Compilation et vérification de programmes LOTOS

Hubert Garavel

I.N.R.I.A. et L.G.I.-I.M.A.G.

Plan

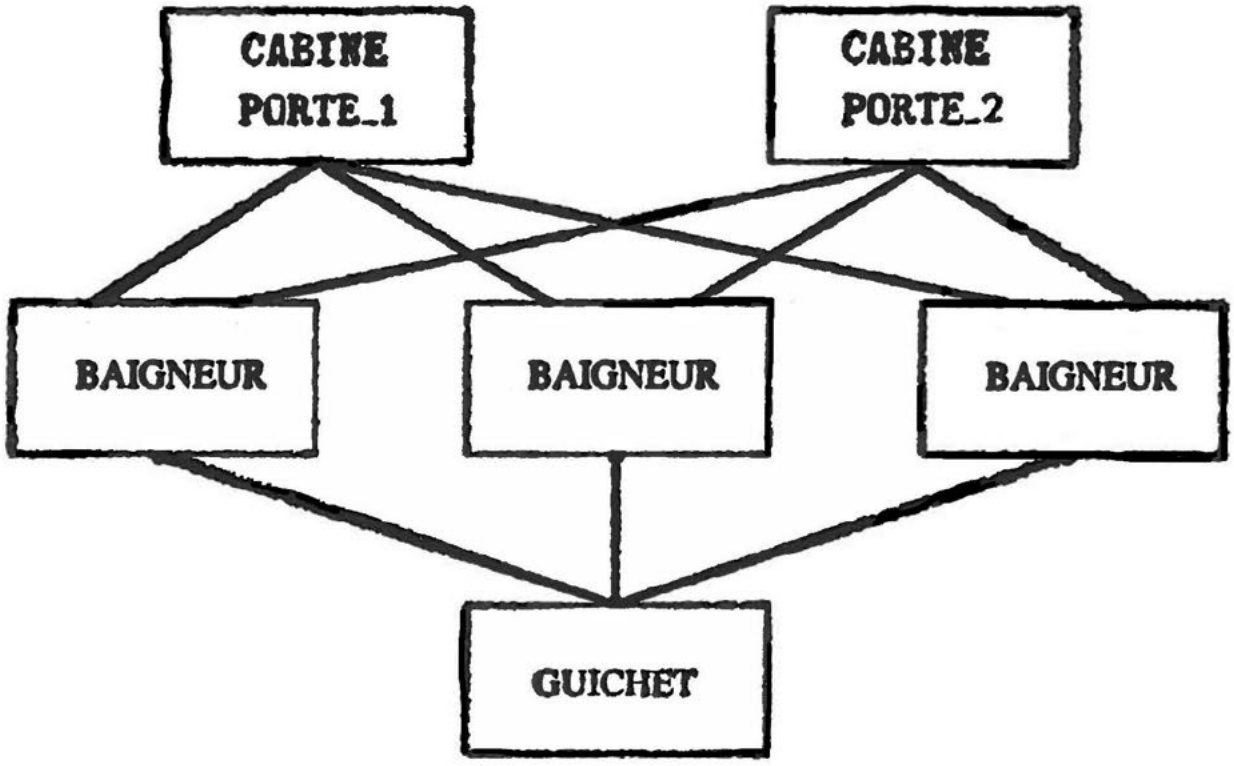
Le langage LOTOS

Language **O**f **T**emporal **O**rdering **S**pecification

- langage de description de **systèmes distribués asynchrones**
- **normalisé** par l'ISO et le CCITT
- **protocoles et services de télécommunications** OSI

Deux parties orthogonales :

- **partie contrôle :**
  - processus parallèles communicants
  - rendez-vous symétrique n-aire
  - algèbre de processus
    - CCS [Milner]
    - TCSP [Brookes-Hoare-Roscoe]
- **partie données :**
  - types abstraits algébriques
  - inspiré du langage ACT-ONE [Ehrig-Mahr-Fey-Hansen]




```
process CABINE [PORTE] ... endproc

process BAIGNEUR [PORTE_1, PORTE_2, GUICHET] ... endproc

process VESTIAIRE [GUICHET] (...) ... endproc

(CABINE [P1] ||| CABINE [P2])
|[P1, P2]|
(BAIGNEUR [P1, P2, G] ||| ... ||| BAIGNEUR [P1, P2, G])
|[G]|
VESTIAIRE [G]
```

```
process CABINE [PORTE] ...
   PORTE !ENTRER;
      PORTE !SORTIR;
         CABINE [PORTE]
endproc


process BAIGNEUR [PORTE_1, PORTE_2, GUICHET] ...
   choice P in [PORTE_1, PORTE_2] []
      P !ENTRER;
         P !SORTIR;
            GUICHET !DEPOSER ?N:NUMERO_PANIER;
               i;
                  GUICHET !REPRENDRE !N;
                     choice P in [PORTE_1, PORTE_2] []
                        P !ENTRER;
                           P !SORTIR;
                              stop
endproc

process VESTIAIRE [GUICHET] (PANIERS_VIDES:ENSEMBLE) ...
   choice N:NUMERO_PANIER []
      [N ∈ PANIERS_VIDES] ->
         GUICHET !DEPOSER !N;
            VESTIAIRE [GUICHET] (PANIERS_VIDES - {N})
   []
   GUICHET !REPRENDRE ?N:NUMERO_PANIER
      VESTIAIRE [GUICHET] (PANIERS_VIDES ∪ {N})
endproc
```

inaction

stop

préfixage

$G\ O_1, \ldots\ O_n\ [[V_0]]\ ;\ B_0$

choix non-déterministe

$B_1\ []\ B_2$

composition parallèle

$B_1\ |[\ G_1, \ldots\ G_n\ ]|\ B_2$

abstraction

hide $G_0, \ldots\ G_n$ **in** B_0

garde

$[V_0]$ -> B_0

définition de variables

let $\widehat{X_0}:S_0{=}V_0, \ldots\ \widehat{X_n}:S_n{=}V_n$ **in** B_0

choix généralisé sur les variables

choice $\widehat{X_0}:S_0, \ldots\ \widehat{X_n}:S_n\ []\ B_0$

choix généralisé sur les portes

choice $\widehat{G_0}$ **in** $[\widehat{G'_0}], \ldots\ \widehat{G_n}$ **in** $[\widehat{G'_n}]\ []\ B_0$

composition parallèle généralisée

par $\widehat{G_0}$ **in** $[\widehat{G'_0}], \ldots\ \widehat{G_n}$ **in** $[\widehat{G'_n}]\ |[\ G_1, \ldots\ G_n\ ]|\ B_0$

terminaison

exit $(R_1, \ldots\ R_n)$

composition séquentielle

B_1 >> **accept** $\widehat{X_1}:S_1, \ldots\ \widehat{X_n}:S_n$ **in** B_2

interruption

$B_1\ [>\ B_2$

appel de processus

$P\ [G_1, \ldots\ G_n]\ (V_1, \ldots\ V_m)$

avec comme non-terminaux :

B : comportement (*behaviour*) G : porte (*gate*)

X : variable V : valeur S : sorte

- relation de dérivation

$$B \overset{G\ !V_1 \ldots !V_n}{\longrightarrow} B'$$

- définie par induction structurelle sur les comportements

$$\frac{true}{(G;\ B_0) \xrightarrow{G} B_0}$$

$$\frac{B_1 \xrightarrow{L} B_1'}{(B_1\ []\ B_2) \xrightarrow{L} B_1'} \qquad \frac{B_2 \xrightarrow{L} B_2'}{(B_1\ []\ B_2) \xrightarrow{L} B_2'}$$

- **graphe** : système de transitions correspondant
  - **état** = comportement LOTOS
  - **arc** = dérivation

```
CABINE [PORTE] =
 PORTE !ENTREE;
  PORTE !SORTIE;
   CABINE [PORTE]
        ↓ PORTE !ENTREE
 PORTE !SORTIE;
  CABINE [PORTE]
        ↓ PORTE !SORTIE
 CABINE [PORTE]
        ↓ PORTE !ENTREE
        ...
```

Problème

- vérifier qu'un programme LOTOS satisfait certaines propriétés
  - démonstrateurs
  - générateurs de modèle :
    * convertir le programme LOTOS en graphe
    * valider les propriétés sur ce graphe

Utilisation d'équivalences

- critères d'abstraction sur le graphe à vérifier
- réduction modulo équivalences
- résultat : $\text{graphe}_1 \sim \text{graphe}_2$
- outils : ALDEBARAN [LGI-IMAG], AUTO [INRIA], ...

Utilisation de logiques temporelles

- propriétés décrites par des modalités
- résultat : graphe $\models$ formule
- outils : CESAR [LGI-IMAG], ...

Les outils de vérification CESAR

Le principe CESAR

- systèmes décrits en **langage de haut niveau**
- validation par **génération de modèle**
- propriétés exprimées en **logique temporelle**

outil	langage à vérifier	logique employée
QUASAR	CSP	CTL
XESAR	ESTELLE/R	LTAC
CÆSAR	LOTOS	?

Une logique temporelle pour LOTOS : R$\overset{\star}{\text{I}}$CO

- formule : ensemble d'arcs du graphe
- logique arborescente
- expressivité des logiques linéaires (séquences)
- aspects temporels décrits par des expressions régulières
- quantificateurs du 1er ordre sur les valeurs et sur les arcs

Implémentation "dynamique" de LOTOS [9]

Outils existants

- interprète : HIPPO [Twente], UOLOTOS [Ottawa], ...
- compilateur : LOTOS→C [Madrid]
- démonstrateur : Boyer-Moore [British Telecom]
- générateur de modèles : Squiggles [Pise]

Principe commun

- construction (exhaustive ou partielle) du graphe
- par application directe des règles sémantiques

Inefficacité

- coût en mémoire :
  - chaque état du graphe est un terme algébrique
- coût en temps :
  - détermination des règles à appliquer
  - comparaison des états pour détecter les circuits

Conséquences

- en théorie, on peut traiter tout LOTOS
- en pratique, seulement une classe très restreinte de programmes
- domaine d'application : interprète en pas à pas (mise au point)
- **crédibilité de LOTOS ?**

Implémentation "statique" de LOTOS

Motivations

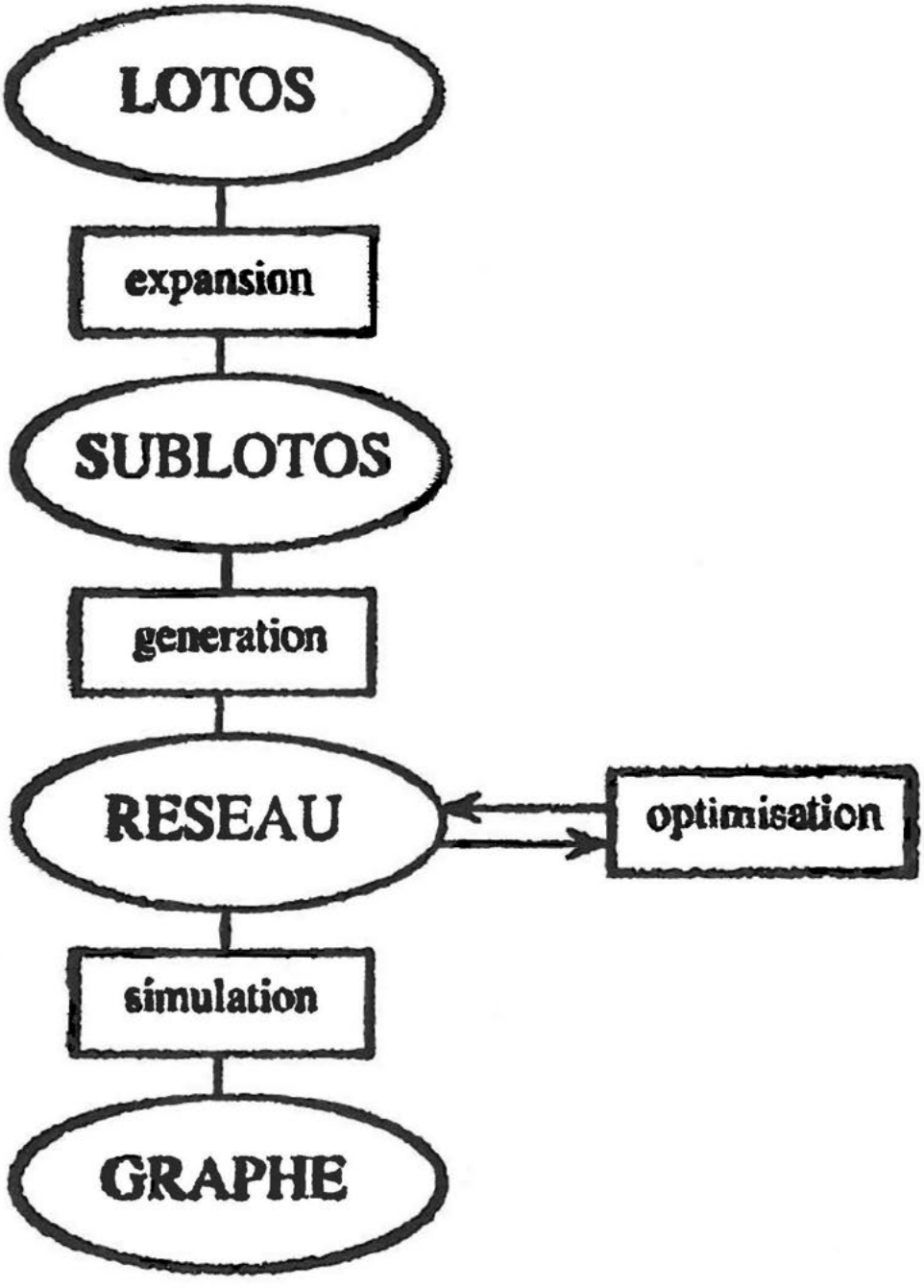
- traiter des programmes LOTOS non triviaux
- système de vérification par génération de modèles
- produire **rapidement** des graphes **complexes**

Principes

- **compiler** et non **interpréter**
- séparer compilation et exécution (**simulation**)
- déplacer le maximum de calculs de l'exécution vers la compilation
  - détermination des rendez-vous possibles
  - évaluation des gardes
- conserver les résultats dans des **formes intermédiaires**

Conséquences

- ne pas traduire directement LOTOS → graphe, mais par étapes
- choisir judicieusement des formes intermédiaires :
  - capables de modéliser contrôle et données
  - économes en mémoire
  - efficaces à manipuler
  - simples et générales


LOTOS
expansion
SUBLOTOS
generation
RESEAU
optimisation
simulation
GRAPHE

Le langage SUBLOTOS

Problème

Traduire LOTOS, langage **dynamique** vers un modèle **statique**.

LOTOS	SUBLOTOS
architecture dynamique	**architecture statique**
création/destruction de processus pendant l'exécution	processus constants et fixés à la compilation
création/destruction de portes pendant l'exécution	portes constatntes et fixées à la compilation
langage fonctionnel	**langage impératif**
"constantes dynamiques" variables locales gérées en pile	"variables d'état" variables globales
passage de paramètre	affectation

Restrictions

- cette traduction n'est pas possible pour la totalité de LOTOS
- hypothèse du **contrôle statique**
  - interdire les appels récursifs de processus à travers "|||"

```
process P [...] ...
   ... ||| P [...]
endproc
```

  - interdire les appels récursifs à gauche de "[>" et ">>"
  - (tous les autres appels récursifs sont permis)
- contrôle statique et domaines finis $\Longrightarrow$ graphe fini

SUBLOTOS

- algèbre de processus, contenue dans LOTOS
- syntaxe et sémantique opérationnelle définies formellement
- sémantique analogue à LOTOS pour le contrôle
- mais plus "impérative" pour les données

$$\frac{eval(V_0\ ,\ C') = true}{\langle G\ O_1, \ldots\ O_n[V_0]\ ;\ B_0\ ,\ C\rangle \xrightarrow{G\ V'_1, \ldots\ V'_n} \langle B_0\ ,\ C'\rangle}$$

où :

$$(\forall i \in \{1, \ldots\ n\})\ V'_i = \begin{cases} \textbf{si } O_i = !V_i \textbf{ alors } eval(V_i\ ,\ C) \\ \textbf{si } O_i = ?X_i{:}S_i \textbf{ alors } oneof(domain(S_i)) \end{cases}$$

$$C' = C \oslash \bigoplus_{i \in \{1, \ldots\ n\}} C'_i$$

$$(\forall i \in \{1, \ldots\ n\})\ C'_i = \begin{cases} \textbf{si } O_i = !V_i \textbf{ alors } \bot \\ \textbf{si } O_i = ?X_i{:}S_i \textbf{ alors } X_i \rightsquigarrow V'_i \end{cases}$$

$$\frac{\langle B_1\ ,\ C\rangle \xrightarrow{L} \langle B'_1\ ,\ C'\rangle}{\langle B_1 \ \square\ B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_1\ ,\ C'\rangle}$$

$$\frac{\langle B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_2\ ,\ C'\rangle}{\langle B_1 \ \square\ B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_2\ ,\ C'\rangle}$$

$$\frac{\langle B_1\ ,\ C\rangle \xrightarrow{\mathbf{i}\bullet 1} \langle B'_1\ ,\ C'\rangle}{\langle B_1 \ ||\ B_2\ ,\ C\rangle \xrightarrow{\mathbf{i}\bullet 1} \langle B'_1 \ ||\ B_2\ ,\ C'\rangle}$$

$$\frac{\langle B_2\ ,\ C\rangle \xrightarrow{\mathbf{i}\bullet 1} \langle B'_2\ ,\ C'\rangle}{\langle B_1 \ ||\ B_2\ ,\ C\rangle \xrightarrow{\mathbf{i}\bullet 1} \langle B_1 \ ||\ B'_2\ ,\ C'\rangle}$$

$$\frac{(\langle B_1\ ,\ C\rangle \xrightarrow{L} \langle B'_1\ ,\ C'_1\rangle) \wedge (\langle B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_2\ ,\ C'_2\rangle) \wedge (gate(L) \neq \mathbf{i}\bullet 1)}{\langle B_1 \ ||\ B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_1 \ ||\ B'_2\ ,\ merge\ (C, C'_1, C'_2)\rangle}$$

$$\frac{(\langle B_1\ ,\ C\rangle \xrightarrow{L} \langle B'_1\ ,\ C'\rangle) \wedge (gate(L) \notin \{G_0, \ldots\ G_n\})}{\langle B_1 \ |[G_0, \ldots\ G_n]|\ B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_1 \ |[G_0, \ldots\ G_n]|\ B_2\ ,\ C'\rangle}$$

$$\frac{(\langle B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_2\ ,\ C'\rangle) \wedge (gate(L) \notin \{G_0, \ldots\ G_n\})}{\langle B_1 \ |[G_0, \ldots\ G_n]|\ B_2\ ,\ C\rangle \xrightarrow{L} \langle B_1 \ |[G_0, \ldots\ G_n]|\ B'_2\ ,\ C'\rangle}$$

$$\frac{(\langle B_1\ ,\ C\rangle \xrightarrow{L} \langle B'_1\ ,\ C'_1\rangle) \wedge (\langle B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_2\ ,\ C'_2\rangle) \wedge (gate(L) \in \{G_0, \ldots\ G_n\})}{\langle B_1 \ |[G_0, \ldots\ G_n]|\ B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_1 \ |[G_0, \ldots\ G_n]|\ B'_2\ ,\ merge\ (C, C'_1, C'_2)\rangle}$$

$$\frac{(\langle B_0\ ,\ C\rangle \xrightarrow{L} \langle B'_0\ ,\ C'\rangle) \wedge (gate(L) \in \{G_0, \ldots\ G_n\})}{\langle \textbf{hide } G_0, \ldots\ G_n \textbf{ in } B_0\ ,\ C\rangle \xrightarrow{\mathbf{i}\bullet 1} \langle \textbf{hide } G_0, \ldots\ G_n \textbf{ in } B'_0\ ,\ C'\rangle}$$

$$\frac{(\langle B_0\ ,\ C\rangle \xrightarrow{L} \langle B'_0\ ,\ C'\rangle) \wedge (gate(L) \notin \{G_0, \ldots\ G_n\})}{\langle \textbf{hide } G_0, \ldots\ G_n \textbf{ in } B_0\ ,\ C\rangle \xrightarrow{L} \langle \textbf{hide } G_0, \ldots\ G_n \textbf{ in } B'_0\ ,\ C'\rangle}$$

$$\frac{(\langle B_0\ ,\ C\rangle \xrightarrow{L} \langle B'_0\ ,\ C'\rangle) \wedge (eval(V_0\ ,\ C) = true)}{(\langle [V_0]\ \text{->}\ B_0\ ,\ C\rangle) \xrightarrow{L} \langle B'_0\ ,\ C'\rangle}$$

$$\frac{\underbrace{\langle \textbf{let } \widehat{X_0}{:}S_0{=}V_0, \ldots\ \widehat{X_n}{:}S_n{=}V_n \textbf{ in } B_0\ ,\ C\rangle}}{\langle B_0\ ,\ C \oslash \bigoplus_{i \in \{0, \ldots\ n\}} (\widehat{X_i} \rightsquigarrow eval(V_i\ ,\ C))\rangle}$$

$$\frac{\underbrace{\langle \textbf{choice } \widehat{X_0}{:}S_0, \ldots\ \widehat{X_n}{:}S_n \ \square\ B_0\ ,\ C\rangle}}{\langle B_0\ ,\ C \oslash \bigoplus_{i \in \{0, \ldots\ n\}} C_i\rangle}$$

où :

$$\textbf{si } \widehat{X_i} \equiv X_0, \ldots\ X_p \textbf{ alors } C_i = \bigoplus_{j \in \{0, \ldots\ p\}} (X_j \rightsquigarrow oneof(domain(S_i)))$$

$$\frac{\langle B_1\ ,\ C\rangle \xrightarrow{L} \langle B'_1\ ,\ C'\rangle \wedge (gate(L) \neq G_0)}{\langle B_1\ [G_0\text{>}\ B_2\ ,\ C\rangle \xrightarrow{L} \langle B'_1\ [G_0\text{>}\ B_2\ ,\ C'\rangle}$$

$$\frac{\underbrace{\langle P\ [G_0, \ldots\ G_m]\ (V_1, \ldots\ V_n)\ ,\ C\rangle}}{\langle behaviour(P)\ ,\ C \oslash \bigoplus_{i \in \{1, \ldots\ n\}} (var_i(P) \rightsquigarrow eval(V_i\ ,\ C))\rangle}$$

Expansion

Objectifs

- traduire LOTOS en SUBLOTOS
- obtenir des processus/portes statiques et des variables globales

Simplification des opérateurs

- élimination de "**choice**", "**par**", "`exit`", "`>>`" et "`|||`"

Développement des processus

- processus non récursifs
- processus récursifs
  - développement **limité** pour éviter la récursion à l'infini
  - critère d'arrêt : *portes effectives = portes formelles*

```
process CABINE [ENTREE, SORTIE] ...
   ENTREE;
      CABINE [SORTIE, ENTREE]
endproc
```

devient :

```
process CABINE [ENTREE, SORTIE] ...
   ENTREE;
      SORTIE;
         CABINE [ENTREE, SORTIE]
endproc
```

- preuve de terminaison sous l'hypothèse du contrôle statique
- après terminaison tous les objets sont statiques et globaux

Les opérateurs LOTOS éliminés par l'expansion

inaction

stop

préfixage

$G\ O_1, \ldots\ O_n\ [[V_0]]\ ;\ B_0$

choix non-déterministe

$B_1\ []\ B_2$

composition parallèle

$B_1\ |[\ G_1, \ldots\ G_n\ ]|\ B_2$

abstraction

hide $G_0, \ldots\ G_n$ **in** B_0

garde

$[V_0]\ \text{->}\ B_0$

définition de variables

let $\widehat{X}_0 : S_0 = V_0, \ldots\ \widehat{X}_n : S_n = V_n$ **in** B_0

choix généralisé sur les variables

choice $\widehat{X}_0 : S_0, \ldots\ \widehat{X}_n : S_n\ []\ B_0$

choix généralisé sur les portes

choice $\widehat{G}_0$ **in** $[\widehat{G}'_0], \ldots\ \widehat{G}_n$ **in** $[\widehat{G}'_n]\ []\ B_0$

composition parallèle généralisée

par $\widehat{G}_0$ **in** $[\widehat{G}'_0], \ldots\ \widehat{G}_n$ **in** $[\widehat{G}'_n]\ |[\ G_1, \ldots\ G_n\ ]|\ B_0$

terminaison

exit $(R_1, \ldots\ R_n)$

composition séquentielle

B_1 **>> accept** $\widehat{X}_1 : S_1, \ldots\ \widehat{X}_n : S_n$ **in** B_2

interruption

$B_1\ [>\ B_2$

appel de processus

$P\ [G_1, \ldots\ G_n]\ (V_1, \ldots\ V_m)$

Le modèle réseau

Objectifs

- ne pas traduire directement SUBLOTOS en graphe
- rechercher une seconde forme intermédiaire
  - sémantique impérative au niveau du contrôle
  - modélisation concise du parallélisme
  - deux solutions : { automates communicants / **réseaux de Petri**

Modélisation du contrôle

- ensemble de places
- ensemble de transitions
  - un ensemble de places d'entrée et de sortie
  - une porte et une liste d'offres ("`!V`" ou "`?X:S`") afin d'exprimer les rendez-vous
- contrôle statique $\Longrightarrow$ marquage sauf

Modélisation des données

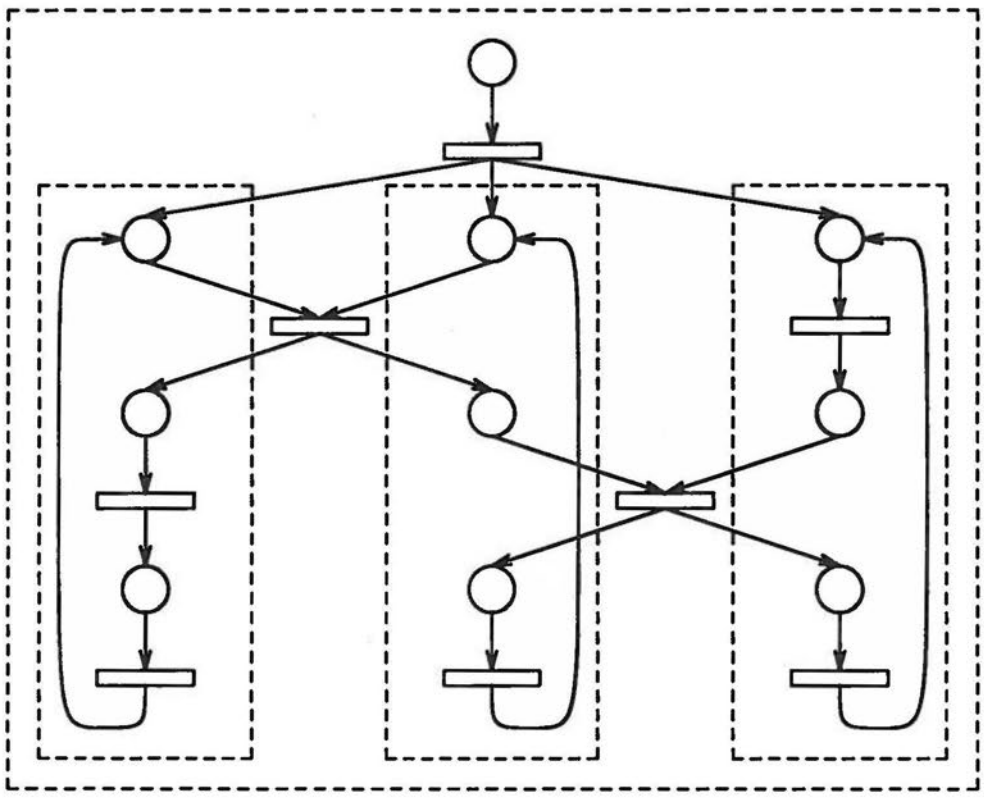
- ensemble de variables d'état
- actions attachées aux transitions (affectations, conditions, itérations, ...)

ε-transitions

- rôle compositionnel dans la construction du réseau
  - transitions **fork** ("| |")
  - actions sur les données ("->", "**let**", "**choice**")
  - transferts de contrôle ("[>", récursion)
- franchissement invisible
- franchissement atomique

Structuration en unités

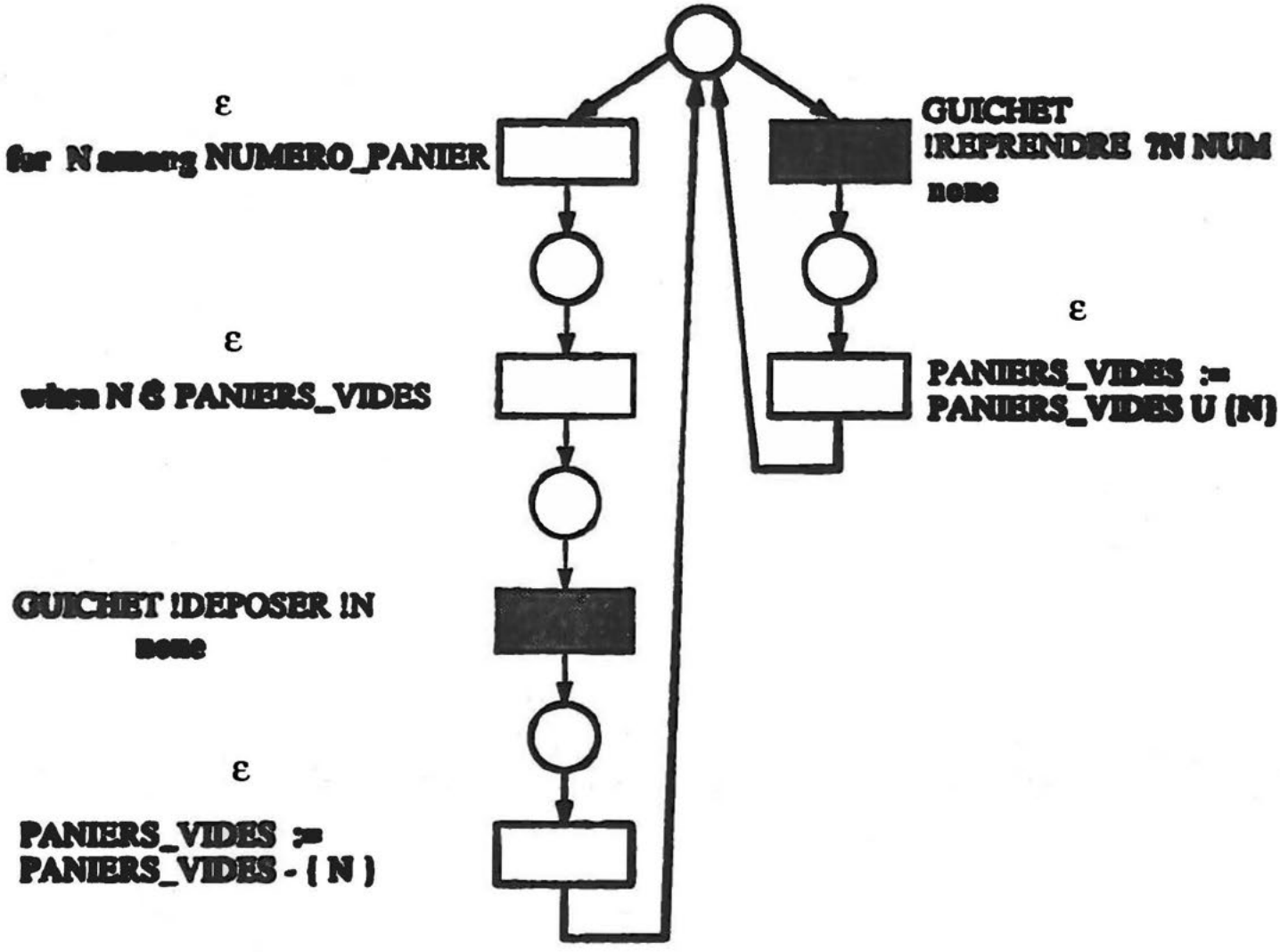
- unité : ensemble de places définissant une tâche séquentielle
- unités hiérarchiquement imbriquées



Sémantique opérationnelle

- définie par une relation de transition
- basée sur les règles d'évolution des réseaux de Petri interprétés
- règles additionnelles pour les ε-transitions

Le modèle réseau



Le modèle réseau

$$\frac{(M_1 \supseteq M_{in}) \wedge (M_2 = (M_1 - M_{in}) \cup M_{out})}{[M_1\ ,\ M_{in}\ ,\ M_{out}] \xrightarrow{m} M_2}$$

$$\frac{C_1 = C_2}{[C_1\ ,\ \mathbf{none}] \xrightarrow{c} C_2}$$

$$\frac{(eval(V\ ,\ C_1) = true) \wedge (C_1 = C_2)}{[C_1\ ,\ \mathbf{when}\ V] \xrightarrow{c} C_2}$$

$$\frac{C_2 = C_1 \oslash \oplus_{i \in \{0,\ldots\ n\}}(\widehat{X_i} \rightsquigarrow eval(V_i\ ,\ C_1))}{[C_1\ ,\ \widehat{X_0}, \ldots\ \widehat{X_n}\ :=\ V_0, \ldots\ V_n] \xrightarrow{c} C_2}$$

$$\frac{C_2 = C_1 \oslash \oplus_{i \in \{0,\ldots\ n\}}(\oplus_{X_j \in \widehat{X_i}}(X_j \rightsquigarrow oneof(S_i)))}{[C_1\ ,\ \mathbf{for}\ \widehat{X_0}, \ldots\ \widehat{X_n}\ \mathbf{among}\ S_0, \ldots\ S_n] \xrightarrow{c} C_2}$$

$$\frac{([C_1\ ,\ A_1] \xrightarrow{c} C) \wedge ([C\ ,\ A_2] \xrightarrow{c} C_2)}{[C_1\ ,\ A_1\ ;\ A_2] \xrightarrow{c} C_2}$$

$$\frac{([C_1\ ,\ A_1\ ;\ A_2] \xrightarrow{c} C_2) \wedge ([C_1\ ,\ A_2\ ;\ A_1] \xrightarrow{c} C_2)}{[C_1\ ,\ A_1\ \&\ A_2] \xrightarrow{c} C_2}$$

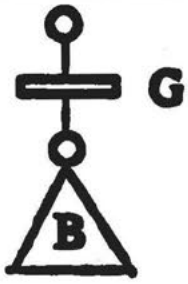
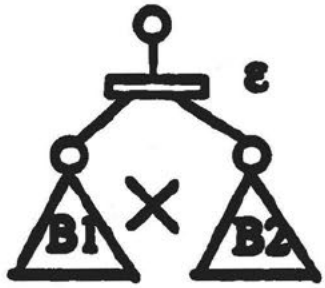
$$\frac{\begin{array}{c}([M_1\ ,\ in(T)\ ,\ out(T)] \xrightarrow{m} M_2) \wedge \\ ([C_1\ ,\ action(T)] \xrightarrow{c} C_2) \wedge \\ (L = eval\_label(gate(T), offer(T), C_2))\end{array}}{[\langle M_1, C_1\rangle\ ,\ T] \xrightarrow{p} [\langle M_2, C_2\rangle\ ,\ L]}$$

$$\frac{\begin{array}{c}(\exists T)\ (\exists \pi)\ (gate(T) \neq \varepsilon) \wedge ([\pi\ ,\ T] \xrightarrow{p} [\pi_2\ ,\ L]) \wedge \\ (\textit{il existe une } \varepsilon\textit{-chaîne allant de } \pi_1 \textit{ à } \pi) \wedge \\ (\textit{pour toute } \varepsilon\textit{-chaîne } \pi'_1, \ldots\ \pi'_n \textit{ allant de } \pi_1 \textit{ à } \pi)\ (\forall i \in \{1, \ldots\ n-1\}) \\ (in(T) \not\subseteq marking(\pi'_i))\ \vee\ (marking(\pi'_i) = marking(\pi))\end{array}}{\pi_1 \xrightarrow{L} \pi_2}$$

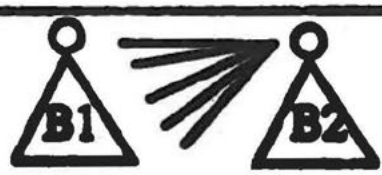
Génération

Objectifs

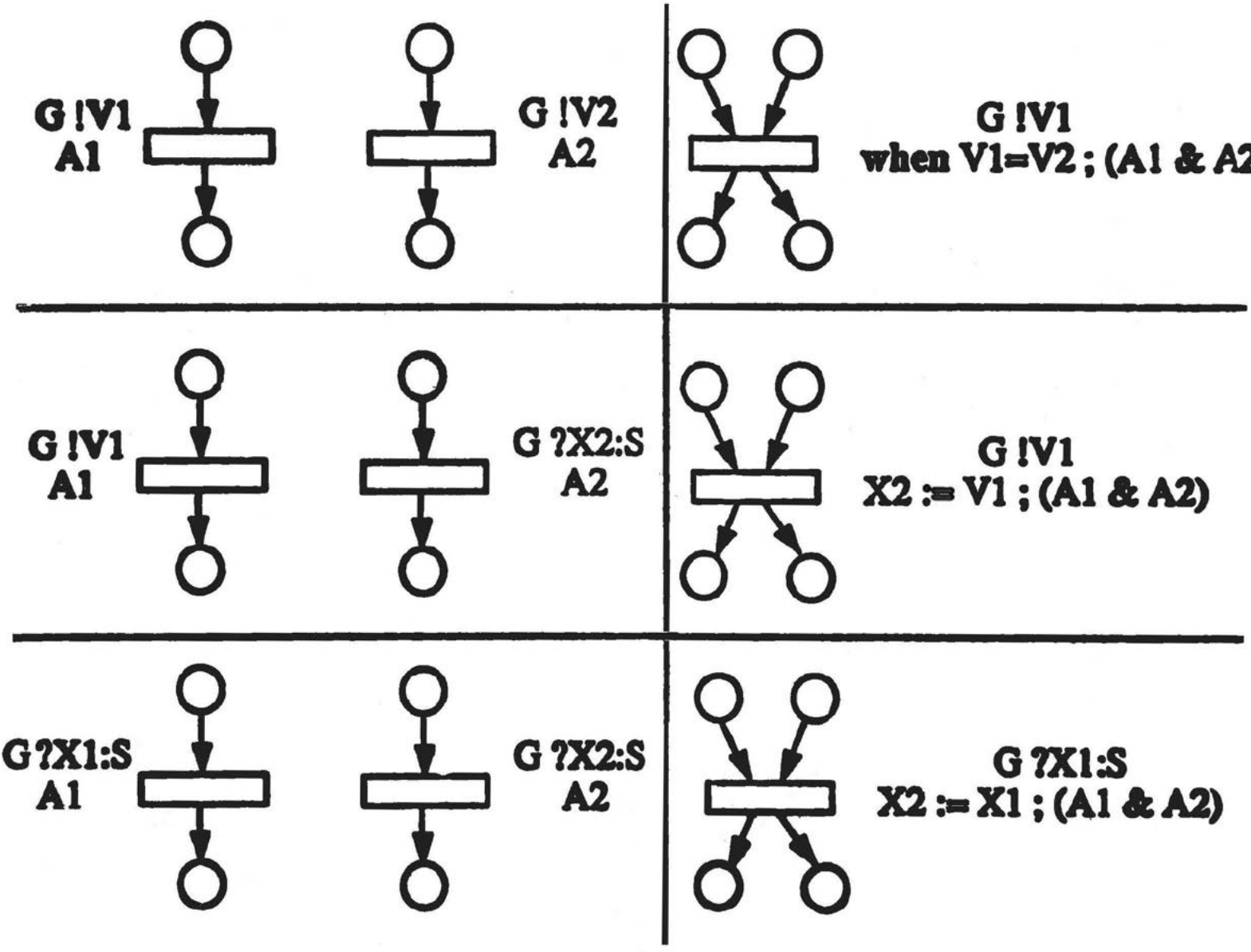
- construire le réseau correspondant à un programme SUBLOTOS
- par induction structurelle sur les comportements

stop	
G ; B	
B1 [] B2	
B1 || B2	

Génération

[V] -> B	 ε when V
let X:S=V in B	 ε X := V
choice X:S [] B	 ε for X among S
B1 [> B2	
instanciation récursive	boucle dans le réseau

Génération : couplage des transitions



Optimisation

Motivations

- algorithme de génération simple et systématique
- sans considérer les cas particuliers $\Longrightarrow$ inefficacités locales
- optimiser la taille du réseau (places, transitions, variables)

Catalogue d'optimisations

- **contrôle**
  - fusion d'ε-transitions
  - places simultanément marquées
  - places et transitions inaccessibles et improductives
- **données**
  - affectations de la forme $X := X$
  - variables constamment égales
  - variables inutilisées

Résultats

- gains importants (40% des places, transitions, variables)
- deux catégories d'optimisations :
  - celles qui corrigent la génération (ε-transitions)
  - celles qui sont propres à LOTOS (variables)
- effet d'amplification à la simulation :
  petite réduction du réseau $\Longrightarrow$ réduction massive du graphe
- concentrer les efforts sur le modèle réseau

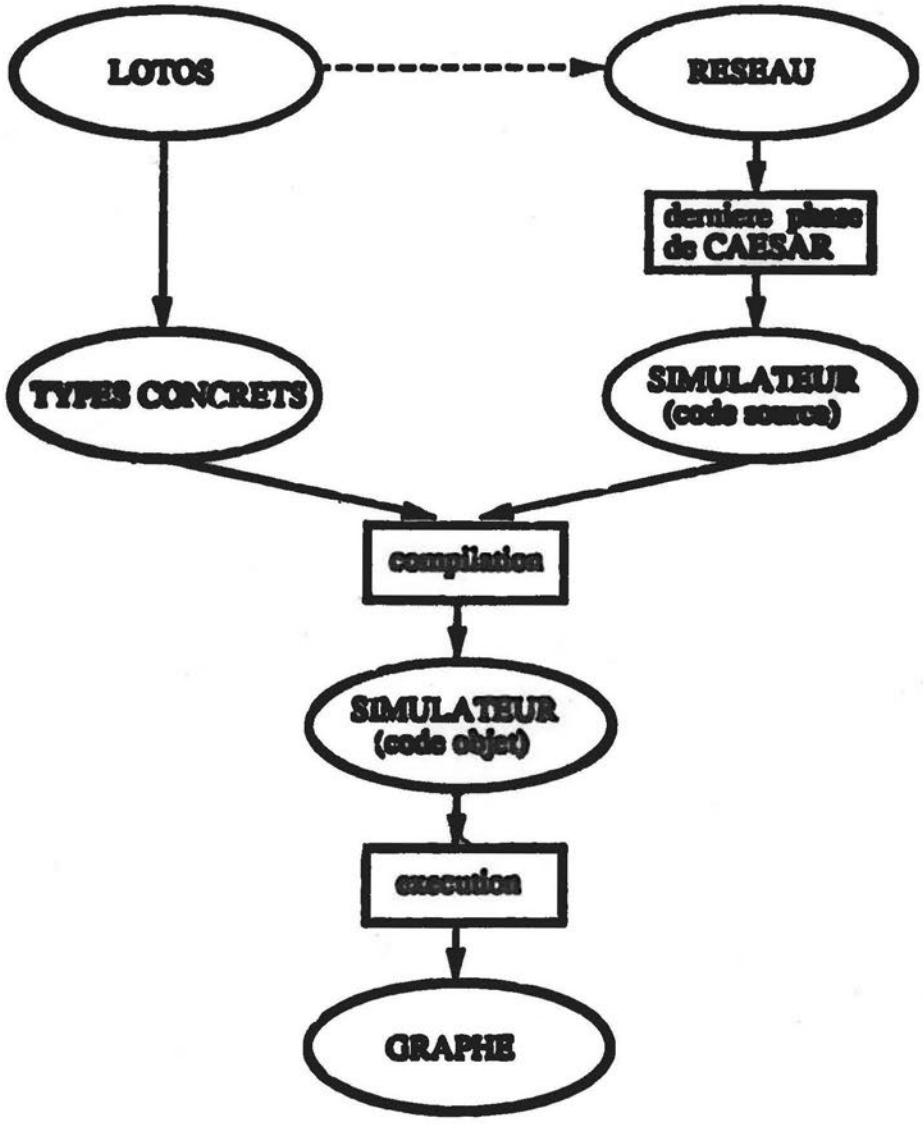
Simulation

Objectifs

- traduire un réseau en graphe

Le programme simulateur

- les types abstraits LOTOS doivent être compilés en C
- il est impossible d'avoir un traducteur universel réseau → graphe
- production d'un programme simulateur en langage C



Fonctionnement du simulateur

- point commun entre QUASAR, XESAR et CÆSAR
- parcours en largeur du graphe
- archivage des états rencontrés
- représentation des états
  - **marquage** : ensemble de places du réseau
  - **contexte** : valeurs des variables d'état
  - **état** : couple ⟨marquage, contexte⟩
- tir des transitions
  - évolution des marquages : règles des réseaux de Petri
  - évolution des contextes : actions des transitions
- impression du graphe sous différents formats

Comparaison entre les approches dynamique et statique

Statistiques

exemple	*nombre d'états*	SQUIGGLES	CÆSAR
`max2`	5	6s	2s
`max3`	11	7s	2s
`max5`	47	47s	3s
`max7`	191	31mn	5s
`max9`	767	4h 41mn	14s

Pourquoi l'approche statique est-elle meilleure ?

	approche dynamique	approche statique
représentation des états	termes algébriques	⟨marquage, contexte⟩
franchissement des transitions	règles de dérivation	lois d'évolution des réseaux de Petri interprétés
détection des circuits	équivalence entre termes	comparaison des états sous forme normale

Bilan

Nouvelle approche de la compilation de LOTOS

- traitement d'un sous-ensemble significatif de LOTOS
- génération efficace du modèle graphe
- introduction de formes intermédiaires :
  - le langage SUBLOTOS
  - le modèle réseau
- méthode de traduction définie formellement
- proposition d'une logique temporelle originale

Réalisation du système CÆSAR

- implémentation complète de la méthode de traduction
- traitement d'exemples complexes (800 000 états et 3 500 000 arcs)
- débit élevé (entre 40 et 250 états/seconde sur SUN 3/60 8 Mo)
- interface avec 7 outils existants :
  ALDEBARAN (LGI-IMAG), AUTO (INRIA), MEC (Bordeaux), PIPN (LAAS), SCAN (BULL/ADI), SQUIGGLES (Pise), XESAR (LGI-IMAG)
- environ 25 000 lignes de code (langage C, SYNTAX©)
- prototype CÆSAR.ADT pour compiler les types abstraits en C

Suite de ce travail

- preuve formelle du compilateur
  - montrer la correction des transformations successives
  - démonstration par induction sur les comportements
- nouvelles optimisations
  - élimination des ε-transitions
  - techniques propres aux réseaux de Petri
  - techniques d'analyse du flux des données
- évaluateur pour la logique temporelle R$\overset{\star}{\text{I}}$CO

De nouvelles applications

- au niveau du modèle **graphe** :
  - exécution interactive
  - génération de tests
  - vérification partielle
- au niveau du modèle **réseau** :
  - évaluation de performances
  - génération de prototypes (code séquentiel ou parallèle)
  - visualisation graphique de l'architecture

LOTOS : un "vrai" langage de programmation

Un espoir pour la vérification formelle ? [28]

- problème : explosion du nombre d'états
- conserver l'approche "génération de modèles"
- mais réduire **avant** la simulation et non **après**
- appliquer des critères d'abstraction au réseau
  pour des équivalences plus faibles que l'équivalence forte
- réduction minime du réseau $\Longrightarrow$ réduction importante du graphe