Extending nuXmv with Timed Transition Systems and Timed Temporal Properties

Alessandro Cimatti, Alberto Griggio, Enrico Magnago, Marco Roveri, Stefano Tonetta

Fondazione Bruno Kessler

untimed nuXmv

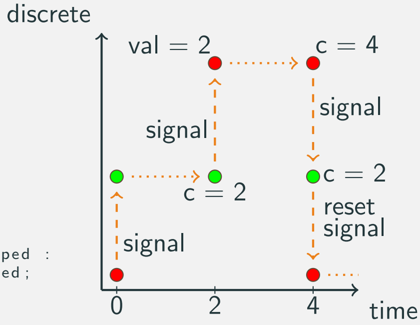
- **Symbolic** model checker for **infinite state** transition systems.
- Verification of **invariants** and **LTL** properties, also **CTL** for finite state.
- **SMT-based** techniques, tight integration with **MathSAT5**: IC3-ia, L2S, K-liveness ...

https://nuxmv.fbk.eu

Modelling Timed Transition Systems (TTS)

Beyond Timed automata, e.g. stopwatch.

```
@TIME_DOMAIN continuous

MODULE stopwatch(signal, reset)
VAR
  c : clock;
  val : real;
  stopped : boolean;

DEFINE out := stopped? val : c;

INIT stopped & c = 0 & val = 0;

-- signal starts/stops the timer.
TRANS next(stopped) = signal? !stopped :
                                stopped;

TRANS
  case
    reset   : next(c) = 0   & next(val) = 0;
    stopped : next(c) = val & next(val) = val;
    TRUE    : next(c) = c   & next(val) = c;
  esac;

 ...
```



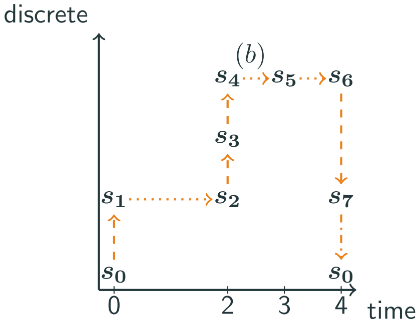
● stopped = TRUE

● stopped = FALSE

Timed Temporal properties

The following properties hold on the trace represented in the plot. b is a boolean atom that holds only between s_4 and s_5.

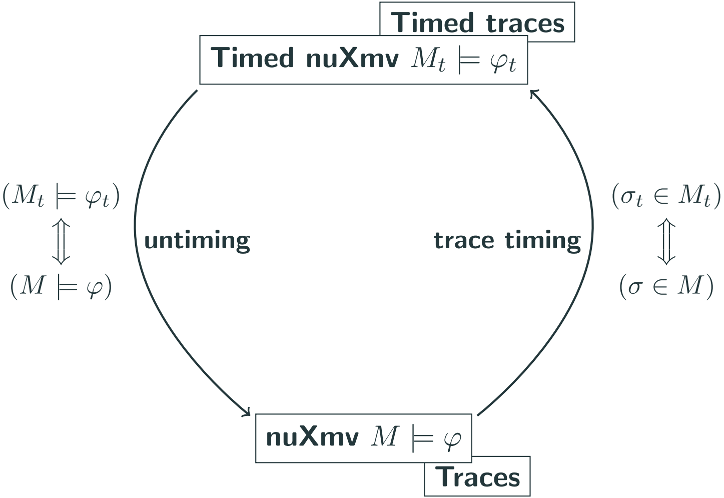
- $G(s_0 \rightarrow time\_until(b) = 2)$
- $G(s_4 \rightarrow \tilde{X} b)$
- $G(s_5 \rightarrow \tilde{Y} b)$
- $GF(s_6 \; @\tilde{F} \; time\_since(b) = 1)$
- $GF_{[0,4]} b$



More details: SMT-Based Satisfiability of First-Order LTL with Event Freezing Functions and Metric Operators, Information and Computation Special Issue GandALF 2017

Architecture overview

Complete reduction to infinite state invariant/LTL model checking.
Lift counter-example, if any.



Issue with liveness properties

Lasso-shaped counter-examples

- nuXmv, as many tools, provides only lasso-shaped counter-examples for liveness properties;
- incomplete for infinite state systems;
- in timed systems we always have a diverging variable: *time*.

Proposed approach

- Extend representation of trace such that some variables have a lasso-shaped behaviour, while some clock variables eventually are never reset.
- Extend BMC encoding with extra constraints to ensure that the trace belongs to the system.

Infinite and timed traces

Representation

- Represent traces with shape: $\alpha\beta_0\beta_1\beta_2\ldots$.
- Assignment at state s is function of the previous assignment.

Example

```
@TIME_DOMAIN continuous

MODULE main
FROZENVAR k : real;
VAR c : clock;

INIT c = 0;
TRANS next(c) = c;

LTLSPEC G F c < k;
```

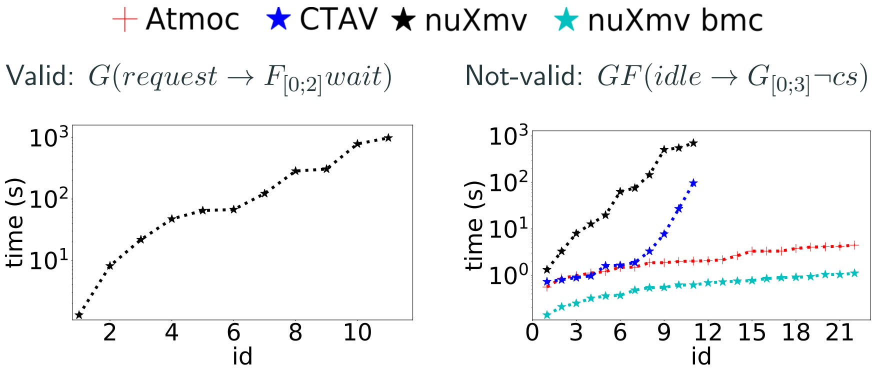
```
 Infinite Trace diverging symbols: c
-> State: 1.1 <-
   k = 2
   c = 0
-- [ time elapse: time = 0; delta = 3 ] --
-> State: 1.2 <-
   c = 3
-- Loop starts here
-- [ time elapse: time = diverging; delta = 1 ] --
-> State: 1.3 <-
   c = c + 1
-- [ time elapse: time = diverging; delta = 3 ] --
-> State: 1.4 <-
   c = c + 3
-- [ time elapse: time = diverging; delta = 1 ] --
-> State: 1.5 <-
   c = c + 1
```

State 1.5 represents the *loop-back state*: first state of the next β.

Experimental evaluation: tools

Tool	INVARIANT		LTL		$\mathbf{MTL}_{0,\infty}$	
	TA	**TTS**	**TA**	**TTS**	**TA**	**TTS**
ATMOC	✓	✗	✓	✗	✓$_{\perp}$	✗
CTAV	✓	✗	✓	✗	✓	✗
DiVinE	✓	✗	✓	✗	✗	✗
LTSmin-Opaal	✓	✗	✓	✗	✗	✗
Uppaal	✓	✗	$\sim$	✗	✗	✗
Timed nuXmv	✓	✓	✓	✓	✓	✓

MTL$_{0,\infty}$ properties



Conclusions

Summary

- nuXmv input language extended for timed transition systems.
- More expressive than timed automata, undecidable.
- Complete reduction to infinite discrete transition system.
- Infinite non-lasso counter-examples.
- nuXmv is able to prove and find counter-examples for $\mathrm{MTL}_{0,\infty}$ specifications.

Future work

- Improve encoding in infinite state transition system: specification size blowup.
- Support for integer time domain.
- Handle broader class of infinite executions.
- Support parameter synthesis on TTS.

The End

Thank you for your attention, questions?

Input language example

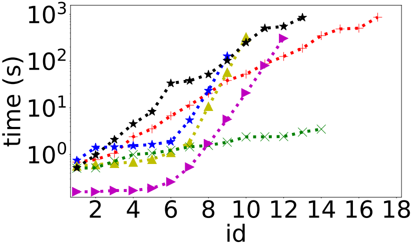
```
-- annotation to specify the time semantics, in this case dense time
@TIME_DOMAIN continuous
MODULE main
 FROZENVAR p: real;
 INIT p > 0  -- parameter
 VAR i: real;  -- input of the sensor
 VAR s: Sensor(i);
 VAR m: Monitor(s.o,p);
 -- any fault is detected in p timed units
 LTLSPEC G ( s.fault -> F [0,p] m.alarm )

MODULE Sensor(i)
 VAR o: real;
 VAR fault: boolean;
 -- if not faulty, the sensor provides in output directly the input
 TRANS !fault -> next(o) = i
 -- if faulty, the sensor output is stuck at the last value
 TRANS fault -> next(o) = o
 TRANS fault -> next(fault)  -- the fault is permanent

MODULE Monitor (i,p)
 VAR previous_value: real;
 VAR c: clock;
 VAR alarm: boolean;
 INIT c=0 & previous_value = i & !alarm
 INVAR TRUE -> c <= p
 TRANS time <= p | time >= p
 -- the monitor reads the sensor every p time units
 TRANS (c = p & next(c) = 0 & next(previous_value) = i) |
       (c <= p & next(c) = c & next(previous_value) = previous_value)
 -- alarm raised when the same value read twice consecutively
 TRANS next(alarm) <-> (alarm | i=previous_value)
```

Fischer mutual exclusion protocol: invariant

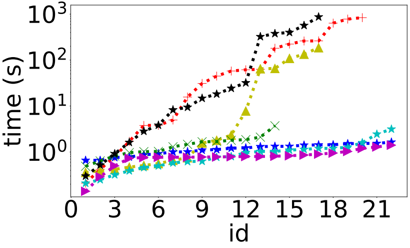
Mutual exclusion property

+ Atmoc ★ CTAV ▲ Divine × LTSmin ★ nuXmv ★ nuXmv bmc ▶ Uppaal

Correct:

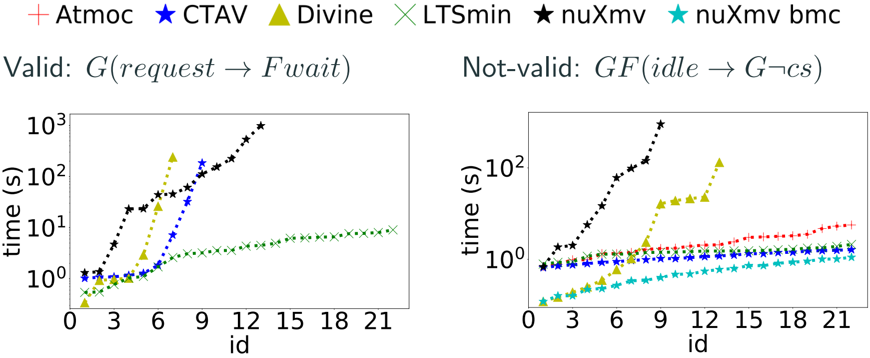


Buggy:



Fischer mutual exclusion protocol: LTL

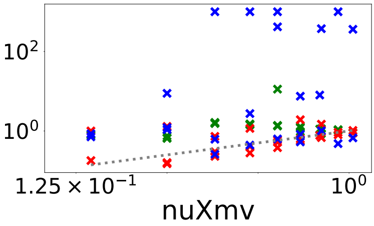
LTL properties

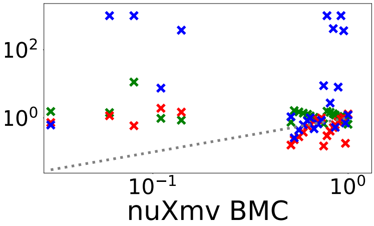


Pure $\mathrm{MTL}_{0,\infty}$

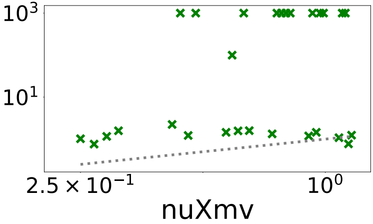
Atmoc CTAV Zot

Not-valid:

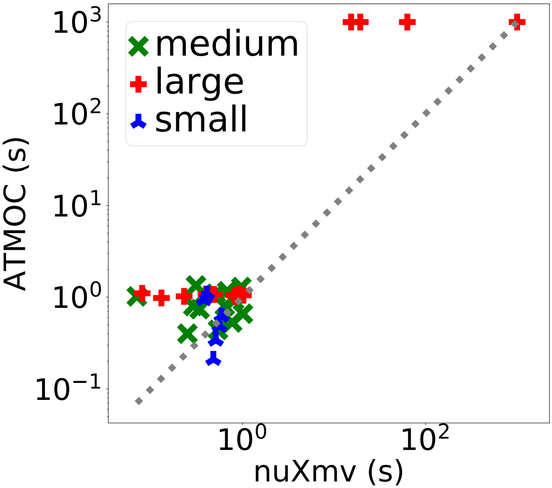




Valid:



Emergency diesel generator, invariants



Model simulation

- List of initial states:

$$I \wedge N \wedge \bigwedge_{j=0}^{i-1} \neg s_j$$

- List of possible time elapses from s:

$$s \wedge \delta > 0 \wedge T \wedge N' \wedge \bigwedge_{j=0}^{i-1} \neg s'_j$$

- List of possible discrete transitions from s:

$$s \wedge \delta = 0 \wedge T \wedge N' \wedge \bigwedge_{j=0}^{i-1} \neg s'_j$$

Trace execution

- Initial state validity:

$$s_0 \wedge I \wedge N$$

- Trace prefix (non-loop):

$$s_i \wedge s'_{i+1} \wedge T \wedge N'$$

- Trace loop from s_l to s_k is valid iff this formula is unsatisfiable:

$$\exists i_{loop} \in \mathbb{N}\ [\exists l \leq j < k : ((s_j^{i_{loop}} \wedge s_{j+1}^{i_{loop}}) \vee \neg T \vee \neg N_{j+1}) \vee \\ ((s_k^{i_{loop}} \wedge s_{l+1}^{i_{loop}+1}) \vee \neg T \vee \neg N_{l+1})]$$

Trace completion

Complete and sound SMT encoding:

$$
\begin{aligned}
\exists s'_l, \ldots, s'_k : \ & \bigwedge_{j=l}^{k} completes(s'_j, s_j) \ \wedge \\
& \forall i_{loop} \in \mathbb{N} \ [\ (\forall l \leq j < k : ({s'}_j^{i_{loop}} \wedge {s'}_{j+1}^{i_{loop}}) \rightarrow (T \wedge N_{j+1})) \ \wedge \\
& \qquad ((\,{s'}_k^{i_{loop}} \wedge {s'}_{l+1}^{i_{loop}+1}) \rightarrow (T \wedge N_{l+1})) \ ]
\end{aligned}
$$

Second and third line of the formula is exactly the loop validity encoding used in execution.

Fischer protocol

```
procedure FISCHER(pid, c, id)
    loop
        while id ≠ 0 do
            skip
        wait_at_most(c)
        id ← pid
        wait_at_least(c)
        if id = pid then
            Critical Section
            id ← 0
```