nuXmv: model checking timed systems

Enrico Magnago

University of Trento,
Fondazione Bruno Kessler

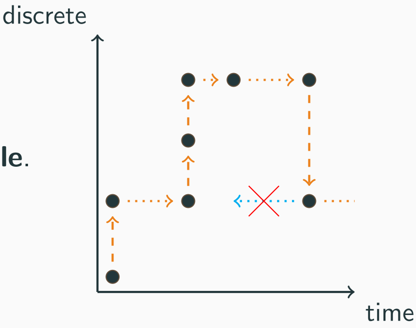
Timed systems

Real time systems

- Correctness depends not only on the logical result but also on the time required to compute it.
- Common in safety-critical domains like: defense, transportation, health-care, space and avionics.

Timed Transition System (TTS)

transitions are either discrete or time-elapses,
all clocks increase of the same amount in time-elapses.
Model checking for TTS is **undecidable**.



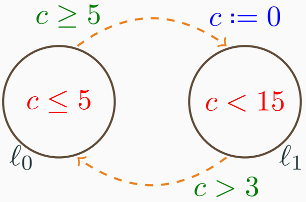
Timed Automata (TA)

decidable restriction of TTS,
finite time abstraction:
clocks compared only to constants.

Timed systems: representation

Timed Automata (TA)

Explicit graph representation of discrete states (nodes) and transitions (edges).
Symbolic representation of temporal aspects via (convex) constraints (location invariants, transition guards and resets).



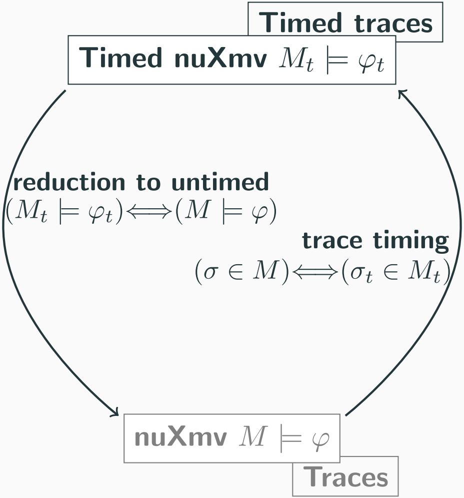
Symbolic TTS

Logical formulae represent sets of states: $p := \{s \mid s \models p)\}$.
Transition system symbolically represented by formula $\varphi(X, X')$.
There is a discrete transition from s_0 to s_1 iff $s_0(X), s_1(X') \models \varphi(X, X')$.

$$
\begin{aligned}
&l = \ell_0 \rightarrow c \leq 5 \quad \wedge \\
&l = \ell_1 \rightarrow c < 15 \quad \wedge \\
&(l = \ell_1 \wedge l' = \ell_0) \rightarrow c > 3 \;\; \wedge \\
&(l = \ell_0 \wedge l' = \ell_1) \rightarrow (c \geq 5 \wedge c' = 0)
\end{aligned}
$$

Timed nuXmv

nuXmv for timed system: architecture



Timed nuXmv: input language [1/4]

Overview

- Must start with `TIME_DOMAIN continuous`;
- Symbolic description of infinite transition system using: `INIT`, `INVAR` and `TRANS` to specify initial, invariant and transition conditions.
- Model described as a synchronous composition of `MODULE` instances.
- Clock variables,
- `time`: built-in clock variable,
- convex invariants over clocks,
- `URGENT`: forbid time elapse.

Timed nuXmv: input language [2/4]

Timed nuXmv adds

- `clock` variable type, all `clocks` increase of the same amount during timed transitions;
- `time`: built-in `clock`, can be used only in comparisons with constants;
- `non continuous` type modifier: symbol can change its assignment during timed transitions;
- `URGENT`: freeze time: when one of the `URGENT` conditions is satisfied only discrete transitions are allowed;
- $\mathrm{MTL}_{0,\infty}$ specifications, by "extending" LTL;

Timed nuXmv: input language [3/4]

Timed nuXmv updates

- `TRANS` constrain the discrete behaviour only,
- `INVAR`: `clocks` allowed in invariants with shape: `no_clock_expr -> convex_clock_expr`;
- LTL operators: X, Y, U, S,
- Bounded LTL operators.

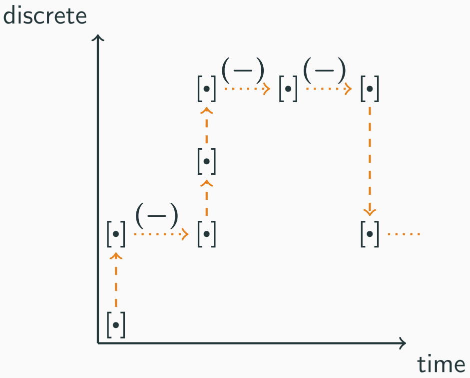
Timed nuXmv: input language [4/4]

Specification

- Different operators to refer to the *timed* next and *discrete* next: `X`, `X~`; symmetrically for the past: `Y`, `Y~`.
- Time interval semantic to handle open intervals: a predicate p might hold in an interval $(a, b]$ for $a, b \in \mathbb{R}$.
- Operators to retrieve value of expression the next/last time an expression will hold/held: `time_until`, `time_since`, `@F~` and `@O~`.

Timed to untimed model

- `clock` symbols and `time`: variables of type `real`.
- δ: continuous positive variable, prescribes the amount of time elapse for every transition.
- ι: prescribes the alternation of singular $[\bullet]$ and open $(-)$ time intervals.



Properties rewriting

MTL $_{fragment}$

$$F_{[0,5]}\ p$$

↓ rewrite

LTL $_{timed}$

$$((\neg p U p) \wedge time\_until(p) \leq 5) \vee$$
$$((\neg p U \tilde{X} p) \wedge time\_until(p) < 5)$$

↓ untime

LTL $_{untimed}$

$$((\neg p U p) \wedge (time@\tilde{F}p) - time \leq 5) \vee$$
$$((\neg p U ((\neg \iota \wedge p) \vee X(\neg \iota \wedge p))) \wedge (time@\tilde{F}p) - time < 5)$$

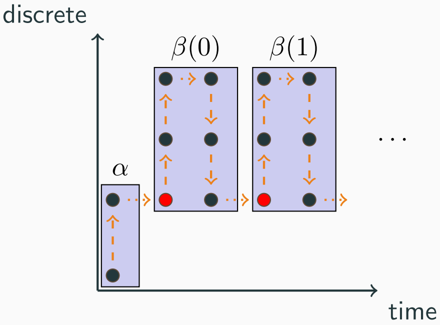
Timed and infinite traces

Timed and infinite traces

From untimed model execution to timed trace.

Issue
nuXmv traces must have shape: $\alpha\beta^\omega$,
α and β sequences of states.
Complete for finite state systems.
TTS: time monotonically increasing, infinite state system, **undecidable**.
Identify traces expressible as: $\alpha\beta(i)^\omega$.
Same problem can be found in infinite state transition systems.

Solution
Value assigned to variables at state s is function of the previous configuration assignments.
e.g. $next(time) := time + \delta$



Timed and infinite traces: operations

Three main operations on traces: **simulation**, **execution** and **completion**.

Simulation

Build a possible execution of the model. The trace can be built automatically by the system or the user can choose each state from the list of possible ones.
Exploit SMT-solver to perform a discrete transition or time-elapse to obtain next configuration.

Timed and infinite traces: operations

Execution

Check if a trace belongs to the language of the model.
Exploit SMT-solver to prove that **for all** possible iterations all prescribed transition can be performed.

Completion

A partial trace is completed so that it belongs to the model language.
Sound and complete technique requires to check if there **exists** a possible completion so that the completed trace belongs to the model language: quantifier alternation ($\exists\forall$).
Adopt sound but incomplete approach.

How to run: model [1/3]

- `./nuXmv -time -int`: start nuXmv interactively and enable commands for timed models.
- `go_time`: process the model.
- `write_untimed_model`: dump SMV model corresponding to the input timed system.

How to run: verify [2/3]

- `timed_check_invar`: check invariants.
- `timed_check_ltlspec`: check LTL.

Mostly the same command line options of the corresponding commands for untimed models.

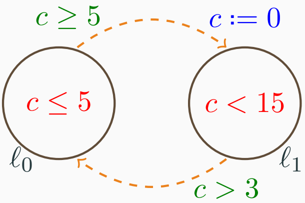
How to run: simulation and traces [3/3]

- `timed_pick_state`: pick initial state.
- `timed_simulate`: simulate the model starting from a given state.
- `execute_traces`: re-execute stored traces.
- `execute_partial_traces`: try to complete stored traces.

Exercises

Simple timed automaton

Write the SMV model corresponding to the timed automaton in the figure.



Properties

- from location ℓ_0 we always reach ℓ_1 within 5 time units;
- if we are in ℓ_1 then for the next 3 time units we remain in ℓ_1;
- if just arrived in ℓ_1 then for the next 3 time units we remain in ℓ_1.

Timed thermostat

- a thermostat has 2 states: *on* and *off*;
  - if the temperature is below 18 degrees the thermostat switches *on*.
  - if the temperature is above 18 degrees the thermostat switches *off*.
- at every time unit the temperature increases (if *on*) or decreases (if *off*) by 1;
- the thermostat measures the temperature at most ($<$) every max_dt time units.
- the temperature initially is in $[18 - max\_dt; 18 + max\_dt]$.

Verify that the temperature is always in $[18 - 2max\_dt; 18 + 2max\_dt]$

Fischer mutual exclusion protocol

procedure $\textsc{Fischer}(pid, c, id)$
    loop
        while $id \neq 0$ **do**
            $skip$
        $x \leftarrow random(0, c)$
        $wait\_at\_most(c)$
        $id \leftarrow pid$
        $wait\_at\_least(c)$
        if $id = pid$ **then**
            Critical Section
            $id \leftarrow 0$

Verify the mutual exclusion property.
NUXMV does not support asynchronous composition: model scheduler explicitly.