FASTer acceleration of counter automata in practice

Sébastien Bardin

Joint work with Jérôme Leroux and Alain Finkel

LSV - CNRS & ENS de Cachan

Laboratoire Spécification et Vérification

Outline

1. Counter system model-checking
   (a) Presburger sets and automata
   (b) Acceleration
   (c) Heuristic
2. The tool FAST
   (a) Overview
   (b) Related tools
   (c) In practice
3. Verification of the TTP protocol with FAST
   (a) Presentation of the protocol
   (b) Verification for 1 fault and N stations
   (c) Polyhedral acceleration
   (d) Verification for 2 faults and N stations
4. Conclusion and future work

Laboratoire Spécification et Vérification

Counter systems model checking - 1

We focus on counter systems, which are automata extended with integer variables.
Counter systems allow to model a large range of complex systems:

- Abstract multi-threaded java programs,
- Embedded systems (TTP/C),
- All Broadcast Protocols,
- ...

Laboratoire Spécification et Vérification

Counter systems model checking - 1

We focus on counter systems, which are automata extended with integer variables.
Counter systems allow to model a large range of complex systems:

- Abstract multi-threaded java programs,
- Embedded systems (TTP/C),
- All Broadcast Protocols,
- ...

But checking safety properties is undecidable for counter systems!!

Counter systems model checking - 2

To overcome this problem, we have chosen:

- A symbolic representation of integer vectors by automata.
- An acceleration technique to help convergence:

$$\sigma^*(X_0) = \bigcup_{i=0}^{\infty} \sigma^i(X_0)$$

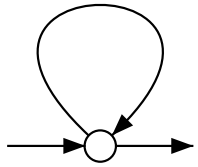
Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

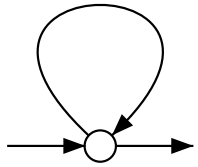
If $x \geq 0$ then $x := x + 2$



Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

If $x \geq 0$ then $x := x + 2$

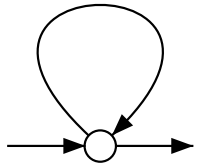


With the classical algorithm

Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

If $x \geq 0$ then $x := x + 2$



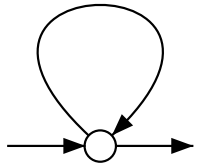
With the classical algorithm

If $S_0 = \{0\}$ then $Reach \supseteq \{0\}$.

Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

If $x \geq 0$ then $x := x + 2$



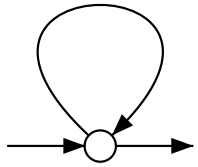
With the classical algorithm

If $S_0 = \{0\}$ then $Reach \supseteq \{0, 2\}$.

Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

If $x \geq 0$ then $x := x + 2$



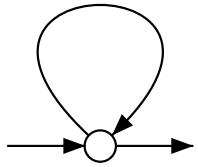
With the classical algorithm

If $S_0 = \{0\}$ then $Reach \supseteq \{0, 2, 4\}$.

Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

If $x \geq 0$ then $x := x + 2$



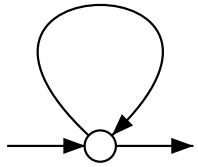
With the classical algorithm

If $S_0 = \{0\}$ then $Reach \supseteq \{0, 2, \ldots, 2.k\}$ and so on!!

Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

If $x \geq 0$ then $x := x + 2$

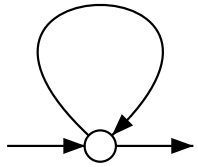


With Acceleration

Notion of Acceleration

acceleration to compute in one operation the iteration of a transition.

If $x \geq 0$ then $x := x + 2$



With Acceleration

If $S_0 = \{0\}$ then $Reach = 2.\mathbb{N}$.

Related work

- FAST: Bardin, Finkel, Leroux, Petrucci [FSTTCS02], [CAV03],
- LASH: Boigelot, Rassart, Wolper [CAV94], [SAS95], [CAV98], [TACAS00], [CAV03],
- TREX: Asarin, Bouajjani, Collomb-Annichini, Lakhnech, Sighireanu, [SPIN00], [SAS01], [CAV01].

Presburger sets and automata

- Presburger arithmetics is the first order additive theory $< \mathbb{N}^m, \leq, + >$, defined by

$$\phi ::= t \leq t | \neg\phi | \phi \vee \phi | \exists x.\phi | true$$
$$t ::= 0 | 1 | y | t - t | t + t.$$

Presburger sets and automata

- Presburger arithmetics is the first order additive theory $< \mathbb{N}^m, \leq, + >$, defined by

$$\phi ::= t \leq t | \neg\phi | \phi \vee \phi | \exists x.\phi | true$$
$$t ::= 0 | 1 | y | t - t | t + t.$$

- This theory is decidable, and Presburger sets can be represented symbolically by automata:
  - DFA [Boudet, Comon CAAP96],
  - NDD [Wolper, Boigelot TACAS00],
  - UBA [Leroux, INFINITY03].

Presburger sets and automata

- Presburger arithmetics is the first order additive theory $< \mathbb{N}^m, \leq, + >$, defined by

$$\phi ::= t \leq t | \neg\phi | \phi \vee \phi | \exists x.\phi | true$$
$$t ::= 0 | 1 | y | t - t | t + t.$$

- This theory is decidable, and Presburger sets can be represented symbolically by automata:
  - DFA [Boudet, Comon CAAP96],
  - NDD [Wolper, Boigelot TACAS00],
  - UBA [Leroux, INFINITY03].
- This representation is closed under $\cup, \cap, \ ^c$ and $\emptyset, \subseteq$ are decidable.
- Moreover the image of a Presburger set by an affine function is still a Presburger set.

Presburger sets and automata

- Presburger arithmetics is the first order additive theory $< \mathbb{N}^m, \leq, + >$, defined by

  $$\phi ::= t \leq t | \neg\phi | \phi \vee \phi | \exists x.\phi | true$$
  $$t ::= 0 | 1 | y | t - t | t + t.$$

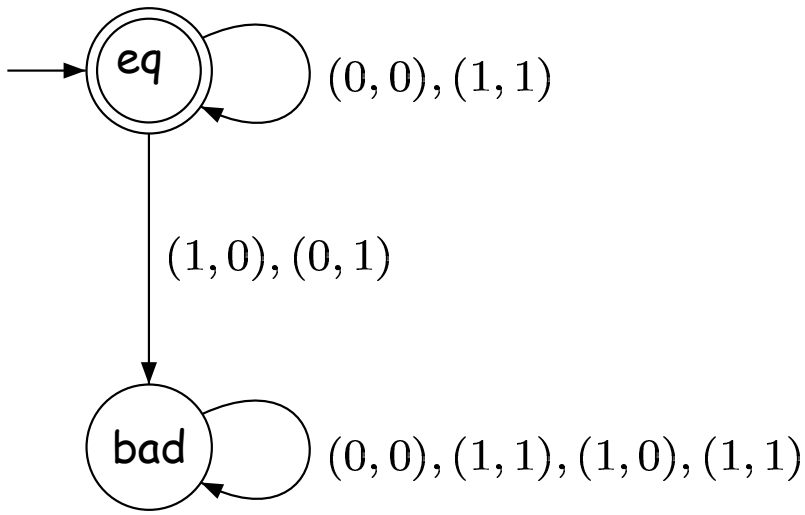
- This theory is decidable, and Presburger sets can be represented symbolically by automata:
  - DFA [Boudet, Comon CAAP96],
  - NDD [Wolper, Boigelot TACAS00],
  - UBA [Leroux, INFINITY03].
- This representation is closed under $\cup, \cap, \ ^c$ and $\emptyset, \subseteq$ are decidable.
- Moreover the image of a Presburger set by an affine function is still a Presburger set.

The automata representation provides an efficient framework to check safety properties on counter systems!!

Laboratoire Spécification et Vérification

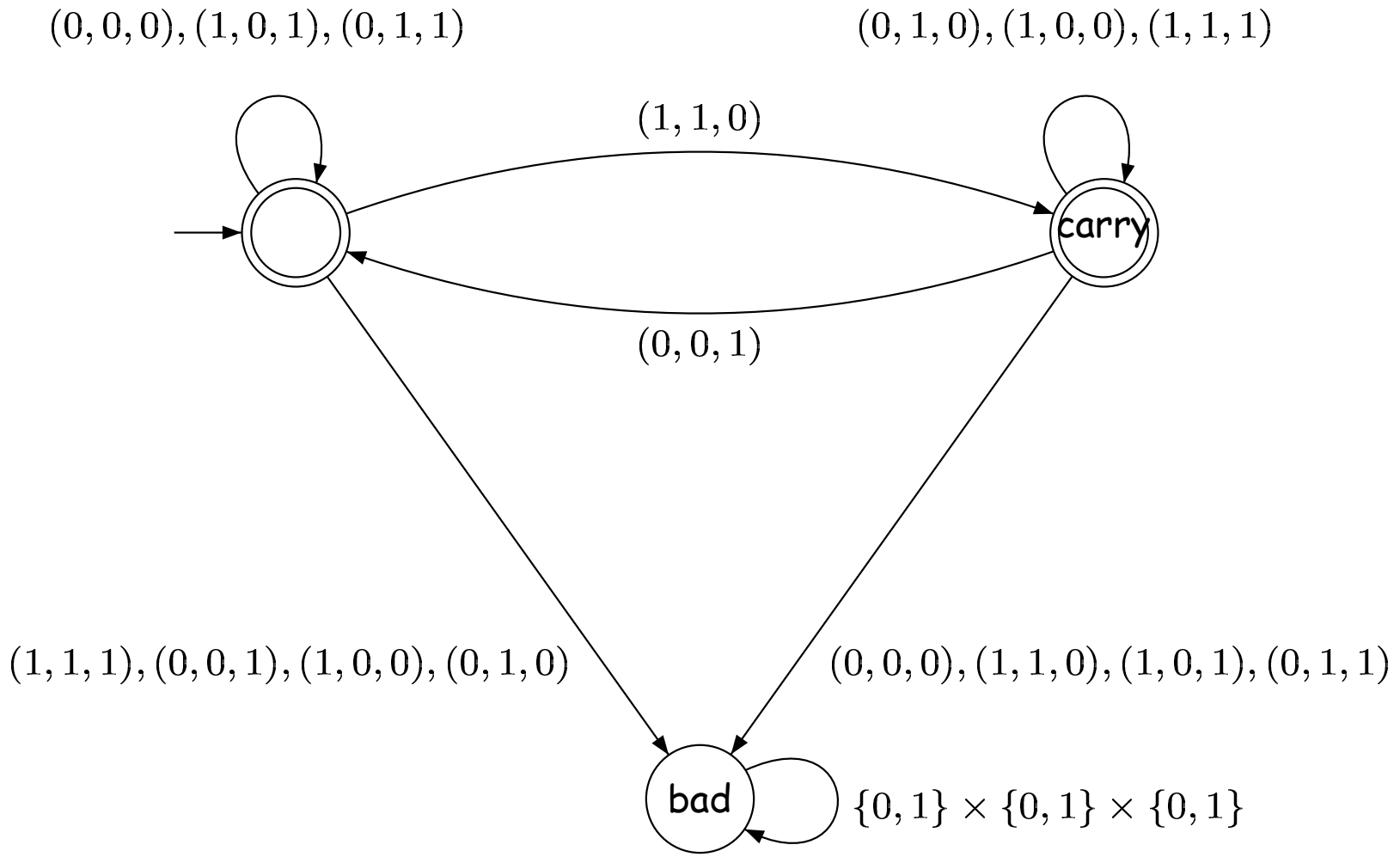
Automata representation in practice - 1

An automaton to represent $\{(x, y), x = y\}$ in basis 2.



Automata representation in practice - 2

An automaton to represent $\{(x, y, z), x + y = z\}$ in basis 2.



Laboratoire Spécification et Vérification

Counter systems

- A Presburger-linear function $f = (M, v, \mathcal{D})$ is defined by $\forall x \in \mathcal{D}, f(x) = M.x + v$ where the guard $\mathcal{D} \subseteq \mathbb{N}^m$ is a Presburger set.

Counter systems

- A Presburger-linear function $f = (M, v, \mathcal{D})$ is defined by $\forall x \in \mathcal{D}, f(x) = M.x + v$ where the guard $\mathcal{D} \subseteq \mathbb{N}^m$ is a Presburger set.
- A counter system L is a tuple $L = (\Sigma, f_\Sigma)$ where Σ is a finite alphabet of actions and $f_\Sigma = \{f_a; a \in \Sigma\}$ is a set of Presburger-linear functions.

Counter systems

- A Presburger-linear function $f = (M, v, \mathcal{D})$ is defined by $\forall x \in \mathcal{D}, f(x) = M.x + v$ where the guard $\mathcal{D} \subseteq \mathbb{N}^m$ is a Presburger set.
- A counter system L is a tuple $L = (\Sigma, f_\Sigma)$ where Σ is a finite alphabet of actions and $f_\Sigma = \{f_a; a \in \Sigma\}$ is a set of Presburger-linear functions.
- $\mathcal{M}_L$ is the multiplicative monoid generated by the set of square matrices $\{M_a;\ a \in \Sigma\}$ of a counter system L.

Counter systems

- A Presburger-linear function $f = (M, v, \mathcal{D})$ is defined by $\forall x \in \mathcal{D}, f(x) = M.x + v$ where the guard $\mathcal{D} \subseteq \mathbb{N}^m$ is a Presburger set.
- A counter system L is a tuple $L = (\Sigma, f_\Sigma)$ where Σ is a finite alphabet of actions and $f_\Sigma = \{f_a; a \in \Sigma\}$ is a set of Presburger-linear functions.
- $\mathcal{M}_L$ is the multiplicative monoid generated by the set of square matrices $\{M_a;\ a \in \Sigma\}$ of a counter system L.

Counter systems with a finite monoid have nice acceleration properties and appear to be well-spread in practice (transfer/reset/inhibitors Petri Nets, Broadcast protocols, ...)

Acceleration for counter systems

Let f be a function, and S a set, we define the acceleration of f by $f^*(S) = \bigcup_{i \in \mathbb{N}} f^i(S)$.

.

Acceleration for counter systems

Let f be a function, and S a set, we define the acceleration of f by $f^*(S) = \bigcup_{i \in \mathbb{N}} f^i(S)$.

- R_f^* is the relation associated with f^*.

Acceleration for counter systems

Let f be a function, and S a set, we define the acceleration of f by $f^*(S) = \bigcup_{i \in \mathbb{N}} f^i(S)$.

- R_f^* is the relation associated with f^*.
- Theorem [Finkel Leroux, FSTTCS02] For a Presburger-linear function $f = (M, v, \mathcal{D})$ with a finite monoid, R_f^* can be computed as a Presburger formula, of the form

$$R_f^* = \{(x, x') | x \in \mathcal{D} \wedge \exists k \geq 0 . x' = g^k(x) \wedge \forall i . 0 \leq i < k, g^i(x) \in \mathcal{D}\}$$

Idea of the construction

- $f = (M, v, \mathcal{D})$ with $< M >$ finite.
- $g : \mathbb{Q}^m \to \mathbb{Q}^m, \forall x \in \mathbb{Q}^m, g(x) = M.x + v$

Idea of the construction

- $f = (M, v, \mathcal{D})$ with $< M >$ finite.
- $g : \mathbb{Q}^m \to \mathbb{Q}^m, \forall x \in \mathbb{Q}^m, g(x) = M.x + v$
- $< M >$ is finite, so there exists $(a, b) \in \mathbb{N} \times \mathbb{N}$ such that $M^{a+b} = M^a$

Idea of the construction

- $f = (M, v, \mathcal{D})$ with $< M >$ finite.
- $g : \mathbb{Q}^m \to \mathbb{Q}^m, \forall x \in \mathbb{Q}^m, g(x) = M.x + v$
- $< M >$ is finite, so there exists $(a, b) \in \mathbb{N} \times \mathbb{N}$ such that $M^{a+b} = M^a$
- Notice that $\forall n \in \mathbb{N}, \forall x \in \mathbb{Q}^m, g^{a+n.b} = g^a(x) + n.M^a.g^b(0)$

Idea of the construction

- $f = (M, v, \mathcal{D})$ with $< M >$ finite.
- $g : \mathbb{Q}^m \to \mathbb{Q}^m, \forall x \in \mathbb{Q}^m, g(x) = M.x + v$
- $< M >$ is finite, so there exists $(a, b) \in \mathbb{N} \times \mathbb{N}$ such that $M^{a+b} = M^a$
- Notice that $\forall n \in \mathbb{N}, \forall x \in \mathbb{Q}^m, g^{a+n.b} = g^a(x) + n.M^a.g^b(0)$
- $G = \{(i, x, x') \in \mathbb{N} \times \mathbb{Z}^m \times \mathbb{Z}^m, x' = g^i(x)\} \Longleftrightarrow$

$$\bigvee_{r=0}^{a-1} [x' = g^r(x) \wedge i = r] \bigvee_{r=0}^{b-1} \exists n \geq 0 [(x' = g^{a+r}(x) + n.M^{a+r}.g^b(0)) \wedge (i = a + r + n.b)]$$

Idea of the construction

- $f = (M, v, \mathcal{D})$ with $< M >$ finite.
- $g : \mathbb{Q}^m \to \mathbb{Q}^m, \forall x \in \mathbb{Q}^m, g(x) = M.x + v$
- $< M >$ is finite, so there exists $(a, b) \in \mathbb{N} \times \mathbb{N}$ such that $M^{a+b} = M^a$
- Notice that $\forall n \in \mathbb{N}, \forall x \in \mathbb{Q}^m, g^{a+n.b} = g^a(x) + n.M^a.g^b(0)$
- $G = \{(i, x, x') \in \mathbb{N} \times \mathbb{Z}^m \times \mathbb{Z}^m, x' = g^i(x)\} \Longleftrightarrow$

$$\bigvee_{r=0}^{a-1} [x' = g^r(x) \wedge i = r] \bigvee_{r=0}^{b-1} \exists n \geq 0 [(x' = g^{a+r}(x) + n.M^{a+r}.g^b(0)) \wedge (i = a + r + n.b)]$$

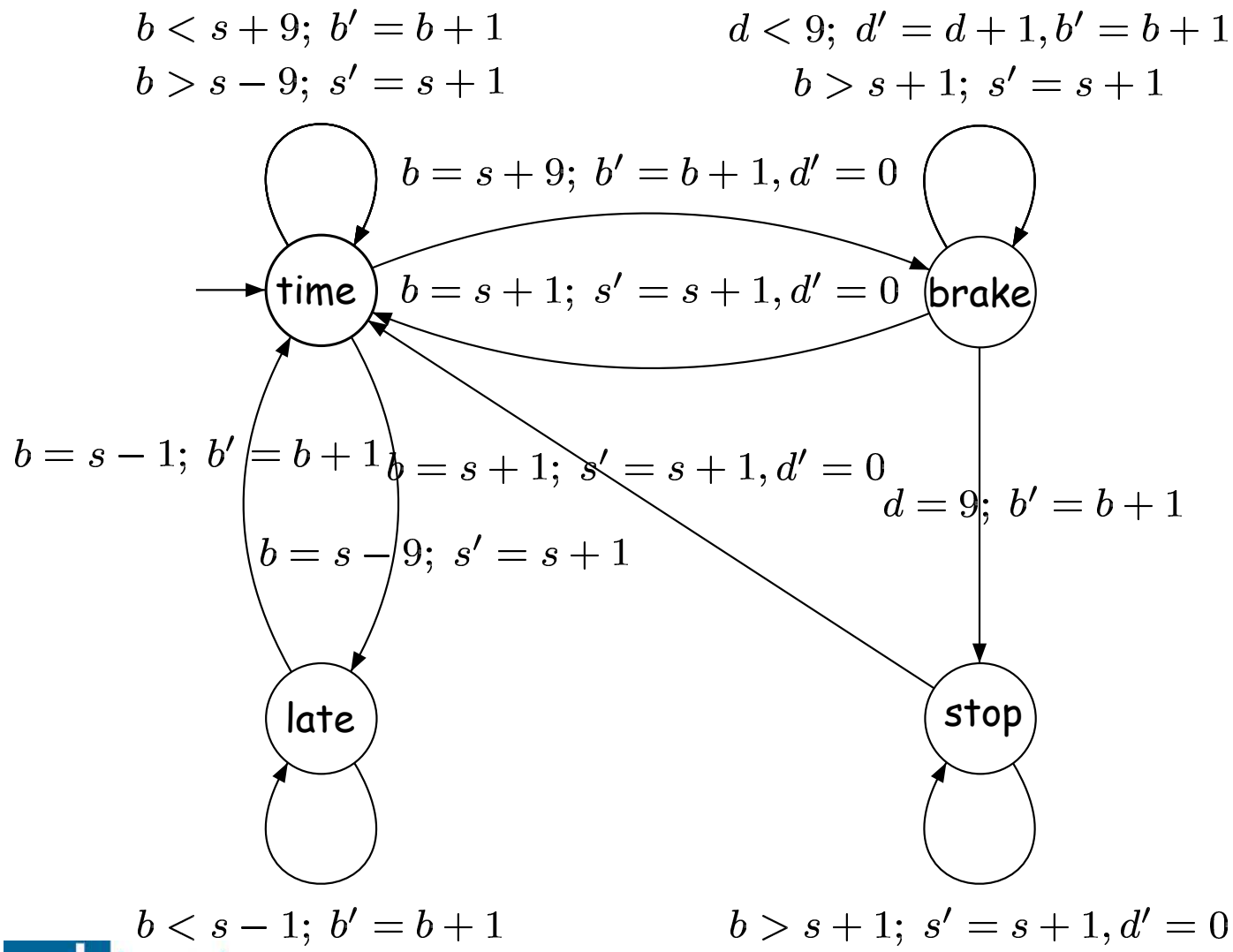
- Finally we have $R_f^* = \{(x, x'), \exists i \geq 0, x' = f^i(x)\} \Longleftrightarrow$

$$\{(x, x'), \exists i \geq 0 [(i, x, x') \in G \wedge (\forall k (0 \leq k < i), \exists x'' \in \mathcal{D}, (k, x, x'') \in G)]\}$$

R_f^* is a Presburger set!!

Laboratoire Spécification et Vérification

How to find out the accelerations ?



Initial configuration:
state=time $\wedge\ b = s \wedge s = d \wedge d = 0$

Property to check :
$|b - s| \leq 20$
always holds.

Laboratoire Spécification et Vérification

Reduction result

Theorem [reduction, Finkel Leroux FSTTCS02]

Any acceleration of functions in a finite set C of Presburger-linear functions can be reduced to the acceleration of functions in a reduced set $[C]$, such that the cardinal of $[C_k]$ is polynomial in k.

$$f_1 = (M, v, \mathcal{D}_1), f_2 = (M, v, \mathcal{D}_2) \longrightarrow f_{1\otimes 2} = (M, v, \mathcal{D}_1 \cup \mathcal{D}_2)$$

Heuristic

- Extension of the classic algorithm, adding cycles (*meta-transitions*).
- 2 problems:
  - find good cycles
  - avoid automata explosion

Heuristic

- Extension of the classic algorithm, adding cycles (*meta-transitions*).
- 2 problems:
  - find good cycles $\rightarrow$ incremental computation and reduction
  - avoid automata explosion

Heuristic

- Extension of the classic algorithm, adding cycles (*meta-transitions*).
- 2 problems:
  - find good cycles $\rightarrow$ incremental computation and reduction
  - avoid automata explosion $\rightarrow$ minimization step

Heuristic

- Extension of the classic algorithm, adding cycles (*meta-transitions*).
- 2 problems:
  - find good cycles $\rightarrow$ incremental computation and reduction
  - avoid automata explosion $\rightarrow$ minimization step

1. $k \leftarrow 1$
2. Compute C_k, the reduced set of cycles of length $\leq k$
3. Use the search algorithm with S_0 and $L \cup C_k$
4. if a fixpoint S is found then return S
   else (the stop criterion is met) do $k \leftarrow k+1, goto\ (2)$

Heuristic -2

The search algorithm: 2 nested greedy algorithms

$S \leftarrow S_0$

while there exists f such that $f^*(S)$ reaches new states do

    $S \leftarrow f^*(S)$

end while

return S

Heuristic -2

The search algorithm: 2 nested greedy algorithms

$S \leftarrow S_0$
while there exists f such that $f^*(S)$ reaches new states do
    $S \leftarrow f^*(S)$
    while there exists f such that $|\mathcal{A}(f^*(S))| < |\mathcal{A}(S)|$ do
        $S \leftarrow f^*(S)$
    end while
end while
return S

Fast

We implement our results in the tool FAST.

FAST is a tool:

- with a powerful model,
- that automatically computes the reachability set in most practical cases,
- easy to use thanks to the GUI interface.

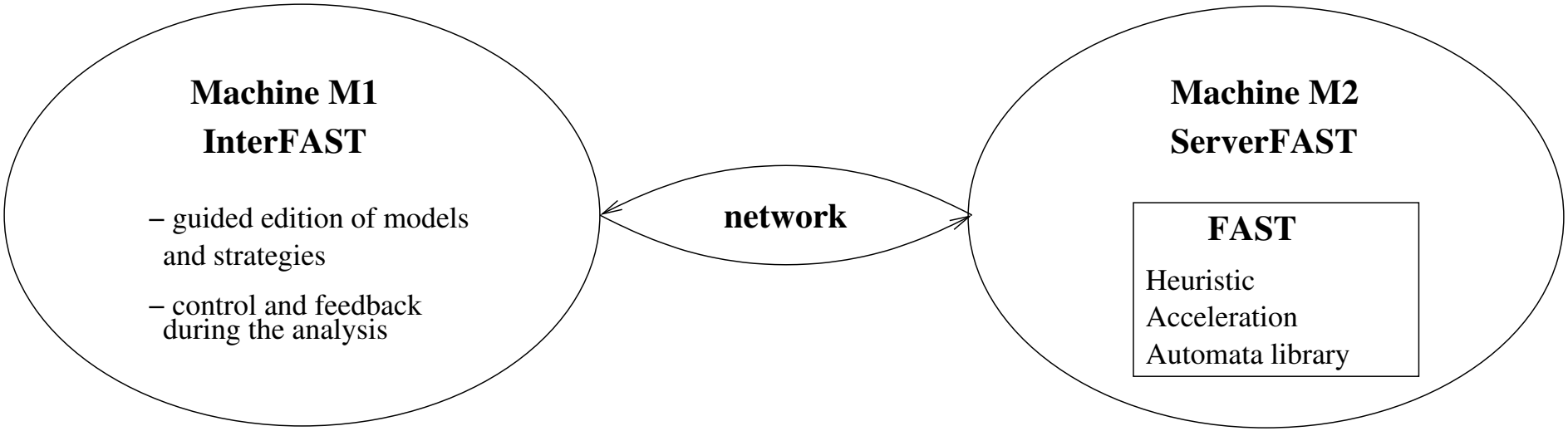
Laboratoire Spécification et Vérification

Tools with acceleration and counters

	variable type	guards	actions	acceleration	auto. cycle search
FAST	$\mathbb{N}$	Presburger	$\vec{x}' = M.\vec{x} + \vec{v}$	yes	yes
LASH	$\mathbb{Z}$	convex sets	$\vec{x}' = M.\vec{x} + \vec{v}$	yes	no
	$\mathbb{R}$	convex sets	$\vec{x}' = M.\vec{x} + \vec{v}$	no	
TREX	$\mathbb{Z}$	$\bigwedge \begin{cases} x_i \leq x_j + c \\ x_i \leq c \\ x_i \geq c \end{cases}$	$\bigwedge \begin{cases} x_i = x_j + c \\ x_i = c \end{cases}$	yes	yes
	$\mathbb{R}$		$\bigwedge \begin{cases} x_i = x_j \\ x_i = 0 \end{cases}$	yes	yes

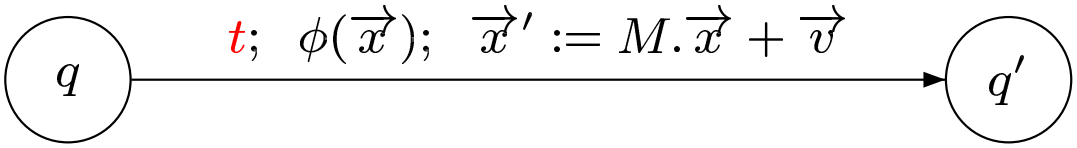
Laboratoire Spécification et Vérification

Fast architecture



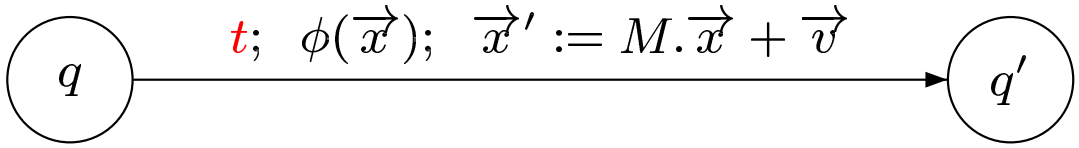
Fast Inputs

Input Model: A counter system such that each transition t is:



Fast Inputs

Input Model: A counter system such that each transition t is:



Input Strategy: A high level query language with

- Automatic computation of reachability sets,
- Presburger solver,
- Modular analyzer.

Case Studies

80% of 40 counter systems (mainly taken from ALV, BABYLON, TREX) have been automatically analysed.

In particular:

- Abstract multi-threaded java programs,
- Embedded systems (TTP/C),
- All Broadcast Protocols,
- Complex toy examples (Swimming Pool),

The TTP protocol - overview

- From car industry.
- Communications between embedded microprocessors (*stations*).
- Clique avoidance mechanism to prevent the partitioning of valid stations after a failure.

The TTP protocol - overview

- From car industry.
- Communications between embedded microprocessors (*stations*).
- Clique avoidance mechanism to prevent the partitioning of valid stations after a failure.
- N stations communicating through a shared bus
  - ➤ messages are broadcast,
  - ➤ static time slots to send and receive messages

The TTP protocol - overview

- From car industry.
- Communications between embedded microprocessors (*stations*).
- Clique avoidance mechanism to prevent the partitioning of valid stations after a failure.
- N stations communicating through a shared bus
  - ➤ messages are broadcast,
  - ➤ static time slots to send and receive messages
- Idea:
  - ➤ a station which considers itself as faulty becomes inactive.
  - ➤ a station which receives more invalid messages than valid ones must be faulty.

The TTP protocol

- M a boolean matrix of size $N \times N$
- C_a(*ack*) and C_f(*fail*) integer vectors of size N
- station i receiving message m_j from station j
- station i sending

The TTP protocol

- M a boolean matrix of size $N \times N$
- C_a(*ack*) and C_f(*fail*) integer vectors of size N
- station i receiving message m_j from station j
  - if m_j correctly received then $C_a[i]++$
  - else $C_f[i]++, M[i][j] := 0$
- station i sending

The TTP protocol

- M a boolean matrix of size $N \times N$
- C_a(*ack*) and C_f(*fail*) integer vectors of size N
- station i receiving message m_j from station j
  - if m_j correctly received then $C_a[i]++$
  - else $C_f[i]++, M[i][j] := 0$
- station i sending
  - if $C_a[i] > C_f[i]$ then $C_a[i] := 0, C_f[i] := 0, !M[i]$
  - else $M[i][i] := 0$, becomes inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	1	1	1	4	0
s_1	1	1	1	1	3	0
s_2	1	1	1	1	2	0
• s_3	1	1	1	1	1	0

ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	1	1	1	4	0
s_1	1	1	1	1	3	0
s_2	1	1	1	1	2	0
s_3	1	1	1	1	1	0

ack fail inactive

A failure occurs while s_0 is sending.

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	1	1	1	1	0
s_1	0	1	1	1	3	1
s_2	1	1	1	1	3	0
s_3	0	1	1	1	1	1

ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	1	1	1	1	0
s_1	0	1	1	1	3	1
s_2	1	1	1	1	3	0
s_3	0	1	1	1	1	1

- ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	1	1	1
s_1	0	1	1	1	1	0
s_2	1	0	1	1	3	1
s_3	0	1	1	1	2	1

- ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	1	1	1
s_1	0	1	1	1	1	0
s_2	1	0	1	1	3	1
s_3	0	1	1	1	2	1

- ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	1	2	1
s_1	0	1	0	1	1	1
s_2	1	0	1	1	1	0
s_3	0	1	0	1	2	2

- ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	1	2	1
s_1	0	1	0	1	1	1
s_2	1	0	1	1	1	0
s_3	0	1	0	1	2	2

- ack fail inactive

$C_a[s_3] < C_f[s_3]$ then s_3 becomes inactive.

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	0	2	1
s_1	0	1	0	0	1	1
s_2	1	0	1	0	1	0
s_3	0	0	0	0	0	0

- ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	0	2	1
s_1	0	1	0	0	1	1
s_2	1	0	1	0	1	0
s_3	0	0	0	0	0	0

ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	0	1	0
s_1	0	1	0	0	1	2
s_2	1	0	1	0	2	0
s_3	0	0	0	0	0	0

ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	0	1	0
s_1	0	1	0	0	1	2
s_2	1	0	1	0	2	0
s_3	0	0	0	0	0	0

ack fail inactive

$C_a[s_1] < C_f[s_1]$ then s_1 becomes inactive.

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	0	1	0
s_1	0	0	0	0	0	0
s_2	1	0	1	0	2	0
s_3	0	0	0	0	0	0

- ack fail inactive

The TTP protocol - In practice

stations	s_0	s_1	s_2	s_3	C_a	C_f
s_0	1	0	1	0	1	0
s_1	0	0	0	0	0	0
s_2	1	0	1	0	2	0
s_3	0	0	0	0	0	0

- ack fail inactive

Valid stations belongs to the same clique!!

Validation of the TTP protocol

- A protocol difficult to validate.
- Merceron and Bouajjani (FTRTFT'02):

Validation of the TTP protocol

- A protocol difficult to validate.
- Merceron and Bouajjani (FTRTFT'02):
  - ➤ Manual proof of correctness (N stations, k faults).
  - ➤ Provide a family of abstractions depending on the number of faults.
  - ➤ Semi-automatic verification with tools LASH and ALV (N stations, 1 fault).

Validation of the TTP protocol

- A protocol difficult to validate.
- Merceron and Bouajjani (FTRTFT'02):
  - ➤ Manual proof of correctness (N stations, k faults).
  - ➤ Provide a family of abstractions depending on the number of faults.
  - ➤ Semi-automatic verification with tools LASH and ALV (N stations, 1 fault).
    - large parametric counter automaton (16 transitions)
    - complex guards

Validation of the TTP protocol

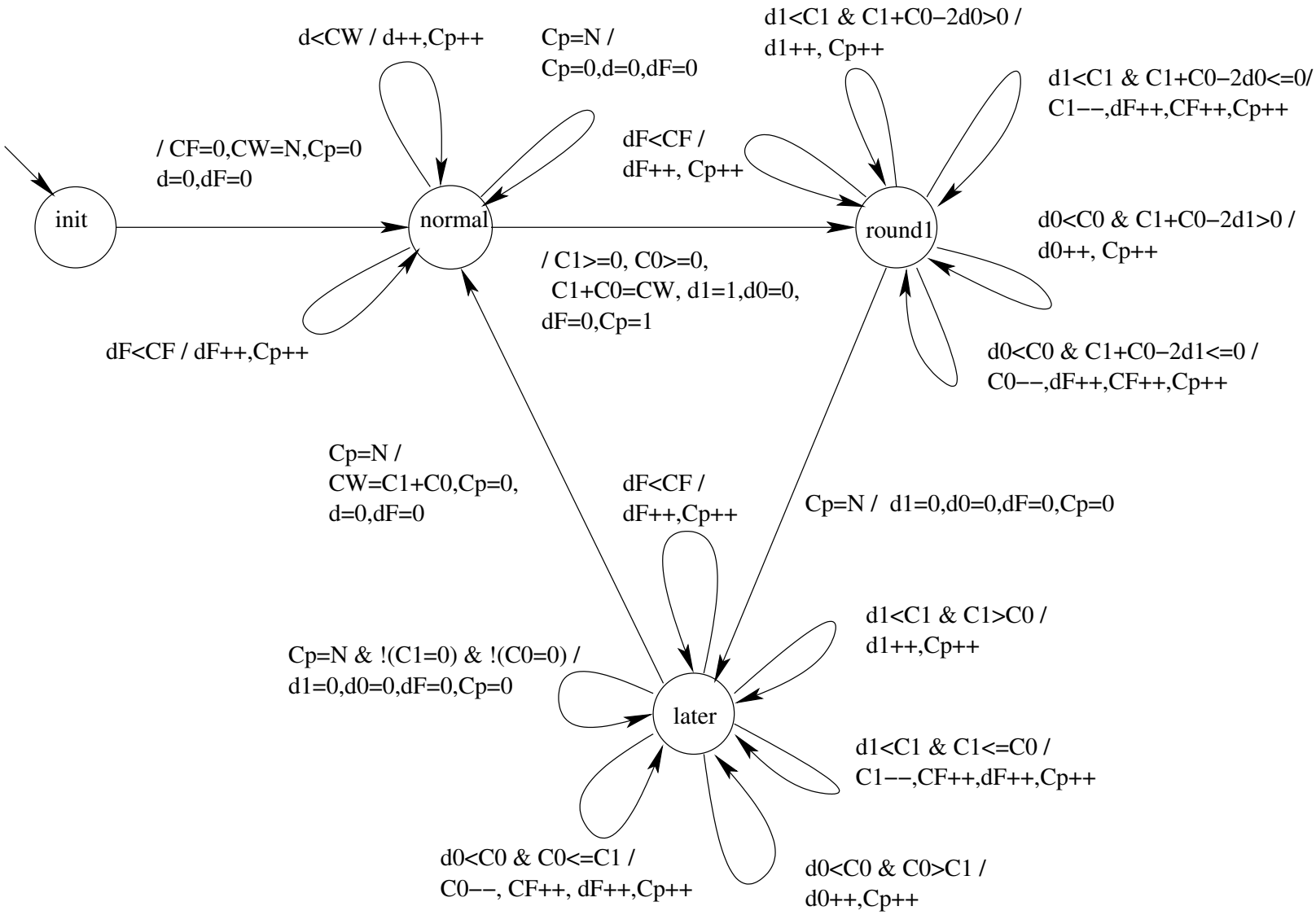
- A protocol difficult to validate.
- Merceron and Bouajjani (FTRTFT'02):
  - ➤ Manual proof of correctness (N stations, k faults).
  - ➤ Provide a family of abstractions depending on the number of faults.
  - ➤ Semi-automatic verification with tools LASH and ALV (N stations, 1 fault).
    - large parametric counter automaton (16 transitions)
    - complex guards
- Few tools are adapted.

Validation of the TTP protocol

- A protocol difficult to validate.
- Merceron and Bouajjani (FTRTFT'02):
  - ➤ Manual proof of correctness (N stations, k faults).
  - ➤ Provide a family of abstractions depending on the number of faults.
  - ➤ Semi-automatic verification with tools LASH and ALV (N stations, 1 fault).
    - large parametric counter automaton (16 transitions)
    - complex guards
- Few tools are adapted.

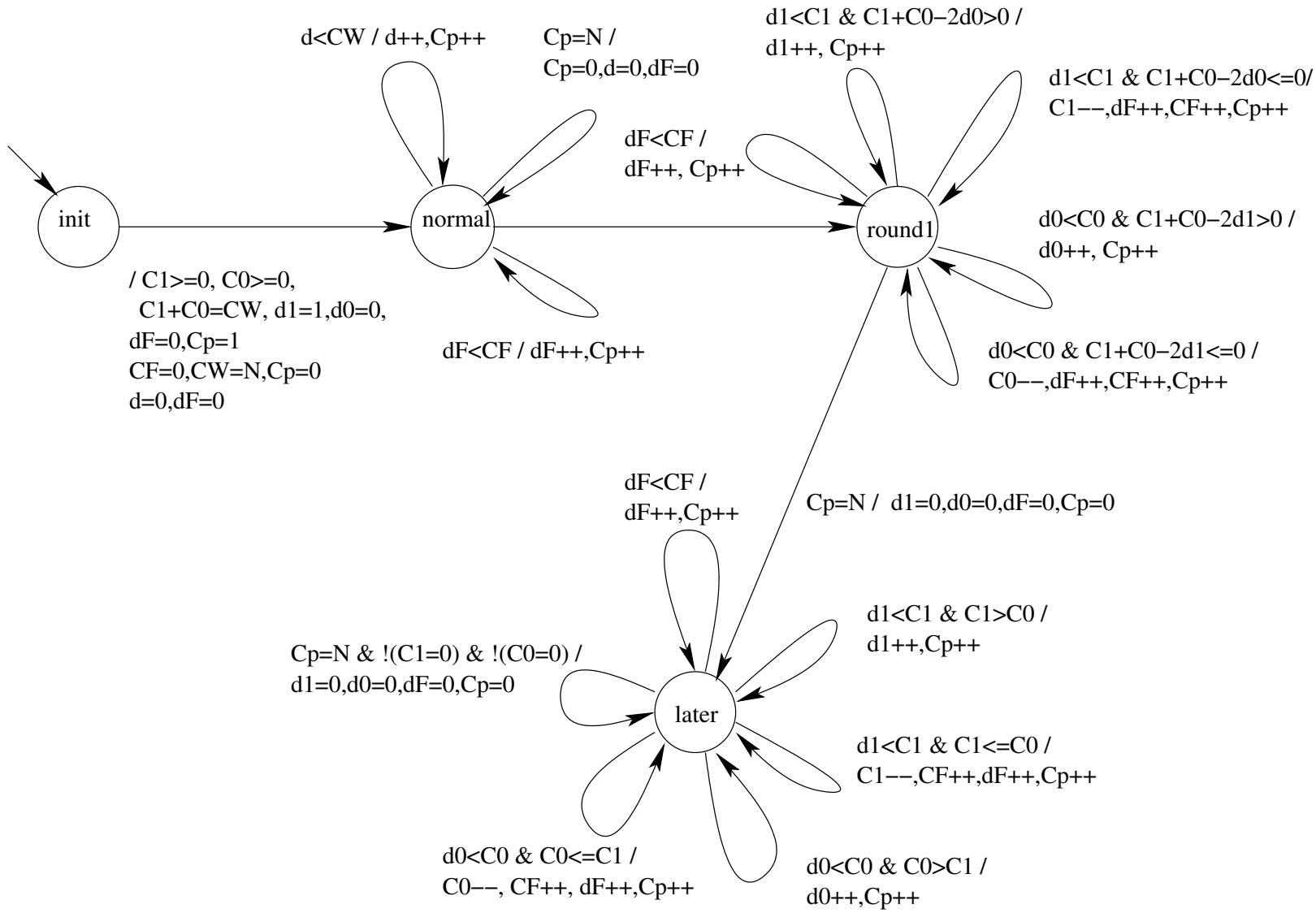
Interesting to test FAST on the TTP.

Model for the TTP, 1 fault N stations



Laboratoire Spécification et Vérification

Model for the TTP, 1 fault N stations



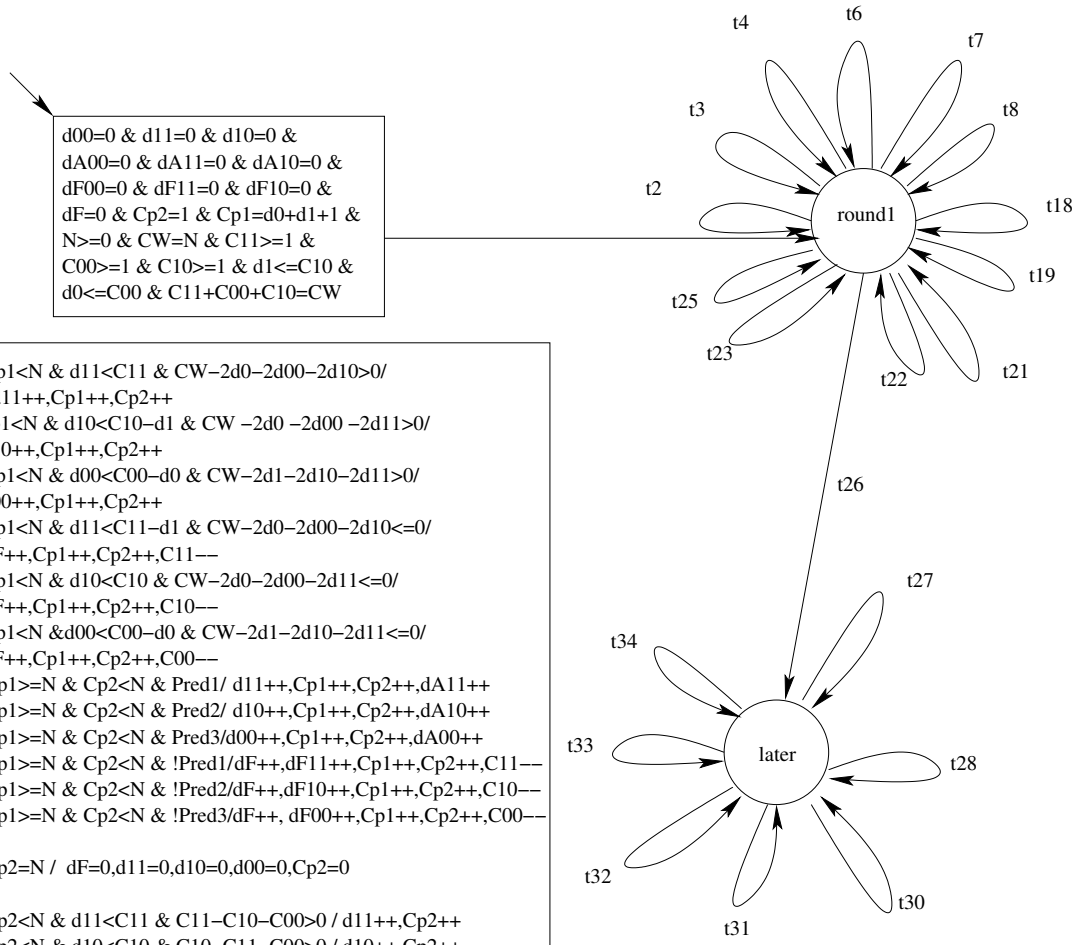
Laboratoire Spécification et Vérification

Verification with Fast, 1 fault

A large model: 16 transitions, 9 variables

- easy to describe in FAST input model,
- full automatic verification (no intermediate property)
  - ➤ the exact reachability set is computed
  - ➤ the property is verified
- cycles of length 1, the reachability set has 27,932 nodes
- on a pentium 4 (2.4 GHz) with 1 Gbyte RAM, computation takes 940 sec. and 73 Mbytes.

Model for the TTP, 2 faults N stations



t2 : Cp1<N & d11<C11 & CW−2d0−2d00−2d10>0/
d11++,Cp1++,Cp2++
t3: Cp1<N & d10<C10−d1 & CW −2d0 −2d00 −2d11>0/
d10++,Cp1++,Cp2++
t4 : Cp1<N & d00<C00−d0 & CW−2d1−2d10−2d11>0/
d00++,Cp1++,Cp2++
t6 : Cp1<N & d11<C11−d1 & CW−2d0−2d00−2d10<=0/
dF++,Cp1++,Cp2++,C11−−
t7 : Cp1<N & d10<C10 & CW−2d0−2d00−2d11<=0/
dF++,Cp1++,Cp2++,C10−−
t8 : Cp1<N &d00<C00−d0 & CW−2d1−2d10−2d11<=0/
dF++,Cp1++,Cp2++,C00−−
t18 : Cp1>=N & Cp2<N & Pred1/ d11++,Cp1++,Cp2++,dA11++
t19 : Cp1>=N & Cp2<N & Pred2/ d10++,Cp1++,Cp2++,dA10++
t21 : Cp1>=N & Cp2<N & Pred3/d00++,Cp1++,Cp2++,dA00++
t22 : Cp1>=N & Cp2<N & !Pred1/dF++,dF11++,Cp1++,Cp2++,C11−−
t23 : Cp1>=N & Cp2<N & !Pred2/dF++,dF10++,Cp1++,Cp2++,C10−−
t25 : Cp1>=N & Cp2<N & !Pred3/dF++, dF00++,Cp1++,Cp2++,C00−−

t26 : Cp2=N / dF=0,d11=0,d10=0,d00=0,Cp2=0

t27 : Cp2<N & d11<C11 & C11−C10−C00>0 / d11++,Cp2++
t28 : Cp2<N & d10<C10 & C10−C11−C00>0 / d10++,Cp2++
t30 : Cp2<N & d00<C00 & C00−C10−C11>0 / d00++, Cp2++
t31 : Cp2<N & d11<C11 & C11−C10−C00<=0 /
C11−−,Cp2++,dF++,CF++
t32 : Cp2<N & d10<C10 & C10−C11−C00<=0 /
C10−−,Cp2++,CF++,dF++
t33 : Cp2<N & d00<C00 & C00−C10−C11<=0 /
C00−−,Cp2++,CF++,dF++
t34 : Cp2<N & dF<CF / Cp2++,dF++

Pred1 :
d1+d11−dA11−dF11−dA10−dF10−d0−d10−d00+dA00+dF00>0
Pred2 :
d1+d10−dA10−dF10−dA11−dF11−d0−d11−d00+dA00+dF00>0
Pred3 :
d0+d00−dA00−dF00−d1−d11−d10+dA11+dA10+dF11+dF10>0

Laboratoire Spécification et Vérification

Verification with Fast, 2 faults

- A very large model: 20 transitions, 18 variables
- Guards are very complex.

Verification with Fast, 2 faults

- A very large model: 20 transitions, 18 variables
- Guards are very complex.
- When computing the acceleration relation of transition t_{25}, the internal representation exceeds its limits and FAST stops.

Verification with Fast, 2 faults

- A very large model: 20 transitions, 18 variables
- Guards are very complex.
- When computing the acceleration relation of transition t_{25}, the internal representation exceeds its limits and FAST stops.
  - ➤ Intermediate automata have more than 2^{24} states!!

Verification with Fast, 2 faults

- A very large model: 20 transitions, 18 variables
- Guards are very complex.
- When computing the acceleration relation of transition t_{25}, the internal representation exceeds its limits and FAST stops.
  - ➤ Intermediate automata have more than 2^{24} states!!

Our acceleration formula is too expensive in this case!!

Faster acceleration

- Almost all the transitions are translations over convex polyhedra

Faster acceleration

- Almost all the transitions are translations over convex polyhedra
  - ➤ Don't need to test if all the predecessors are in the guard.

Faster acceleration

- Almost all the transitions are translations over convex polyhedra
  - ➤ Don't need to test if all the predecessors are in the guard.
- We can use a simpler acceleration formula:

Faster acceleration

- Almost all the transitions are translations over convex polyhedra
  - ➤ Don't need to test if all the predecessors are in the guard.
- We can use a simpler acceleration formula:
  - ➤ $R_f^* = \{(x, x') | x \in \mathcal{D} \wedge \exists k \geq 0 . x' = g^k(x) \wedge \forall i . 0 \leq i < k, g^i(x) \in \mathcal{D}\}$ (1)

Faster acceleration

- Almost all the transitions are translations over convex polyhedra
  - ➤ Don't need to test if all the predecessors are in the guard.
- We can use a simpler acceleration formula:
  - ➤ $R_f^* = \{(x, x') | x \in \mathcal{D} \wedge \exists k \geq 0 . x' = g^k(x) \wedge \forall i . 0 \leq i < k, g^i(x) \in \mathcal{D}\}$ (1)
  - ➤ $R_f^* = \{(x, x') | x \in \mathcal{D} \wedge \exists k \geq 0 . x' = g^k(x) \wedge k > 0 \Rightarrow g^{k-1}(x) \in \mathcal{D}\}$ (2)

Faster acceleration

- Almost all the transitions are translations over convex polyhedra
  - ➤ Don't need to test if all the predecessors are in the guard.
- We can use a simpler acceleration formula:
  - ➤ $R_f^* = \{(x, x') | x \in \mathcal{D} \wedge \exists k \geq 0 . x' = g^k(x) \wedge \forall i . 0 \leq i < k, g^i(x) \in \mathcal{D}\}$ (1)
  - ➤ $R_f^* = \{(x, x') | x \in \mathcal{D} \wedge \exists k \geq 0 . x' = g^k(x) \wedge k > 0 \Rightarrow g^{k-1}(x) \in \mathcal{D}\}$ (2)
  - ➤ $R_f^* = \{(x, x') \in D \times (D + v);\ x' - x \in \mathbb{N}.v\}$ $(polyhedral\ acceleration)$

Faster acceleration

- Almost all the transitions are translations over convex polyhedra
  - ➤ Don't need to test if all the predecessors are in the guard.
- We can use a simpler acceleration formula:
  - ➤ $R_f^* = \{(x, x') | x \in \mathcal{D} \wedge \exists k \geq 0 . x' = g^k(x) \wedge \forall i . 0 \leq i < k, g^i(x) \in \mathcal{D}\}$ (1)
  - ➤ $R_f^* = \{(x, x') | x \in \mathcal{D} \wedge \exists k \geq 0 . x' = g^k(x) \wedge k > 0 \Rightarrow g^{k-1}(x) \in \mathcal{D}\}$ (2)
  - ➤ $R_f^* = \{(x, x') \in D \times (D + v);\ x' - x \in \mathbb{N}.v\}$ $(polyhedral\ acceleration)$

The polyhedral acceleration is quadratic in the size of the function while the generic formula (1) is at most elementary in the size of the function.

Polyhedral acceleration in practice

We use the polyhedral acceleration on the TTP with 2 faults.

Polyhedral acceleration in practice

We use the polyhedral acceleration on the TTP with 2 faults.

- Acceleration relations are computed.

Polyhedral acceleration in practice

We use the polyhedral acceleration on the TTP with 2 faults.

- Acceleration relations are computed.
  - For t_{25} it takes 18 sec, 460 Mbytes (413,447 states!!)

Polyhedral acceleration in practice

We use the polyhedral acceleration on the TTP with 2 faults.

- Acceleration relations are computed.
  - For t_{25} it takes 18 sec, 460 Mbytes (413,447 states!!)
- For a small fixed number of stations (about 10), the reachability set is computed.

Polyhedral acceleration in practice

We use the polyhedral acceleration on the TTP with 2 faults.

- Acceleration relations are computed.
  - ➤ For t_{25} it takes 18 sec, 460 Mbytes (413,447 states!!)
- For a small fixed number of stations (about 10), the reachability set is computed.
- For an arbitrary value of N, the intermediate automata exceed the limit.

Polyhedral acceleration in practice

We use the polyhedral acceleration on the TTP with 2 faults.

- Acceleration relations are computed.
  - ➤ For t_{25} it takes 18 sec, 460 Mbytes (413,447 states!!)
- For a small fixed number of stations (about 10), the reachability set is computed.
- For an arbitrary value of N, the intermediate automata exceed the limit.
- We have to use an overapproximation for $N \geq 0$.

Polyhedral acceleration in practice

We use the polyhedral acceleration on the TTP with 2 faults.

- Acceleration relations are computed.
  - For t_{25} it takes 18 sec, 460 Mbytes (413,447 states!!)
- For a small fixed number of stations (about 10), the reachability set is computed.
- For an arbitrary value of N, the intermediate automata exceed the limit.
- We have to use an overapproximation for $N \geq 0$.
  - simplify some guards,
  - remove some variables,
  - modular analysis.

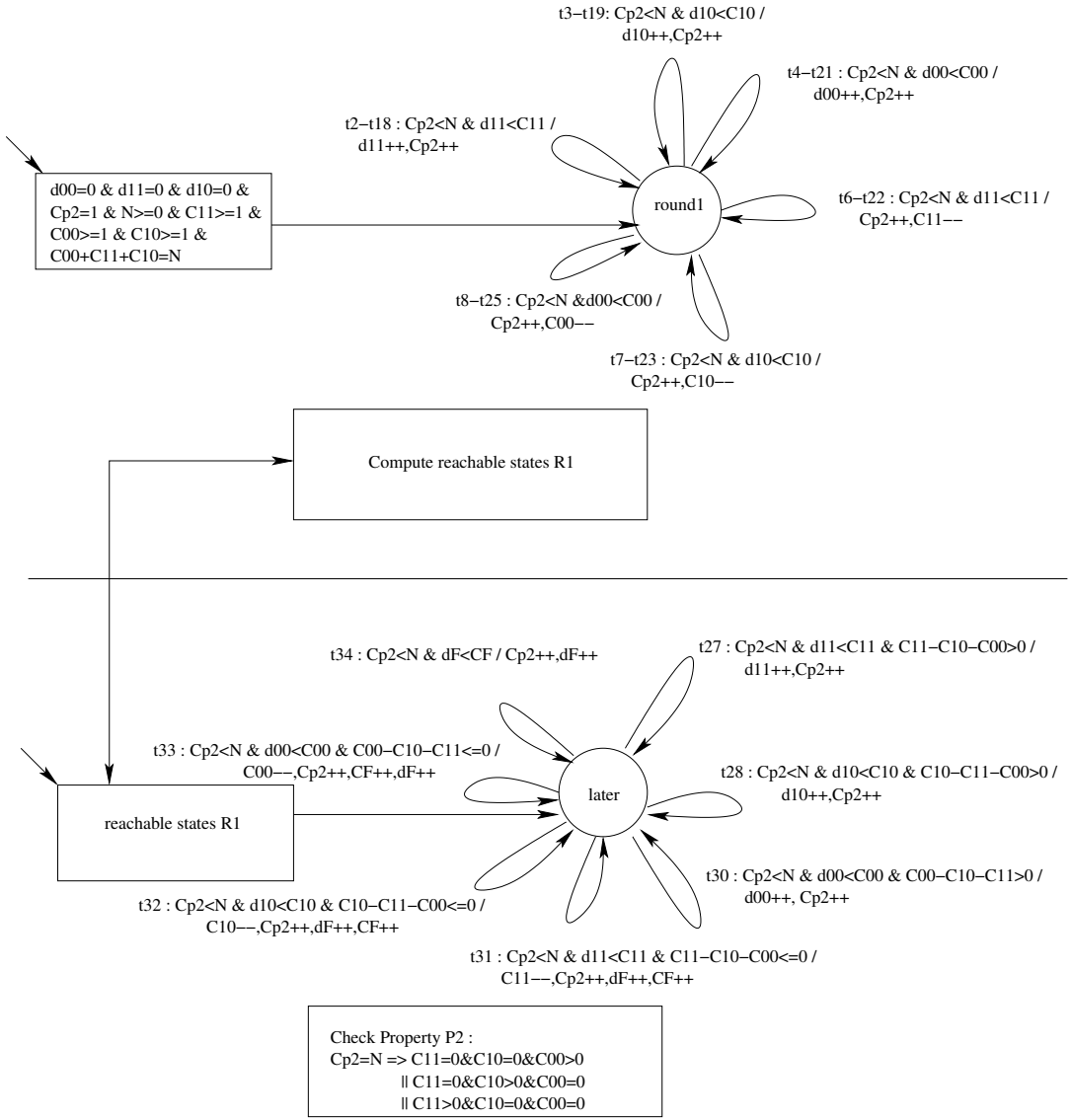
Polyhedral acceleration in practice

We use the polyhedral acceleration on the TTP with 2 faults.

- Acceleration relations are computed.
  - For t_{25} it takes 18 sec, 460 Mbytes (413,447 states!!)
- For a small fixed number of stations (about 10), the reachability set is computed.
- For an arbitrary value of N, the intermediate automata exceed the limit.
- We have to use an overapproximation for $N \geq 0$.
  - simplify some guards,
  - remove some variables,
  - modular analysis.

The protocol is verified with FAST for 2 fauts and N stations.

Abstraction for the TTP with 2 faults



Laboratoire Spécification et Vérification

Results

	Presburger acceleration		polyhedral acceleration		
	time1 seconds	memory1 Mbytes	time2 seconds	memory2 Mbytes	number of states
1 fault, N stations	940	73	600	63	27,932
2 faults, 5 stations	↑	↑	446	588	5,684
2 faults, 10 stations	↑	↑	12,365	588	273,427
2 faults, 15 stations	↑	↑	↑	↑	↑
2 faults, N stations	↑	↑	↑	↑	↑
2 faults, N stations (abstraction)	210	200	175	200	11,036

Conclusion and Future Works

Conclusion:

- Polyhedral acceleration appears to be interesting in practice,
- But for complex systems like the TTP, we are never far from the limits of the tool.

Future Works:

- Other specific acceleration formula,
- More efficient Presburger library to scale up to wider systems.