# Automated Reasoning and Formal Verification
# Module II: Formal Verification
# Ch. 10: **SMT-Based Model Checking**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
URL: https://disi.unitn.it/rseba/DIDATTICA/arfv2025/
Teaching assistant: Gabriele Masina – gabriele.masina@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems
Academic year 2024-2025

last update: Friday 21st February, 2025, 11:21

# Outline

# Outline

# Motivations

- Model Checking for Timed Systems:
  - relevant improvements and results over the last decades
  - historically, “explicit-state” search style, based on DBMs
    - notable examples: Kronos, Uppaal
  - More recently, *symbolic* verification techniques:
    - extensions of decision diagrams
    - CDD, DDD, RED, ...
- Key problem: potential blow up in size
- A more recent and viable alternative to Binary Decision Diagrams: SAT-based MC
  - Bounded Model Checking (BMC), K-induction, IC3/PDR, ...

# Context

**First Idea: SMT-based BMC of Timed Systems**

[Audemard et al. 2002], [Sorea, MTCS'02], [Niebert et al.,FTRTFT'02]

Leverage the SAT-based BMC approach to Timed Systems by means of SMT Solvers

**Extensions**

- SMT eventually applied to other SAT-based MC techniques
  - K-Induction
  - interpolant-based
  - IC3/PDR
- SMT applied to a variety of domains:
  - hybrid systems
  - verification of SW (loop invariants/proof obbligations, ...)
  - hardware verification
- Nowadays SMT leading backend technology for FV

We restrict to BMC for Timed/Hybrid Systems only

# Outline

# Bounded Model Checking [Biere et al., TACAS'99]

- Given a Kripke Structure $M$, an LTL property $f$ and an integer bound $k$, is there an execution path of $M$ of length (up to) $k$ satisfying $f$? ($M \models_k \mathbf{E}f$)
- Problem converted into the satisfiability of the Boolean formula:

$$[[M]]^f_k := I(s^{(0)}) \wedge \bigwedge_{i=0}^{k-1} R(s^{(i)}, s^{(i+1)}) \wedge (\neg L_k \wedge [[f]]^0_k) \vee \bigvee_{l=0}^{k} ({}_lL_k \wedge {}_l[[f]]^0_k)$$

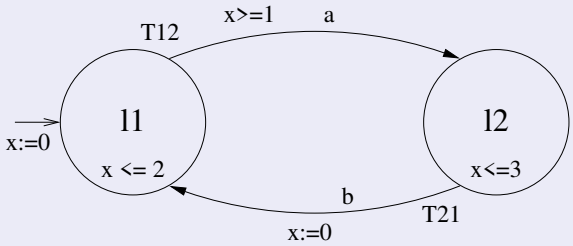
  s.t. ${}_lL_k \stackrel{\text{def}}{=} R(s^{(k)}, s^{(l)})$, $L_k \stackrel{\text{def}}{=} \bigvee_{l=0}^{k} {}_lL_k$
- A satisfying assignment represents a satisfying execution path.
- Test repeated for increasing values of $k$
- Incomplete
- Very effective for debugging, alternative to OBDDs
- Complemented with K-Induction [Sheeran et al. 2000]
- Further developments: IC3/PDR [Bradley, VMCAI 2011]

# General Encoding for LTL Formulae

| $f$ | $[[f]]^i_k$ | ${}_l[[f]]^i_k$ |
|---|---|---|
| $p$ | $p^{(i)}$ | $p^{(i)}$ |
| $\neg p$ | $\neg p^{(i)}$ | $\neg p^{(i)}$ |
| $h \wedge g$ | $[[h]]^i_k \wedge [[g]]^i_k$ | ${}_l[[h]]^i_k \wedge {}_l[[g]]^i_k$ |
| $h \vee g$ | $[[h]]^i_k \vee [[g]]^i_k$ | ${}_l[[h]]^i_k \vee {}_l[[g]]^i_k$ |
| $\mathbf{X}g$ | $\begin{cases} [[g]]^{i+1}_k & \text{if } i < k \\ \bot & \text{otherwise.} \end{cases}$ | $\begin{cases} {}_l[[g]]^{i+1}_k & \text{if } i < k \\ {}_l[[g]]^i_k & \text{otherwise.} \end{cases}$ |
| $\mathbf{G}g$ | $\bot$ | $\bigwedge^k_{j=min(i,l)} {}_l[[g]]^j_k$ |
| $\mathbf{F}g$ | $\bigvee^k_{j=i} [[g]]^j_k$ | $\bigvee^k_{j=min(i,l)} {}_l[[g]]^j_k$ |
| $h\mathbf{U}g$ | $\bigvee^k_{j=i} \left( [[g]]^j_k \wedge \bigwedge^{j-1}_{n=i} [[h]]^n_k \right)$ | $\bigvee^k_{j=i} \left( {}_l[[g]]^j_k \wedge \bigwedge^{j-1}_{n=l} {}_l[[h]]^n_k \right) \vee$ $\bigvee^{i-1}_{j=l} \left( {}_l[[g]]^j_k \wedge \bigwedge^k_{n=i} {}_l[[h]]^n_k \wedge \bigwedge^{j-1}_{n=l} {}_l[[h]]^n_k \right)$ |
| $h\mathbf{R}g$ | $\bigvee^k_{j=i} \left( [[h]]^j_k \wedge \bigwedge^{j}_{n=i} [[g]]^n_k \right)$ | $\bigwedge^k_{j=min(i,l)} {}_l[[g]]^j_k \vee$ $\bigvee^k_{j=i} \left( {}_l[[h]]^j_k \wedge \bigwedge^{j}_{n=i} {}_l[[g]]^n_k \right) \vee$ $\bigvee^{i-1}_{j=l} \left( {}_l[[h]]^j_k \wedge \bigwedge^k_{n=i} {}_l[[g]]^n_k \wedge \bigwedge^{j}_{n=l} {}_l[[g]]^n_k \right)$ |

# Timed Automata [Alur and Dill, TCS'94; Alur, CAV'99]

- Clocks: real variables (ex. $x$)
- Locations:
  - label: (ex. $l_1$),
  - invariants: (conjunctive) constraints on clocks values (ex. $x \leq 2$)
- Switches:
  - event labels (ex. $a$),
  - clock constraints (ex. $x \geq 1$),
  - reset statements (ex. $x := 0$)
- Time elapse: all clocks are increased by the same amount



# $\mathcal{LRA}$-Formulae

[Audemard et al., CADE'02]; [Sorea, MTCS'02]; [Niebert et al.,FTRTFT'02]

- $\mathcal{LRA}$-formulae are Boolean combinations of
  - Boolean variables and
  - linear constraints over real variables (equalities and differences)
    - e.g., $(x - 2 \cdot y \geq 4) \wedge ((x = y) \vee \neg A)$
- An interpretation $\mathcal{I}$ for a $\mathcal{LRA}$ formula assigns
  - truth values to Boolean variables
  - real values to numerical variables and constants
    - e.g., $\mathcal{I}(x) = 3, \mathcal{I}(y) = -1, \mathcal{I}(A) = \bot$
- $\mathcal{I}$ satisfies a $\mathcal{LRA}$-formula $\phi$, written "$\mathcal{I} \models \phi$", iff $\mathcal{I}(\phi)$ evaluates to true under the standard semantics of Boolean and mathematical operators.
  - E.g., $\mathcal{I}((x - 2 \cdot y \geq 4) \wedge ((x = y) \vee \neg A)) = \top$

# The MATHSAT Solver [Audemard et al., CADE'02]

- Bottom level: a $\mathcal{T}$-Solver for sets of $\mathcal{LRA}$ constraints
  - E.g. $\{..., z_1 - x_1 \leq 6, z_2 - x_2 \geq 8, x_1 = x_2, z_1 = z_2, ...\} \Longrightarrow$ unsat.
  - Combination of symbolic and numerical algorithms (equivalence class building, Belman-Ford, Simplex)
- Top level: a CDCL procedure for propositional satisfiability
  - mathematical predicates treated as propositional atoms
  - invokes $\mathcal{T}$-Solver on every assignment found
  - used as an enumerator of assignments
  - lots of enhancements

(see chapter on SMT)

# Outline

# Outline

# SMT-Based BMC for Timed Systems

## Independently developed approaches (2002):

- [Audemard et al. FORTE'02]: encoding into $\mathcal{LRA}$
  - all LTL properties
- [Sorea, MTCS'02]: encoding into $\mathcal{LRA}$
  - based on automata-theoretic approach for LTL
- [Niebert et al.,FTRTFT'02]: encoding into $\mathcal{DL}$
  - limited to reachability

## Disclaimer

These slides are adapted from [Audemard et al. FORTE'02]:

> G. Audemard, A. Cimatti, A. Kornilowicz, R. Sebastiani
> *Bounded Model Checking for Timed Systems*,
> proc. FORTE 2002, Springer
> freely available as `https://disi.unitn.it/rseba/publist.html`

(with some simplification in the notation).

# BMC for Timed Systems

**Basic ingredients:**

- An extension of propositional logic expressive enough to represent timed information: "$\mathcal{LRA}$-formulae"
- A SMT($\mathcal{LRA}$) solver for deciding $\mathcal{LRA}$-formulae
  $\Longrightarrow$ e.g., the MATHSAT solver
- An encoding from timed BMC problems into $\mathcal{LRA}$-formulae
  - $\mathcal{LRA}$-satisfiable iff an execution path within the bound exists

# Outline

# The encoding

Given a timed automaton $A$ and a LTL formula $f$:

- The encoding $[[A, f]]_k$ is obtained following the same schema as in propositional BMC:

$$[[A, f]]_k := I(s^{(0)}) \wedge \bigwedge_{i=0}^{k-1} R(s^{(i)}, s^{(i+1)}) \wedge (\neg L_k \wedge [[f]]_k^0) \vee \bigvee_{l=0}^{k} (\, _lL_k \wedge \, _l[[f]]_k^0)$$

- $[[M, f]]_k$ is a $\mathcal{LRA}$-formula, where
  - Boolean variables encode the discrete part of the state of the automaton
  - constraints on real variables represent the temporal part of the state

# Encoding: Boolean Variables

- Locations: an array $\underline{l}$ of $n \stackrel{\text{def}}{=} \lceil log_2(|L|) \rceil$ Boolean variables
  - $\underline{l_i}$ holds iff the system is in the location $l_i$
  - ex: "$\neg \underline{l_i}[3] \wedge \underline{l_i}[2] \wedge \neg \underline{l_i}[1] \wedge \underline{l_i}[0]$" means "the system is in location $\underline{l_5}$"
  - "$(\underline{l_i} = \underline{l_j})$" stands for "$\bigwedge_n (\underline{l_i}[n] \leftrightarrow \underline{l_j}[n])$",
  - "primed" variables $\underline{l_i}'$ to represent location after transition
- Events: for each event $a \in \Sigma$, a Boolean variable $\underline{a}$
  - $\underline{a}$ holds iff the system executes a switch with event $a$.
- Switches: for each switch $\langle l_i, a, \varphi, \lambda, l_j \rangle \in E$, a Boolean variable $T$,
  - $T$ holds iff the system executes the corresponding switch
- Time elapse and null transitions: two variables $T_\delta$ and $T^j_{null}$
  - $T_\delta$ holds iff time elapses by some $\delta > 0$
  - $T^j_{null}$ holds if and only $A_j$ does nothing (specific for automaton $A_j$)

Note: also for events, switches&transitions it is possible to use arrays of Boolean variables of size $\lceil log_2(|\Sigma|) \rceil$, $\lceil log_2(|E| + 2) \rceil$ respectively

# Encoding: Clock Values and Constraints

- Clocks values $x$ are "normalized" wrt absolute time $(t - x)$:
  - a clock value $x$ is written as difference $t - x$
  - $t$ represents the absolute time
  - "offset" variable $x$ represents the absolute time when the clock was reset last time
- Clock constraints $(x \bowtie c)$ reduce to $(t - x \bowtie c)$, $\bowtie \in \{\leq, \geq, <, >\}$, $c \in \mathbb{Z}$
- Clock reset conditions $(x := 0)$ reduce to $(x := t)$
- Clock equalities like $(x_k = x_l)$ reduce to $(t_k - x_k = t_l - x_l)$
  - appear only in loops
  - only place where full $\mathcal{LRA}$ is needed (rather than $\mathcal{DL}$)

  $\Longrightarrow$ for invariant checking (no loops) $\mathcal{DL}$ suffices
- Encoding the effect of transitions:
  - with a time-elapse transition:
    - $t' > t$, and $x' = x$
  - otherwise:
    - $t' = t$, absolute time does not elapse
    - $x' = t'$, if the clock is reset
    - $x' = x$, if the clock is not reset

# Encoding: Initial Conditions

## Initial condition $I(s)$:

- Initially, the automaton is in an initial location:

$$\bigvee_{l_i \in L^0} \underline{l_i}$$

- Initially, clocks have a null value:

$$\bigwedge_{x \in X} (x = t)$$

## Remark

Here and hereafter: in the encoding, when we write a formula $\varphi$, we implicitly mean "any formula logically equivalent to $\varphi$"

- in particular when encoding symbolically the discrete part of the system
- e.g., there is probably a much more compact formula equivalent to $\bigvee_{l_i \in L^0} \underline{l_i}$

# Encoding: Invariants

## Transition relation $R(s, s')$: Invariants

- Always, being in a location implies the corresponding invariant constraints:

$$\bigwedge_{l_i \in L} (\underline{l_i} \rightarrow \bigwedge_{\psi \in I(l_i)} \psi),$$

# Encoding: Transitions

Transition relation $T(s, s')$:

- Switches:

$$\bigwedge_{T \stackrel{\text{def}}{=} \langle l_i, a, \varphi, \lambda, l_j \rangle \in E} T \rightarrow \left( \underline{l_i} \wedge \underline{a} \wedge \varphi \wedge \underline{l_j}' \wedge (t' = t) \wedge \bigwedge_{x \in \lambda} (x' = t') \wedge \bigwedge_{x \notin \lambda} (x' = x) \right)$$

- Time elapse:

$$T_\delta \rightarrow \left( (\underline{l}' = \underline{l}) \wedge (t' - t > 0) \wedge \bigwedge_{x \in X} (x' = x) \wedge \bigwedge_{a \in \Sigma} \neg \underline{a} \right)$$

- Null transition:

$$T^j_{null} \rightarrow \left( (\underline{l}' = \underline{l}) \wedge (t' = t) \wedge \bigwedge_{x \in X} (x' = x) \wedge \bigwedge_{a \in \Sigma} \neg \underline{a} \right)$$

# Encoding: Relations between Transitions

- Mutual exclusion between events:

$$\bigwedge_{a_k, a_r \in \Sigma, a_k \neq a_r} (\neg \underline{a}_k \vee \neg \underline{a}_r)$$

- At least one transition takes place:

$$T^j_{null} \vee T_\delta \vee \bigvee_{T \in E} T$$

- Mutual exclusion between transitions:

$$\bigwedge_{T_k, T_r \in E \cup \{T^j_{null}\} \cup \{T_\delta\}, T_k \neq T_r} (\neg T_k \vee \neg T_r)$$

If events and transitions are encoded via arrays of Booleans, mutual exclusion constraints are not needed

# Automata Product Construction

- The encoding is compositional wrt. product of automata
- The encoding of $A = A_1 || A_2$ is given by the conjunction of the encodings of $A_1$ and $A_2$, plus a few extra axioms
- Mutual exclusion between events that are local

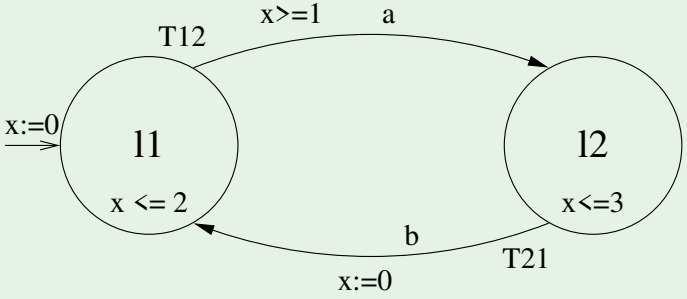
$$\bigwedge_{\substack{a_1 \in \Sigma_1 \setminus \Sigma_2 \\ a_2 \in \Sigma_2 \setminus \Sigma_1}} (\neg \underline{a}_1 \vee \neg \underline{a}_2)$$

- Forcing system activity:

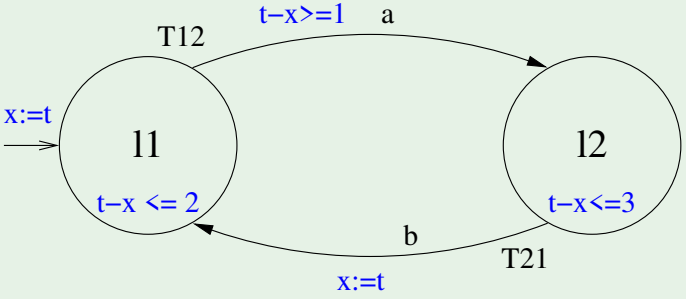
$$\bigvee_{j=0}^{N-1} \neg T^j_{null}$$

  - one distinct $T^j_{null}$ for each automaton $A_j$
  - $T_\delta$ is common to all automata $A_j$

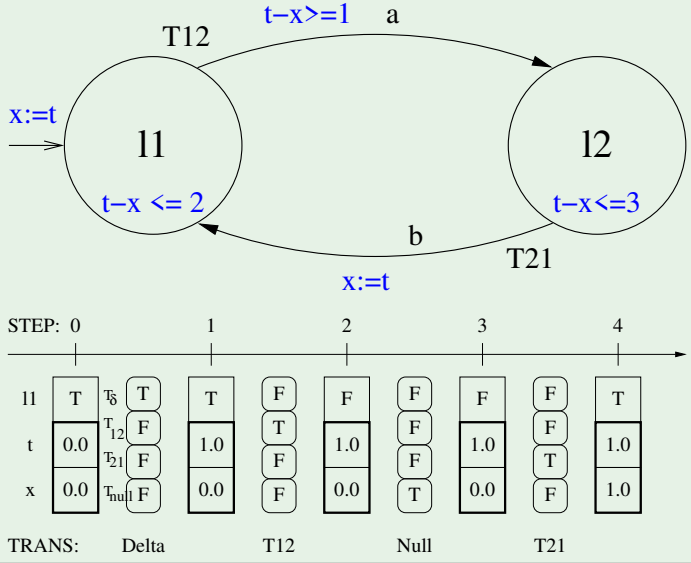
# A Simple Example



# A Simple Example



# A Simple Example



# Outline

# Encoding: Extension

## Adding Global Variables

Dealing with some global variable $v$ on discrete domain:

- A switch $T \stackrel{\text{def}}{=} \langle l_i, a, \varphi, \lambda, l_j \rangle$ can
  - be subject to a condition $\psi(v)$

  $\Longrightarrow$ add $T \rightarrow \psi(v)$
  - assign $v$ to some value n or keep its value

  $\Longrightarrow$ add $T \rightarrow (v' = n)$ or add $T \rightarrow (v' = v)$
- $T_\delta$ mantains the value of $v$:

  $\Longrightarrow$ add $T_\delta \rightarrow (v' = v)$
- $T^j_{null}$ imposes no constraint on $v$:

  $\Longrightarrow$ add nothing (for $A_j$)

# MATHSAT: Optimizations

**Customization of MATHSAT**

- Limit Boolean variable-selection heuristic to pick transition variables, in forward order

# Encoding: Optimizations

## Boolean Propagation of Math Constraints:

Idea: add small and mathematically-obvious lemmas

$$
\begin{array}{rcl}
\neg(t' = t) & \leftrightarrow & (t' - t > 0) \\
\bigwedge_{x \in X}(\neg(x = t) & \leftrightarrow & (t - x > 0)) \\
\bigwedge_{x \in X} \neg(x' = x) & \leftrightarrow & (x' - x > 0)
\end{array}
$$

$$
\begin{array}{lcccl}
\bigwedge_{x \in X}(\ (x = t) & \wedge & (x' = x) & \wedge & (t' = t)) \rightarrow (x' = t') \\
\bigwedge_{x \in X}(\neg(x = t) & \wedge & (x' = x) & \wedge & (t' = t)) \rightarrow \neg(x' = t') \\
\bigwedge_{x \in X}(\ (x = t) & \wedge & \neg(x' = x) & \wedge & (t' = t)) \rightarrow \neg(x' = t') \\
\bigwedge_{x \in X}(\ (x = t) & \wedge & (x' = x) & \wedge & \neg(t' = t)) \rightarrow \neg(x' = t') \\
\bigwedge_{x \in X}(\ (x' = x) & \wedge & (t' - t > 0) & \wedge & (t - x > 0)) \rightarrow (t' - x' > 0) \\
\bigwedge_{x \in X}(\ (t' = t) & \wedge & \neg(t - x > 0) & \wedge & (x' - x > 0)) \rightarrow \neg(t' - x' > 0) \\
\bigwedge_{x \in X}(\ (t - x \bowtie c) & \wedge & (x' = x) & \wedge & (t' = t)) \rightarrow (t' - x' \bowtie c) \\
\bigwedge_{x \in X}(\neg(t - x \bowtie c) & \wedge & (x' = x) & \wedge & (t' = t)) \rightarrow \neg(t' - x' \bowtie c)
\end{array}
$$

$\Longrightarrow$ force assignments by unit-propagation,
$\Longrightarrow$ saves calls to the $\mathcal{T}$-Solvers
Hint: Why loose your time to learn what you already know?

# Encoding Variants

## Shortening counter-examples:

- Collapsing consequent time elapsing transitions:
  - $s \xmapsto{\delta} s$, $s \xmapsto{\delta'} s$ reduced to $s \xmapsto{\delta+\delta'} s$
  - add $\neg T_\delta \vee \neg T'_\delta$ to transition relation $R(s, s')$

  $\Longrightarrow$ implements the notion of "non-Zeno-ness" (see previous chapter)
- Allow multiple parallel transitions
  - remove mutex between labels which are local to processes

  $\Longrightarrow$ allows a form of parallel progression

Remark: may change the notion of "next step"
$\Longrightarrow$ only if no "**X**" operators occurs in the property!

# Encoding Variants (cont.)

## A limited form of symmetry reduction

If N automata are symmetric (frequent with protocol verification):

- Intuition: restrict executions s.t.
  - At step 0 only $A_0$ can move
  - At step 1 only $A_0, A_1$ can move
  - At step 2 only $A_0, A_1, A_2$ can move
  - ...

  $\Longrightarrow$ we name "0" the first automata who acts, "1" the second one, etc.
- for step $i < N-1$, we drop the disjunct $\neg T_{null}^{i+1\ (i)} \vee \ldots \vee \neg T_{null}^{N-1\ (i)}$:

$$\textit{set} \bigvee_{j=0}^{min(i,N-1)} \neg T_{null}^{j\ (i)} \quad \textit{rather than} \bigvee_{j=0}^{N-1} \neg T_{null}^{j\ (i)}$$

$\Longrightarrow$ drops "symmetric" executions

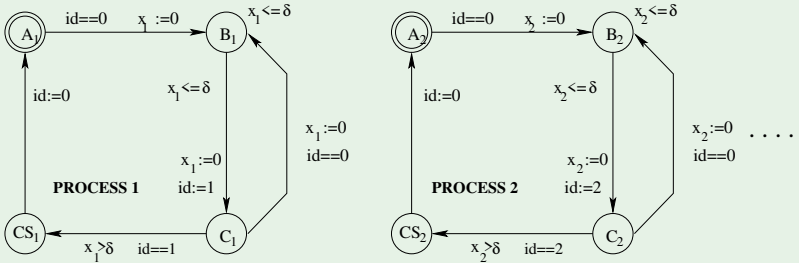
$\Longrightarrow$ reduces the search space of a up to $2^{N(N-1)/2}$ factor!

# Outline

# A Case-study: Fischer's Protocol

## A Mutual-Exclusion Real-Time Protocol

- N identical processes accessing one critical section
- shared variable $id \in \{0, 1, 2, ..., N\}$: process identifier (0: none)
  - when entering wait state $C_j$, agent $A_j$ writes its code on id
  - if $id = j$ after $\delta$, then $A_j$ can enter the critical session
- Two properties under test
  - Reachability: **EF** $\bigwedge_i P_i.C$ (reached in N+1 steps)
  - Fairness: **E**$\neg$(**GF**$P_i.B \rightarrow$ **GF**$P_i.CS$) (reached in N+5 steps)



# Fischer's protocol: (cont.)

Exercise:

- Why is $\mathbf{EF} \bigwedge_i P_i.C$ reached in N+1 steps?
- Why is $\mathbf{E}\neg(\mathbf{GF}P_i.B \rightarrow \mathbf{GF}P_i.CS)$ reached in N+5 steps?

(See [Audemard et al, FORTE'02] for the solution.)

# Fischer's protocol: (reachability)

$$M \models_k \mathbf{EF} \bigwedge_i P_i.C$$

| | MATHSAT | | MATHSAT,Sym | | DDD | | UPPAL | | KRONOS | | RED | | RED,Sym | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N | Time | Size | Time | Size | Time | Size | Time | Size | Time | Size | Time | Size | Time | Size |
| 3 | 0.05 | 2.9 | 0.04 | 2.9 | 0.11 | 106 | 0.01 | 1.7 | 0.01 | 0.8 | 0.23 | 2.0 | 0.19 | 2.0 |
| 4 | 0.09 | 3.0 | 0.08 | 3.0 | 0.14 | 106 | 0.02 | 1.9 | 0.02 | 2.2 | 1.00 | 2.1 | 0.70 | 2.1 |
| 5 | 0.20 | 3.2 | 0.16 | 3.2 | 0.24 | 106 | 0.21 | 1.9 | 0.09 | 19 | 3.70 | 2.2 | 2.00 | 2.4 |
| 6 | 0.60 | 3.7 | 0.23 | 3.7 | 0.47 | 106 | 3.44 | 6.7 | 0.39 | 236 | 12.00 | 2.7 | 5.20 | 3.1 |
| 7 | 3.20 | 4.2 | 0.36 | 4.2 | 1.30 | 106 | 153 | 54 | | MEM | 38 | 4.0 | 12 | 4.7 |
| 8 | 29 | 4.9 | 0.52 | 4.9 | 3.96 | 106 | TIME | | | | 121 | 7.6 | 26 | 7.8 |
| 9 | 343 | 5.9 | 0.75 | 5.9 | 14 | 106 | | | | | 416 | 16.6 | 49 | 13.3 |
| 10 | 3331 | 6.5 | 1.01 | 6.5 | 62 | 106 | | | | | 1382 | 39 | 90 | 23 |
| 11 | TIME | | 1.39 | 7.0 | 691 | 106 | | | | | TIME | | 157 | 38 |
| 12 | | | 1.89 | 7.5 | | MEM | | | | | | | 266 | 63 |
| 13 | | | 2.44 | 8.2 | | | | | | | | | 439 | 100 |
| 14 | | | 3.24 | 8.9 | | | | | | | | | 709 | 155 |
| 15 | | | 4.11 | 9.7 | | | | | | | | | 1118 | 225 |
| 16 | | | 5.10 | 10.7 | | | | | | | | | 1717 | 342 |
| 17 | | | 6.30 | 11.7 | | | | | | | | | 2582 | 492 |
| 18 | | | 8.00 | 12.9 | | | | | | | | | TIME | |
| 19 | | | 9.50 | 14.2 | | | | | | | | | | |

(MATHSAT times are sum of all instances up to $k$)

# Fischer's protocol (liveness violation)

$$M \models_k \mathbf{E}\neg(\mathbf{GF}P_i.B \rightarrow \mathbf{GF}P_i.CS)$$

| | MATHSAT | | | | | MATHSAT with Boehm heuristic | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $k \backslash N$ | 2 | 3 | 4 | 5 | 6 | 2 | 3 | 4 | 5 | 6 |
| 2 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 |
| 3 | 0.01 | 0.02 | 0.01 | 0.01 | 0.03 | 0.01 | 0.01 | 0.02 | 0.03 | 0.04 |
| 4 | 0.01 | 0.02 | 0.02 | 0.02 | 0.04 | 0.01 | 0.02 | 0.04 | 0.07 | 0.17 |
| 5 | 0.02 | 0.03 | 0.05 | 0.09 | 0.18 | 0.01 | 0.03 | 0.09 | 0.30 | 1.16 |
| 6 | 0.03 | 0.10 | 0.21 | 0.54 | 1.35 | 0.02 | 0.07 | 0.31 | 1.52 | 7.74 |
| 7 | 0.04 | 0.26 | 0.97 | 3.20 | 9.83 | 0.02 | 0.18 | 1.19 | 7.14 | 45.00 |
| 8 | | 0.65 | 4.80 | 19.72 | 70.70 | | 0.06 | 4.70 | 33.50 | 242.00 |
| 9 | | | 5.55 | 112.17 | 478.00 | | | 0.61 | 165.90 | 1348.00 |
| 10 | | | | 303.17 | 3086.00 | | | | 9.92 | 7824.00 |
| 11 | | | | | 5002.00 | | | | | 252.00 |
| $\Sigma$ | 0.12 | 1.08 | 11.62 | 438.93 | 8648.15 | 0.07 | 0.37 | 6.98 | 218.40 | 9720.13 |

# Outline

# The encoding

Given a Linear hybrid automaton $A$ and a LTL formula $f$:

- The encoding $[[A, f]]_k$ is obtained following the same schema as in propositional BMC:

$$[[A, f]]_k := I(s^{(0)}) \wedge \bigwedge_{i=0}^{k-1} R(s^{(i)}, s^{(i+1)}) \wedge (\neg L_k \wedge [[f]]_k^0) \vee \bigvee_{l=0}^{k} ( {}_l L_k \wedge {}_l[[f]]_k^0)$$

- $[[M, f]]_k$ is a $\mathcal{LRA}$-formula, where
  - Boolean variables encode the discrete part of the state of the automaton
  - a real variable $t$ (rational for rectangular automata) encodes absolute time elapse
  - real (rational) variables $x \in X$ encode continuous variables
  - constraints on real (rational) variables represent the continuous flow part of the state

# Encoding: Boolean Variables

- Locations: $\underline{l}$, as with timed systems
- Events: $a \in \Sigma$, as with timed systems
- Switches: $T$, as with timed systems
- Time elapse and null transitions: $T_\delta$ and $T^j_{null}$, as with timed systems

# Encoding: Continuous variables and constraints

- Continuous variables:
  - $t$ represents the absolute time
  - real (rational) variables $x$ represent continuous values
- Continuous constraints (initial, guards, invariants) reduce to linear constraints on $X$:
  $\sum_{x_i \in X} a_i x_i \bowtie c$ s.t. $\bowtie \in \{\leq, \geq, <, >\}$, $c \in \mathbb{Q}$
  - $x_i \bowtie c_i$ with rectangular automata
- Encoding the effect of discrete transitions:
  - $t' = t$, absolute time does not elapse
  - Jump relations reduce to linear transformations: $\bigwedge_j x'_j := \sum_i a_{ij} x_i + c_j$
    - $\bigwedge_{x_i \in X} (x'_i := c_i)$ with rectangular automata
- Encoding the effect of time-elapse transitions:
  - $t' > t$
  - $\bigwedge_j \Psi_j(X, t, X', t) \geq 0$

    where $\Psi_j(X, t, X', t) \stackrel{\text{def}}{=} \sum_i a_{ij}(x'_i - x_i) + c_j(t' - t) \geq 0$, given $\bigwedge_j \sum_i a_{ij} \frac{dx_i}{dt} + c_j \geq 0$
    - with rectangular automata:

      $(x'_i - x_i \leq c_i^M (t' - t) + b_i^M)$, $(x'_i - x_i \geq c_i^m (t' - t) + b_i^m)$ s.t. $c_i^M \stackrel{\text{def}}{=} max\{\frac{dx_i}{dt}\}$, $c_i^m \stackrel{\text{def}}{=} min\{\frac{dx_i}{dt}\}$,

# Encoding: Initial Conditions and Invariants

## Initial condition $I(s)$:

- Initially, the automaton is in an initial location:

$$t = 0 \rightarrow \bigvee_{l_i \in L^0} \underline{l_i}$$

- Initially, clocks comply with initial conditions:

$$t = 0 \rightarrow \bigwedge_{l_i \in L^0} (\underline{l_i} \rightarrow \mathit{Init}_l(X))$$

## Transition relation $R(s, s')$: Invariants

- Always, being in a location implies the corresponding invariant constraints:

$$\bigwedge_{l_i \in L} (\underline{l_i} \rightarrow \bigwedge_{\psi \in I(l_i)} \psi),$$

# Encoding (linear automata): Transitions

Transition relation $T(s, s')$:

- Switches:

$$\bigwedge_{T \stackrel{\text{def}}{=} \langle l_i, a, \varphi, J, l_j \rangle \in E} T \rightarrow \left( \underline{l_i} \wedge \underline{a} \wedge \varphi \wedge \underline{l_j}' \wedge (t' = t) \wedge \bigwedge_{x_j \in X} (x_j' := \sum_i a_{ij} x_i + c_j) \right)$$

- Time elapse:

$$T_\delta \rightarrow \left( (\underline{l'} = \underline{l}) \wedge (t' - t > 0) \wedge (\bigwedge_j \Psi_j(X, t, X', t) \geq 0) \wedge \bigwedge_{a \in \Sigma} \neg \underline{a} \right)$$

- Null transition:

$$T^j_{null} \rightarrow \left( (\underline{l'} = \underline{l}) \wedge (t' = t) \wedge \bigwedge_{x_i \in X} (x_i' = x_i) \wedge \bigwedge_{a \in \Sigma} \neg \underline{a} \right)$$

# Encoding (rectangular automata): Transitions

Transition relation $T(s, s')$:

- Switches:

$$\bigwedge_{T \stackrel{\text{def}}{=} \langle l_i, a, \varphi, \lambda, l_j \rangle \in E} T \rightarrow \left( \underline{l_i} \wedge \underline{a} \wedge \varphi \wedge \underline{l_j}' \wedge (t' = t) \wedge \bigwedge_{x_i \in X} (x_i' := c_i) \right)$$

- Time elapse:

$$T_\delta \rightarrow \left( (\underline{l}' = \underline{l}) \wedge (t' - t > 0) \wedge \bigwedge_{x_i \in X} (x_i' - x_i \leq c_i^M (t' - t) + b_i^M) \wedge (x_i' - x_i \geq c_i^m (t' - t) + b_i^m) \wedge \bigwedge_{a \in \Sigma} \neg \underline{a} \right)$$

- Null transition:

$$T^j_{null} \rightarrow \left( (\underline{l}' = \underline{l}) \wedge (t' = t) \wedge \bigwedge_{x_i \in X} (x_i' = x_i) \wedge \bigwedge_{a \in \Sigma} \neg \underline{a} \right)$$

# Outline

# Proposed Exercise

## Proposed Exercise

- Consider the Train-gate-controller example from [Alur CAV'99] (see previous chapter)
  - Encode the Initial state formula
  - Encode the transition relation
  - Encode the BMC problem for the formula $\mathbf{G}(s_2 \rightarrow t_2)$
- As above, reducing the delay time for the controller from 1 to 0.5
  - what happens?
  - in how many steps?
- Encode the above into MathSAT

# Proposed Exercise

## Proposed Exercise

- Consider the rectangular automaton of the Train-gate example (see previous chapter)
  - Encode the Initial state formula $I(s^{(0)})$
  - Encode the transition relation $R(s^{(i)}, s^{(i+1)})$