

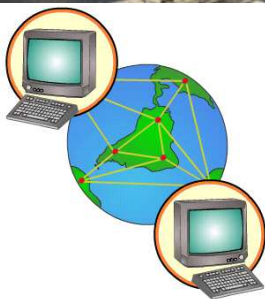
FAST : Théorie et mise en œuvre de l'accélération

Sébastien Bardin

LSV - CNRS & ÉNS de Cachan

28 février 2006

Vérification de systèmes réactifs



Systèmes réactifs

- Logiciels ou matériels
- Autonomes
- **Souvent critiques**

S'assurer du bon fonctionnement des systèmes réactifs est crucial

Mes travaux :

- méthodes formelles
- vérification automatique
- vérification de modèles



Processeurs embarqués dans les voitures pour remplacer les transmissions mécaniques

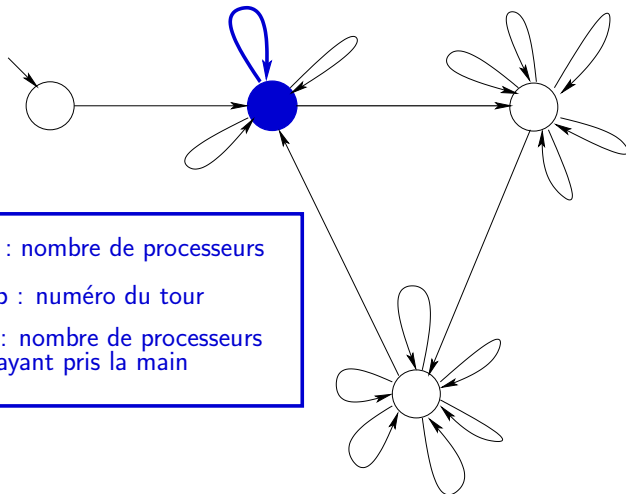
protocole TTP

- tolérance aux pannes
- assure qu'une erreur ne se propage pas

Le TTP est supporté par Audi, PSA, Renault, ...

Modèle du protocole TTP [Bouajjani-Merceron 2002]

```
Si  $d < N$  Faire  
   $d := d + 1; C_p := C_p + 1$   
Fin Si
```

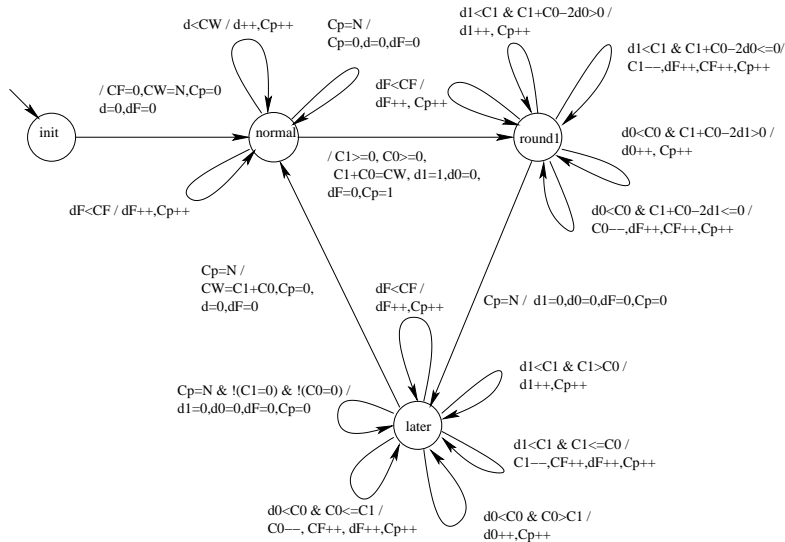


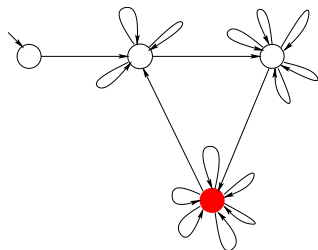
N : nombre de processeurs

C_p : numéro du tour

d : nombre de processeurs
ayant pris la main

Modèle du protocole TTP [Bouajjani-Merceron 2002]





Question

Dans la location **rouge**, est-ce que

$C_p = N \Rightarrow (C_0 = 0 \vee C_1 = 0)$?

On veut le vérifier pour toutes les valeurs du paramètre N !!

Systèmes à compteurs

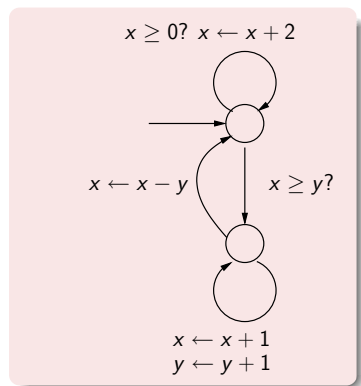
- Nous vérifions des modèles mathématiques
- Automates étendus par des variables entières **non bornées**

Problèmes

- Vérification indécidable pour 2 compteurs avec $(+1, -1, \stackrel{?}{=} 0)$
- Entre autre : Ensemble infini de configurations accessibles

Systèmes à compteurs

- Automate de contrôle fini
 - Q : locations
 - T : transitions
- Nombre fini de variables à valeurs dans \mathbb{N} .
- Variables testées et modifiées par les transitions
- Configuration
 - $c = (q, v) \in Q \times \mathbb{N}$
- $\text{post}^*(c)$: configurations accessibles à partir de c

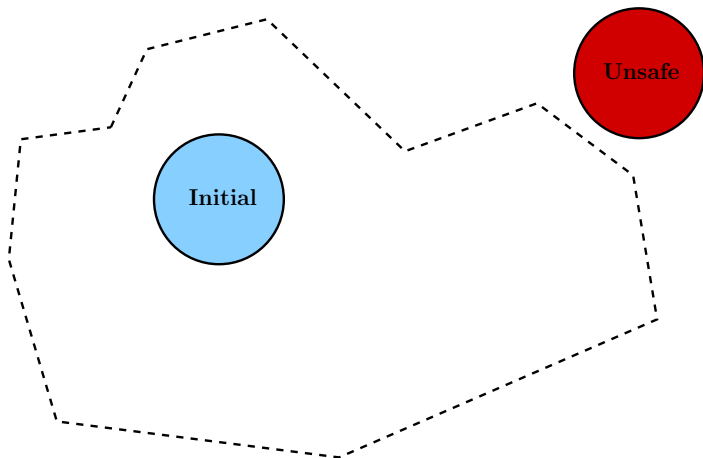


Propriétés à vérifier

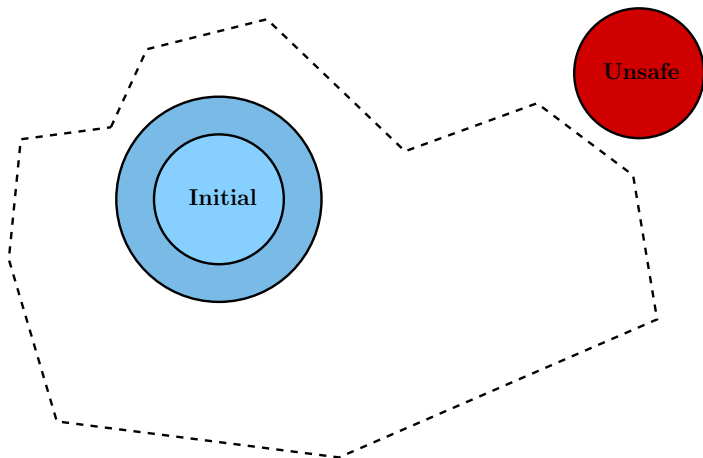
Propriétés d'accessibilités = propriétés sur les configurations atteignables.

- utile : exclusion mutuelle, absence de blocage, ...
- facile à vérifier à partir de l'ensemble d'accessibilité.

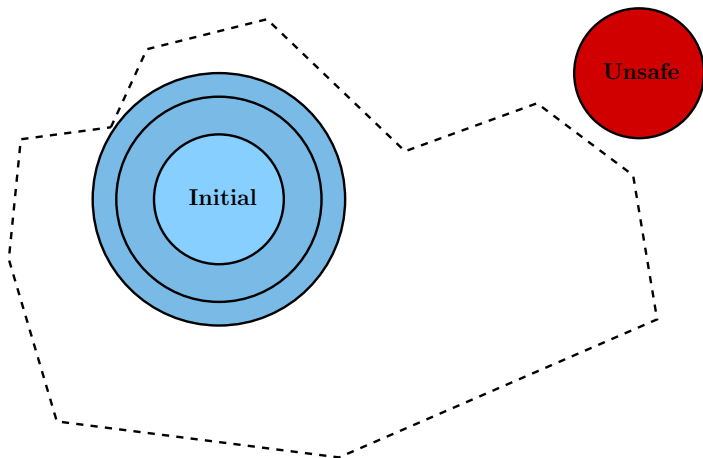
Rappel : vérification dans le cas fini



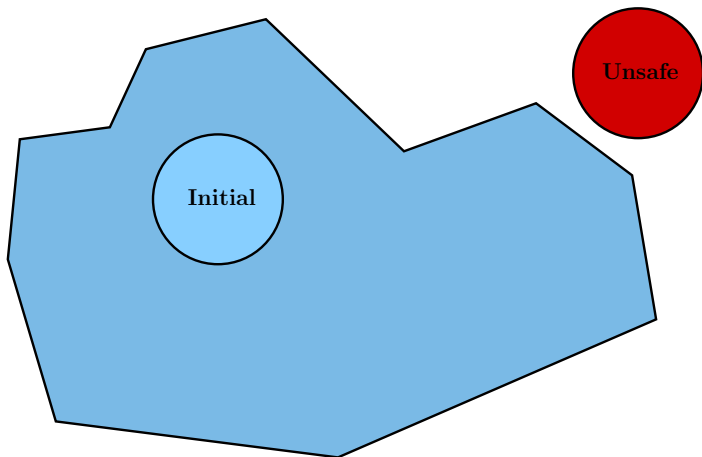
Rappel : vérification dans le cas fini



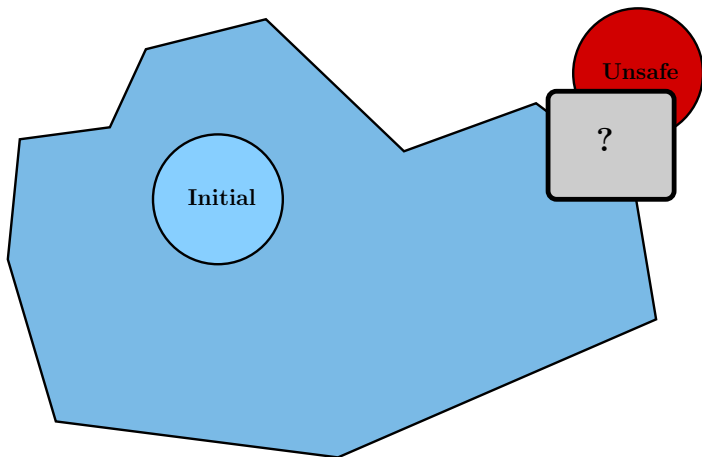
Rappel : vérification dans le cas fini



Rappel : vérification dans le cas fini



Rappel : vérification dans le cas fini



Systèmes infinis

méthodes énumératives ne fonctionnent plus

Algorithmes sur des sous-classes décidables

- réseaux de Petri,
- automates temporisés, ...

Semi-algorithmes de calcul de l'ensemble d'accessibilité

- Classes plus expressives
- Pas de terminaison théorique, on espère terminaison pratique
- Adaptation du calcul itératif au cas infini
- (model checking symbolique)

Systèmes infinis

méthodes énumératives ne fonctionnent plus

Algorithmes sur des sous-classes décidables

- réseaux de Petri,
- automates temporisés, ...

Semi-algorithmes de calcul de l'ensemble d'accessibilité

- Classes plus expressives
- Pas de terminaison théorique, on espère terminaison pratique
- Adaptation du calcul itératif au cas infini
- (model checking symbolique)

Systèmes infinis

méthodes énumératives ne fonctionnent plus

Algorithmes sur des sous-classes décidables

- réseaux de Petri,
- automates temporisés, ...

Semi-algorithmes de calcul de l'ensemble d'accessibilité

- Classes plus expressives
- Pas de terminaison théorique, on espère terminaison pratique
- Adaptation du calcul itératif au cas infini
- (model checking symbolique)

Idée : étendre les méthodes énumératives du model checking.

Problème 1 : Les ensembles de configurations sont infinis.

Solution = Gérer des ensembles infinis de configurations

- **Ensembles infinis** représentés symboliquement.
- Nécessite les opérations POST , \sqcup , \sqsubseteq sur la représentation.

Exemple : représentation par intervalles d'ensembles d'entiers

- La formule $X = \{x \geq 5\}$ indique que x peut prendre toutes les valeurs plus grandes que 5
- Après la transition $\xrightarrow{y:=x+1}$ les valeurs possibles de y sont dans $Y = \{y \geq 6\}$
- L'inclusion est décidable sur de tels ensembles

Idée : étendre les méthodes énumératives du model checking.

Problème 1 : Les ensembles de configurations sont infinis.

Solution = Gérer des ensembles infinis de configurations

- **Ensembles infinis** représentés symboliquement.
- Nécessite les opérations POST , \sqcup , \sqsubseteq sur la représentation.

Exemple : représentation par intervalles d'ensembles d'entiers

- La formule $X = \{x \geq 5\}$ indique que x peut prendre toutes les valeurs plus grandes que 5
- Après la transition $\xrightarrow{y:=x+1}$ les valeurs possibles de y sont dans $Y = \{y \geq 6\}$
- L'inclusion est décidable sur de tels ensembles

Procédure basique de model checking symbolique

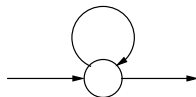
Procédure itérative de calcul de $\text{post}_S^*(X_0)$

- 1 $X \leftarrow X_0$
- 2 Si $\text{POST}(X) \sqsubseteq X$ Aller à 5
- 3 $X \leftarrow \text{POST}(X) \sqcup X$
- 4 Aller à 2
- 5 Retourner X

La terminaison est très rare

Problème 2 = circuits du graphe de contrôle

Si $x \geq 0$ faire $x \leftarrow x + 2$



Si $X_0 = \{0\}$ alors

Procédure basique de model checking symbolique

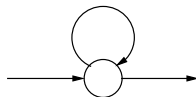
Procédure itérative de calcul de $\text{post}_S^*(X_0)$

- 1 $X \leftarrow X_0$
- 2 Si $\text{POST}(X) \subseteq X$ Aller à 5
- 3 $X \leftarrow \text{POST}(X) \sqcup X$
- 4 Aller à 2
- 5 Retourner X

La terminaison est très rare

Problème 2 = circuits du graphe de contrôle

Si $x \geq 0$ faire $x \leftarrow x + 2$



Si $X_0 = \{0\}$ alors $X = \{0\}$

Procédure basique de model checking symbolique

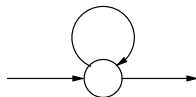
Procédure itérative de calcul de $\text{post}_S^*(X_0)$

- 1 $X \leftarrow X_0$
- 2 Si $\text{POST}(X) \subseteq X$ Aller à 5
- 3 $X \leftarrow \text{POST}(X) \sqcup X$
- 4 Aller à 2
- 5 Retourner X

La terminaison est très rare

Problème 2 = circuits du graphe de contrôle

Si $x \geq 0$ faire $x \leftarrow x + 2$



Si $X_0 = \{0\}$ alors $X = \{0, 2\}$

Procédure basique de model checking symbolique

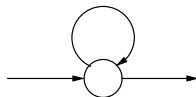
Procédure itérative de calcul de $\text{post}_S^*(X_0)$

- 1 $X \leftarrow X_0$
- 2 Si $\text{POST}(X) \subseteq X$ Aller à 5
- 3 $X \leftarrow \text{POST}(X) \sqcup X$
- 4 Aller à 2
- 5 Retourner X

La terminaison est très rare

Problème 2 = circuits du graphe de contrôle

Si $x \geq 0$ faire $x \leftarrow x + 2$

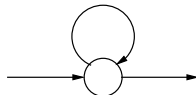


Si $X_0 = \{0\}$ alors $X = \{0, 2, \dots, 2k\}$

Accélération de circuits

Améliorer la convergence du calcul itératif en calculant en un coup l'**itération** d'une **séquence de transitions** (circuits).

Si $x \geq 0$ faire $x \leftarrow x + 2$

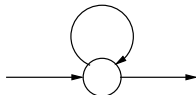


Si $X_0 = \{0\}$ alors

Accélération de circuits

Améliorer la convergence du calcul itératif en calculant en un coup l'**itération** d'une **séquence de transitions** (circuits).

Si $x \geq 0$ faire $x \leftarrow x + 2$



Si $X_0 = \{0\}$ alors $\text{post}^*(X_0) = 2\mathbb{N}$, en un coup.

Accélérations de systèmes à compteurs (en 2002)

(*Karp-Miller 1969*)

(*Fribourg 1990*)

[Boigelot-Wolper 1994],

[Boigelot-Wolper 1998],

[Annichini-Asarin-Bouajjani 2000], (+ temps)

[Finkel-Leroux 2002],

Remarques

- 1 les techniques sont très diverses, pas de cadre unifié : difficile de comparer les différents algorithmes d'accélération
- 2 il manque un lien entre l'algorithme d'accélération et le calcul de point fixe (comment choisir les circuits)

- Représentation symbolique : automates binaires
 - DFA [Boudet-Comon 1996],
 - NDD [Wolper-Boigelot 2000]).
- Accélération de circuits
 - $f(x) = M.x + v$, avec *monoïde fini* et garde G convexe [Boigelot 1998]
 - $f(x) = M.x + v$ avec *monoïde fini* et garde G Presburger [Finkel-Leroux 2002]
 - fonctions “à la automates temporisés” [Annichini-Asarin-Bouajjani 2000]
- Heuristique : pas d'heuristique argumentée
- Outils
 - ALV (sans accélération) [Bultan]
 - LASH (accélération mais pas de recherche de circuits) [Wolper]
 - TRES (accélération et recherche de circuits, mais pas argumentée) [Bouajjani]

Problème : Techniques trop diverses

Résultat : Cadre unifié englobant la plupart des résultats connus
[ATVA'05]

Problème : Quelle recherche de circuits ?

Résultat : Heuristique maximale et efficace en pratique
[CAV'03,ATVA'05]

Problème : Renforcer l'efficacité des techniques d'accélération

Résultat : Algorithme d'accélération convexe [TACAS'04]

Problème : Applications, expérimentations

Résultat : Codage dans FAST [CAV'03],
Vérification du TTP [TACAS'04] et du CES

Ces travaux s'inscrivent dans le cadre de l'ACI PERSÉE.

Problème : Techniques trop diverses

Résultat : Cadre unifié englobant la plupart des résultats connus
[ATVA'05]

Problème : Quelle recherche de circuits ?

Résultat : Heuristique maximale et efficace en pratique
[CAV'03,ATVA'05]

Problème : Renforcer l'efficacité des techniques d'accélération

Résultat : Algorithme d'accélération convexe [TACAS'04]

Problème : Applications, expérimentations

Résultat : Codage dans FAST [CAV'03],
Vérification du TTP [TACAS'04] et du CES

Ces travaux s'inscrivent dans le cadre de l'ACI PERSÉE.

Problème : Techniques trop diverses

Résultat : Cadre unifié englobant la plupart des résultats connus
[ATVA'05]

Problème : Quelle recherche de circuits ?

Résultat : Heuristique maximale et efficace en pratique
[CAV'03,ATVA'05]

Problème : Renforcer l'efficacité des techniques d'accélération

Résultat : Algorithme d'accélération convexe [TACAS'04]

Problème : Applications, expérimentations

Résultat : Codage dans FAST [CAV'03],
Vérification du TTP [TACAS'04] et du CES

Ces travaux s'inscrivent dans le cadre de l'ACI PERSÉE.

Problème : Techniques trop diverses

Résultat : Cadre unifié englobant la plupart des résultats connus
[ATVA'05]

Problème : Quelle recherche de circuits ?

Résultat : Heuristique maximale et efficace en pratique
[CAV'03,ATVA'05]

Problème : Renforcer l'efficacité des techniques d'accélération

Résultat : Algorithme d'accélération convexe [TACAS'04]

Problème : Applications, expérimentations

Résultat : Codage dans FAST [CAV'03],
Vérification du TTP [TACAS'04] et du CES

Ces travaux s'inscrivent dans le cadre de l'ACI PERSÉE.

Problème : Techniques trop diverses

Résultat : Cadre unifié englobant la plupart des résultats connus
[ATVA'05]

Problème : Quelle recherche de circuits ?

Résultat : Heuristique maximale et efficace en pratique
[CAV'03,ATVA'05]

Problème : Renforcer l'efficacité des techniques d'accélération

Résultat : Algorithme d'accélération convexe [TACAS'04]

Problème : Applications, expérimentations

Résultat : Codage dans FAST [CAV'03],
Vérification du TTP [TACAS'04] et du CES

Ces travaux s'inscrivent dans le cadre de l'ACI PERSÉE.

- 1 Introduction
- 2 **Systemes à compteurs**
- 3 Accélération de circuits
- 4 Sélection des circuits
- 5 L'outil FAST
- 6 Applications
- 7 Conclusion

Arithmétique de Presburger

Arithmétique du premier ordre “sans la multiplication”

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

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

Ensemble de Presburger = ensemble de solutions d'une formule de Presburger.

Système à compteurs $S = (m, Q, T)$

- domaine des variables $D = \mathbb{N}^m$
- transitions avec P-fonctions affines $f = (M, v, G)$

$G \subseteq \mathbb{N}^m$ Presburger-définissable

M matrice carrée, v vecteur

$x' = f(x)$ ssi $x \in G$ et $x' = M.x + v$

Exemple : $x + y \leq z \rightarrow x' = 2.x + y$

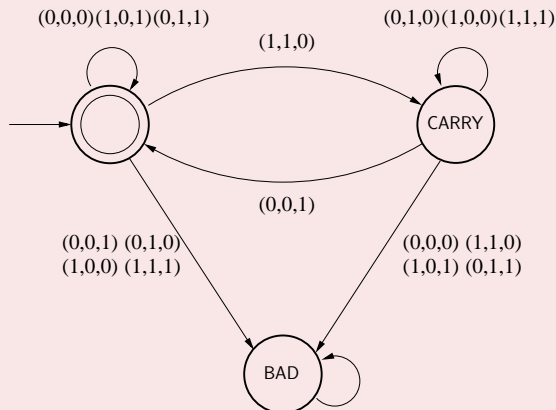
Les automates reconnaissent des ensembles d'entier

- un entier en base 2 est un mot sur l'alphabet $\{0, 1\}$;
- les automates reconnaissent des ensembles de mots
- extensions
 - entiers relatifs : complément à 2
 - vecteurs : tuples de lettres ou entrelacement

Les ensembles de Presburger (et un peu plus) sont représentables par des automates.

Bonne représentation symbolique car les opérations ensemblistes sont les opérations standards sur les automates.

Représentation symbolique : Automates binaires



- DFA [Boudet-Comon 1996],
- NDD [Wolper-Boigelot 2000]).

- 1 Introduction
- 2 Systèmes à compteurs
- 3 **Accélération de circuits**
- 4 Sélection des circuits
- 5 L'outil FAST
- 6 Applications
- 7 Conclusion

Monoïde associé à la fonction $f = (M, v, G)$: ensemble des matrices $\{1, M, M^2, \dots, M^n, \dots\}$

R_f^* est la clôture transitive de f .

Théorème [Finkel-Leroux 2002]

Soit $f = (M, v, G)$ une P-fonction affine à **monoïde fini**. Alors R_f^* est défini par une formule de Presburger de la forme

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

La construction est 3-EXP en $|\mathcal{A}(G)|$, $|v|_{max}$, $|M|_{max}$ et m .

Idee de la construction

- $f = (M, v, G)$ avec $\langle M \rangle$ fini.
- $\bar{f} : \mathbb{Z}^m \rightarrow \mathbb{Z}^m, \forall x \in \mathbb{Z}^m, \bar{f}(x) = M.x + v$

- $\langle M \rangle$ fini, donc $\exists (a, b) \in \mathbb{N} \times \mathbb{N}$ such that $M^{a+b} = M^a$
- On déduit que $\forall n \in \mathbb{N}, \forall x \in \mathbb{Z}^m, \bar{f}^{a+n.b} = \bar{f}^a(x) + n.M^a.\bar{f}^b(0)$
- Soit $\bar{F} = \{(i, x, x') \in \mathbb{N} \times \mathbb{Z}^m \times \mathbb{Z}^m, x' = \bar{f}^i(x)\} \iff$
 $\bigvee_{r=0}^{a-1} \{(i, x, x') \mid x' = \bar{f}^r(x) \wedge i = r\} \bigvee_{r=0}^{b-1} \{(i, x, x') \mid \exists n \geq 0 (x' =$
 $\bar{f}^{a+r}(x) + n.M^{a+r}.\bar{f}^b(0)) \wedge (i = a + r + n.b)\}$

$$R_f^* = \{(x, x'), \exists i \geq 0, x' = f^i(x)\} \iff$$
$$\{(x, x'), \exists i \geq 0 [(i, x, x') \in \bar{F} \wedge (\forall k (0 \leq k < i), \exists x'' \in G, (k, x, x'') \in \bar{F})]\}$$

Translations convexes [TACAS'04]

$f = (I, v, G)$ avec I matrice identité et G convexe

- Pas besoin de tester si les prédécesseurs sont dans la garde.
- On peut simplifier la formule

Théorème [TACAS'04]

L'accélération convexe est quadratique en $|\mathcal{A}(G)|$.

Comparaison des complexités (bornes sup.)

paramètre	ordre de grandeur	f standard	f convexe
$ \mathcal{A}(G) $	100.000	3-EXP	quadratique
m	5-50	3-EXP	EXP
$ V _{max}$	≤ 10	3-EXP	poly. en m
$ M _{max}$	≤ 10	3-EXP	= 1

Comparaison sur des études de cas

f tirée de	$ \mathcal{A}(f^*) $	Temps (secondes) Standard/Convexe	Mémoire (Mo) Standard/Convexe
Dekker, 22 var	1.536	0,7/0,8	4,6/4
Mesh32, 52 var	1.614	2,1/2,5	8/7,8
Mesh32, 52 var	16.766	10,35/7,4	31/13
TTP2, 19 var	26.409	5,6/2,3	17/18
Dekker, 22 var	41.950	18/10,2	52/30
TTP2, 19 var	190.986	50/9	400/140
TTP2, 19 var	380.332	↑↑↑/34	↑↑↑/534
Dekker, 22 var	?	↑↑↑/ >900	↑↑↑/ >500

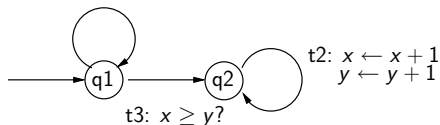
- 1 Introduction
- 2 Systèmes à compteurs
- 3 Accélération de circuits
- 4 **Sélection des circuits**
- 5 L'outil FAST
- 6 Applications
- 7 Conclusion

Que permet de calculer l'accélération de circuits ?

Première réponse

Système plat = au plus 1 circuit élémentaire par nœud de contrôle

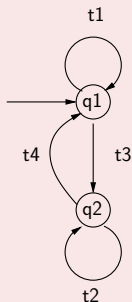
t1: $x \geq 0?$ $x \leftarrow x + 2$



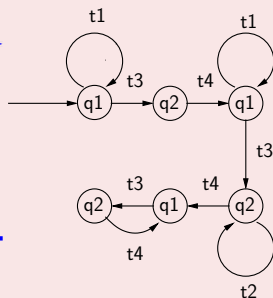
Accélération de circuits + S plat = calcul de $\text{post}_S^*(X_0)$

Aplatissements de systèmes non plats

Système S



Système S'



S' est un **aplatissement** de S si
 S' est **plat** et S **simule** S'

Un système (S, X_0) est aplatissable ssi il existe un aplatissement S' de S tel que S et S' équivalents pour l'accessibilité.

Théorème 1 [ATVA'05]

$\text{post}^*(X_0)$ est calculable par accélération de circuits **ssi** (S, X_0) est aplatissable.

Autre caractérisation

Expression régulière linéaire restreinte (rlre) sur T

$$\rho = w_1^* w_2^* \dots w_n^*, \text{ où } w_i \in T^*.$$

$\text{post}(\rho, X) =$ configurations accessibles en suivant les mots de ρ

Théorème 2 [ATVA'05]

$\text{post}^*(X_0)$ est calculable par accélération de circuits **ssi** il existe une rlre ρ sur T telle que $\text{post}^*(X_0) = \text{post}(\rho, X_0)$.

Heuristique (I)

Entrée : (S, X_0)

- 1 $X \leftarrow X_0; k \leftarrow 0$
- 2 $k \leftarrow k + 1$
- 3 Exécuter
- 4 Si $\text{post}(X) \subseteq X$ aller à 10
- 5 Énumérer le prochain ρ rlre sur T
- 6 $X \leftarrow \text{post}(\rho, X)$
- 7 aller à 4
- 8 Avec
- 9 Quand *Watchdog* stoppe aller à 2
- 10 Retourner X

procédure maximale : termine sur (S, X_0) aplatissable

PB trouver vite un bon rlre (temps)

PB éviter au maximum les mauvais sous-calculs (espace)

Heuristique (I)

Entrée : (S, X_0)

- 1 $X \leftarrow X_0; k \leftarrow 0$
- 2 $k \leftarrow k + 1$
- 3 **Exécuter**
- 4 Si $\text{post}(X) \subseteq X$ aller à 10
- 5 *Choisir équitablement* $w \in T^{\leq k}$
- 6 $X \leftarrow \text{post}(w^*, X)$
- 7 aller à 4
- 8 **Avec**
- 9 **Quand** *Watchdog* stoppe **aller à 2**
- 10 Retourner X

procédure toujours MAXIMALE

trouver vite les bons circuits : partitionnement + *Watchdog*

éviter au maximum les mauvais sous-calculs : *Choisir*

Résultat

On restreint la recherche d'une heuristique à la mise au point de

- *Choisir* (une solution type est proposée)
- *Watchdog* (une solution type est proposée)

Procédure

- **maximale**
- **efficace** : bons résultats sur les compteurs (cf. FAST)

Problème = $|T|^{\leq k}$ peut être exponentiel en k .

Idée = réduire $|T|^{\leq k}$ en enlevant certaines fonctions redondantes.

Trois réductions :

- réduction par union [Finkel-Leroux 2002]
 - $[(M, v, G_1) + (M, v, G_2)]^* = [(M, v, G_1 \vee G_2)]^*$
- réduction par commutation [CAV'03]
 - si f et g commutent alors $f^*g^* = (f \cdot g)^*$
- réduction par conjugaison [ATVA'05]
 - $(f_2 \cdot f_3 \cdot f_1)^* = I_d + f_2 \cdot f_3 \cdot (f_1 \cdot f_2 \cdot f_3)^* \cdot f_1$

système	$ T $	k	$ C^{\leq k} $	U	Cm	Cj	U+Cm
csm	13	1	14	14	14	14	14
	13	2	183	103	57	99	35
consistency	8	1	9	9	9	9	9
	8	2	68	45	44	39	30
	8	3	484	172	299	178	98
swimming pool	6	1	7	7	7	7	7
	6	2	43	21	24	25	16
	6	3	259	56	114	97	28
	6	4	1555	126	614	421	47
	6	5	9331	252	3591	1977	86

U, Cm ,Cj : réduction par union, commutation, conjugaison

- 1 Introduction
- 2 Systèmes à compteurs
- 3 Accélération de circuits
- 4 Sélection des circuits
- 5 **L'outil FAST**
- 6 Applications
- 7 Conclusion

Ces résultats sont intégrés dans FAST :

- calcul des accessibles pour des systèmes à compteurs,
- accélération de circuits + recherche de circuits.

FAST fonctionne très bien en pratique

- 80% de réussite sur les 40 systèmes testés [CAV'03].
- 1ère vérification automatique du TTP [TACAS'04]
- 1ère vérification automatique du CES

Comparaison des technologies

	ALV	LASH	FAST	TREX
système	relationnel	affine		restreint
rep. symb	automates binaires			arith. + pdbm (indéc. \sqsubseteq)
accélération	non	circuits		circuits (partiel.réc.)
rech. circuits	non	non	oui	oui, $\leq k$

Expérimentations

Système	ALV	LASH	FAST	k	TREX
RTP (borné)	T	T	T	1	T
Lamport (borné)	T	T	T	1	T
Dekker (borné)	T	T	T	1	T
ticket 2	T	T	T	1	T
kanban	↑	T	T	1	T
multipoll	↑	T	T	1	↑
prod/cons (2)	↑	T	T	1	-
ttp	↑	T	T	1	-
prod/cons (N)	↑	↑	T	2	-
lift control, N	↑	↑	T	2	T
train	↑	↑	T	2	T
csm, N	↑	↑	T	2	↑
consistency	↑	↑	T	3	-
swimming pool	↑	↑	T	4	↑
pncsa	↑	↑	↑	?	↑
incdec	↑	↑	↑	?	↑
bigjava	↑	↑	↑	?	↑

T : réussite en moins de 20 minutes,

↑ : calcul pas terminé en moins de 20 minutes

Exemple	variables	transitions	temps (s)	mémoire (Mo)	k
Producer/Consumer	5	3	0.41	2.37	1
Lampport 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	≥ 1800	-	-
Mesh3x2	52	54	≥ 1800	-	-
Manufacturing system	7	6	≥ 1800	-	-
PNCSA	31	38	≥ 1800	-	-
extended ReaderWriter	24	22	≥ 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	≥ 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

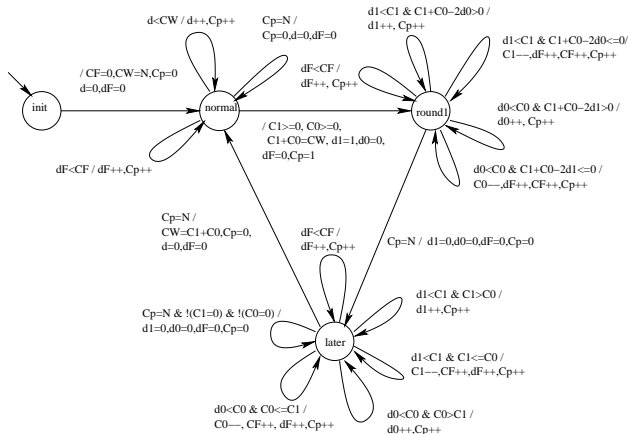
Résultats sur un Intel Pentium 933 Mhz avec 512 Mo

Exemple	variables	transitions	temps (s)	mémoire (Mo)	k
2-Producer/2-Consumer with Java threads <i>2 types of producers and 2 types of consumers</i>	44	38	≥ 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	≥ 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

Résultats sur un Intel Pentium 933 Mhz avec 512 Mo

- 1 Introduction
- 2 Systèmes à compteurs
- 3 Accélération de circuits
- 4 Sélection des circuits
- 5 L'outil FAST
- 6 **Applications**
- 7 Conclusion

Vérification du TTP par FAST



1 erreur [TACAS'04]

16 transitions, 9 variables, gardes complexes

- vérification complètement automatique
- sur un P4 2.4 GHz, 1 Go de RAM : 940 sec. et 73 Mo.

Autres outils :

- ALV ne termine pas
- LASH termine si on lui donne les bons circuits
- Le TTP n'est pas un modèle d'entrée de T_{REX}.

2 erreurs [TACAS'04]

20 transitions, 18 variables, gardes plus complexes

- l'accélération standard ne suffit
- nous utilisons l'accélération convexe et une surapproximation du modèle.

- Supporté par Philips
- diffusion multimedia streaming
- communication sans perte sur des réseaux non fiables

Jonathan Billington et Lin Liu [Billington-Liu 2002]

- modélisation par réseau de Petri coloré,
- système infini, compteurs et **files de tailles paramétrées**
- preuves (complexes) à la main de nombreuses propriétés du CES (ex : aspect de l'ensemble d'accessibilité)

Problème de modélisation

FAST ne gère pas les files.

- files simulées par des compteurs,
- la correction de la simulation s'exprime comme une propriété du système à compteurs et est vérifiée par FAST automatiquement.

Résultats

Les propriétés de [Billington-Liu 2002] sont vérifiées facilement.

Gestion à la main des ressources mémoires (langage C)

on manipule : variables de pointeurs + tas mémoire

- tas mémoire = collection de cellules mémoires
- une cellule contient : des données ou des adresses
- adresse $\in \{\text{valide, invalide, NULL}\}$
- primitives : `new` et `free`

Erreurs fréquentes

- violation mémoire
- fuite mémoire

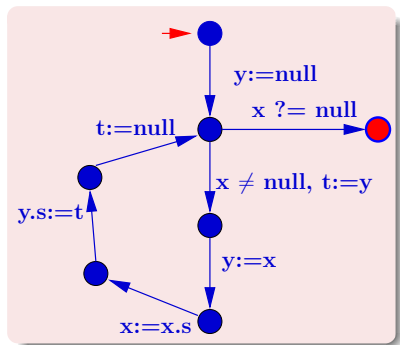
Travail dans le cadre d'un contrat EDF/LSV,
poursuivi en projet RNTL AVÉRILES

Systèmes à pointeurs

Programmes :

- un seul successeur (listes)
- pas de données, juste pointeurs

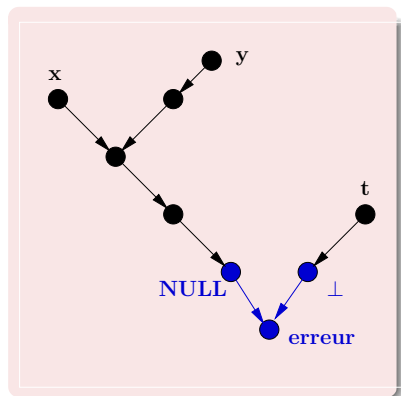
```
List reverse(List x) {  
  List y,t;  
  y =NULL;  
  while (x!=NULL) {  
    t=y;  
    y=x;  
    x=x->n;  
    y->n=t;  
    t=NULL;  
  }  
  return y;  
}
```



Configurations concrètes

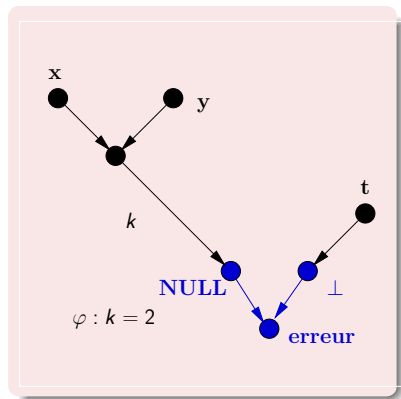
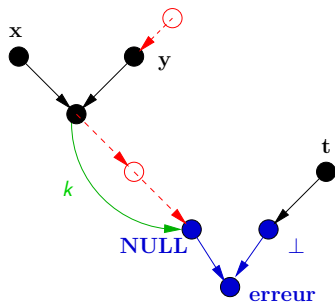
Graphes mémoires

- nœuds = cellules mémoires
- arc = “pointé par”
- étiquette : variables de pointeurs contenant l’adresse de la cellule
- \perp = adresses invalides



Résultats sur les systèmes à pointeurs [AVIS'04]

- représentation d'ensembles infinis de graphes mémoires
- = graphes mémoire (shape) + compteurs
- forme canonique des shapes
- **nombre fini de shape**



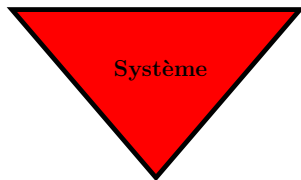
Vérification de systèmes à pointeurs [AVIS'06]

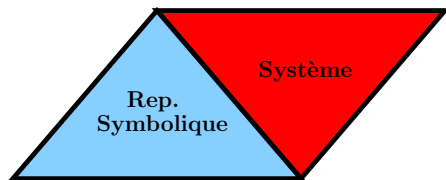
- codage des graphes mémoires dans du contrôle
- bisimulation du système à pointeurs par un système à compteurs
- vérification dans FAST

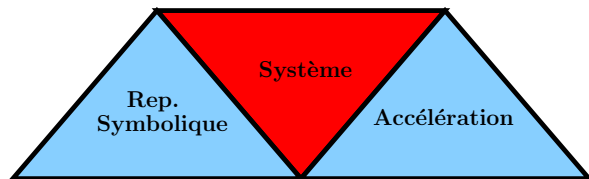
Début de prototype (avec Arnaud Sangnier et Étienne Lozes)

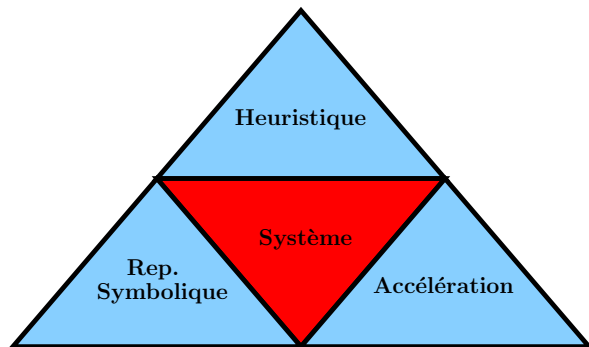
- Fonctionne bien sur ≈ 15 petits exemples classiques
- Propriétés qualitatives et quantitatives
- Permet de manipuler compteurs + pointeurs

- 1 Introduction
- 2 Systèmes à compteurs
- 3 Accélération de circuits
- 4 Sélection des circuits
- 5 L'outil FAST
- 6 Applications
- 7 **Conclusion**









1. Méthodologie générique [ATVA'05]
 - cadre de l'accélération
 - puissance et limite (systèmes aplatissables)
 - heuristique maximale
 - optimisations (réductions)
2. Instanciation aux systèmes à compteurs
 - deux algorithmes d'accélération
 - [Finkel-Leroux 2002]
 - [TACAS'04]
 - une réduction spécifique
 - l'outil FAST

3. Nombreuses expérimentations

- **Systemes à compteurs**
 - 40 systèmes infinis [CAV'03]
 - le TTP [TACAS'04]
- **Systemes à compteurs + files**
 - le CES (dans ma thèse, travail avec Laure Petrucci)
 - les Stop and Wait Protocol [Billington-Gallasch-Petrucci 2005]
- **Systemes à pointeurs**
 - travaux en cours [AVIS'06, AVIS'04]
 - 15 exemples classiques (travail avec Étienne Lozes et Arnaud Sangnier)

Limites théoriques

- $\text{post}_S^*(X_0)$ non représentable
- $\text{post}_S^*(X_0)$ non applatissable
- monoïde fini

Limites pratiques

- taille des automates avec le nombre de compteurs
- nombre de circuits

Court terme

Nouvelle version `FASTER` en chantier [soumission CAV'06]

- moteur indépendant de la bibliothèque de Presburger
- langage utilisateur plus riche
- automates binaires partagés (Couvreur)

Mixer BDD et automates binaires

Plus long terme

Passer à l'échelle : abstractions et raffinement

Compteurs temporisés ? (TPN, TA + compteurs, ...)

Model checking "borné" sur des ensembles infinis

Interface, langage d'entrée