

# Assemblage et tissage de composants

Stephane HONG TUAN HA

Encadreur : Pascal Fradet

Inria/Irisa, projet Lande

# Contexte

- ▶ Composants logiciels
  - ▶ nous considérons un logiciel comme des composants interconnectés
  - ▶ un composant est un processus déterministe avec des ports d'entrée et de sortie
  - ▶ un logiciel est un réseau de processus interconnectés via leur ports par des canaux de communication
- ▶ AOP (Aspect Oriented Programming)

# Plan

- ▶ Décrire le modèle de composants
- ▶ Assemblage performant de composants
- ▶ Tisser des propriétés de sécurité

# **Modèle de composants basé sur les réseaux de Kahn**

# Le langage [2]

- ▶ Un composant est un ensemble de commandes :

$$P ::= \{C_1, \dots, C_n\}$$

- ▶ Les commandes ont la forme suivante :

$$C ::= l_1 : G \mid A \rightsquigarrow l_2$$

où  $l_1, l_2$  sont des labels ( $\in \mathbf{Label}$ ),  $G$  une garde et  $A$  une action.

- ▶ Une action est soit une opération de lecture, une opération d'écriture, ou une action interne

$$A ::= f?x \mid f!x \mid I \quad f \in \mathbf{Channel}, x \in \mathbf{Var}$$

# Exemple de composant

2ByteSwap = {

$l_0 : (eofIn) \mid skip \rightsquigarrow l_{fin}$

$l_0 : \overline{(eofIn)} \mid in?x \rightsquigarrow l_1$

$l_1 : true \mid in?y \rightsquigarrow l_2$

$l_2 : true \mid out!y \rightsquigarrow l_3$

$l_3 : true \mid out!x \rightsquigarrow l_0$

$l_{fin} : false \mid skip \rightsquigarrow l_{fin}$

}

# Quelques définitions

- ▶ Un store est une fonction qui fait correspondre les variables d'un composant à leurs valeurs  
( $\text{Var} \rightarrow \text{Val}_\perp$ )
- ▶ La sémantique des actions internes et des gardes sont donnés par les fonctions

$$\mathcal{I} : I \rightarrow \text{Store} \rightarrow \text{Store} \text{ and } \mathcal{G} : G \rightarrow \text{Store} \rightarrow \mathbb{B}$$

- ▶ La sémantique est exprimée par une fonction de transition entre des états constitués d'un label et d'un store

# Réseau de processus de Kahn (KPN) [1]

- ▶ Ensemble de processus communicants de manière asynchrone par des files FIFO
- ▶ Sémantique d'un réseau de Kahn :
  - ▶ lecture bloquante
  - ▶ écriture non bloquante
  - ▶ pas d'instruction pour tester la présence ou l'absence d'éléments dans une file
- ▶ Propriétés intéressantes des KPNs :
  - ▶ déterministe
  - ▶ modèle hiérarchique
  - ▶ sémantique dénotationnelle

# Quelques définitions

- ▶ Un canal interne est écrit par exactement un composant et lu par exactement un composant
- ▶ Un canal d'entrée est lu par exactement un composant (et écrit par aucun)
- ▶ Un canal de sortie est écrit par exactement un composant (et lu par aucun)

# Éléments de formalisation

- ▶ Représentation d'un état du KPN par une paire  $(\pi, \gamma)$  avec :
  - ▶  $\pi : \mathbf{Prog} \rightarrow \mathbf{State}$  : une fonction qui associe à chaque processus  $pp \in \mathbf{Prog}$  son état courant  $q_p$ .
  - ▶  $\gamma : \mathbf{Channel} \rightarrow \mathbf{Val}^*$  : une fonction qui associe à chaque canal interne  $c$  son état (*i.e.* une chaîne finie sur  $\mathbf{Val}$ )

# Sémantique d'un réseau de composants [3]

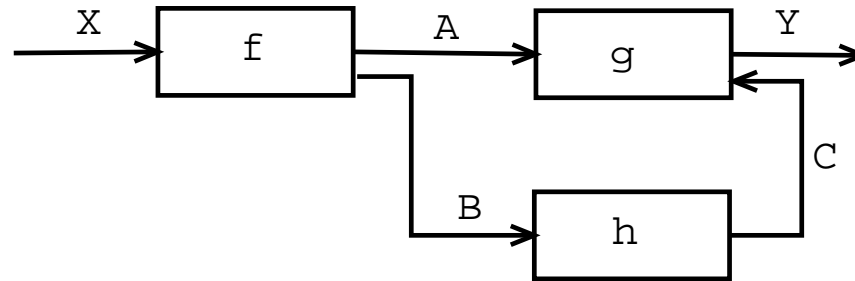
- ▶ La relation de transition  $\longrightarrow_K$  d'un KPN  $K$  est définie par des règles d'inférences ; par exemple :

$$\text{Lecture interne} \quad \frac{\pi(p) \xrightarrow{f?v}_P (l, s), \gamma(f) = v\sigma}{(\pi, \gamma) \xrightarrow{f?v}_K (\pi[p \mapsto (l, s)], \gamma[f \mapsto \sigma])}$$

- ▶ Une trace est une suite d'états de la forme :

$$(\pi_0, \gamma_0) \xrightarrow{\alpha_1}_K (\pi_1, \gamma_1) \xrightarrow{\alpha_2}_K \dots$$

# Exemple d'assemblage



- ▶ Soulève un problème d'efficacité de l'implémentation de l'assemblage
  - ▶ Soit en séquence avec une utilisation non bornée de la mémoire
  - ▶ Soit par quantum de temps avec une perte de performance due aux changements de contexte

# Problématique de la performance

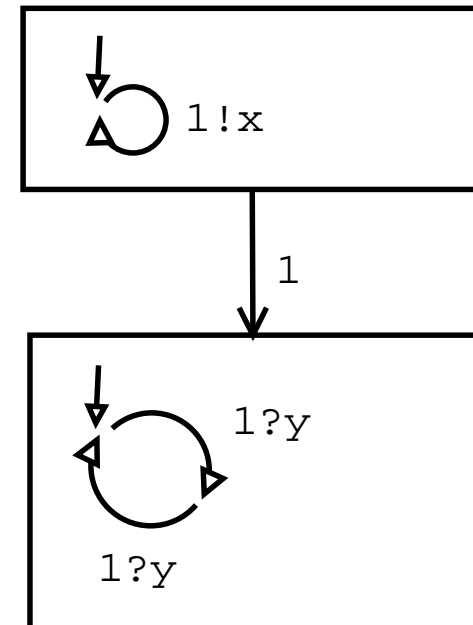
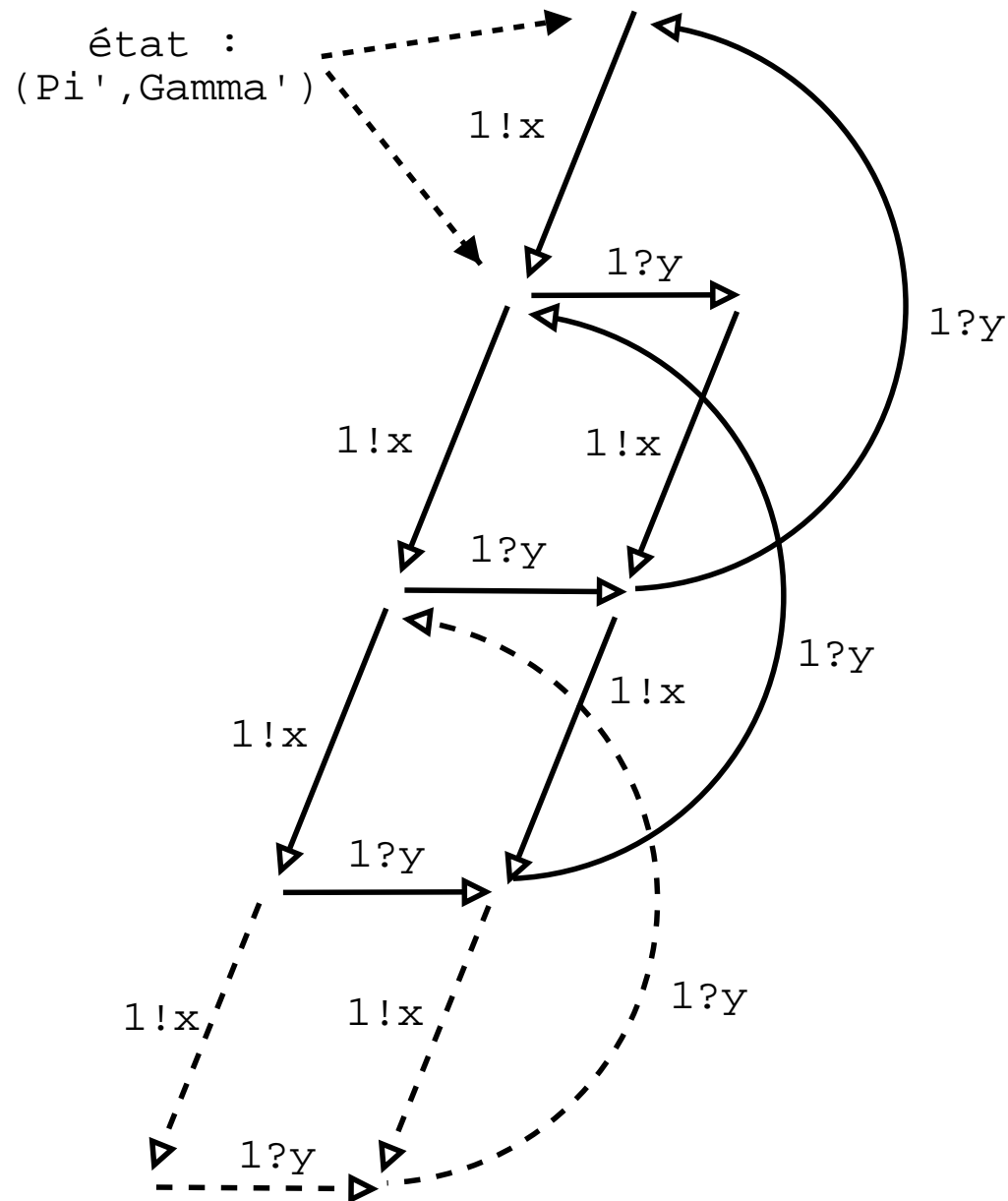
- ▶ Composition invasive / fusion :
  - ▶ fournir à l'utilisateur plusieurs techniques pour préciser l'assemblage
  - ▶ rechercher automatiquement un ordonnancement
    - ▶ Objectif technique : séquentialiser totalement l'exécution parallèle d'un réseau de Kahn
      - passer d'un réseau de composants à un unique composant équivalent
      - enlever les communications internes et les remplacer par des affectations

# Recherche automatique d'un ordonnancement

# Graphe abstrait des exécutions

- ▶ Présentation de l'abstraction :
  - ▶ abstraction des files par leur taille exacte ou par un intervalle de taille
  - ▶ pas de prise en compte des valeurs
- ▶ Graphe infini qui représente une sur-approximation des exécutions
  - ▶ pas d'évaluation des conditions

# Exemple d'un graphe abstrait

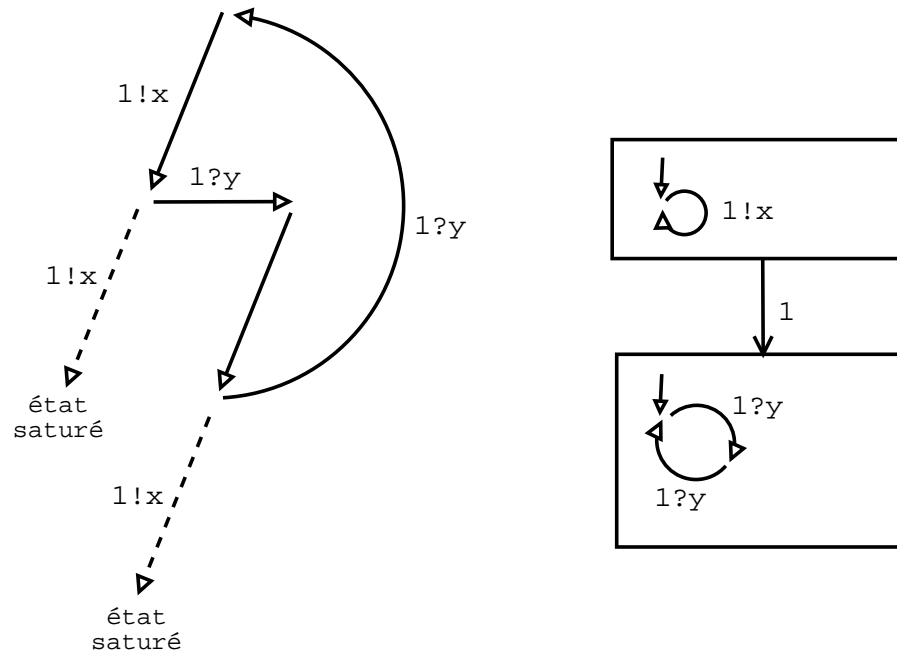


# Propriétés d'un ordonnancement

- ▶ Respect de la sémantique du KPN
- ▶ Séquentialité
- ▶ Maximalité
- ▶ Équité (fairness)
  - ▶ propriété sur les boucles

# Critère de saturation

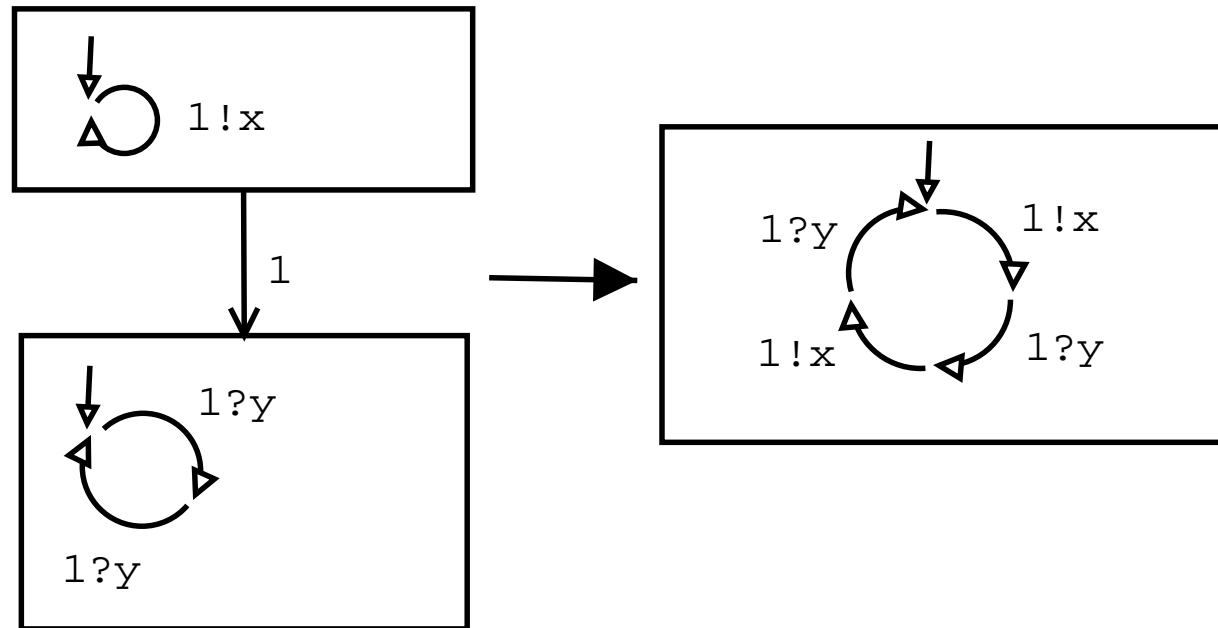
- ▶ Saturation de l'état  $(\pi, \gamma_2)$  sur le graphe des exécutions  $G$ , si le graphe contient un autre état  $(\pi, \gamma_1)$  tel que :
  - ▶  $\exists$  un chemin de  $(\pi, \gamma_1)$  à  $(\pi, \gamma_2)$
  - ▶  $\forall c, \gamma_1(c) \leq \gamma_2(c)$
  - ▶  $\exists c, \gamma_1(c) < \gamma_2(c) \wedge \gamma_1(c) \geq 1$
- ▶ exemple :



# Recherche d'un ordonnancement borné

- ▶ Restreindre le graphe abstrait avec le critère de saturation
- ▶ Recherche d'un sous-graphe vérifiant la définition d'un ordonnancement
- ▶ Terminaison assurée par la taille finie du graphe

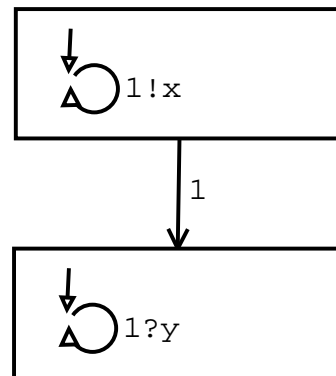
# Exemple pour un réseau borné



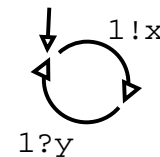
# **Tissage de propriétés de sécurité dans ce cadre**

# Plusieurs approches possibles

1. Tisser la propriété sur le programme fusionné (tissage classique)
  - ▶ "Enforcing Trace Properties (...)"
2. Tisser la propriété de sécurité sur les composants
  - ▶ par ajout de canaux de communication
3. Tisser la propriété pendant la fusion
  - ▶ contraindre le graphe abstrait des exécutions

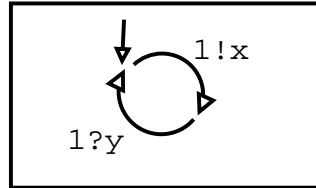


Propriété  
sur les traces :

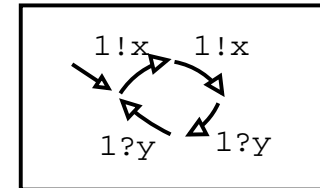


# Exemple simple

tisser  
la propriété  
après la fusion :

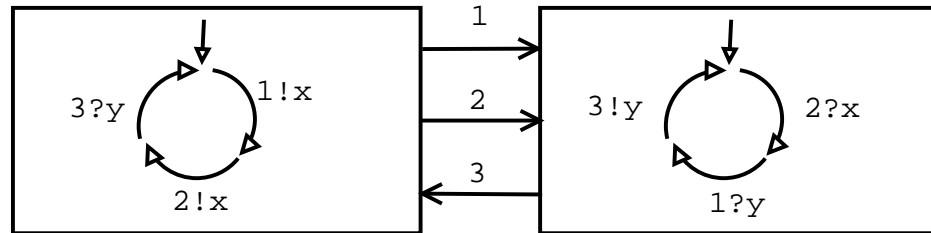


programme accepté

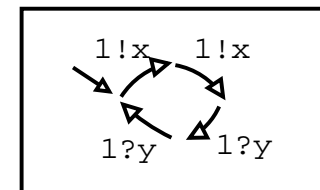
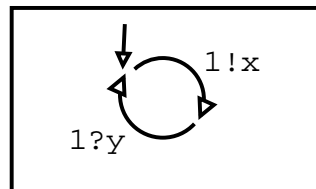


programme rejeté

tisser  
les composants  
indépendamment :



prendre en compte  
la propriété  
pendant la fusion :



cas impossible

# Bibliographie

## References

- [1] G. Kahn, The semantics of a simple language for parallel programming, In *Information Processing '74: Proceedings of the IFIP Congress*, 1974.
- [2] P. Cousot and R. Cousot. Systematic Design of Program Transformation Frameworks by Abstract Interpretation. In *Conference Record of Symposium on Principles of Programming Languages*, 2002.
- [3] M. Geilen and T. Basten, Requirements on the Execution of Kahn Process Networks, In *12th European Symposium on Programming*, 2003.