

# FAST, theory and practice of acceleration.

Sébastien Bardin  
Joint work with Alain Finkel

LSV - CNRS & ÉNS de Cachan

École Polytechnique Fédérale de Lausanne, 02/12/2004

# Counter systems

## Counter systems

- We do not check programs but mathematical models.
- Automata extended with **unbounded** integer variables.
- **Many applications.**

- Communication protocols,
- Abstract multi-threaded java programs,
- Embedded systems

- Petri Nets and many extensions,
- Discrete clocks,
- Abstractions of queue

# Counter systems

## Problem

- the state space is infinite.
- checking safety properties is undecidable.

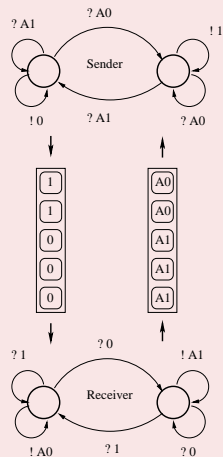
# Model-checking finite systems (1)

## finite system

- $n$  finite variables  
 $(x_1, \dots, x_n) \in F_1 \times \dots \times F_n$
- a finite control structure
  - finite set  $Q$  of locations  
 $q_1, \dots, q_m$
  - finite set  $T$  of transitions  
 $(Q \times (F_1 \times \dots \times F_n))^2$

## Example

booleans, enumerations, bounded queues, bounded integers, ...



## Model-checking finite systems (2)

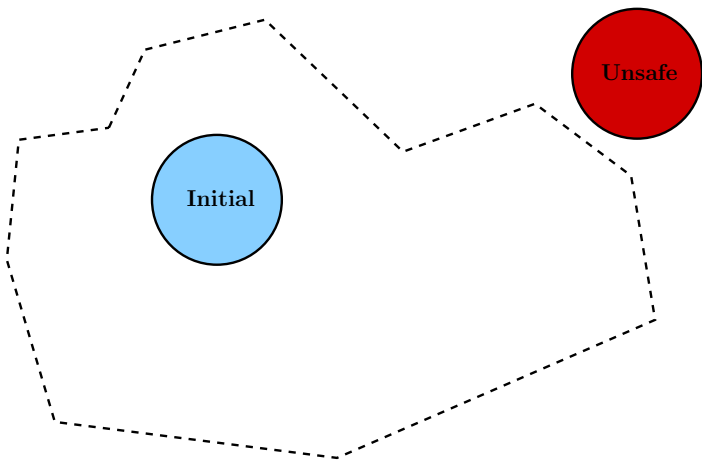
### Reachability set

- configurations  $c \in Q \times (F_1 \times \dots \times F_n)$
- $c' \in \text{post}_{\mathcal{S}}(c)$  iff there exists a transition  $t$  such that  $c \xrightarrow{t} c'$ .
- $c' \in \text{post}_{\mathcal{S}}^*(c)$  iff there exist  $t_1, \dots, t_n$  such that  $c \xrightarrow{t_1} c_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} c'$ .
- $\text{post}_{\mathcal{S}}^*(c_0)$  is the reachability set from initial configuration  $c_0$ .

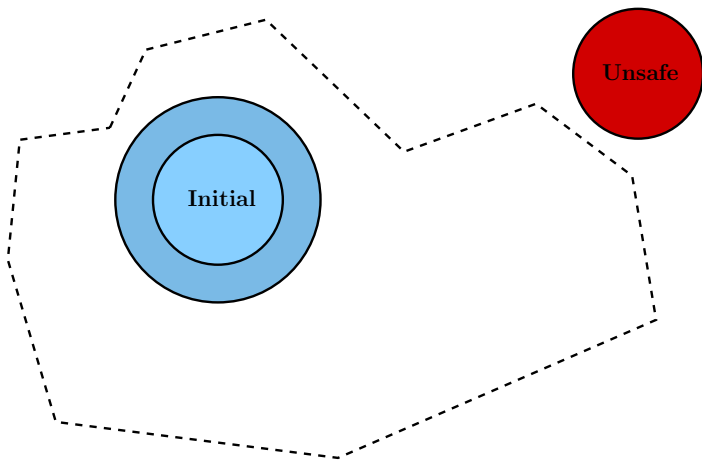
Safety properties = properties on reachable configurations.

- interesting: mutual exclusion, deadlock freedom, ...
- can be checked easily from the reachability set.

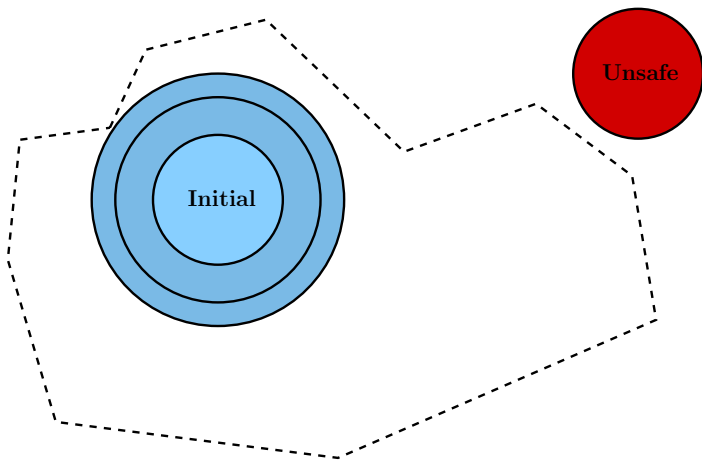
## Model-checking finite systems (3)



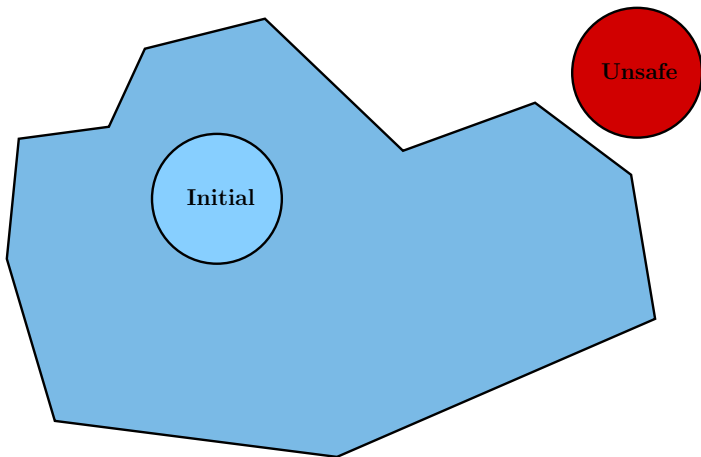
## Model-checking finite systems (3)



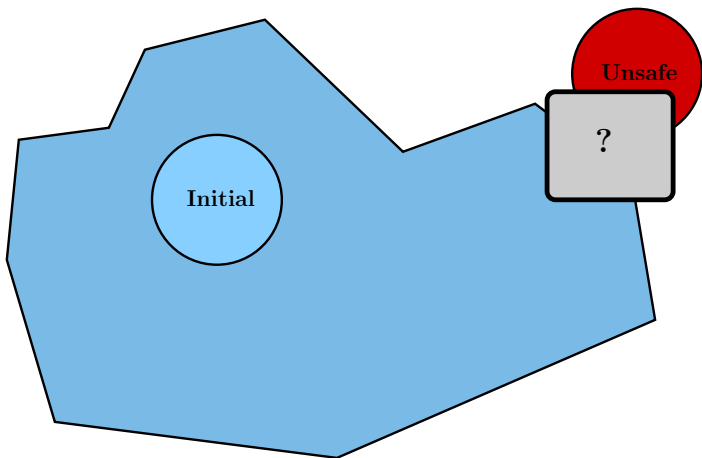
## Model-checking finite systems (3)



## Model-checking finite systems (3)



## Model-checking finite systems (3)



## Extend model-checking to infiniteness

### Problem

Does not work for infinite reachability sets.

# Extend model-checking to infiniteness

Active research area

## Decidable subclasses

- Rarely expressive enough
- OR decidable procedures are too expensive (Petri Nets, timed automata)

## Other approach

- finite abstractions
- semi-algorithms, efficient in practice (widening, automated abstractions)

# Symbolic model-checking

Infinite reachability set is one of the problems

## Solution

- manipulate **symbolic configurations**, i.e. infinite sets of configurations.
- symbolic configurations are encoded by **symbolic representations**.
- symbolic representations must provide symbolic operations for  $\text{post}_g$ ,  $\cup$ ,  $\subseteq$ .

# Symbolic reachability set problem

## Symbolic reachability set problem

### INPUT

- A *system*  $\mathcal{S}$ ,
- A *symbolic representation*  $\mathcal{C}$  fit to  $\mathcal{S}$
- A symbolic encoding  $c_0 \in \mathcal{C}$  of initial configurations  $X_0$

### OUTPUT

- the encoding  $c' \in \mathcal{C}$  of  $\text{post}_{\mathcal{S}}^*(X_0)$  if it exists,
- NO otherwise

## Problems

The iterative firing of transitions does not converge to the reachability set in general.

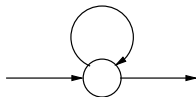
# Acceleration

## Acceleration

Compute in one step the infinite iteration of some transitions of the system.

- exact widening
- computation of the transitive closure of a transition

If  $x \geq 0$  then  $x \leftarrow x + 2$



Standard iteration

If  $S_0 = \{0\}$  then

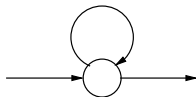
# Acceleration

## Acceleration

Compute in one step the infinite iteration of some transitions of the system.

- exact widening
- computation of the transitive closure of a transition

If  $x \geq 0$  then  $x \leftarrow x + 2$



Standard iteration

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

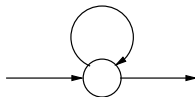
# Acceleration

## Acceleration

Compute in one step the infinite iteration of some transitions of the system.

- exact widening
- computation of the transitive closure of a transition

If  $x \geq 0$  then  $x \leftarrow x + 2$



Standard iteration

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

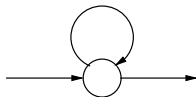
# Acceleration

## Acceleration

Compute in one step the infinite iteration of some transitions of the system.

- exact widening
- computation of the transitive closure of a transition

If  $x \geq 0$  then  $x \leftarrow x + 2$



Standard iteration

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

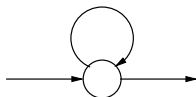
# Acceleration

## Acceleration

Compute in one step the infinite iteration of some transitions of the system.

- exact widening
- computation of the transitive closure of a transition

If  $x \geq 0$  then  $x \leftarrow x + 2$



With **acceleration**

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

# Framework

## what you need

- 1 a symbolic representation
- 2 an acceleration function
- 3 find cycles

## Problems

- 1  $\text{post}_g^*(X_0)$  not representable
- 2  $\text{post}_g^*(X_0)$  not computable with acceleration
- 3 practical (cycles, memory)

# Objectives

## Issue

Efficient semi-algorithm to solve the symbolic reachability set problem in practice.

## Results

Instantiation of the framework to counters

- representation (UBA, [Leroux, INFINITY03]).
- acceleration ([Finkel-Leroux, FSTTCS02], [BFL, TACAS04] )
- heuristics (described succinctly in [TACAS04])

Implementation in FAST

- Many non trivial case studies
- Despite theoretical results (undecidability, complexity of acceleration), the tool behaves very well.

## Related work

- FAST: Bardin, Finkel, Leroux, Petrucci [FSTTCS02], [CAV03], [TACAS04],
- LASH: Boigelot, Rassart, Wolper [CAV94], [SAS95], [CAV98], [TACAS00], [CAV03],
- TREX: Asarin, Bouajjani, Collomb-Annichini, Lakhnech, Sighireanu, [SPIN00], [SAS01], [CAV01].

# Outline

- 1 Motivations
- 2 Counter systems
- 3 Symbolic model-checking for counter systems
- 4 The tool FAST
- 5 Experimentation

# Presburger arithmetic.

## Presburger arithmetics

First order additive theory  $\langle \mathbb{N}^m, \leq, + \rangle$ , defined by

$$\phi ::= t \leq t \mid \neg \phi \mid \phi \vee \phi \mid \exists x. \phi \mid \text{true}$$

$$t ::= 0 \mid 1 \mid y \mid t - t \mid t + t.$$

Decidable, and Presburger sets can be represented symbolically by automata

- DFA [Boudet, Comon CAAP96],
- NDD [Wolper, Boigelot TACAS00],
- UBA [Leroux, INFINITY03].

# Counter systems

## Counter system

A  $n$ -dim counter system  $\mathcal{S} = (Q, T, \alpha, \beta, (G_t)_{t \in T})$ ,

- $Q$  finite set of *locations*,
- $T$  finite set of *transitions*,
- $\alpha : T \rightarrow Q$  and  $\beta : T \rightarrow Q$  the *source* and *target* mappings,
- $(G_t)_{t \in T}$  a sequence of Presburger-definable binary relations on  $\mathbb{N}^n$ , called *actions*.

## Semantics

- *set of configurations*  $\mathcal{C}_{\mathcal{S}}$  of  $\mathcal{S}$  is  $Q \times \mathbb{N}^n$ ,
- *one-step reachability relation*  
 $(q, x) \mathcal{R}_{\mathcal{S}}(t) (q', x')$  if  $q = \alpha(t)$ ,  $q' = \beta(t)$  and  $x G_t x'$

# Restriction

## Linear counter system

A  $n$ -dim linear counter system  $\mathcal{S} = (Q, T, \alpha, \beta, (G_t)_{t \in T})$

- $\mathcal{S}$  is a counter system,
- for all  $t \in T$ ,  $(x, x') \in G_t$  if and only if  $x \in D_t$  and  $x' = M_t \cdot x + v_t$ , where  $D_t \subseteq \mathbb{N}^n$  Presburger set,  $M_t \in \mathcal{M}_n(\mathbb{Z})$ ,  $v_t \in \mathbb{Z}^n$ .

## Associated monoid

The set of matrices

$$\{M \in \mathcal{M}_n(\mathbb{Z}) \mid \exists M_1, \dots, M_k \in (M_t)_{t \in T}, M = M_1 \times \dots \times M_k\}$$

## About linear counter systems

### About the transitions

- Stability by composition
- Expressivity and conciseness (*nested loops*)
- Closed under union (*reduction*)

# Outline

- 1 Motivations
- 2 Counter systems
- 3 Symbolic model-checking for counter systems
- 4 The tool FAST
- 5 Experimentation

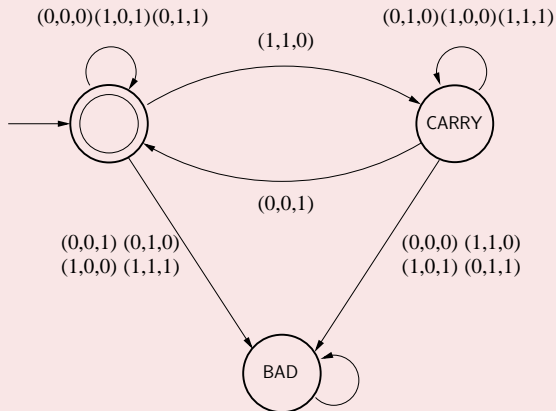
# Automata

## Automata as integer accepters

- an integer written in basis  $r$  is a word on  $\{0, \dots, r - 1\}$
- automata recognize sets of words.

Presburger sets (and a little bit more) can be represented by automata.

## Automata



# Automata

- 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.

## Automata as a symbolic representation

Presburger sets (encoded by automata) provides an efficient framework to check safety properties on counter systems.

# Presburger acceleration

$R_f^*$  is the transitive closure of  $f$ .

Theorem (Finkel Leroux, FSTTCS02)

*For a function  $f = (M, v, \mathcal{D})$  with a finite monoid,  $R_f^*$  can be computed as a Presburger formula, of the form*

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

This construction is at most elementary in  $|\mathcal{A}(\mathcal{D})|$ ,  $|v|_{\max}$ ,  $|M|_{\max}$  and  $m$ .

## Idea of the construction

- $f = (M, v, \mathcal{D})$  with  $\langle M \rangle$  finite.
- $\bar{f} : \mathbb{Z}^m \rightarrow \mathbb{Z}^m, \forall x \in \mathbb{Z}^m, \bar{f}(x) = M.x + v$

- $\langle M \rangle$  finite, so  $\exists(a, b) \in \mathbb{N} \times \mathbb{N}$  such that  $M^{a+b} = M^a$
- Deduce  $\forall n \in \mathbb{N}, \forall x \in \mathbb{Z}^m, \bar{f}^{a+n.b} = \bar{f}^a(x) + n.M^a.\bar{f}^b(0)$
- $G = \{(i, x, x') \in \mathbb{N} \times \mathbb{Z}^m \times \mathbb{Z}^m, x' = \bar{f}^i(x)\} \iff$   
 $\bigvee_{r=0}^{a-1} \{(i, x, x') \mid x' = g^r(x) \wedge i = r\} \bigvee_{r=0}^{b-1} \{(i, x, x') \mid \exists n \geq 0 (x' = \bar{f}^{a+r}(x) + n.M^{a+r}.\bar{f}^b(0)) \wedge (i = a + r + n.b)\}$

$$R_f^* = \{(x, x'), \exists i \geq 0, x' = f^i(x)\} \iff$$

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

# Faster acceleration

## Restricted case

Function  $f$  is now a **translation over a convex polyhedron**.

- Don't need to test if all the predecessors are in the guard.
- We can use a simpler acceleration formula

## Convex acceleration [Bardin-Finkel-Leroux TACAS04]

- $R_f^* = \{(x, x') \mid x \in \mathcal{D} \wedge \exists k \geq 0. x' = \bar{f}^k(x) \wedge \forall i. 0 \leq i < k, \bar{f}^i(x) \in \mathcal{D}\}$  (1)
- $R_f^* = \{(x, x') \mid x \in \mathcal{D} \wedge \exists k \geq 0. x' = \bar{f}^k(x) \wedge k > 0 \Rightarrow \bar{f}^{k-1}(x) \in \mathcal{D}\}$  (2)
- $R_f^* = \{(x, x') \in D \times (D + v); x' - x \in \mathbb{N}.v\}$

# Faster acceleration

The convex acceleration has a complexity bounded by

$$|\mathcal{A}(D)|^2 \cdot 4 \cdot (4 \cdot m \cdot |v|_{\max} + 1)^{3 \cdot m}$$

Big improvement compared to Presburger acceleration

- Presburger acceleration is elementary in both  $\mathcal{A}(D)$  and  $|v|_{\max}$ .
- $|\mathcal{A}(D)|$  very large while  $m$  and  $|v|_{\max}$  small.

## Fixed-point computation

- More experimental than previous solutions.
- Work done in FAST is not yet completely formalized.
- A paper is being written on the subject (submission CAV05)

### Idea

- Acceleration is sufficient for **flat systems**.
- $\mathcal{S}$  is flat iff  $\text{post}_{\mathcal{S}}^*(X) = \mathcal{R}_{\mathcal{S}}(\rho)(X)$  with  $\rho$  a SLRE on  $T^*$ .
- The main issue is to find  $\rho$ .

## Fixed-point computation

```
1: while  $\text{post}_g(X) \not\subseteq X$  do  
2:   choose  $w \in T^*$   
3:    $X \leftarrow \mathcal{R}_g^*(w)(X)$   
4: end while  
5: return  $X$ 
```

# Fixed-point computation

```
1: Main procedure
2:  $k \leftarrow 0$ 
3:  $k \leftarrow k + 1$ 
4:  $\Sigma \leftarrow T^{\leq k}$ 
5: Search( $\Sigma$ )
6:    $\text{post}_g^*(X)$  found  $\rightarrow$  goto 8
7:   Abort  $\rightarrow$  goto 3
8: return  $\text{post}_g^*(X)$ 
```

```
1: procedure Search(  $\Sigma$  )
2:   while  $\text{post}_g(X) \not\subseteq X$  do
3:      $w \leftarrow \text{Next} \in \Sigma$ 
4:      $X \leftarrow \mathcal{R}_g^*(w)(X)$ 
5:     if Abort then
6:       return Abort
7:     end if
8:   end while
9: return  $X$ 
```

# Fixed-point computation

## Practical improvements

- reduction of the number of cycles [FSTTCS02]
- procedures Next and Abort.

# Outline

- 1 Motivations
- 2 Counter systems
- 3 Symbolic model-checking for counter systems
- 4 **The tool FAST**
- 5 Experimentation

# Overview

We implement our results in the tool FAST [CAV03].

## FAST provides

- a powerful input model (linear counter systems with unbounded integer),
- acceleration and heuristics to automatically compute the reachability set in most practical cases.

## FAST works well in practice

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

## Architecture

Three (almost independent) layers

- Symbolic representation (Presburger sets encoded by automata)
- Acceleration algorithms
- Heuristics

# Input/Output

## INPUT

- 1 A linear counter system
- 2 A strategy
  - script language, sequence of computations
  - variables: regions, transitions and booleans. All usual operators on regions + primitives to compute the reachability set.
  - standard procedures easy to specify, the user can help the tool

## OUTPUT

Computation of the query.

# Outline

- 1 Motivations
- 2 Counter systems
- 3 Symbolic model-checking for counter systems
- 4 The tool FAST
- 5 **Experimentation**

## The TTP protocol - overview

- Supported by transport industry (Airbus, Audi, EADS, PSA, Renault, ...)
- Fault-tolerant 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

# Validation of the TTP protocol

## A protocol difficult to validate

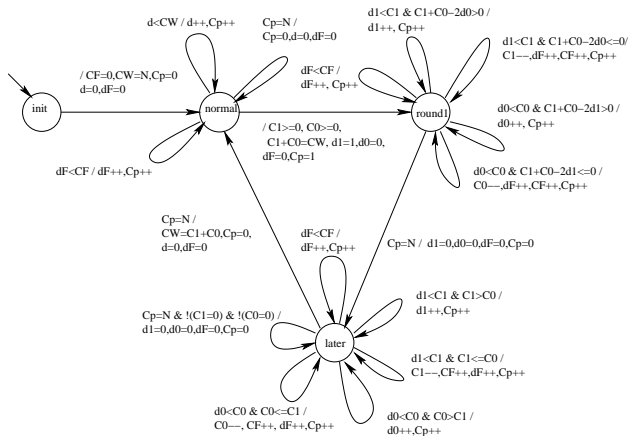
Merceron and Bouajjani [FTRTFT'02]

- Manual proof of correctness ( $N$  stations,  $k$  faults).
- Provide a family of counting abstractions depending on the number of faults.
  - large parametric counter automata
  - complex guards

## Difficult to check automatically

Semi-automatic verification with tools LASH and ALV ( $N$  stations, 1 fault).

# Model for the TTP, N stations



# Verification with FAST

## 1 fault [TACAS04]

16 transitions, 9 variables, complex guards

- full automatic verification (no intermediate property)
- reachability set has 27,932 nodes
- P4 2.4 GHz, 1 Gbyte RAM: 940 sec. and 73 Mbytes.

## 2 faults [TACAS04]

20 transitions, 18 variables, more complex guards

- standard acceleration fails, we have to use convex acceleration.
- $N = 15$ : 273,427 nodes, 12,000 sec., 588 Mbytes.
- $N$  stations: manual abstractions and check the property.

## The CES protocol - overview

- Supported by Philips
- multimedia streaming
- reliable communications on lossy channels
- a sender and a receiver communicating through a lossy channel (stop & wait protocols)

## The CES protocol - validation

Jonathan Billington and Lin Liu [ICATPN02]

- Colored Petri Net model,
- infinite state space, counters and parameterized queues
- (complex) proofs by hand of many properties (ex: aspect of the state space)

## Verification with FAST

### Rich model

Lossy queues are beyond counter systems.

- queues are translated into counters.
- correctness of the translation: a sufficient condition is checked automatically on the counter system.

### Results

All properties of Billington-Liu'02 are checked automatically.

Case study	variables	transitions	time (s)	memory (MB)	cycle length
Producer/Consumer	5	3	0.41	2.37	1
Lamport ME	11	9	2.70	2.88	1
Dekker ME	22	22	21.72	5.48	1
RTP	9	12	2.24	2.76	1
Peterson ME	14	12	4.97	3.78	1
Reader/Writer	13	9	9.68	23.14	1
CSM	13	13	45.57	6.31	2
FMS	22	20	157.48	8.02	2
Multipoll	17	20	22.96	5.13	1
Kanban	16	16	10.43	6.54	1
Mesh2x2	32	32	$\geq 1800$	-	-
Mesh3x2	52	54	$\geq 1800$	-	-
Manufacturing system	7	6	$\geq 1800$	-	-
PNCSA	31	38	$\geq 1800$	-	-
extended ReaderWriter	24	22	$\geq 1800$	-	-
SWIMMING POOL	9	6	111	29.06	4
Last-in First-served	17	10	1.89	2.74	1
Esparza-Finkel-Mayr	6	5	0.79	2.55	1
Inc/Dec	32	28	$\geq 1800$	-	-
Producer/Consumer with Java threads - 2	18	14	13.27	3.81	1
Producer/Consumer with Java threads - N	18	14	723.27	12.46	2

Results using an Intel Pentium 933 Mhz with 512 Mbytes

Case study	variables	transitions	time (s)	memory (MB)	cycle length
2-Producer/2-Consumer with Java threads <i>2 types of producers and 2 types of consumers</i>	44	38	$\geq 1800$	-	-
Central Server system	13	8	20.82	6.83	2
Consistency Protocol	12	8	275	7.35	3
M.E.S.I. Cache Coherence Protocol	4	4	0.42	2.44	1
M.O.E.S.I. Cache Coherence Protocol	4	5	0.56	2.49	1
Synapse Cache Coherence Protocol	3	3	0.30	2.23	1
Illinois Cache Coherence Protocol	4	6	0.97	2.64	1
Berkeley Cache Coherence Protocol	4	3	0.49	2.75	1
Firefly Cache Coherence Protocol	4	8	0.86	2.59	1
Dragon Cache Coherence Protocol	5	8	1.42	2.72	1
Futurebus+ Cache Coherence Protocol	9	10	2.19	3.38	1
lift controller - N	4	5	4.56	2.90	3
bakery	8	20	$\geq 1800$	-	-
barber m4	8	12	1.92	2.68	1
ticket 2i	6	6	0.88	2.54	1
ticket 3i	8	9	3.77	3.08	1
TTP	10	17	1186.24	73.24	1

Results using an Intel Pentium 933 Mhz with 512 Mbytes

# Comparison with Other Tools

	system	symb. rep	acceleration	heuristics	exact?
FAST	linear CS	UBA	yes	cycle, red.	yes
LASH	convex CS	NDD	yes	loop	yes
TREX	~ timed automata	CPDBM	yes	cycles $\leq k$	?
BRAIN	convex CS	period/basis	no	backward	yes
HYTECH	convex CS	polyhedra	widening	-	no
ALV	full CS	NDDs	no	-	yes

# Conclusion

## Results

- Instantiation of accelerated symbolic model-checking to counters.
- Implementation in `FAST`.
- Works well.

# Perspective

## FAST

- symbolic representation [Couvreur]
- acceleration
- heuristics [Bardin, Leroux, Sutre]
- abstraction [Bultan & co. CAV04] [Bouajjani & co. CAV04]
- formula from the automaton [Leroux]
- model specification [PERSÉE project]

## Other questions

- why does FAST terminate? [Bardin, Leroux, Sutre]
- liveness
- other data types [PERSÉE project]