. **Signal Sets (Monitor)**: $S_{i,T} = \langle \hat{s}_0, \hat{s}_1, \hat{s}_2, \ldots \rangle, i \in 1 \ldots n$
2. **Measurement Sequence**: $M_{i,T} = \langle m_0, m_1, m_2, \ldots \rangle, m_j \in \mathbb{R}$
3. **Error Tolerance**: $\Delta \in \mathbb{R}$

Verification Algorithm

- Let $\mathbb{E}$ be the set of all **noise symbols** that are used in the AADDs of the Signal set as well as from the affine forms of $\vec{p}$.
- **Goal**: The goal of the verification algorithm is to show that there exists an assignment of all $\epsilon \in \mathbb{E}$ such that all the AADDs in $S_{i,T}$ evaluate to their corresponding values in $M_{i,T}$, $+$-Δ.
- **Idea**: Transform the question of the existence of such an assignment into a linear inequality equation system and try to find a solution using a Linear Programming solver.

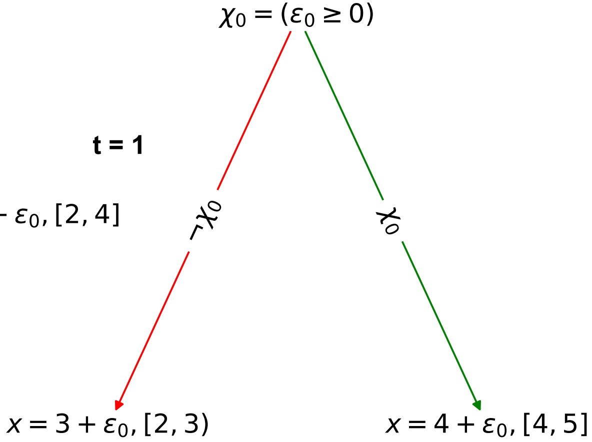
Verification Algorithm

- The linear inequality equation system that we are creating in the algorithm consists of:
  1. Inequality equations of the control flow path (**Internal Nodes**).
  2. Inequality equations that are a result of checking if the measurement value is contained in the value range of the variable (**Leaf Nodes**).
  3. Inequality equations of the noise symbols constraining them to the range $[-1, 1]$ (**Affine Arithmetic**).
- **We don't need to consider every control flow path**!
- If one of the linear inequality systems of the potential control flow paths has a solution, then there exists an assignment of $\mathbb{E}$ under which the $S_{i,T}$ evaluate to our $M_{i,T}$.

Verification Algorithm

t = 0 **t = 1**

$x = 3 + \varepsilon_0, [2, 4]$



Verification Algorithm

- Measurement Sequence $M = \langle 3.5, 4.5 \rangle$, Error Tolerance $\Delta = 0.1$
- Two possible control flow paths lead to two inequality equation systems:

1.

$$
\begin{aligned}
\epsilon_0 \geq 0, \\
3 + \epsilon_0 \geq 3.5 - \Delta, 3 + \epsilon_0 \leq 3.5 + \Delta, \\
\boxed{4 + \epsilon_0 \geq 4.5 - \Delta, 4 + \epsilon_0 \leq 4.5 + \Delta}, \\
\epsilon_0 \geq -1, \epsilon_0 \leq 1
\end{aligned}
$$

2.

$$
\begin{aligned}
\epsilon_0 < 0, \\
3 + \epsilon_0 \geq 3.5 - \Delta, 3 + \epsilon_0 \leq 3.5 + \Delta, \\
\boxed{3 + \epsilon_0 \geq 4.5 - \Delta, 3 + \epsilon_0 \leq 4.5 - \Delta}, \\
\epsilon_0 \geq -1, \epsilon_0 \leq 1
\end{aligned}
$$

Verification Algorithm

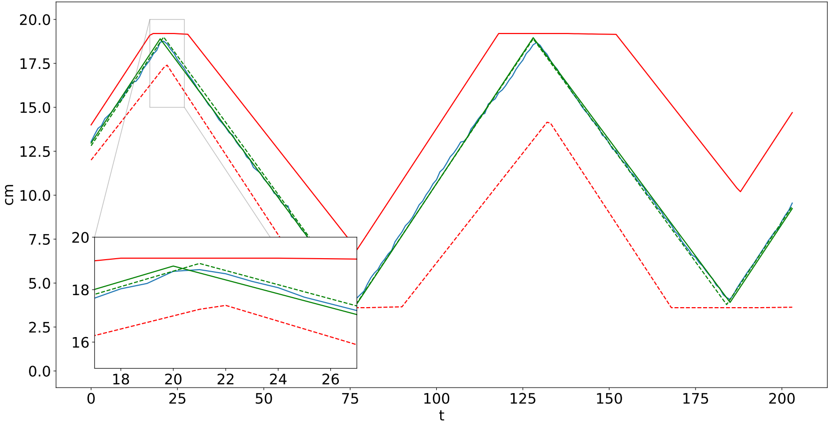
- Since the signal sets are an **over-approximation** of the behaviour, **false positives can result**.
- If we **can't find an assignment for any possible control path** then **we can be certain that the measured behaviour does not correspond to the modeled behaviour**.

Results: Water Tank

- Water tank with two pumps.
- Water can be pumped in or out of the tank.
- Parameters: Flow rate ($[0.043, 0.051]\frac{cm^3}{s}$ modelled by $0.047 + 0.004\epsilon_0$).
- State variables: Water height (initial $[12.0, 13.0]$ cm modelled by AADD Leaf $12.5 + 0.5\epsilon_1$).
- $\Delta = 0.4$
- Measurement sequence from experiment on a real system.

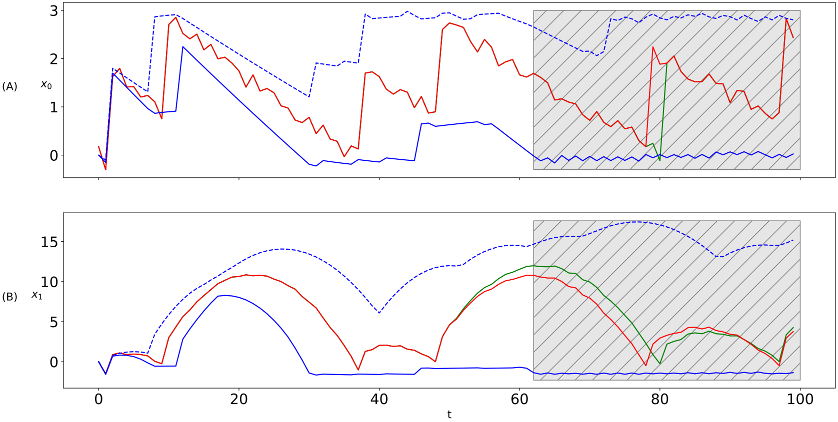
Results: Water Tank

Positive Verification Result



Results: Σ-Δ-Modulator

Parametric Error Detection



Outlook

- Change of the simulation framework used for the symbolic execution to SystemC AMS.
- Adding of further constraints into the signal sets.
- Implementation and evaluation for real time use.