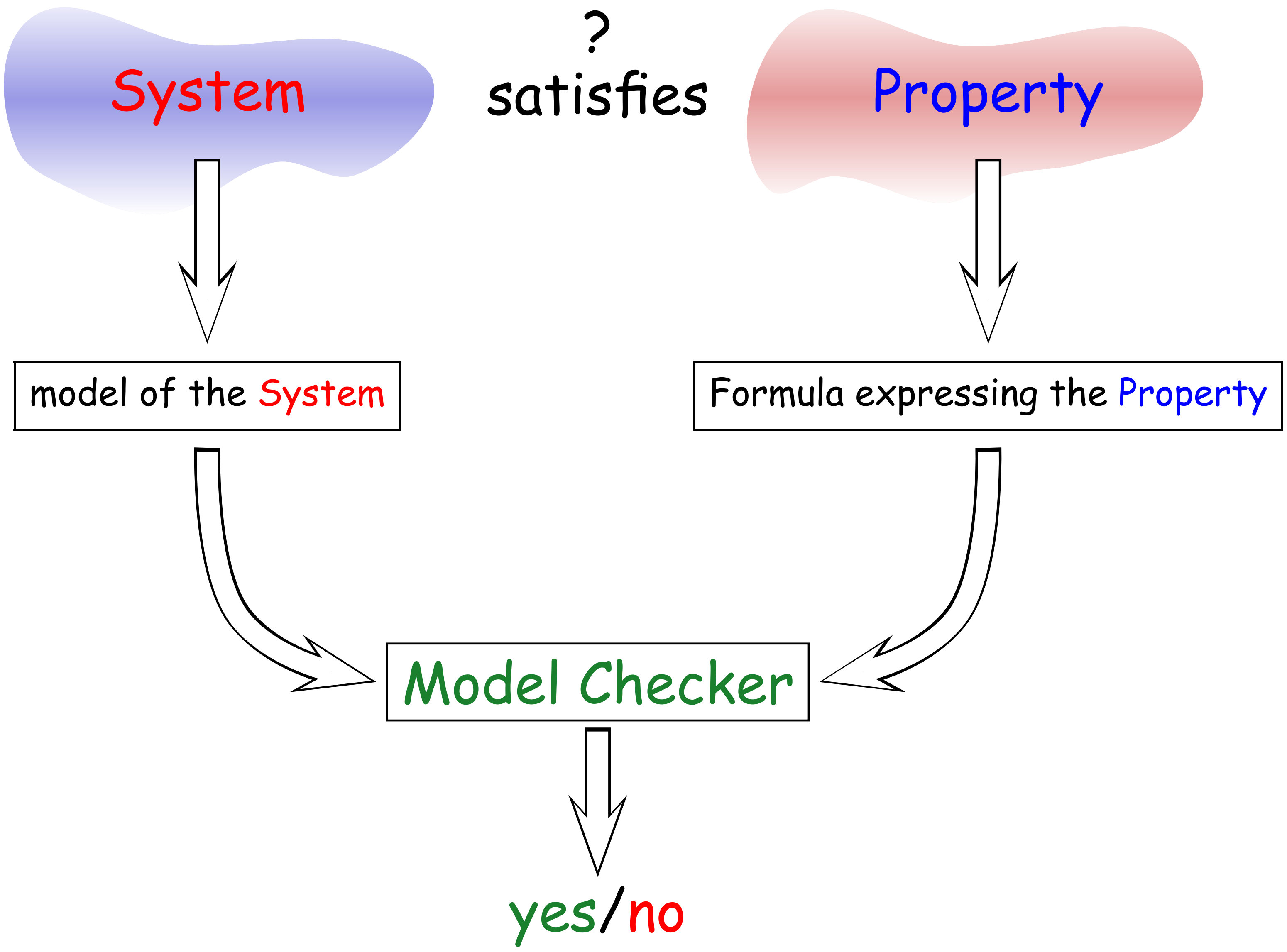
Symbolic Model Checking for Simply Timed Systems

Nicolas MARKEY and Philippe SCHNOEBELEN

{markey,phs}@lsv.ens-cachan.fr

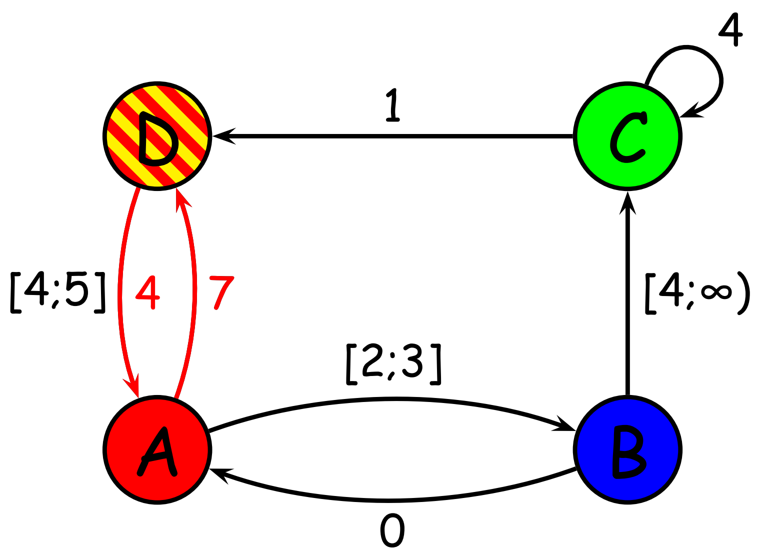
Lab. Specification & Verification, CNRS & ENS Cachan

Model Checking



Timed Kripke Structures

Timed Kripke Structure = Kripke Structure with integer durations on transitions:

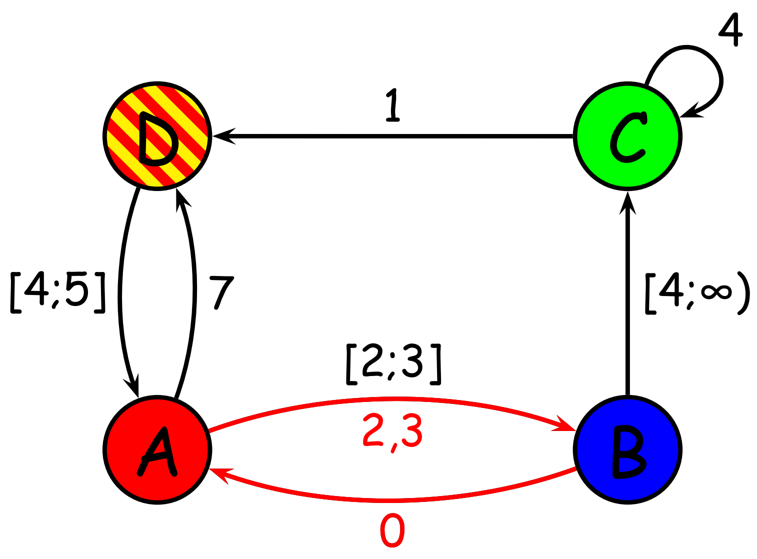


Semantics: instantaneous steps, no intermediate state.

$$A \models E\,(\text{true}\ U_{=11}\ \text{red})$$

Timed Kripke Structures

Timed Kripke Structure = Kripke Structure with
integer durations on transitions:

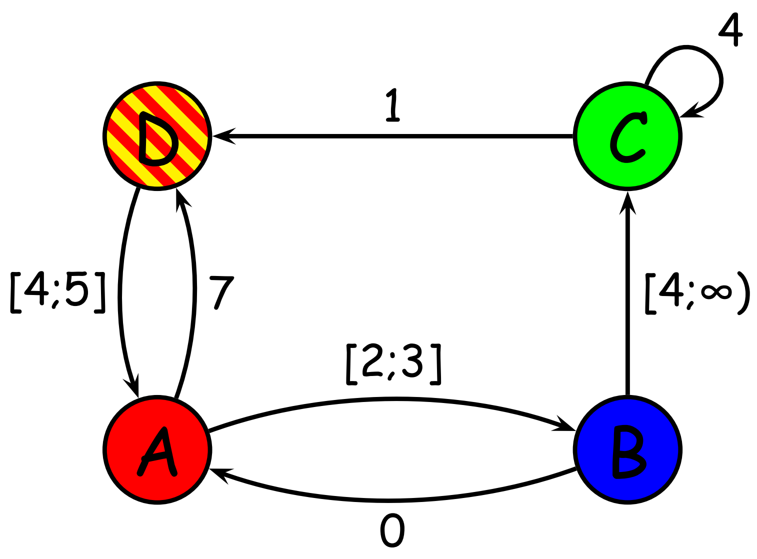


Semantics: instantaneous steps, no intermediate state.

$A \models EF_{=5}\ \text{red}$

Timed Kripke Structures

Timed Kripke Structure = Kripke Structure with integer durations on transitions:



Semantics: instantaneous steps, no intermediate state.

$$A \not\models AG\,(EF_{9..10}\ \mathrm{green})$$

Algorithm for $EU_{=0}$ and $AU_{=0}$

For $E[\varphi\ U_{=0}\ \psi]$:

either ψ holds, or φ holds and some successor satisfies $E[\varphi\ U\ \psi]$ in the sub-TKS containing only transitions whose duration is zero.

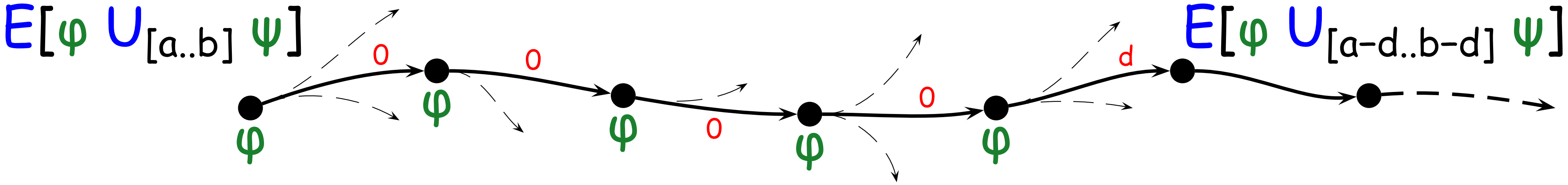
For $EG_{=0}\ \varphi$:

either there is a path where φ always holds, or there is a path verifying φ until a non-zero transition is eventually reached.

For $A[\varphi\ U_{=0}\ \psi]$:

both $A[\varphi\ U\ \psi]$ and $AF_{=0}\ \psi$ must hold.

Algorithm for $EU_{[a..b]}$



At each step i between 0 and b, we iteratively compute $max\_duration + 1$ sets of states s.t.

$$Set_i[j] = E[(\varphi\ U_{[i+j-1+a-b..i+j-1]}\ \psi) \wedge \text{first non-neg. duration} \geq j]$$

Thus

$$Set_i[1] = E[(\varphi\ U_{[i+a-b..i]}\ \psi)$$

and

$$Set_b[1] = E[(\varphi\ U_{[a..b]}\ \psi)$$

Algorithm for $EU_{\leq a}$

We could use $EU_{[0..a]}$, but it would be expensive: theoretically, $TCTL_{\leq,\geq}$ is easier than TCTL.

We design specific algorithms for $TCTL_{\leq,\geq}$, using only the "minimal duration" transition relation:

$$q \xrightarrow{d}\!\!\!\!\rightharpoondown q' \quad \text{iff} \quad \begin{cases} q \xrightarrow{d} q' \\ \forall d'.\ (d' \geq d \ \vee \ q \overset{d'}{\not\to} q') \end{cases}$$

This relation can be computed efficiently with BDDs.

Minimum and maximum delays

In order to compute the minimum delay between a set S and a set T, we recursively compute $EF_{\leq a}T$ until it intersects with S or it reaches a fixpoint.

We compute maximum delays by computing $EG_{\leq a}\neg T$, until it does not intersect with S anymore, or until it reaches a fixpoint.

What is NuSMV?

(NuSMV = symbolic CTL model-checker.

SMV: "Symbolic Model Verifier".

Developped at CMU (Pittsburgh, USA)

by Ken McMillan et al.

NuSMV: Open-source (LGPL) implementation of SMV

Developped at ITC-IRST (Trento, Italy)

by Alessandro Cimatti et al.

What is NuSMV?

(NuSMV = symbolic CTL model-checker.

$$\mathrm{CTL} \ni \varphi ::= p \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathrm{EX}\ \varphi \mid \mathrm{AX}\ \varphi \mid \mathrm{E}\,[\varphi \mathrm{U} \varphi] \mid \mathrm{A}\,[\varphi \mathrm{U} \varphi]$$

NuSMV also handles TCTL, assuming that one transition takes one time unit.

$$\mathrm{TCTL} \ni \psi ::= \psi \in \mathrm{CTL} \mid \mathrm{E}\,[\psi \mathrm{U}_{[c,d]} \psi] \mid \mathrm{A}\,[\psi \mathrm{U}_{[c,d]} \psi]$$

Coding TKS in NuSMV

We add one special integer variable, named duration.

We encode

$$\boxed{v_1, \ldots, v_n} \xrightarrow{d} \boxed{v'_1, \ldots, v'_n}$$

as

$$\boxed{\begin{array}{c} v_1, \ldots, v_n, \\ \text{duration} = \ldots \end{array}} \longrightarrow \boxed{\begin{array}{c} v'_1, \ldots, v'_n, \\ \text{duration} = d \end{array}}$$

Bridge crossing at night

- 4 persons are to cross the bridge.
- At most two at a time on the bridge.
- They have one lamp, needed when crossing.
- It takes 5, 10, 20 and 25 seconds for each of them to cross.

What is the optimal combination?

Bridge crossing at night

This example can easily be translated into NuSMV, but...

	NuSMV		TSMV	
	time	memory	time	memory
bridge	0.13 sec.	9 112 KB	0.02 sec.	1 280 KB
bridge × 10	4.14 sec.	18 860 KB	0.04 sec.	1 280 KB
bridge × 20	22.44 sec.	19 508 KB	0.07 sec.	1 280 KB
bridge × 50	176.55 sec.	28 640 KB	0.22 sec.	9 920 KB
bridge × 100	844.25 sec.	50 964 KB	0.45 sec.	11 580 KB
bridge × 200	4889.93 sec.	98 212 KB	1.03 sec.	14 956 KB

TSMV is (more or less) scale-insensitive.

Encouraging results

- relatively easy implementation into NuSMV,
- efficient model checking of timed Kripke structures,
- scale-insensitivity, memory and time consumption equivalent to those of NuSMV on classical Kripke Structure (with duration 1).

Future work

- more examples (NUSMV classical examples, UppAal examples, scheduling examples, ...)
- clean up the code, find better algorithms,
- counter-example generation.