

# Vérification et automates symboliques

Laure (Danthony-)Gonnord  
(Rappels et) Stage de DEA

Sous la direction de Nicolas Halbwachs  
Verimag - Grenoble - France  
mars-juin 2003

# Plan

- ➡ La vérification, qu'est-ce que c'est ?
- ➡ État de l'art
- ➡ Stage de DEA “de la logique des durées aux automates symboliques”

# Idée

Cahier des Charges  $\Rightarrow$  Logiciel

# Idée

Cahier des Charges  $\Rightarrow$  Logiciel

Environnement + Logiciel + Système Physique =  
Système Critique

# Idée

Cahier des Charges  $\Rightarrow$  Logiciel

Environnement + Logiciel + Système Physique =  
Système Critique

$\Rightarrow$  BOUM !

↳ Vérifier de façon **automatique** qu'un programme est **correct**

# Domaines

Différents domaines s'y rapportent :

# Domaines

Différents domaines s'y rapportent :

★ Automates

# Domaines

Différents domaines s'y rapportent :

- ★ Automates
- ★ Logique

# Domaines

Différents domaines s'y rapportent :

- ★ Automates
- ★ Logique
- ★ Maths Applis

# Domaines

Différents domaines s'y rapportent :

- ★ Automates
- ★ Logique
- ★ Maths Applis
- ★ Stats

# Domaines

Différents domaines s'y rapportent :

- ★ Automates
- ★ Logique
- ★ Maths Applis
- ★ Stats
- ★ Compilation

# Domaines

Différents domaines s'y rapportent :

- ★ Automates
- ★ Logique
- ★ Maths Applis
- ★ Stats
- ★ Compilation
- ★ . . . .

# Laboratoires de recherche

En France, les labos :

- ★ LSV (Cachan) : Vérification, Automates, . . .
- ★ LIAFA (Jussieu) : Algorithmique, Automates, Mots, Vérification . . .
- ★ Vérimag (Grenoble) : Conception de Systèmes, Vérification . . .
- ★ et aussi : LORIA (Nancy), Irisa (Ker Lahn), . . .

# Systemes étudiés

Différentes modélisations des systèmes :

★ Automates finis (mots finis ou infinis)

# Systemes étudiés

Différentes modélisations des systèmes :

- ★ Automates finis (mots finis ou infinis)
- ★ Automates temporisés = automates + horloges

# Systemes étudiés

Différentes modélisations des systèmes :

- ★ Automates finis (mots finis ou infinis)
- ★ Automates temporisés = automates + horloges
- ★ Automates hybrides = automates + var. dynamiques

# Systemes étudiés

Différentes modélisations des systèmes :

- ★ Automates finis (mots finis ou infinis)
- ★ Automates temporisés = automates + horloges
- ★ Automates hybrides = automates + var. dynamiques
- ★ Traces (systèmes concurrents)

# Systemes étudiés

Différentes modélisations des systèmes :

- ★ Automates finis (mots finis ou infinis)
- ★ Automates temporisés = automates + horloges
- ★ Automates hybrides = automates + var. dynamiques
- ★ Traces (systèmes concurrents)
- ★ Programmes assembleur

# Systemes étudiés

Différentes modélisations des systèmes :

- ★ Automates finis (mots finis ou infinis)
- ★ Automates temporisés = automates + horloges
- ★ Automates hybrides = automates + var. dynamiques
- ★ Traces (systèmes concurrents)
- ★ Programmes assembleur
- ★ . . . .

↳ Différentes méthodes

# Différentes méthodes complémentaires

Méthodes classiques :

- ★ Simulation/Test : pb de “couverture”
- ★ Theorem Proving : Coq, Isabelle, . . .
- ★ Vérification Formelle :
  - Model-checking :  $\pm$  vérification exhaustive des états
  - Analyse approchée : interprétation abstraite

# A l'encontre des idées reçues

2 constats

★ Les problèmes intéressants sont **indécidables**

# A l'encontre des idées reçues

2 constats

- ★ Les problèmes intéressants sont **indécidables**
- ★ Les rares algos sont de complexité **monstrueuse**

**MAIS**

# A l'encontre des idées reçues

2 constats

- ★ Les problèmes intéressants sont **indécidables**
- ★ Les rares algos sont de complexité **monstrueuse**

**MAIS**

- ★ Des semi-algorithmes suffisent souvent

# A l'encontre des idées reçues

2 constats

- ★ Les problèmes intéressants sont **indécidables**
- ★ Les rares algos sont de complexité **monstrueuse**

## MAIS

- ★ Des semi-algorithmes suffisent souvent
- ★ Un algorithme exponentiel peut s'exécuter rapidement en pratique.

# Interprétation abstraite

Méthode (Cousot 72/78) :

- ★ Définition d'un **opérateur d'élargissement**
  - ★ Calcul d'ensembles emboîtés en avant puis en arrière
  - ★ Avec de la **chance**, diagnostic **YES**
- ↳ Algorithme **conservatif**

# Interprétation abstraite - 2 - Exemple

`x:=1`

(1)

`while x < 10000 do`

(2)

`x := x + 1`

(3)

`od;`

(4)

# Interprétation abstraite - 2 - Exemple

`x:=1`

(1)

$$X_1 = [1, 1]$$

`while x < 10000 do`

(2)

$$X_2 = (X_1 \cup X_3) \cap [-\infty, 9999]$$

`x := x + 1`

(3)

$$X_3 = X_2 \oplus [1, 1]$$

`od;`

(4)

$$X_4 = (X_1 \cup X_3) \cap [10000, +\infty]$$

# Interprétation abstraite - 2 - Exemple

`x:=1`

(1)

$$X_1 = [1, 1]$$

`while x < 10000 do`

(2)

$$X_2 = (X_1 \cup X_3) \cap [-\infty, 9999]$$

`x := x + 1`

(3)

$$X_3 = X_2 \oplus [1, 1]$$

`od;`

(4)

$$X_4 = (X_1 \cup X_3) \cap [10000, +\infty]$$

↳ Et maintenant, **calculons!**

# Interprétation abstraite - 3 - exemple

Itérations croissantes et décroissantes :

# Interprétation abstraite - 3 - exemple

Itérations croissantes et décroissantes :

$X_1$	$\emptyset$	$[1, 1]$	$[1, 1]$	$[1, 1]$	$[1, 1]$	$[1, 1]$	$[1, 1]$	$[1, 1]$
$X_2$	$\emptyset$	$\emptyset$	$[1, 1]$	$[1, 1]$	$[1, 2]$	$[1, +\infty]$	$[1, +\infty]$	$[1, 9999]$
$X_3$	$\emptyset$	$\emptyset$	$\emptyset$	$[2, 2]$	$[2, 2]$	$[2, 2]$	$[2, 10000]$	$[2, 10000]$
$X_4$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$[10000, 10000]$

## Interprétation abstraite - 3 - exemple

Itérations croissantes et décroissantes :

$X_1$	$\emptyset$	$[1, 1]$	$[1, 1]$	$[1, 1]$	$[1, 1]$	$[1, 1]$	$[1, 1]$	$[1, 1]$
$X_2$	$\emptyset$	$\emptyset$	$[1, 1]$	$[1, 1]$	$[1, 2]$	$[1, +\infty]$	$[1, +\infty]$	$[1, 9999]$
$X_3$	$\emptyset$	$\emptyset$	$\emptyset$	$[2, 2]$	$[2, 2]$	$[2, 2]$	$[2, 10000]$	$[2, 10000]$
$X_4$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$[10000, 10000]$

↳ **Sur-approximation** du point fixe  $\Rightarrow$  pas de dépassement de capacité.

# Et maintenant

Le stage de DEA ...

# Cadre de l'Etude

- Automates symboliques

# Cadre de l'Etude

- Automates symboliques
- Variables numériques

# Cadre de l'Etude

- Automates symboliques
- Variables numériques
- ↳ Comme d'habitude, vérification **conservative** !

# Le langage Lustre

★ Modèle synchrone (E/S synchronisées)

# Le langage Lustre

- ★ Modèle synchrone (E/S synchronisées)
- ★ Notion de flot de variables

# Le langage Lustre

- ★ Modèle synchrone (E/S synchronisées)
- ★ Notion de flot de variables



- ★ Définitions par équation  $s_1 = e_1 \wedge e_2 \wedge \text{pre}(e_3)$   
( $\forall n, s_1^{(n)} = e_1^{(n)} \wedge e_2^{(n)} \wedge e_3^{(n-1)}$ )

# Le langage Lustre

- ★ Modèle synchrone (E/S synchronisées)
- ★ Notion de flot de variables

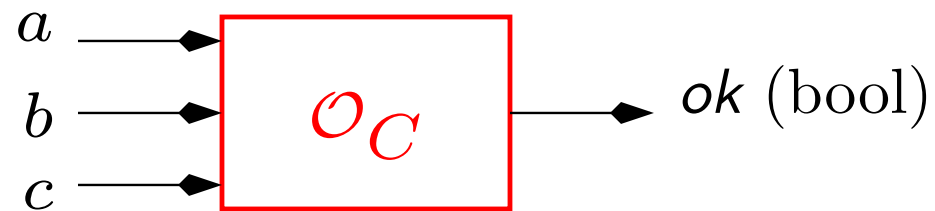


- ★ Définitions par équation  $s_1 = e_1 \wedge e_2 \wedge \text{pre}(e_3)$   
( $\forall n, s_1^{(n)} = e_1^{(n)} \wedge e_2^{(n)} \wedge e_3^{(n-1)}$ )

↳ automates **symboliques**

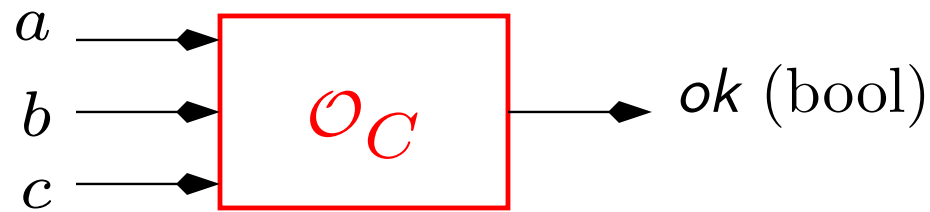
# Observateurs

★ Propriété  $\Rightarrow$  Automate Observateur



# Observateurs

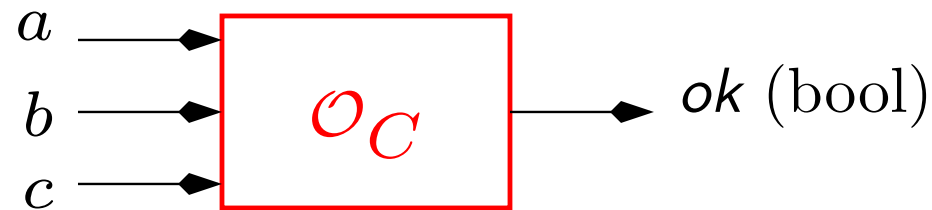
★ Propriété  $\Rightarrow$  Automate Observateur



★ Spécification dans le même langage

# Observateurs

★ Propriété  $\Rightarrow$  Automate Observateur

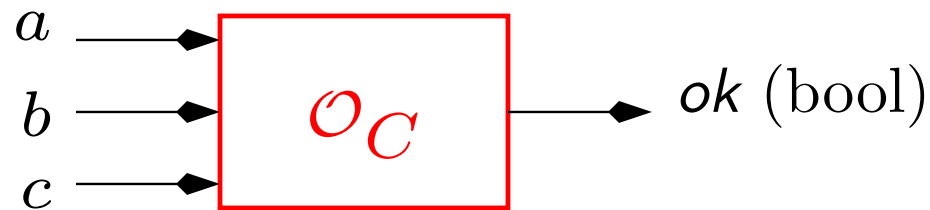


★ Spécification dans le même langage

★ Exécution **en parallèle**

# Observateurs

★ Propriété  $\Rightarrow$  Automate Observateur



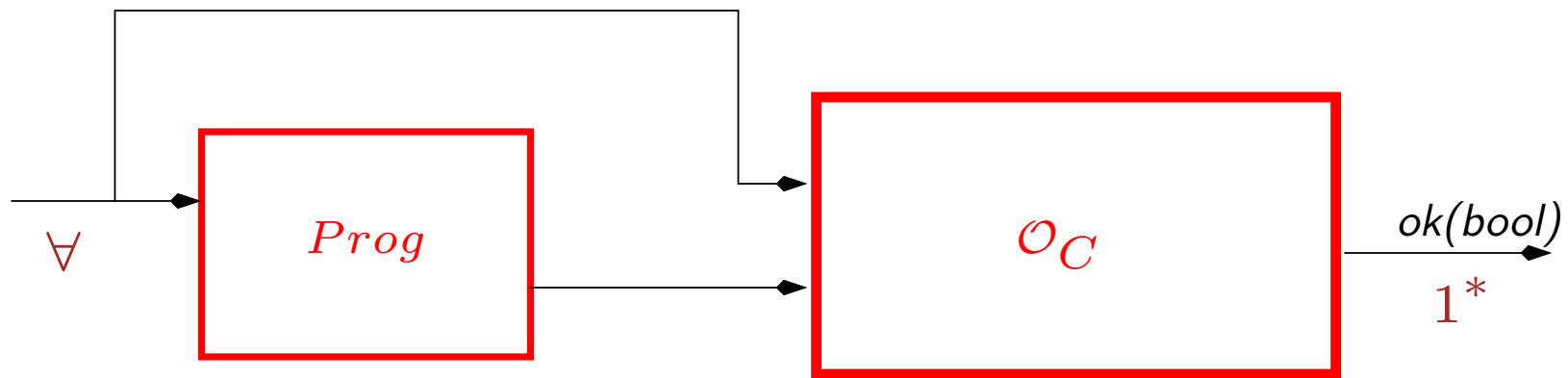
★ Spécification dans le même langage

★ Exécution en parallèle

↳ But : prouver  $L(\neg C) \cap L(Prog) = \emptyset$

# Observateurs - 2

Schéma :



# Autour de Lustre

Des outils de vérification :

★ **Lésar** : contrôle uniquement : méthode énumérative ou symbolique.

# Autour de Lustre

Des outils de vérification :

- ★ **Lésar** : contrôle uniquement : méthode énumérative ou symbolique.
- ★ **NBac** : contrôle + approximation supérieure des états atteignables des variables numériques (**polyèdres convexes**).

# Autour de Lustre

Des outils de vérification :

- ★ **Lésar** : contrôle uniquement : méthode énumérative ou symbolique.
- ★ **NBac** : contrôle + approximation supérieure des états atteignables des variables numériques (**polyèdres convexes**).

↳ Utilisation des outils existants

# Une logique temporelle

“Quantified discrete duration calculus” [Pandya] :  
★ Une logique temporelle

# Une logique temporelle

“Quantified discrete duration calculus” [Pandya] :

- ★ Une logique temporelle
- ★ Une logique d’intervalles

# Une logique temporelle

“Quantified discrete duration calculus” [Pandya] :

- ★ Une logique temporelle
- ★ Une logique d’intervalles
- ★ Modèles : suites finies de fonctions booléennes

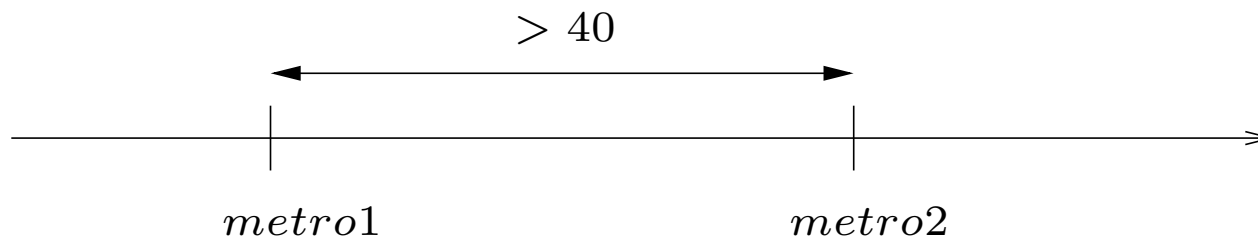
# Une logique temporelle

“Quantified discrete duration calculus” [Pandya] :

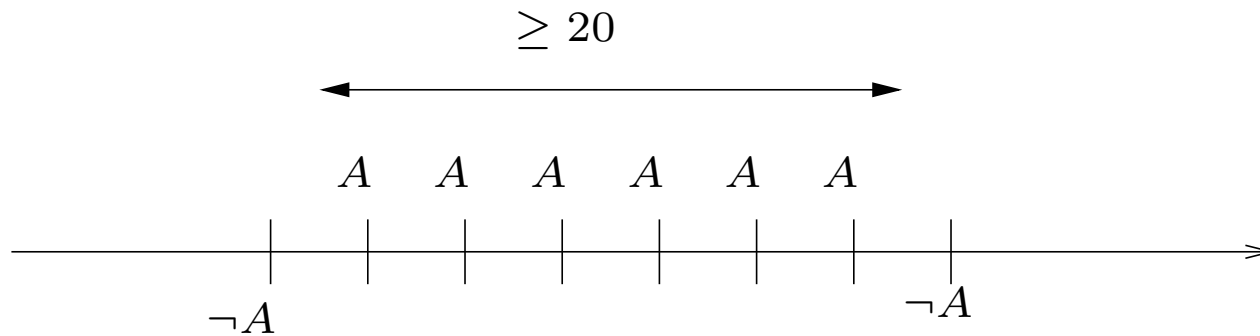
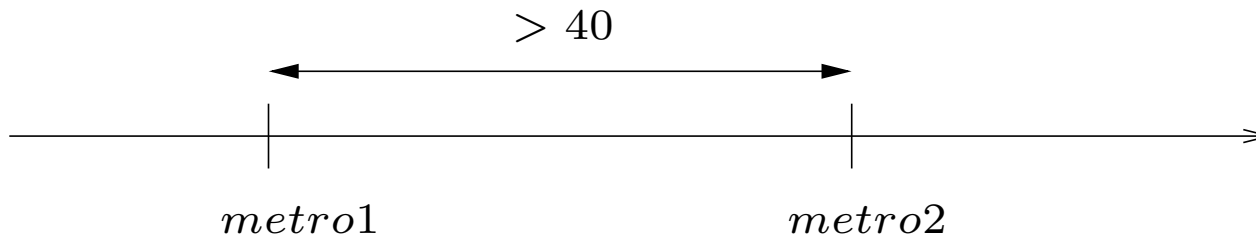
- ★ Une logique temporelle
- ★ Une logique d'intervalles
- ★ Modèles : suites finies de fonctions booléennes

↳ **But** : Construire  $\mathcal{O}_\varphi$

# Exemples de propriétés



# Exemples de propriétés



# Exemples simples d'observateurs

★ Soit  $\varphi_1 = \llbracket p \rrbracket$  :

ok = p  $\rightarrow$  p and pre(ok)

# Exemples simples d'observateurs

★ Soit  $\varphi_1 = \llbracket p \rrbracket$  :

ok = p  $\rightarrow$  p and pre(ok)

★ Soit  $\varphi_2 = \Sigma P \leq \delta$  :

nb\_p = 0  $\rightarrow$  (if p then pre(nb\_p)+1 else pre(nb\_p));

ok = nb\_p  $\leq$  delta

# Exemples simples d'observateurs

★ Soit  $\varphi_1 = \llbracket p \rrbracket$  :

$ok = p \rightarrow p \text{ and } pre(ok)$

★ Soit  $\varphi_2 = \Sigma P \leq \delta$  :

$nb\_p = 0 \rightarrow (\text{if } p \text{ then } pre(nb\_p)+1 \text{ else } pre(nb\_p));$

$ok = nb\_p \leq \delta$

↳ **Automatisons** tout ça !

# Conclusion

- ★ Travail réalisé : identification de deux fragments QDDC-DET et QDDC-INDET(avec oracle)

# Conclusion

- ★ Travail réalisé : identification de deux fragments QDDC-DET et QDDC-INDET(avec oracle)
- ★ Traduction prouvée et **implémentée**

# Conclusion

- ★ Travail réalisé : identification de deux fragments QDDC-DET et QDDC-INDET(avec oracle)
- ★ Traduction prouvée et implémentée
- ★ Études de cas :

# Conclusion

- ★ Travail réalisé : identification de deux fragments QDDC-DET et QDDC-INDET(avec oracle)
- ★ Traduction prouvée et implémentée
- ★ Études de cas :
  - expressivité bonne (Mine Pump)

# Conclusion

- ★ Travail réalisé : identification de deux fragments QDDC-DET et QDDC-INDET(avec oracle)
- ★ Traduction prouvée et **implémentée**
- ★ Études de cas :
  - expressivité **bonne** (Mine Pump)
  - vérification de propriétés paramétrées.

# Conclusion

- ★ Travail réalisé : identification de deux fragments QDDC-DET et QDDC-INDET(avec oracle)
- ★ Traduction prouvée et **implémentée**
- ★ Études de cas :
  - expressivité **bonne** (Mine Pump)
  - vérification de propriétés paramétrées.
  - vérification pas toujours concluante

# Perspectives

- ★ Automates à oracles pour d'autres logiques

# Perspectives

- ★ Automates à oracles pour d'autres logiques
- ★ Meilleurs outils pour les variables numériques

# Perspectives

- ★ Automates à oracles pour d'autres logiques
- ★ Meilleurs outils pour les variables numériques
  - ↳ Poursuite du travail **en thèse**

# Et enfin

<http://laure.gonnord.org>

<http://www-verimag.imag.fr>