Résolution générique “à la volée” de systèmes d’équations booléennes et applications

Radu Mateescu

INRIA Rhône-Alpes / VASY

Plan

- Introduction
- Systèmes d'équations booléennes d'alternance 1
- Algorithmes de résolution "à la volée"
- Vérification par équivalence et logique temporelle
- Implémentation et expérimentation
- Conclusion et travaux futurs

Introduction

- Vérification énumérative :
  - Fiabilité des systèmes concurrents à nombre fini d'états
  - Construction de l'espace d'états par énumération *explicite* des états (≠ vérification *symbolique*)
  - Adaptée aux systèmes asynchrones non-déterministes
- Approches traditionnelles :
  - Vérification par équivalence (*equivalence checking*)
  - Vérification par logique temporelle (*model checking*)
- Solution adoptée :
  - Traduction du problème de vérification vers la résolution d'un *système d'équations booléennes* (SEB)
  - Génération de *diagnostics* (fragments de l'espace d'états) expliquant le résultat de la vérification

Systèmes d'équations booléennes (syntaxe)

Un SEB est un tuple $B = (x, M_1, \ldots, M_n)$, où

- $x \in \boldsymbol{X}$: variable booléenne principale
- $M_i = \{ x_j =\sigma_i \, op_j \, \boldsymbol{X}_j \}_{j \text{ in } [1, mi]}$: blocs d'équations
  - $\sigma_i \in \{ \mu, \nu \}$: signe (point fixe) du bloc i
  - $op_j \in \{ \vee, \wedge \}$: opérateur de l'équation j
  - $\boldsymbol{X}_j \subseteq \boldsymbol{X}$: variables en partie droite de l'équation j
  - F = $\vee\emptyset$ (disjonction vide) et T = $\wedge\emptyset$ (conjonction vide)
  - x_j dépend de x_k ssi $x_k \in \boldsymbol{X}_j$
  - M_i dépend de M_l ssi une x_j de M_i dépend d'une x_k de M_l
  - Bloc *fermé* : ne dépend pas d'autres blocs
- SEB d'*alternance* 1 : M_i ne dépend que de $M_{i+1} \ldots M_n$

Exemple

$$M_1: \begin{cases} x_1 =_\mu x_2 \vee x_3 \\ x_2 =_\mu x_3 \vee x_4 \\ x_3 =_\mu x_2 \wedge x_7 \end{cases}$$

$$M_3: \begin{cases} x_7 =_\nu x_8 \wedge x_9 \\ x_8 =_\nu T \\ x_9 =_\nu F \end{cases}$$

$$M_2: \begin{cases} x_4 =_\mu x_5 \vee x_6 \\ x_5 =_\mu x_8 \vee x_9 \\ x_6 =_\mu F \end{cases}$$

Blocs particuliers

- Bloc *acyclique* :
  - Pas de dépendances cycliques entre les variables du bloc
- Var. x_i disjonctive (conjonctive) : $op_i = \vee$ ($op_i = \wedge$)
- Bloc *disjonctif* :
  - Comporte des variables disjonctives
  - Et des variables conjonctives
    - avec un seul successeur non constant dans le bloc (le dernier en partie droite de l'équation)
    - les autres successeurs sont des constantes ou des variables libres (définies dans d'autres blocs)
- Bloc *conjonctif* : définition duale

Systèmes d'équations booléennes (sémantique)

- Contexte : fonction partielle $\delta : \boldsymbol{X} \rightarrow$ Bool
- Sémantique d'une formule booléenne :
  - $[[\ op\ \{ x_1, \ldots, x_p \}\ ]]\ \delta = op\ (\delta\ (x_1), \ldots, \delta\ (x_p))$
- Sémantique d'un bloc :
  - $[[\ \{ x_j =\sigma\ op_j\ \boldsymbol{X}_j \}_{j \text{ in } [1, m]}\ ]]\ \delta = \sigma\Phi_\delta$
  - $\Phi_\delta : \mathrm{Bool}^m \rightarrow \mathrm{Bool}^m$
  - $\Phi_\delta\ (b_1, \ldots, b_m) = ([[\ op_j\ \boldsymbol{X}_j\ ]]\ (\delta \oplus [b_1/x_1, \ldots, b_m/x_m]))_{j \text{ in } [1, m]}$
- Sémantique d'un SEB :
  - $[[\ (x, M_1, \ldots, M_n)\ ]] = \delta_1\ (x)$
  - $\delta_n = [[\ M_n\ ]]\ []$ (M_n fermé)
  - $\delta_i = ([[\ M_i\ ]]\ \delta_{i+1}) \oplus \delta_{i+1}$ (M_i dépend de $M_{i+1} \ldots M_n$)

Résolution globale

- SEB $B = (x, M_1, \ldots, M_n)$ d'alternance 1
- Primitive : calcul de la sémantique d'un bloc
  - Par ordre inverse des dépendances
    $\delta_n = [[\ M_n\ ]]\ []\quad , \quad \delta_i = ([[\ M_i\ ]]\ \delta_{i+1}) \oplus \delta_{i+1}$
  - Application « brutale » du théorème de Knaster-Tarski
    $[[\ \{\ x_j =\mu\ op_j\ X_j\ \}_{j \text{ in } [1,\, m]}\ ]]\ \delta = \mu\Phi_\delta = \cup_{k\geq 0}\ \Phi_\delta{}^k\ (F, \ldots, F)$
    $[[\ \{\ x_j =\nu\ op_j\ X_j\ \}_{j \text{ in } [1,\, m]}\ ]]\ \delta = \nu\Phi_\delta = \cap_{k\geq 0}\ \Phi_\delta{}^k\ (T, \ldots, T)$
  - Prendre la valeur de x dans M_1 : $\delta_1\ (x)$
- Inconvénients :
  - Nécessite la construction complète du SEB
  - Risque de calculer des informations « inutiles »

Exemple

$$M_1 \quad \left\{ \begin{array}{l} x_1 =_\mu x_2 \vee x_3 \\ x_2 =_\mu x_3 \vee x_4 \\ x_3 =_\mu x_2 \wedge x_7 \end{array} \right.$$

$$M_3 \quad \left\{ \begin{array}{l} x_7 =_\nu x_8 \wedge x_9 \\ x_8 =_\nu \mathrm{T} \\ x_9 =_\nu \mathrm{F} \end{array} \right.$$

$$M_2 \quad \left\{ \begin{array}{l} x_4 =_\mu x_5 \vee x_6 \\ x_5 =_\mu x_8 \vee x_9 \\ x_6 =_\mu \mathrm{F} \end{array} \right.$$

© 2003 Radu Mateescu

Résolution locale

- SEB $B = (x, M_1, \ldots, M_n)$ d'alternance 1
- Primitive : calcul d'une variable d'un bloc
  - Une routine de résolution R_i associée à M_i
  - R_i (x_j) calcule la valeur de xj dans Mi
  - Evaluation des parties droites des équations+ substitution
  - Pile des appels R_1 $(x) \rightarrow \ldots \rightarrow R_n$ (x_k) bornée par la profondeur du graphe de dépendances entre blocs
  - Style « coroutine » : chaque Ri doit garder son contexte
- Avantages :
  - Permet de construire le SEB « à la volée »
  - Calcule uniquement des informations « utiles »

© 2003 Radu Mateescu

Exemple

$$M_1 : \begin{cases} x_1 =_\mu x_2 \vee x_3 \\ x_2 =_\mu x_3 \vee x_4 \\ x_3 =_\mu x_2 \wedge x_7 \end{cases}$$

$$M_2 : \begin{cases} x_4 =_\mu x_5 \vee x_6 \\ x_5 =_\mu x_8 \vee x_9 \\ x_6 =_\mu F \end{cases}$$

$$M_3 : \begin{cases} x_7 =_\nu x_8 \wedge x_9 \\ x_8 =_\nu T \\ x_9 =_\nu F \end{cases}$$

© 2003 Radu Mateescu

Principe des algorithmes locaux

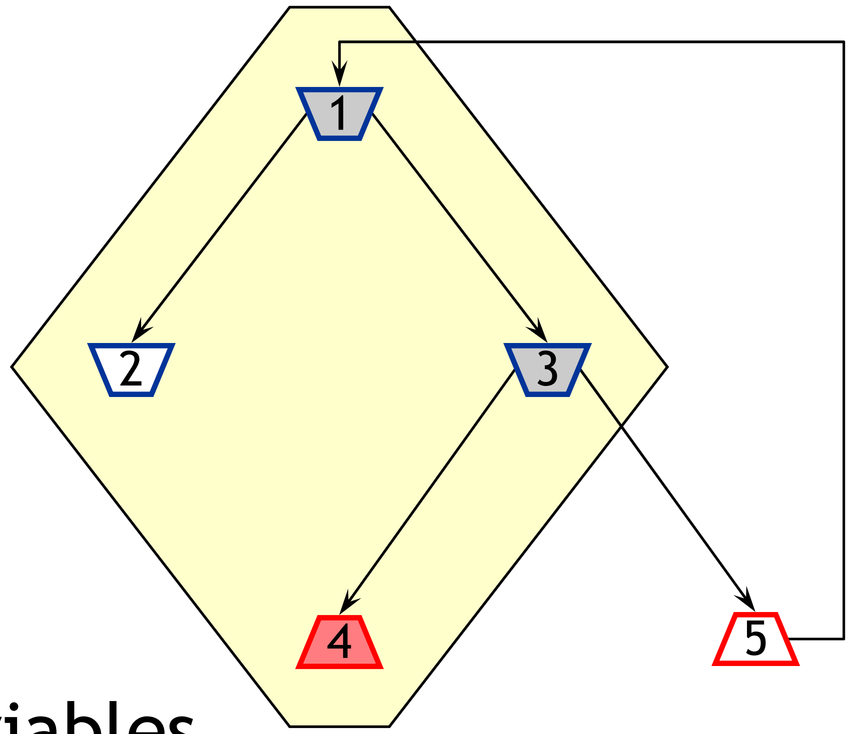
- Représentation des blocs comme *graphes booléens*
- Au bloc $M = \{ x_j =_\mu op_j \, \boldsymbol{X}_j \}_{j \text{ in } [1, m]}$
  on associe le graphe booléen $G = (V, E, L, \mu)$, où :
  - $V = \{ x_1, \ldots, x_m \}$: ensemble de sommets (variables)
  - $E = \{ (x_i, x_j) \mid x_j \in \boldsymbol{X}_i \}$: ensemble d'arcs (dépendances)
  - $L : V \rightarrow \{ \vee, \wedge \}$, $L(x_j) = op_j$: étiquetage des sommets
- Principe des algorithmes :
  - Exploration *en avant* de G en partant de $x \in V$
  - Propagation *en arrière* des variables stables (calculées)
  - Terminaison : x est stable ou G est exploré totalement

Exemple

SEB (μ-bloc)

$$\begin{cases} x_1 =_\mu x_2 \vee x_3 \\ x_2 =_\mu F \\ x_3 =_\mu x_4 \vee x_5 \\ x_4 =_\mu T \\ x_5 =_\mu x_1 \end{cases}$$

graphe booléen



: $\vee$-variables

: $\wedge$-variables

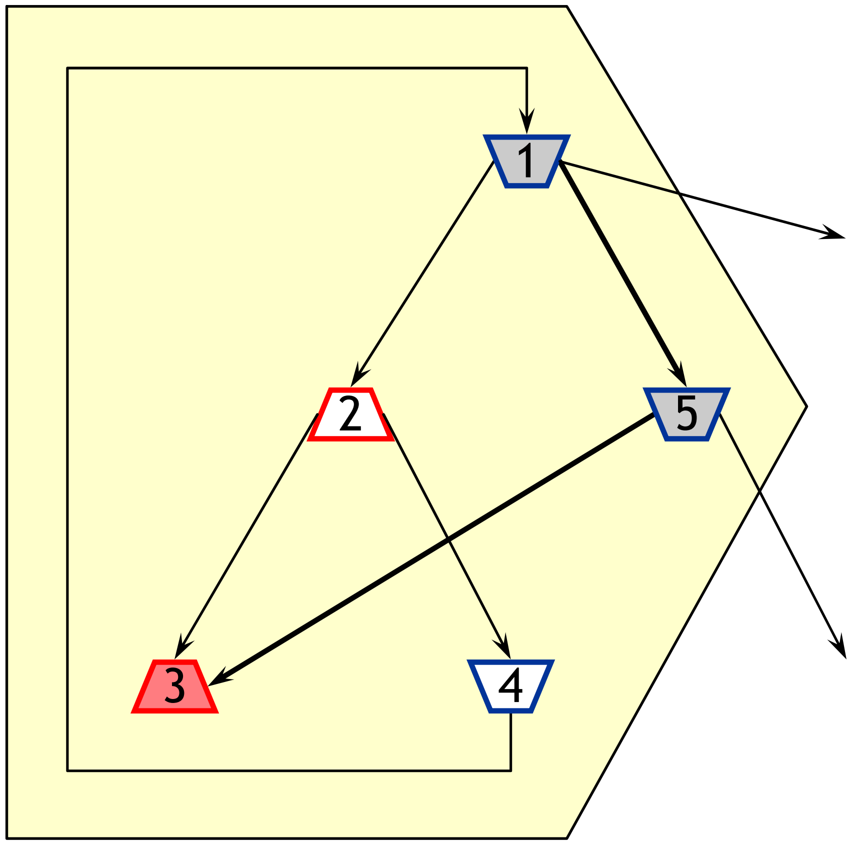
Trois critères d'efficacité

Soit $B = (x, M_1, \ldots, M_n)$ un SEB d'alternance 1.
Pour chaque routine R_i associée à M_i :

- La complexité en temps de R_i (x_j) dans le pire des cas doit être O $(|V|+|E|)$
  ➔ complexité linéaire pour la résolution de B
- Pendant l'exécution de R_i (x_j), chaque nouvelle variable explorée doit être « reliée » à x_j par (au moins) une séquence de variables instables
  ➔ limiter l'exploration du graphe aux variables « utiles »
- Après la fin de R_i (x_j), toutes les variables explorées doivent être stables
  ➔ mémoriser les résultats entre appels successifs de R_i (xj)

Algorithme A1

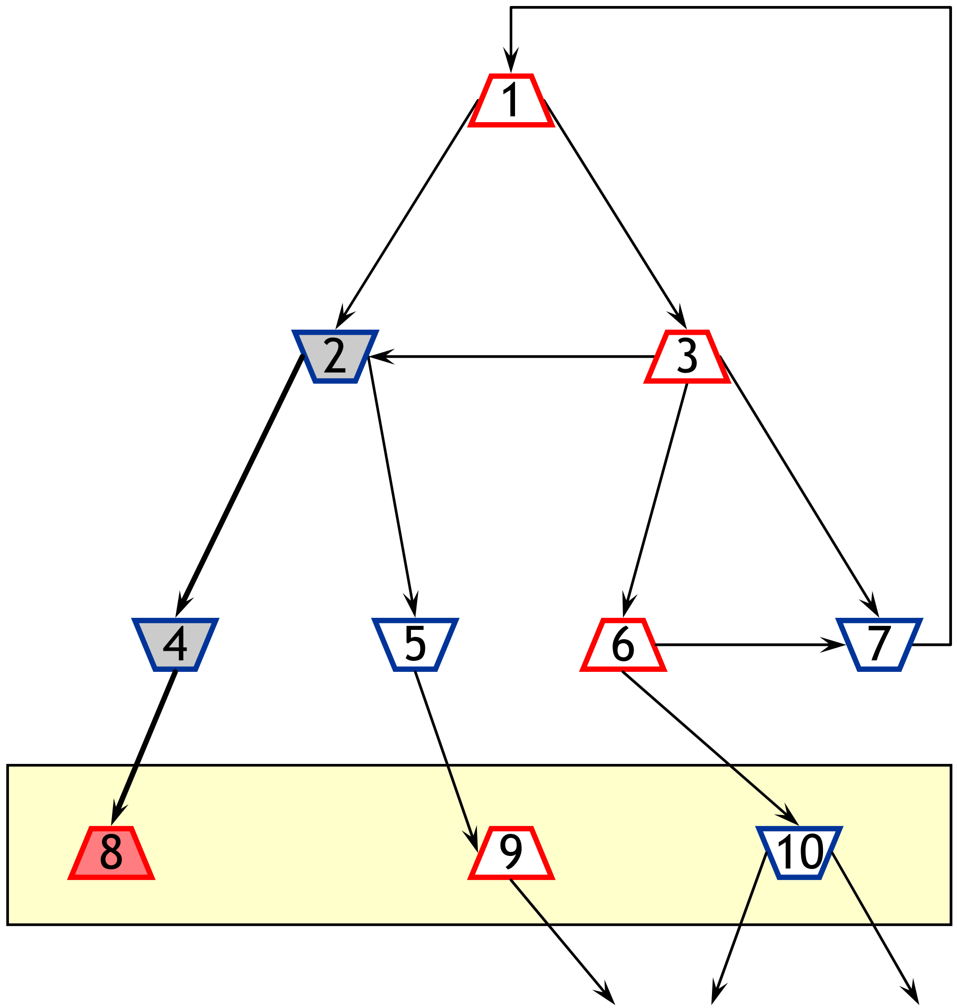
- S'applique à tous les types de blocs
- Parcours en profondeur (DFS) du graphe booléen
- Propagation en arrière des variables stables
- Pré-calcul de diagnostic
- Satisfait A, B, C
- Complexité en mémoire $O\,(|V|+|E|)$
- Version optimisée de [Andersen-94]
- Développé pour le μ-calcul régulier (EVALUATOR 3.0) [Mateescu-Sighireanu-00]



© 2003 Radu Mateescu

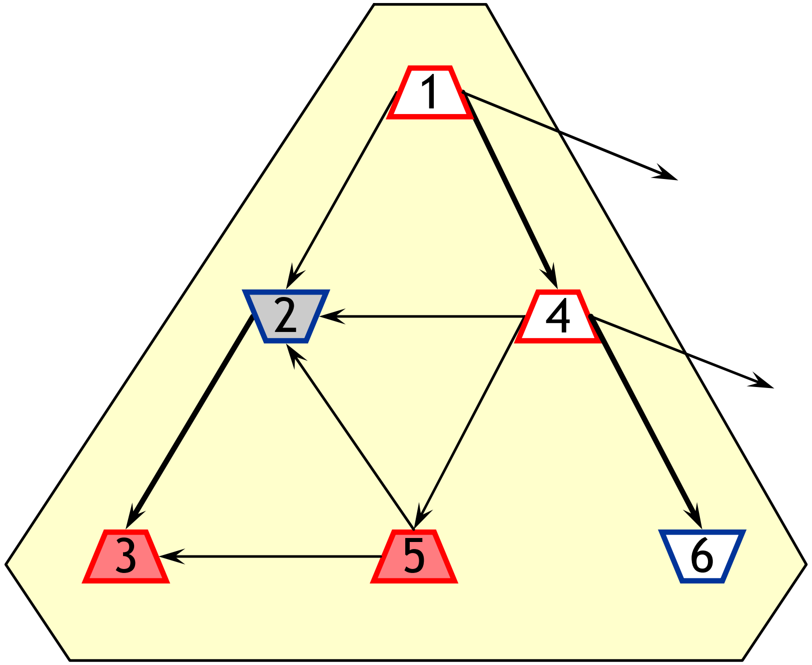
Algorithme A2

- S'applique à tous les types de blocs
- Parcours en largeur (BFS) du graphe booléen
- Propagation en arrière des variables stables
- Pré-calcul de diagnostic
- Satisfait A, C
- Complexité en mémoire $O\ (|V| + |E|)$
- Diagnostics de profondeur réduite



Algorithme A3

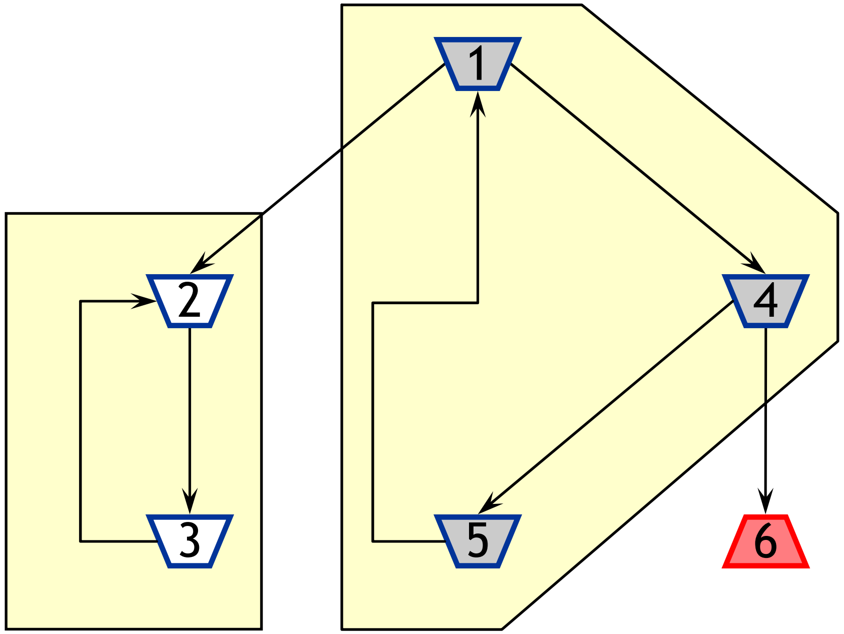
- S'applique uniquement aux blocs *acycliques*
- Parcours DFS du graphe booléen
- Propagation en arrière des variables stables
- Pré-calcul de diagnostic
- Satisfait A, B, C
- Pas de stockage des arcs
- Complexité en mémoire $O(|V|)$
- Développé pour vérifier des traces de simulation [Mateescu-02]



© 2003 Radu Mateescu

Algorithme A4

- S'applique uniquement aux blocs *disjonctifs* ou *conjonctifs*
- Parcours en profondeur (DFS) du graphe booléen
- Propagation en arrière des variables stables
- Détection et stabilisation des CFC
- Pas de stockage des arcs
- Satisfait A, B, C
- Complexité en mémoire $O(|V|)$



CFC de variables F

CFC de variables T

Récapitulatif

- A1 (DFS, général)
  - Satisfait A, B, C
  - Complexité en mémoire $O(|V|+|E|)$
- A2 (BFS, général)
  - Satisfait A, C + diagnostics « petits »
  - Complexité en mémoire $O(|V|+|E|)$
- A3 (DFS, acyclique)
  - Satisfait A, B, C
  - Complexité en mémoire $O(|V|)$
- A4 (DFS, disjonctif)
  - Satisfait A, B, C
  - Complexité en mémoire $O(|V|)$

Complexité en temps $O(|V|+|E|)$

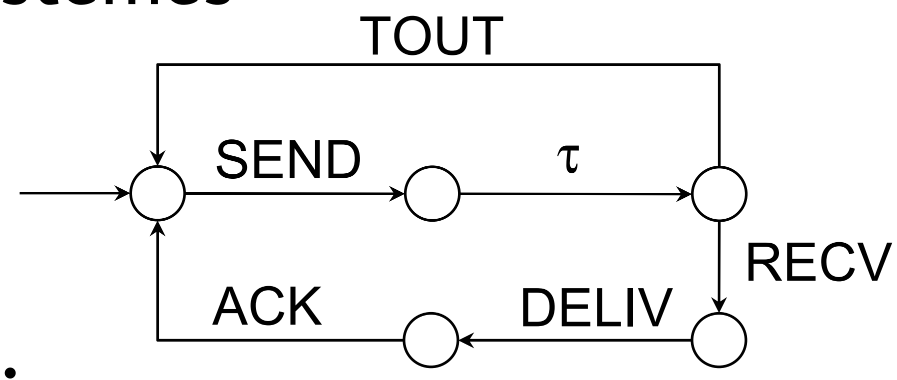
Applications

- Vérification par *équivalences/préordres*
- Vérification par *logiques temporelles*

à la volée

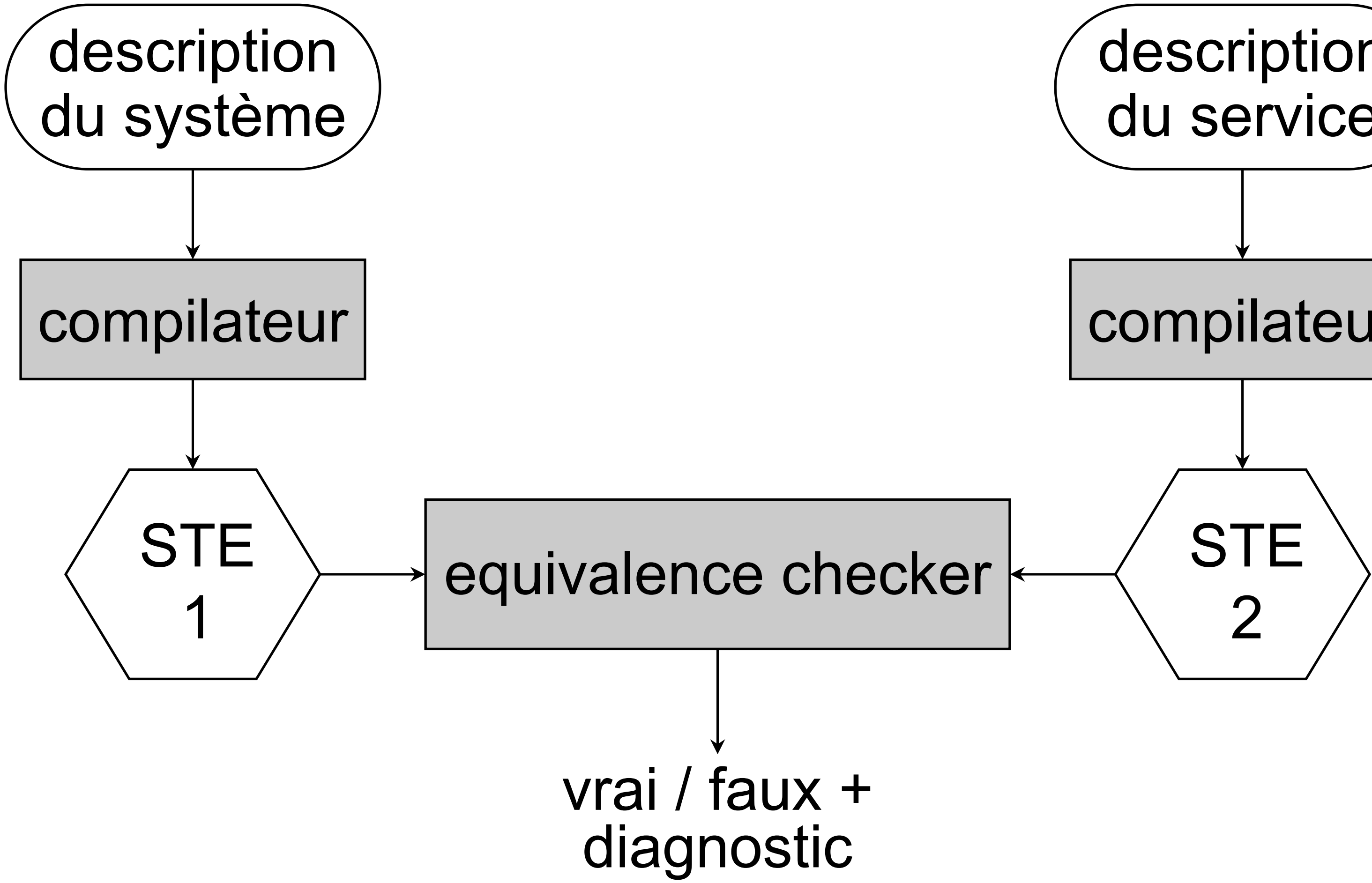
Systèmes de transitions étiquetées

- STEs : modèles pour les systèmes parallèles asynchrones
- Un STE $M = (S, A, T, s_0)$



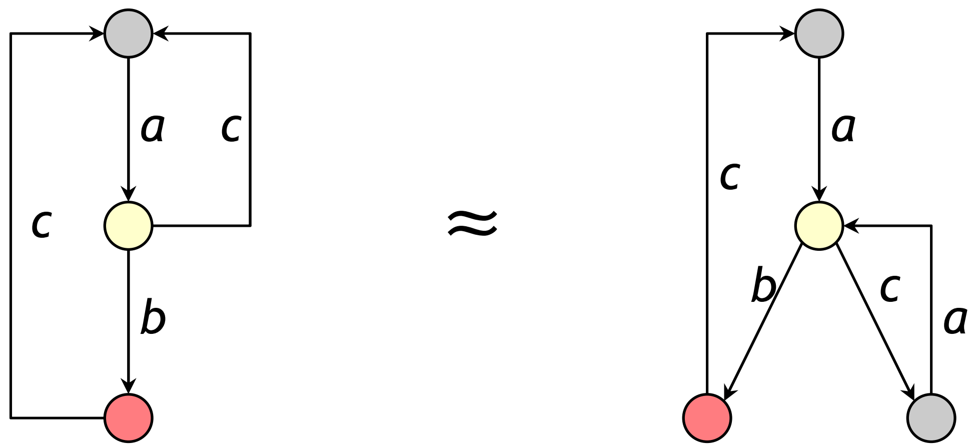
- Représentations d'un STE :
  - *explicite* (fonction « prédécesseur » et/ou « successeur »)
    - Calculs itératifs sur les ensembles d'états
    - Environnement **BCG** (*Binary Coded Graphs*) [Garavel-92]
  - *implicite* (fonction « successeur »)
    - Exploration à la volée de la relation de transition
    - Environnement **Open / Caesar** [Garavel-98]

Vérification par équivalence (equivalence checking)



Equivalence forte

- Soit 2 STE $M_1 = (S_1, A, T_1, s_{01})$ et $M_2 = (S_2, A, T_2, s_{02})$
  $\approx\, \subseteq S_1 \times S_2$ est la plus grande relation t.q. $s_1 \approx s_2$ ssi
  $\forall a \in A \;.\; \forall s_1 \rightarrow_a s_1' \in T_1 \;.\; \exists s_2 \rightarrow_a s_2' \in T_2 \;.\; s_1' \approx s_2'$
  et
  $\forall a \in A \;.\; \forall s_2 \rightarrow_a s_2' \in T_2 \;.\; \exists s_1 \rightarrow_a s_1' \in T_1 \;.\; s_1' \approx s_2'$
- $M_1 \approx M_2$ ssi $s_{01} \approx s_{02}$



Traduction vers un SEB

- Principe : $s_1 \approx s_2$ ssi $X_{s1,s2}$ est vraie
- SEB général :

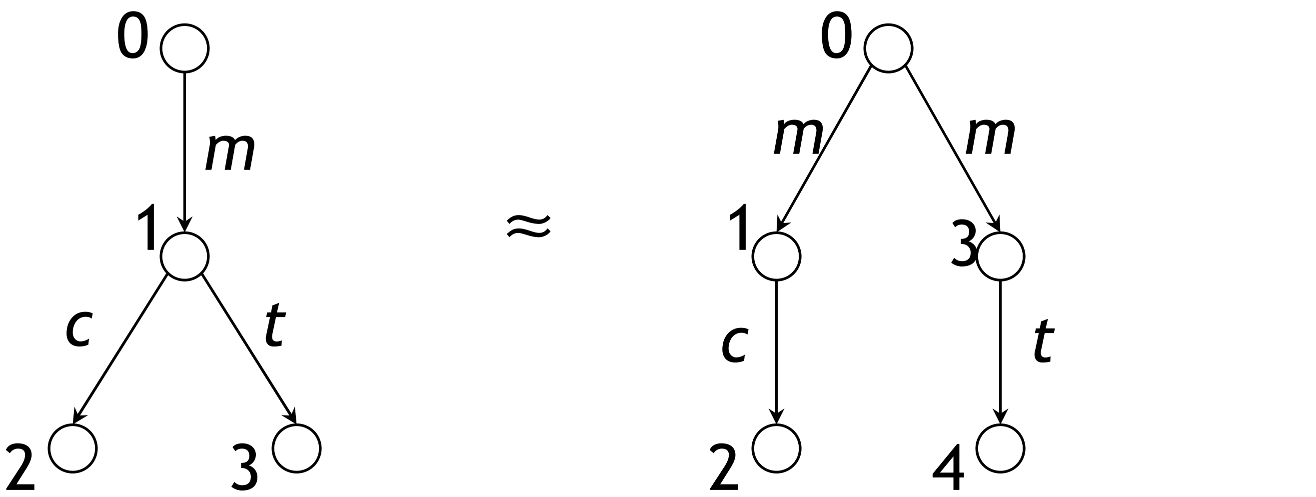
$$\left\{ \begin{array}{l} \boxed{X_{s1,s2} =_\nu (\wedge_{s1 \to a\ s1'} \vee_{s2 \to a\ s2'} X_{s1',s2'})} \\ \qquad\qquad \wedge \\ \qquad\qquad (\wedge_{s2 \to a\ s2'} \vee_{s1 \to a\ s1'} X_{s1',s2'}) \end{array} \right.$$

- SEB simplifié :

$$\left\{ \begin{array}{l} X_{s1,s2} =_\nu (\wedge_{s1 \to a\ s1'} Y_{a,s1',s2}) \wedge (\wedge_{s2 \to a\ s2'} Z_{a,s1,s2'}) \\ Y_{a,s1',s2} =_\nu \vee_{s2 \to a\ s2'} X_{s1',s2'} \\ Z_{a,s1,s2'} =_\nu \vee_{s1 \to a\ s1'} X_{s1',s2'} \end{array} \right.$$

$s_1 \leq s_2$
(préordre)

Exemple



SEB général :

$$\begin{cases} X_{00} =_\nu (X_{11} \vee X_{13}) \wedge X_{11} \wedge X_{13} \\ X_{11} =_\nu (X_{22} \wedge \mathrm{F}) \wedge X_{22} \\ X_{13} =_\nu (\mathrm{F} \wedge X_{34}) \wedge X_{34} \end{cases}$$

SEB simplifié :

$$\begin{cases} X_{00} =_\nu Y \wedge X_{11} \wedge X_{13} \\ Y =_\nu X_{11} \vee X_{13} \\ X_{11} =_\nu (X_{22} \wedge \mathrm{F}) \wedge X_{22} \\ X_{13} =_\nu (\mathrm{F} \wedge X_{34}) \wedge X_{34} \end{cases}$$

SEB acyclique

- Graphe booléen *acyclique* :
  - Un des deux STEs est *acyclique* (séquence, arbre, …)
  - $X_{s1,s2} \rightarrow^{a} X_{s1',s2'} \rightarrow^{b} X_{s1'',s2''} \rightarrow^{c} \ldots$
- Application de l'algorithme A3 (mémoire ↓)
- Utile surtout pour la vérification par *préordre*
- Exemple : relecture de séquences d'exécution
  - STE 1 : modélisation d'un système
  - STE 2 : ensemble de scénarios d'exécution / simulation
  - Vérifier que STE 1 *accepte* STE 2

© 2003 Radu Mateescu

SEB conjonctif

- Graphe booléen *conjonctif* :
  - Préordre STE1 $\leq$ STE2 et STE2 *déterministe* (+ vice-versa)
  - Equivalence et un des deux STEs est *déterministe*
  - $X_{s1,s2} = (\wedge_{s1 \to a\, s1a'} \vee_{s2 \to a\, s2'} X_{s1a',s2'}) \wedge$

$$(\wedge_{s1 \to b\, s1b'} \vee_{s2 \to b\, s2'} X_{s1b',s2'}) \wedge$$

$$\ldots$$

$$(\wedge_{s2 \to l\, s2'} \vee_{s1 \to l\, s1a'} X_{s1l',s2'})$$

$$= (\vee_{s2 \to a\, s2'} X_{s1a',s2'}) \wedge (\vee_{s2 \to b\, s2'} X_{s1b',s2'}) \wedge \ldots \wedge$$

$$(\wedge_{s2 \to l\, s2'} X_{s1l',s2'})$$

$$= \wedge_{s2 \to l\, s2'} X_{s1l',s2'}$$

- Application de l'algorithme A4 (mémoire $\downarrow$)

Equivalences faibles

- Soit 2 STE $M_1 = (S_1, A_\tau, T_1, s_{01})$, $M_2 = (S_2, A_\tau, T_2, s_{02})$
  τ : action interne, $A_\tau = A \cup \{ \tau \}$
- Equivalence $\tau^*.a$:

$$\begin{cases} X_{s1,s2} =_\nu & (\wedge_{s1 \to \tau^* a\, s1'} \vee_{s2 \to \tau^* a\, s2'} X_{s1',s2'}) \\ & \wedge \\ & (\wedge_{s2 \to \tau^* a\, s2'} \vee_{s1 \to \tau^* a\, s1'} X_{s1',s2'}) \end{cases}$$

- Equivalence de sûreté :

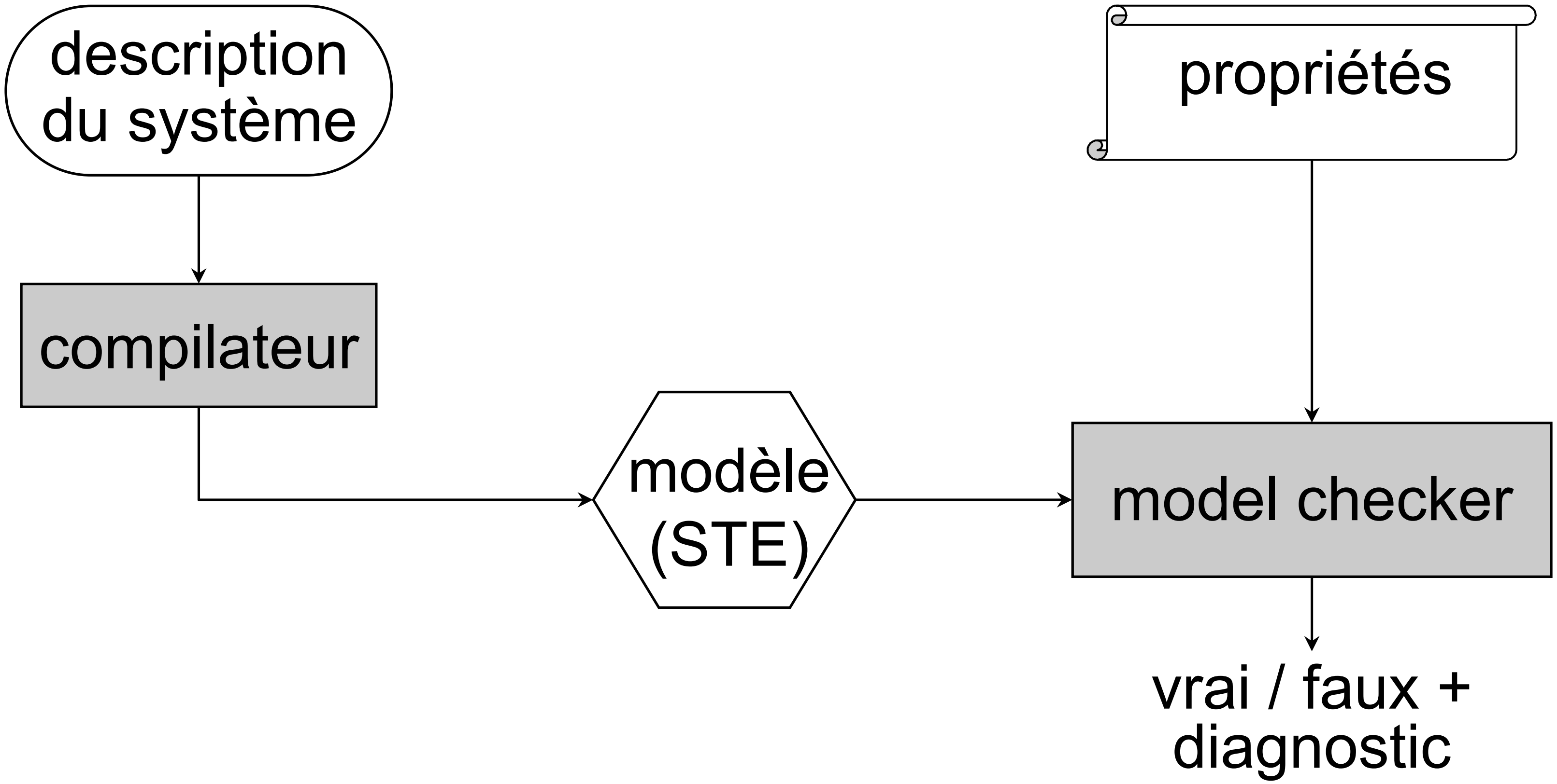
$$\begin{cases} X_{s1,s2} =_\nu Y_{s1,s2} \wedge Y_{s2,s1} \\ Y_{s1,s2} =_\nu (\wedge_{s1 \to \tau^* a\, s1'} \vee_{s2 \to \tau^* a\, s2'} Y_{s1',s2'}) \end{cases}$$

- Schéma similaire :
  - équivalence observationnelle, de branchement, delay, …

Récapitulatif

- Graphe booléen *général* :
  - Toutes les équivalences et leurs préordres
  - Algorithmes A1 et A2
- Graphe booléen *acyclique* :
  - Forte : un des deux STEs acyclique
  - $\tau^*.a$ et sûreté : un STE acyclique (circuits de τ autorisés)
  - Branching et observationnelle : les deux STEs acycliques
  - Algorithme A3 (mémoire ↓)
- Graphe booléen *conjonctif* :
  - Toutes les équivalences : un des deux STE déterministe
  - Algorithme A4 (mémoire ↓)

Vérification par logique temporelle (model checking)



Mu-calcul modal

- Soit $M = (S, A, T, s_0)$ un STE.
- Syntaxe du μ-calcul modal :

Formules sur actions

$$\alpha ::= a \mid \neg\alpha \mid \alpha_1 \vee \alpha_2$$

Formules sur états

$$\varphi ::= \mathrm{F} \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle\, \alpha \,\rangle\, \varphi \mid X \mid \mu X \,.\, \varphi$$

Formules sur actions

Soit $M = (S, A, T, s_0)$. Sémantique $[\![\alpha]\!] \subseteq A$:

- $[\![a]\!] = \{ a \}$
- $[\![\neg\alpha]\!] = A \setminus [\![\alpha]\!]$
- $[\![\alpha_1 \vee \alpha_2]\!] = [\![\alpha_1]\!] \cup [\![\alpha_2]\!]$

Opérateurs dérivés :

- $\mathrm{T} = a \vee \neg a$
- $\mathrm{F} = \neg\mathrm{T}$
- $\alpha_1 \wedge \alpha_2 = \neg(\neg\alpha_1 \vee \neg\alpha_2)$
- $\alpha_1 \Rightarrow \alpha_2 = \neg\alpha_1 \vee \alpha_2$
- $\alpha_1 \Leftrightarrow \alpha_2 = (\alpha_1 \Rightarrow \alpha_2) \wedge (\alpha_2 \Rightarrow \alpha_1)$

© 2003 Radu Mateescu

Formules sur états

Contexte $\rho : Y \rightarrow 2^S$. Sémantique $[[\ \varphi\ ]]\rho \subseteq S$:

- $[[\ F\ ]]\rho = \emptyset$
- $[[\ \neg\varphi\ ]]\rho = S \setminus [[\ \varphi\ ]]\rho$
- $[[\ \varphi_1 \vee \varphi_2\ ]]\rho = [[\ \varphi_1\ ]]\rho \cup [[\ \varphi_2\ ]]\rho$
- $[[\ \langle\ \alpha\ \rangle\ \varphi\ ]]\rho = \{\ s \in S \mid \exists (s, a, s') \in T. \ a \in [[\ \alpha\ ]] \wedge s' \in [[\ \varphi\ ]]\rho\ \}$
- $[[\ Y\ ]]\rho = \rho\ (Y)$
- $[[\ \mu Y\ .\ \varphi\ ]]\rho = \cup_{k \geq 0}\ \Phi_\rho^{\ k}\ (\emptyset)$

  $\Phi_\rho : 2^S \rightarrow 2^S\ ,\ \Phi_\rho\ (U) = [[\ \varphi\ ]]\rho[U / Y]$

Opérateurs dérivés :

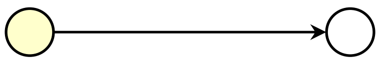
- $[\ \alpha\ ]\ \varphi = \neg\langle\ \alpha\ \rangle\ \neg\varphi$
- $\nu Y\ .\ \varphi = \neg\mu Y\ .\ \neg\varphi\ [\neg Y\ /\ Y]$

© 2003 Radu Mateescu

Exemples

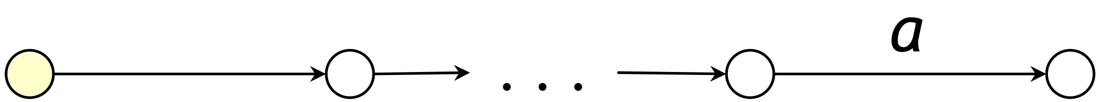
- Absence de blocage sur l'état courant

  $\langle T \rangle T$



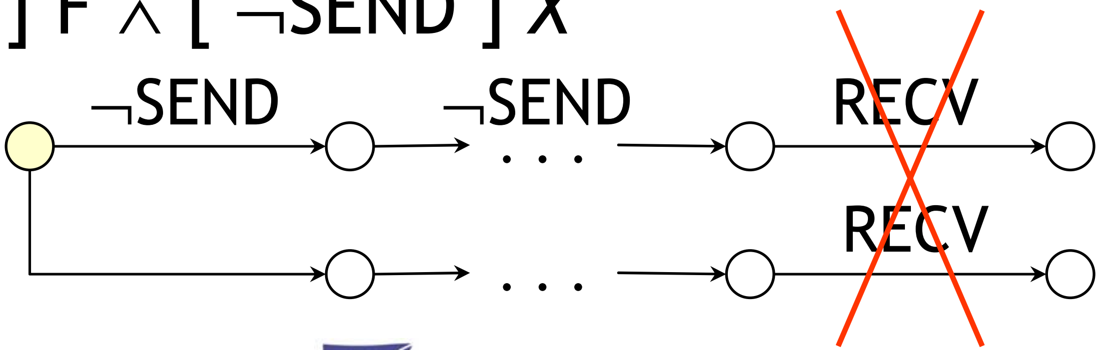
- Accessibilité potentielle d'une action *a*

  $\mu X \, . \, \langle a \rangle T \vee \langle T \rangle X$



- Pas de RECV avant un SEND

  $\nu X \, . \, [\, \mathrm{RECV} \,] \, F \wedge [\, \neg \mathrm{SEND} \,] \, X$



© 2003 Radu Mateescu

Mu-calcul d'alternance 1

- Absence de récursion mutuelle entre les formules de plus petit et de plus grand point fixe
- Exemple :

  "après chaque SEND, il y aura potentiellement un RECV"

  $$\nu X \,.\, [\,\mathrm{SEND}\,]\, (\mu Y \,.\, \langle\,\mathrm{RECV}\,\rangle\, \mathrm{T} \vee \langle\,\mathrm{T}\,\rangle\, Y\,) \wedge [\,\mathrm{T}\,]\, X$$

- Variante équationnelle :

  $$\{\, X =_{\nu} [\,\mathrm{SEND}\,]\, Y \wedge [\,\mathrm{T}\,]\, X \,\}$$

  $$\{\, Y =_{\mu} \langle\,\mathrm{RECV}\,\rangle\, \mathrm{T} \vee \langle\,\mathrm{T}\,\rangle\, Y \,\}$$

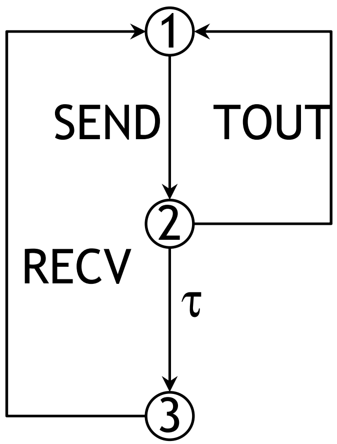
  pas de dépendances cycliques entre les blocs

Traduction vers SEB d'alternance 1

- Principe : $s \models X$ ssi X_s est vraie
- Propriété : $\{ X =_\nu [\ \mathrm{SEND}\ ]\ Y \wedge [\ \mathrm{T}\ ]\ X \}$

  $\{ Y =_\mu \langle\ \mathrm{RECV}\ \rangle\ \mathrm{T} \vee \langle\ \mathrm{T}\ \rangle\ Y \}$
- SEB : $\{ X_s =_\nu (\wedge_{s \xrightarrow{\mathrm{SEND}} s'} Y_{s'}) \wedge (\wedge_{s \rightarrow s'} X_{s'}) \}$

  $\{ Y_s =_\mu (\vee_{s \xrightarrow{\mathrm{RECV}} s'} \mathrm{T}) \vee (\vee_{s \rightarrow s'} Y_{s'}) \}$



$$\begin{cases} X_1 =_\nu Y_2 \wedge X_2 \\ X_2 =_\nu X_1 \wedge X_3 \\ X_3 =_\nu X_1 \end{cases} \qquad \begin{cases} Y_1 =_\mu Y_2 \\ Y_2 =_\mu Y_1 \vee Y_3 \\ Y_3 =_\mu \mathrm{T} \end{cases}$$

SEB acyclique

- Graphe booléen *acyclique* :
  - STE *acyclique* et formules *gardées*
- Traductions des opérateurs de CTL (et ACTL) :
  - $E\,[\varphi_1\ U\ \varphi_2] = \mu X\,.\ \varphi_2 \vee (\varphi_1 \wedge \langle\ T\ \rangle\ X)$
  - $A\,[\varphi_1\ U\ \varphi_2] = \mu X\,.\ \varphi_2 \vee (\varphi_1 \wedge \langle\ T\ \rangle\ T \wedge [\ T\ ]\ X)$
- Réduction pour le μ-calcul complet [Mateescu-02]
  - Elimination des opérateurs de plus grand point fixe ➔ formule d'alternance 1
  - Traduction en forme gardée (taille quadratique)
- Application de l'algorithme A3 (mémoire ↓)

SEB disjonctif

- Graphe booléen *disjonctif* :
  - Opérateur de *potentialité* de CTL

    $E\ [\varphi_1\ U\ \varphi_2] = \mu X\ .\ \varphi_2 \vee (\varphi_1 \wedge \langle\ T\ \rangle\ X)$

    $\{\ X =_\mu \varphi_2 \vee Y\quad ,\ Y =_\mu \varphi_1 \wedge Z\quad ,\ Z =_\mu \langle\ T\ \rangle\ X\ \}$

    $\{\ X_s =_\mu \varphi_{2s} \vee Y_s\quad ,\ Y_s =_\mu \varphi_{1s} \wedge Z_s\quad ,\ Z_s =_\mu \vee_{s \to s'} X_{s'}\ \}$
  - Modalité de *possibilité* de PDL

    $\langle\ (a\ |\ b)^*\ .\ c\ \rangle\ T$

    $\{\ X =_\mu \langle\ c\ \rangle\ T \vee \langle\ a\ \rangle\ X \vee \langle\ b\ \rangle\ X\ \}$

    $\{\ X_s =_\mu (\vee_{s \to c\ s'}\ T) \vee (\vee_{s \to a\ s'}\ X_{s'}) \vee (\vee_{s \to b\ s'}\ X_{s'})\ \}$
- Application de l'algorithme A4 (mémoire ↓)

SEB conjonctif

- Graphe booléen *conjonctif* :
  - Opérateur d'*inévitabilité* de CTL

    $A\ [\varphi_1\ U\ \varphi_2] = \mu X\ .\ \varphi_2 \vee (\varphi_1 \wedge \langle\ T\ \rangle\ T \wedge [\ T\ ]\ X)$

    $\{\ X =_\mu \varphi_2 \vee Y\ \ ,\ Y =_\mu \varphi_1 \wedge Z \wedge [\ T\ ]\ X\ ,\ Z =_\mu \langle\ T\ \rangle\ T\ \}$

    $\{\ X_s =_\mu \varphi_{2s} \vee Y_s\ \ ,\ Y_s =_\mu \varphi_{1s} \wedge Z_s \wedge (\wedge_{s \to s'} X_{s'})\ \ ,\ \ Z_s =_\mu \vee_{s \to s'} T\ \}$
  - Modalité de *nécessité* de PDL

    $[\ (a\ |\ b)^*\ .\ c\ ]\ F$

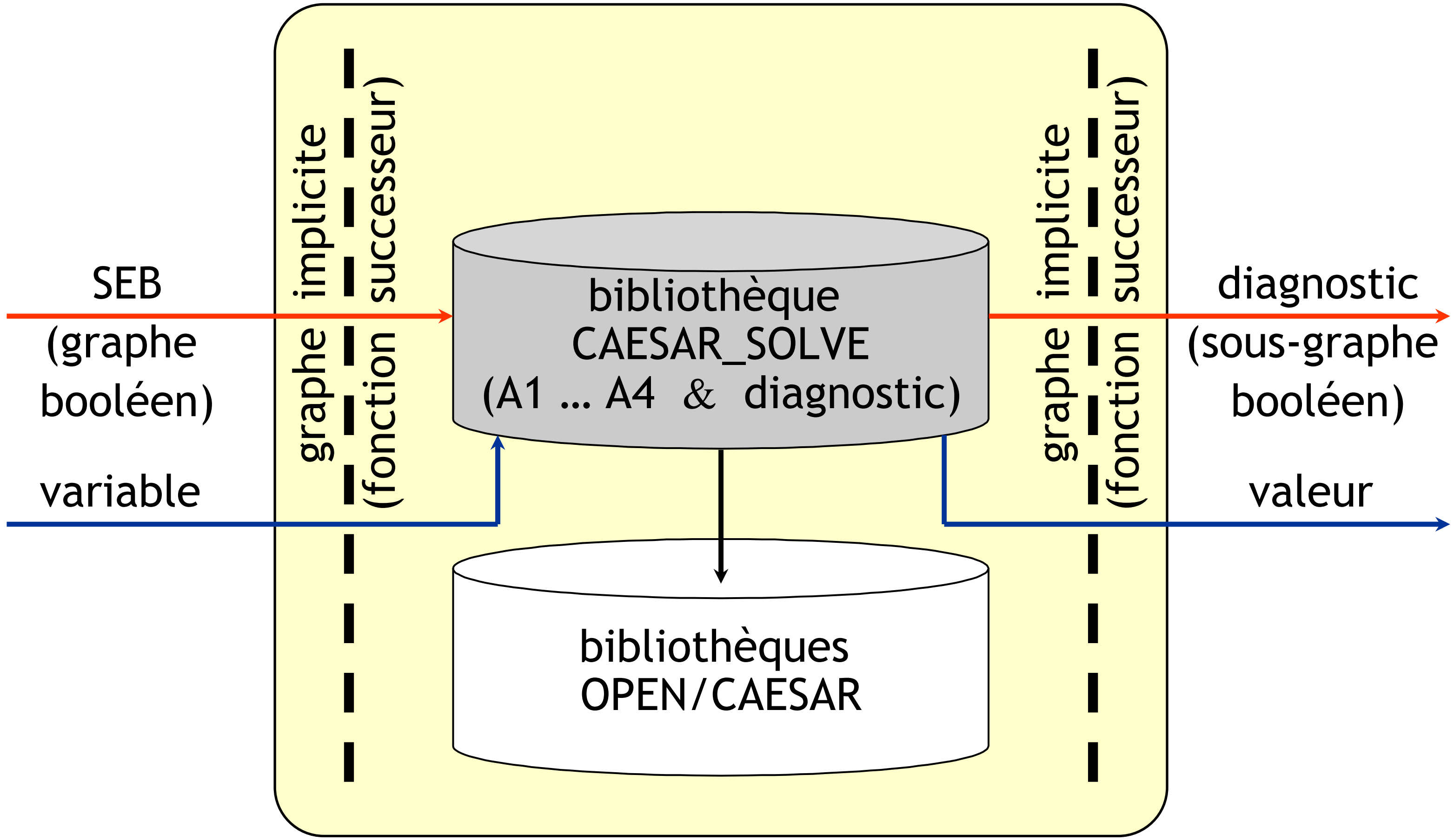
    $\{\ X =_\mu [\ c\ ]\ F \wedge [\ a\ ]\ X \wedge [\ b\ ]\ X\ \}$

    $\{\ X_s =_\mu (\wedge_{s \to c\ s'} F) \wedge (\wedge_{s \to a\ s'} X_{s'}) \wedge (\wedge_{s \to b\ s'} X_{s'})\ \}$
- Application de l'algorithme A4 (mémoire ↓)

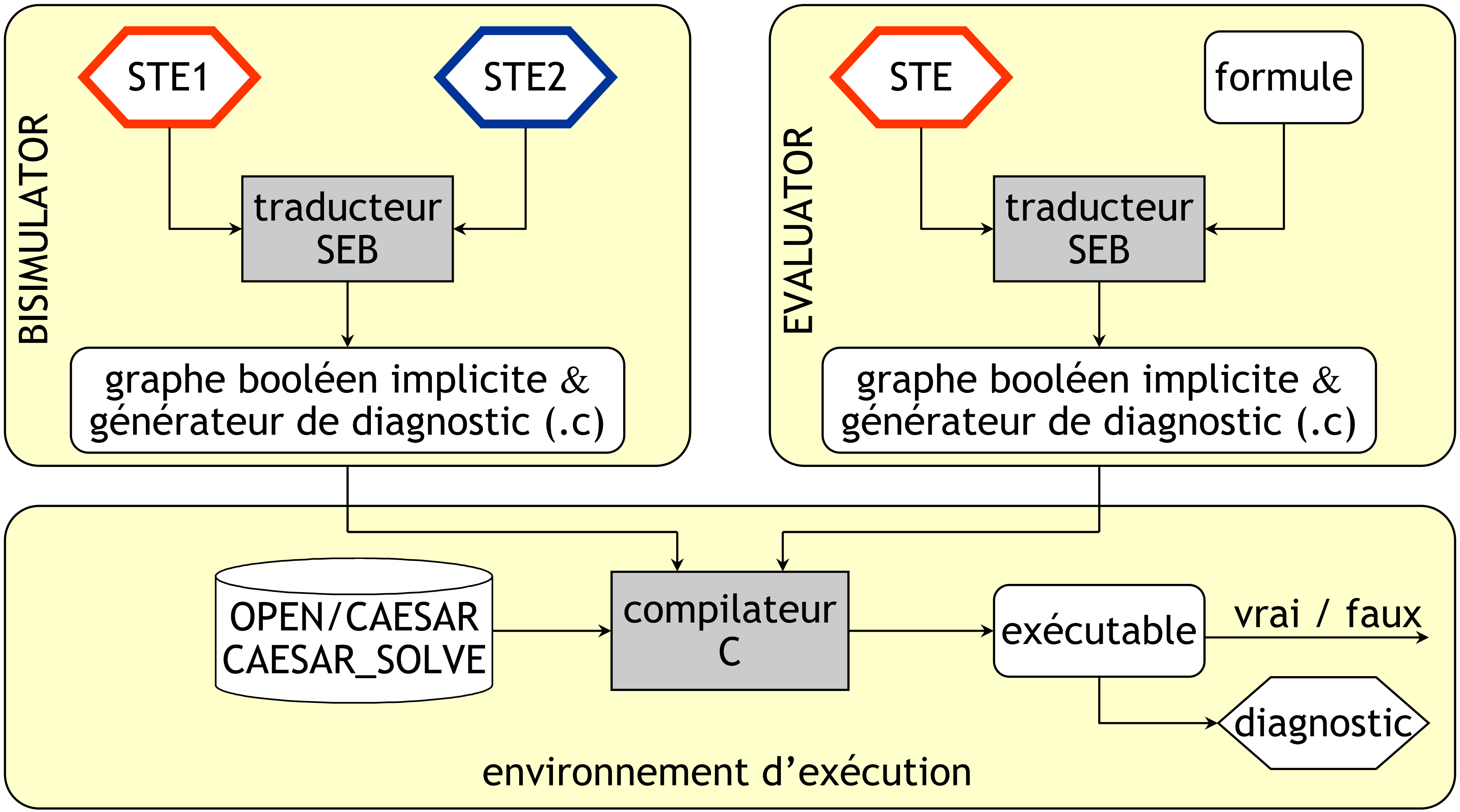
Récapitulatif

- Graphe booléen *général* :
  - STE quelconque et formule de μ-calcul d'alternance 1
  - Algorithmes A1 et A2
- Graphe booléen *acyclique* :
  - STE acyclique et formule gardée (CTL, ACTL)
  - STE acyclique et formule de μ-calcul (+ réduction)
  - Algorithme A3 (mémoire $\downarrow$)
- Graphe booléen *disjonctif* / *conjonctif* :
  - STE quelconque et formules de CTL, ACTL, PDL
  - Algorithme A4 (mémoire $\downarrow$)

Bibliothèque CAESAR_SOLVE



BISIMULATOR et EVALUATOR



Quelques mesures de performances

- Trois protocoles de communication (ABP, BRP, LEP)
- Algorithme A2 versus A1 :
  - Comparaison STE protocole – STE erroné (strong)
  - Vérification propriété fausse sur STE protocole
  - Réductions de 75 % - 99 % en profondeur du diagnostic
- Algorithme A3 versus A1 :
  - Relecture de séquences (100000 transitions) dans le STE
  - Vérification de propriétés sur les séquences
  - Gains de 15 % - 27 % en mémoire
- Algorithme A4 versus A1 :
  - Comparaison STE protocole – STE service ($\tau^*.a$)
  - Vérification de propriétés (ACTL + PDL)
  - Gains de 12 % - 63 % en mémoire

Conclusion et travaux futurs

Bilan :

- Algorithmes de résolution locale des SEBs
- Génération de diagnostics
- Deux applications : BISIMULATOR et EVALUATOR
- Bibliothèque générique CAESAR_SOLVE

Perspectives :

- Nouveaux algorithmes (« single-scan » sur traces)
- Nouvelles applications (génération de tests)
- Parallélisation sur grappes de PC

© 2003 Radu Mateescu