The Impressive Power of Stopwatches

Frank Cassez*, Kim Larson†

Naman Kumar

*IRCCyN/CNRS UMR 6597, France
†Dep. of Computer Science, Aalborg University, Denmark

Delivered at: Chennai Mathematical Institute

November 13, 2022

What is this Presentation About?

1. Verification considers the properties of real-time systems and represents them as ideal mathematical models, running the underlying 'tests' on the model itself
2. One of the most important tests on timed automata is reachability analysis
3. This presentation is about two generalizations of timed automata and their consequences on reachability analysis:
    1. *Linear Hybrid Automata*, a very strong generalization of timed automata that models both discrete and continuous systems
    2. *Stopwatch Automata*, a much weaker extension of timed automata

A Generalization of Timed Automata

1. Linear Hybrid Automata (**HA**) are a strong extension of timed automata that model discrete as well as continuous evaluations of variables
2. **LHA** generalize almost every aspect of timed automata, including
    - clocks
    - guards
    - resets
    - even the passing of time!
3. We will consider each one of these separately and then consider the semantics

States, Clocks and Symbols

1. **LHA** operate over a finite set of **states** N, called **locations**
   - The initial location is represented as l_0
2. Clocks are replaced by **variables** V
   - At any given time, variables have a **valuation** $v \in \mathbb{R}^V$
   - The initial valuation v_0 need not be $\mathbf{0}$
3. Each symbol represents an **action** $a \in A$

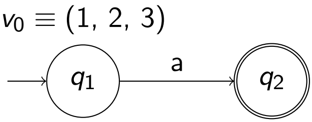

Figure: A basic LHA

Guards

Linear Constraints

A *linear expression* $\phi(v)$ over V is of the form $\sum a_i v_i$ with $v_i \in V$, $a_i \in \mathbb{Z}$. A *linear constraint* is a propositional formula using $\wedge, \vee, \neg$ over the atomic formulae

$$\phi(v) \bowtie c$$

where $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$.

1. **Simple constraints** are of the form $v - v' \bowtie c$ and $v \bowtie c$
2. Timed automata only allow simple constraints
3. **LHA** allow all linear constraints!

Resets

Linear Assigments

A *linear assignment* over V is an expression of the form

$$v := Av + b$$

where $A \in \mathbb{Z}^{n \times n}$ and $b \in \mathbb{Z}^n$.

1. **Simple assigments** only let A have entries 0 and 1 with a single 1 per row
2. Timed automata only allow simple assignments
3. **LHA** allow all linear assignments!

Variable Rate of Change

1. **LHA** allow for variables to move at differing rates of change
2. In a normal timed automaton, each clock moves forward by 1 second if 1 second passes
3. This is not necessarily the case for **LHA**!
   - Each variable has a *derivative* $d_v \in \mathbb{Z}$ associated with it
   - Each location has a range of derivatives $[d_1, d_2]$ for each variable that it allows

Continuous Change

Given $t \in \mathbb{R}^+$, the valuation $v + dt$ is defined as $v(x) + d(x)t$.

Semantics of an LHA

LHAs allow two kinds of transitions:

1. **Location Transitions**: In which the location of the automaton changes.

$$\langle l, v \rangle \xrightarrow{a} \langle l', v' \rangle$$

iff there exists a transition $(l, \gamma, a, \alpha, l')$ such that the guard $\gamma(v)$ is true, a is the action taken, and v' is as per the assignment $v' = \alpha(v)$

2. **Time-Delay Transitions**: In which there is a time delay on a single location.

$$\langle l, v \rangle \xrightarrow{e} \langle l', v' \rangle$$

iff $l = l'$ and $\exists d$ in the set of derivatives for the location l such that $v' = v + dt$.

The Water-Level Monitor

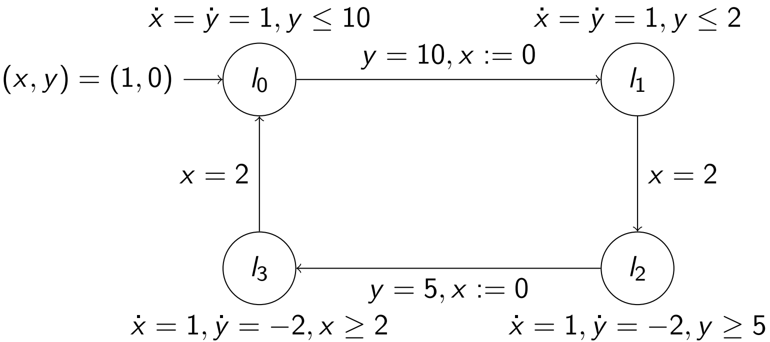

Figure: An LHA for a Water-Level Monitor

Unobservable Transitions

1. LHAs also accomodate for **unobservable** time-delays
2. In this case, at any given location, the variables are delayed as usual but time remains the same
3. Such transitions are represented as $\langle l, v\rangle \xrightarrow{\tau(e)} \langle l', v'\rangle$

Other Classes of Automaton

- **Timed Automata**: **LHA** which only allow simple constraints and simple assignments, and the derivatives of all variables $= 1$
- **Stopwatch Automata**: Timed automata in which the derivative of a variable is allowed to be either 0 or 1, ie. a clock can be stopped on any location
- **Linear Stopwatch Automata**: **LHA** in which the derivative is allowed to be only 0 and 1

We let TL_C be the set of timed languages accepted by class C.

The Main Theorem

Theorem 1

The classes **LHA** and **SWA** are equally expressive in the sense that $\mathrm{TL}_{\mathbf{LHA}} = \mathrm{TL}_{\mathbf{SWA}}$.

- It turns out that **LHA** and **SWA** have the same power
- The only additional power of **LHA** is the ability to stop time
- Everything else can be simulated by a regular timed automaton with unobservable transitions
- We will now look at a proof sketch of theorem 1

What do we Need to Deal With?

There are three differences we need to accomodate for:

- **Negative Values**: **SWA** do not allow for negative valuations, but **LSWA** do (via assignments)
- **Linear Constraints**: **SWA** only allow simple constraints
- **Linear Assigments**: **SWA** only allow simple assignments

We will look at each of these three in order.

Negative Values

- **Idea**: Split each variable x into (x^+, x^-)
- If $x := 0$ do nothing, if $x := 1$ put $(x^+, x^-) := (1, 0)$.
- If x can take a transition by satisfying the guard $\phi(x)$,
  - Set $x^+ := \phi(x)$ if $\phi(x) \geq 0$
  - Set $x^- := -\phi(x)$ otherwise
- Finally, replace x with $x^+ - x^-$

This ensures that $x = x^+ - x^-$ while $x^+, x^- \geq 0$.

Linear Guards

- **Idea**: Write $\sum a_i x_i = \sum b_j x_j - \sum c_k x_k$ where the negative values in a_i are replaced by the $c_i = -a_i$
- Introduce two new variables u, v, which count the two sums, then replace by $u - v \bowtie c$
- To count sums, introduce the following unobservable states:

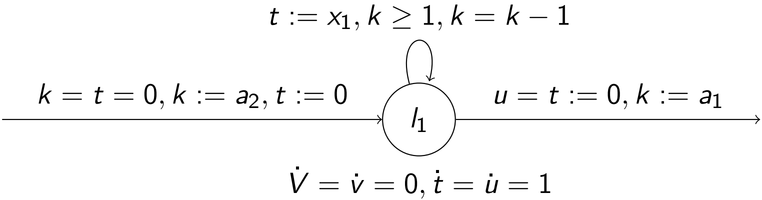

Figure: Replacement States

Linear Assignments

- Same idea as linear guards
- Set up new variables u_i to count $\sum_j a_{ij}x_j$
- Finally, set $x_i := u_i + b_i$ (this can be done using an assignment/reset interplay)
- Add $|V|$ new unobservable states to perform the addition
- **Total Complexity**: For n states, m transitions and k clocks, the total number of stopwatches is $3k + 3$ and number of states is $n + 4m(3k + 3)$

What do we Need to Deal With?

- We have already seen the case of linear guards and assignments
- Next, we reduce **LHA** to **LSWA**
- We thus need to deal with variable derivatives different from 0 and 1
- There are two cases:
  1. *Integer slopes* of the form $\dot{x} := c$
  2. *Interval slopes* of the form $\dot{x} \in [c_1, c_2]$

Integer Slopes

- Complete construction is involved, but we will give a brief sketch
- Slopes are only required for time-delay transitions
- First, add a variable t that measures how long the machine stays at state l
- Second, add an auxiliary variable y used to update x
- Two new unobservable states:
    1. Only a single self-transition when $y = t$ and then decrements t by 1, as a consequence x increases by y t times $\implies$ $v'(x) = v(x) + yt$
    2. Placeholder with all $\dot{x} = \dot{t} = 0$

Interval Slopes

- Same construction with one extra state that can only be nondeterministically reached within the interval $[c_1, c_2]$
- In case of negative slope, similar construction with two variables
- We have shown that **LSWA** = **LHA**

Thus, the expressive power of **LHA** is the same as **SWA**, and $\mathbf{TL}_{\mathbf{LHA}} = \mathbf{TL}_{\mathbf{SWA}}$.

- Reachability in **LHA** is undecidable
- Reachability in **SWA** is also undecidable, but reachability analysis is much easier due to less constraints
- One-one correspondence between the language of LHA and that of the corresponding SWA
- The authors extend the model-checking tool UPPAAL to SWA classes, and have discovered a low price of approximation.

Thank You

Questions?