Proving and Explaining the Unfeasibility of Message Sequence Charts for Hybrid Systems

Alessandro Cimatti  Sergio Mover  Stefano Tonetta

Fondazione Bruno Kessler

October 31, 2011
Motivations

Hybrid Systems

- Mix discrete (e.g. hardware) and continuous (e.g. sensor) behaviors.
- Complex critical systems: train control system (ETCS), airplane traffic control system (TCAS), ...
- Network of components.

Scenario-verification

Is there a run of the system compatible with the scenario?
If such a run exists, the scenario is feasible.
Motivations

Existing approaches:

1. **Reduction to reachability:**
   - Can prove both feasibility and unfeasibility.
   - Inefficient.

2. **Scenario-based encoding [CAV11]:**
   - Cannot prove unfeasibility.
   - Efficient.

Our contribution is a **SMT-based** technique that:

- Efficiently proves **unfeasibility**.
- Extracts **explanations** for the unfeasibility.
1. Background
   - SMT analysis of Hybrid Systems
   - Scenario-Verification

2. Proving the unfeasibility of scenarios

3. Explanations of Unfeasibility

4. Experimental Evaluation

5. Conclusions and future work
Outline

1 Background
   • SMT analysis of Hybrid Systems
   • Scenario-Verification

2 Proving the unfeasibility of scenarios

3 Explanations of Unfeasibility

4 Experimental Evaluation

5 Conclusions and future work
Outline

1. Background
   - SMT analysis of Hybrid Systems
   - Scenario-Verification

2. Proving the unfeasibility of scenarios

3. Explanations of Unfeasibility

4. Experimental Evaluation

5. Conclusions and future work
Hybrid automata ([Henzinger 96]):

- Framework for representing hybrid systems.
- **Discrete** instantaneous mode switches.
- **Continuous** evolution according to flow conditions.
Network of hybrid automata $\mathcal{H} = H_1 \| \ldots \| H_n$:

- Move **asynchronously** on local events ($\tau$).
- Synchronize on shared events.

Different semantics:

1. **Global-time** ([Henzinger 96]).
2. **Local-time** ([Bengtsson 98]).
Local-time semantics

The time evolves **independently** in each automaton:
- Local time scale.
- The continuous evolution is a **local transition**.

The local time of the automata must be the same:
- On synchronizations.
- At the end of a run.

\[ \tau = \text{local event (no stutter or time)}. \]
SMT analysis of Hybrid Systems

- Each automaton is encoded in a symbolic transition system
  \( H_i = \langle \text{Init}_i, \text{Trans}_i \rangle \).
- Bounded model checking:

  \[ \text{BMC}_{H_1}(k) \]
  \[ \begin{array}{cccccc}
  1 & \rightarrow & 2 & \rightarrow & 3 & \rightarrow & \cdots & \rightarrow & k \\
  \end{array} \]

  \[ \text{BMC}_{H_2}(k) \]
  \[ \begin{array}{cccccc}
  1 & \rightarrow & 2 & \rightarrow & 3 & \rightarrow & \cdots & \rightarrow & k \\
  \end{array} \]

- \( k \)-induction.
  - Base case: BMC up to \( k \).
  - Inductive case: BMC and simple path condition up to \( k + 1 \).

- Use SMT solvers as decision procedure.
Outline

1. **Background**
   - SMT analysis of Hybrid Systems
   - Scenario-Verification

2. Proving the unfeasibility of scenarios

3. Explanations of Unfeasibility

4. Experimental Evaluation

5. Conclusions and future work
\[ \langle m, \phi \rangle \]: Message sequence chart \( m \) with constraints \( \phi \).

\( m \): parallel composition of instances.

\( \phi = \phi_g \land \phi_1 \land \ldots \land \phi_n \): formulas over the network variables on synchronization.

- Global \( \phi_g \): over all the network variables.
- Local \( \phi_i \): over variable of \( H_i \).

\[ \begin{align*}
\text{Rod}_1 & \quad \text{Controller} & \quad \text{Rod}_2 \\
& \quad \text{Add}_1 & \quad \text{Add}_2 \\
\text{time} \leq 19 & \quad \text{Rem}_1 & \quad \text{time} \geq 19 \\
& \quad \text{Add}_1 & \\
& \quad \text{Rem}_1 & \quad \text{Rem}_2 \\
\text{time} \geq 80 & \\
\end{align*} \]
MSC verification via reachability

- The CMSC is translated in a monitor automaton $S_m$.
- The automaton is composed with the network.
- Enables off-the-shelf verification techniques:
  - BMC: feasibility.
  - k-induction: unfeasibility.

\[ m = \sigma_1 || \sigma_2 || \sigma_3 || \sigma_4 \]
The CMSC is translated in a monitor automaton $S_m$. The automaton is composed with the network. Enables off-the-shelf verification techniques:
- BMC: feasibility.
- k-induction: unfeasibility.

Cut: $\langle l_1^0, l_2^0, l_3^0, l_4^0 \rangle$
MSC verification via reachability

- The CMSC is translated in a monitor automaton $S_m$.
- The automaton is composed with the network.
- Enables off-the-shelf verification techniques:
  - BMC: feasibility.
  - k-induction: unfeasibility.

Cut: $\langle l^1_1, l^1_2, l^0_3, l^0_4 \rangle$
MSC verification via reachability

- The CMSC is translated in a monitor automaton $S_m$.
- The automaton is composed with the network.
- Enables off-the-shelf verification techniques:
  - BMC: feasibility.
  - $k$-induction: unfeasibility.

![Diagram of automaton]

Cut: $\langle l_1^1, l_2^1, l_3^1, l_4^1 \rangle$
The CMSC is translated in a monitor automaton $S_m$. The automaton is composed with the network. Enables off-the-shelf verification techniques:
- BMC: feasibility.
- k-induction: unfeasibility.

Cut: $\langle l_1^1, l_2^1, l_3^2, l_4^2 \rangle$
MSC verification via reachability

- The CMSC is translated in a monitor automaton $S_m$.
- The automaton is composed with the network.
- Enables off-the-shelf verification techniques:
  - BMC: feasibility.
  - k-induction: unfeasibility.

Cut: $\langle l^1_1, l^3_2, l^3_3, l^2_4 \rangle$
Scenario-based encoding

For all the automata:
- Fix the position of the shared events.
  transitions are simplified wrt shared event

Add
Rem
Add
Rem
Add
Rem
Add

Sergio Mover (FBK)
Unfeasibility and Explanations of MSC
October 31, 2011 14 / 28
Scenario-based encoding

- For all the automata:
  - Fix the position of the shared events.
  - Transitions are simplified wrt shared event
  - Add the synchronization constraints.

Add $1$

Rem $1$

Add $2$

Rod$_1$

Controller

Rod$_2$
Scenario-based encoding

- For all the automata:
  - Fix the position of the shared events.
  - Transitions are simplified wrt shared event.
  - Add the synchronization constraints.
  - Encode the “local segments”.
  - Transitions are simplified wrt $\tau$.

[Diagram showing transitions and events labeled with $\tau$ and arrows indicating additions and removals.]
Outline

1. Background
   - SMT analysis of Hybrid Systems
   - Scenario-Verification

2. Proving the unfeasibility of scenarios

3. Explanations of Unfeasibility

4. Experimental Evaluation

5. Conclusions and future work
<table>
<thead>
<tr>
<th></th>
<th>Reduction to reachability</th>
<th>SMT-based approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Feasibility</strong></td>
<td>BMC</td>
<td>Scenario-driven encoding</td>
</tr>
<tr>
<td></td>
<td>Inefficient</td>
<td>Efficient</td>
</tr>
<tr>
<td><strong>Unfeasibility</strong></td>
<td>K-induction</td>
<td>Partitioned k-induction</td>
</tr>
<tr>
<td></td>
<td>Inefficient</td>
<td>Efficient</td>
</tr>
</tbody>
</table>
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

Unfeasible iff UNSAT.
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.

Unfeasible iff UNSAT.
**Partitioned K-induction - Algorithm**

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

\[
\begin{array}{c}
\text{SAT - new states are reachable}\\
\text{simple path}
\end{array}
\]
Inductive step: proved incrementally following the partial order of the MSC.

Base case: bounded feasibility check.
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

SAT - new states are reachable

simple path

Unfeasible iff UNSAT
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

```
Rod1
  ↓
Add1
  ↓
Rem1     Add2
  ↓
Rod2
```

Unfeasible iff UNSAT

```
τ
```

Simple path

UNSAT - no new states are reachable
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.
Inductive step: proved incrementally following the partial order of the MSC.

Base case: bounded feasibility check.
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

\[ \text{Add} \]

\[ \text{Rem} \]

\[ \text{Add}_1 \]

\[ \text{Rem}_1 \]

\[ \text{Add}_2 \]

\[ \text{SAT} - \text{new states are reachable} \]

\[ \text{simple path} \]
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

---

![Diagram of MSC]
Inductive step: proved incrementally following the partial order of the MSC.

Base case: bounded feasibility check.

SAT - new states are reachable

Simple path
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

![Diagram of MSC with transitions Add, Rem, and Controller nodes between Rod1 and Rod2]

Unfeasible iff UNSAT
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.

- **Add**: 2
- **Rem**: 1
- **Rod**: 2
- **Controller**: Add

Unfeasible iff UNSAT

- **τ**: simple path

**UNSAT**: no new states are reachable

**Simple path**
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.
**Inductive step:** proved incrementally following the partial order of the MSC.

**Base case:** bounded feasibility check.

Unfeasible iff UNSAT.
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

Unfeasible iff UNSAT

Sergio Mover (FBK)
• **Inductive step**: proved incrementally following the partial order of the MSC.
• **Base case**: bounded feasibility check.

\[
\text{Add} \quad 2 \\
\text{Add} \quad 1
\]

\[
\text{Rem} \
\text{Rem} \quad 1
\]

\[
\text{Rod}_1 \quad \text{Controller} \quad \text{Rod}_2
\]

\[
\tau \
\tau
\]

Unfeasible iff UNSAT

\[
\tau \
\tau
\]

Sergio Mover (FBK)
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.

- SAT - new states are reachable
- simple path

Unfeasible iff UNSAT
Inductive step: proved incrementally following the partial order of the MSC.
Base case: bounded feasibility check.
Inductive step: proved incrementally following the partial order of the MSC.

Base case: bounded feasibility check.

SAT - new states are reachable
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

Inductive step: proved incrementally following the partial order of the MSC.

Base case: bounded feasibility check.

Unfeasible iff UNSAT

UNSAT - no new states are reachable

Simple path
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

![Diagram of MSC with Rod 1, Controller, Rod 2, Add 1, Rem 1, Add 2 transitions]

Unfeasible iff UNSAT
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

```
Add_1  τ  τ  Add_1  τ  τ  Rem_1

Rem_1
```

Unfeasible iff UNSAT

Sergio Mover (FBK)

Unfeasibility and Explanations of MSC

October 31, 2011 17 / 28
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.
**Inductive step:** proved incrementally following the partial order of the MSC.

**Base case:** bounded feasibility check.

SAT - new states are reachable

Simple path
Inductive step: proved incrementally following the partial order of the MSC.

Base case: bounded feasibility check.
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.

SAT - new states are reachable

simple path
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.

Sergio Mover (FBK)  
Unfeasibility and Explanations of MSC  
October 31, 2011 17 / 28
**Partitioned K-induction - Algorithm**

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

**Diagram:**
- **Rod**: Rod 1 and Rod 2
- **Controller**:
  - Add 1
  - Rem 1
  - Add 2
- **Edges**:
  - Add 1 → Rod 1 → Add 1 → Rem 1
  - Rem 1 → Rod 2
  - UNSAT - no new states are reachable
  - Simple path

**Notes:**
- Unfeasible iff UNSAT
- October 31, 2011
**Partitioned K-induction - Algorithm**

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

![Diagram showing the inductive step and base case with MSC transitions and actions](attachment:image.png)

- **Unfeasible iff UNSAT**
Partitioned K-induction - Algorithm

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.
**Inductive step**: proved incrementally following the partial order of the MSC.

**Base case**: bounded feasibility check.
Inductive step: proved incrementally following the partial order of the MSC.

Base case: bounded feasibility check.
**Partitioned K-induction - Algorithm**

- **Inductive step**: proved incrementally following the partial order of the MSC.
- **Base case**: bounded feasibility check.

Unfeasible iff UNSAT

---

Sergio Mover (FBK)
Outline

1. Background
   - SMT analysis of Hybrid Systems
   - Scenario-Verification

2. Proving the unfeasibility of scenarios

3. Explanations of Unfeasibility

4. Experimental Evaluation

5. Conclusions and future work
Typical use case:
- We expect that a scenario is feasible.
- The analysis proves that the scenario is unfeasible in the network.
- How do we explain the unfeasibility?

We extract three types of explanations for the unfeasibility.
Unfeasibility due to a component

Explained with a formula that:

- Is **required** by the component when simulating its MSC events.
- Is **not consistent** with the other components when they simulate the events of the MSC.

\[ \text{It is the interpolant of } A \text{ and } B \]

\[ A \text{ is the encoding of the component and its MSC events.} \]

\[ B \text{ is the encoding of the other components and their MSC events.} \]
Unfeasibility due to a component

Explained with a formula that:

- Is required by the component when simulating its MSC events.
- Is not consistent with the other components when they simulate the events of the MSC.

It is the interpolant of $A$ and $B$:

- $A$ is the encoding of the component and its MSC events.
- $B$ is the encoding of the other components and their MSC events.
Unfeasibility due the network

Explained with a formula that:

- Is **required** by the network when simulating the MSC.
- Is **not consistent** with the additional constraints of the MSC.
Unfeasibility due the network

Explained with a formula that:

- Is **required** by the network when simulating the MSC.
- Is **not consistent** with the additional constraints of the MSC.

It is the interpolant of $A$ and $B$:

- $A$ is the encoding of the network and the MSC.
- $B$ are the CMSC constraints.

\[ \mathbf{Ready} \quad \dot{x} \in [0.9, 1.1] \quad \mathbf{TRUE} \]

\[ x = 0 \]

\[ \dot{x} \in [0.9, 1.1] \quad x \leq 5.9 \]

Recovering

\[ \dot{x} \in [0.9, 1.1] \quad x \leq 16 \]

Add $\frac{1}{x} := 0$

Remove $\frac{1}{x} := 0$

\[ x \geq 16 \quad / \quad \frac{x}{\tau} := x \]

\[ \text{Rod1} \]

\[ \text{Ready} \quad \dot{x} \in [0.9, 1.1] \quad \mathbf{TRUE} \]

\[ x = 0 \]

\[ \dot{x} \in [0.9, 1.1] \quad x \leq 5.9 \]

Recovering

\[ \dot{x} \in [0.9, 1.1] \quad x \leq 16 \]

Add $\frac{1}{x} := 0$

Remove $\frac{1}{x} := 0$

\[ x \geq 16 \quad / \quad \frac{x}{\tau} := x \]

\[ \text{Rod2} \]

\[ \text{No Rod} \quad \dot{x} \in [0.9, 1.1] \quad \mathbf{TRUE} \]

\[ x = 0 \]

\[ \dot{x} \in [0.9, 1.1] \quad x \leq 16 \]

Add $\frac{1}{x} := 0$

Remove $\frac{1}{x} := 0$

\[ x \geq 16 \quad / \quad \frac{x}{\tau} := x \]

\[ \text{Controller} \]

\[ \Delta_{\text{time}} \leq \frac{146}{3} \]

\[ \text{time} \leq 19 \quad \text{Add}_2 \]

\[ \text{Rem}_1 \]

\[ \text{time} \geq 19 \quad \text{Add}_2 \]

\[ \text{Rem}_2 \]

\[ \text{time} \leq 80 \quad \text{Add}_1 \]

\[ \text{Rem}_1 \]

\[ \text{time} \geq 80 \quad \text{Add}_1 \]
Inconsistent subset of the CMSC

Subset of the original CMSC that is still unfeasible with the network.
Subset of the original CMSC that is still unfeasible with the network. Extracted from the unsatisfiable core of the encoding.
Outline

1. Background
   - SMT analysis of Hybrid Systems
   - Scenario-Verification

2. Proving the unfeasibility of scenarios

3. Explanations of Unfeasibility

4. Experimental Evaluation

5. Conclusions and future work
Experimental Evaluation

Implementation:
- Approach implemented on top of the NuSMV model checker.
- We use the MathSAT SMT solver.

Settings:
- Linear hybrid automata benchmarks.
- Several handcrafted (unsatisfiable) MSCs.
- We scaled the dimension of the benchmarks (number of automata, length of the MSCs).

Comparison:
- MSC partitioned k-induction.
- Monolithic k-induction on the system composed with the monitor automata.
Partitioned k-induction vs. Monolithic k-induction (run times)
Outline

1 Background
   - SMT analysis of Hybrid Systems
   - Scenario-Verification

2 Proving the unfeasibility of scenarios

3 Explanations of Unfeasibility

4 Experimental Evaluation

5 Conclusions and future work
Conclusions and future work

- Efficient approach for proving the unfeasibility of CMSC.
  - The encoding exploits the structure of the CMSC.
  - Partitioned k-induction.
- Unfeasibility explanations:
  - Useful to localize and correct the errors.
  - Extracted exploiting the SMT solver functionalities.

Future works:
- More expressive MSCs (e.g. partial MSCs specifications).
- Validate the extracted explanations by real users.
- Automatic refinement loop in the abstraction.
- Non-linear hybrid systems.
Thank you for your attention.