

Aspects de synchronisation et disponibilité

Stephane HONG TUAN HA

ACI DISPO

Inria/Irisa, projet Lande

Contexte et Plan

- ▶ ACI Dispo - Projet Lande
 - ▶ Composants logiciels
 - ▶ AOP (Aspect Oriented Programming)
 - ▶ Disponibilité
- ▶ Les 3 points étudiés :
 - ▶ modèle de composants qui est formel et simple
 - ▶ aspect d'ordonnancement
 - ▶ pour fusionner des composants
 - ▶ pour obtenir un assemblage performant de composants
 - ▶ aspect de disponibilité
 - ▶ pour imposer une politique de disponibilité

Modèle de composants

Modèle de composants

- ▶ Modèle basé sur les réseaux de processus de Kahn
- ▶ Ensemble de processus communicants de manière asynchrone par des files FIFO
 - ▶ 1 canal de communication relie au plus 2 composants
- ▶ Sémantique :
 - ▶ lecture bloquante
 - ▶ écriture non bloquante
 - ▶ pas d'instruction pour tester la présence ou l'absence d'éléments dans une file
- ▶ Propriétés intéressantes : déterministe, modèle hiérarchique, et sémantique dénotationnelle

Langage des composants

- ▶ Définition des composants par un ensemble de commandes de la forme :

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

où l_1, l_2 sont des labels, G une garde, et A une action.

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

$$A ::= f?x \mid f!x \mid I$$

où f est un canal et x une variable

- ▶ Définition des connections entre les composants

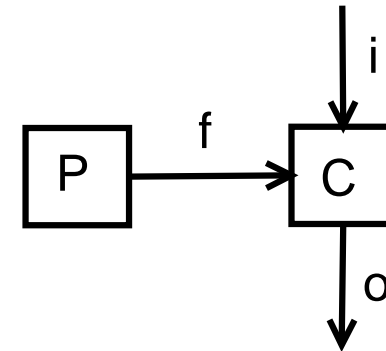
Exemple - Réseau de composants

► Composants :

$$P = \left\{ \begin{array}{l} p_0 : x := x + 1 \quad \rightsquigarrow \quad p_1; \\ p_1 : f!x \quad \rightsquigarrow \quad p_0 \end{array} \right\}$$

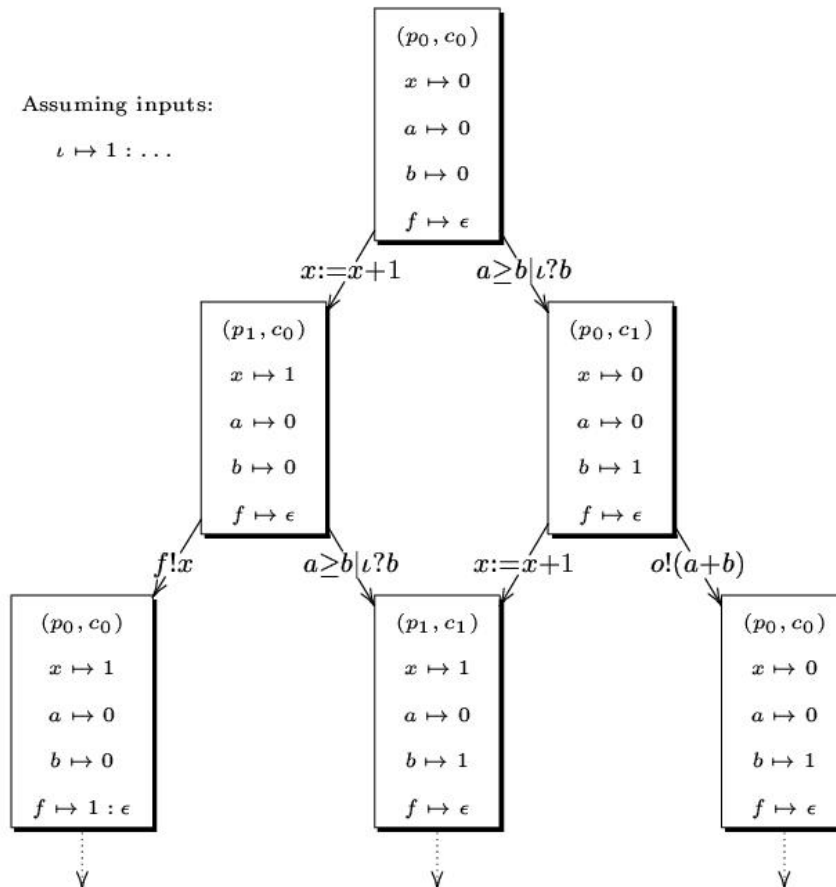
$$C = \left\{ \begin{array}{l} c_0 : a < b \mid f?a \quad \rightsquigarrow \quad c_1; \\ c_0 : a \geq b \mid \iota?b \quad \rightsquigarrow \quad c_1; \\ c_1 : o!(a + b) \quad \rightsquigarrow \quad c_0 \end{array} \right\}$$

► Réseau :



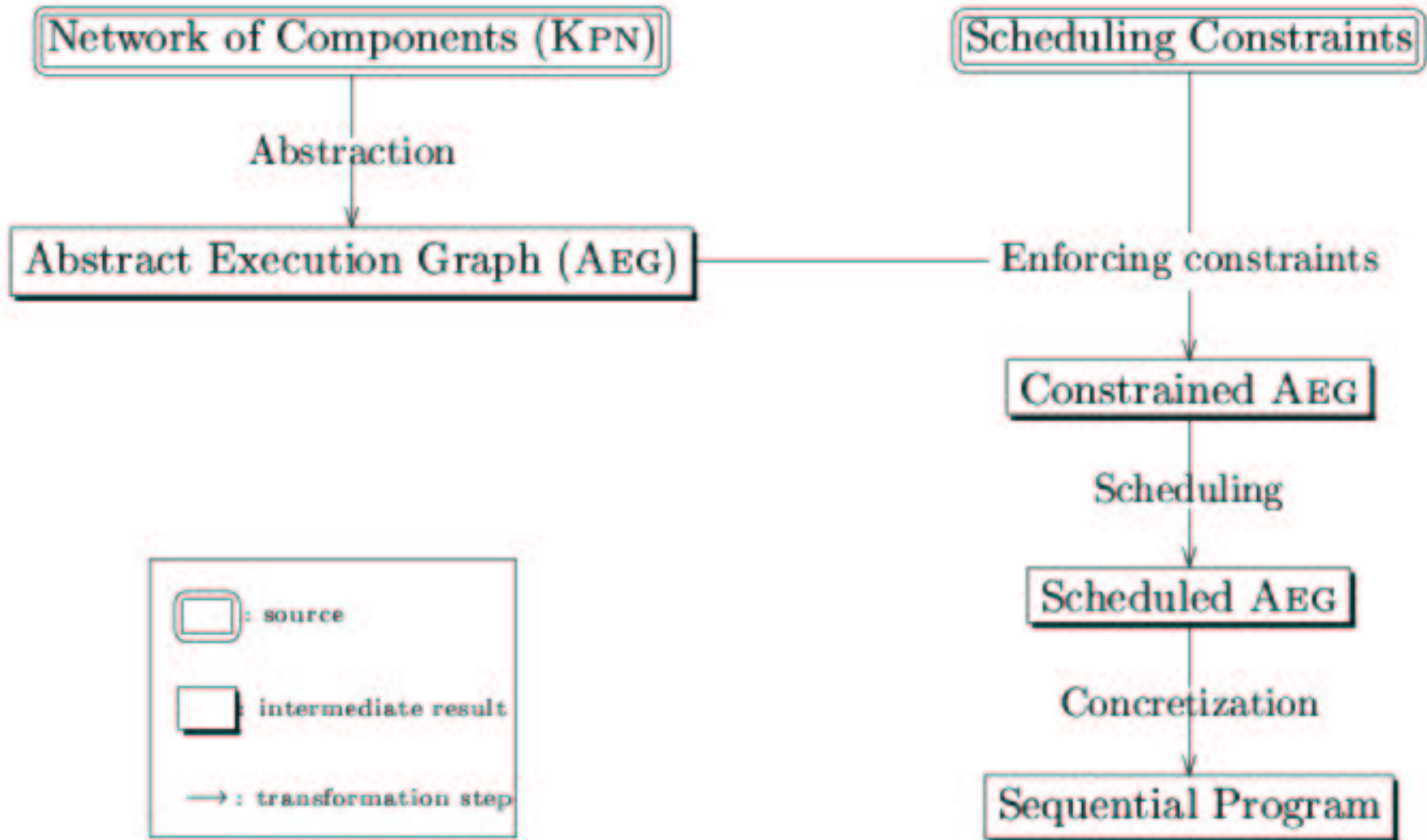
Sémantique du réseau

- ▶ Exprimée comme un LTS
- ▶ Exemple



Aspect d'ordonnancement

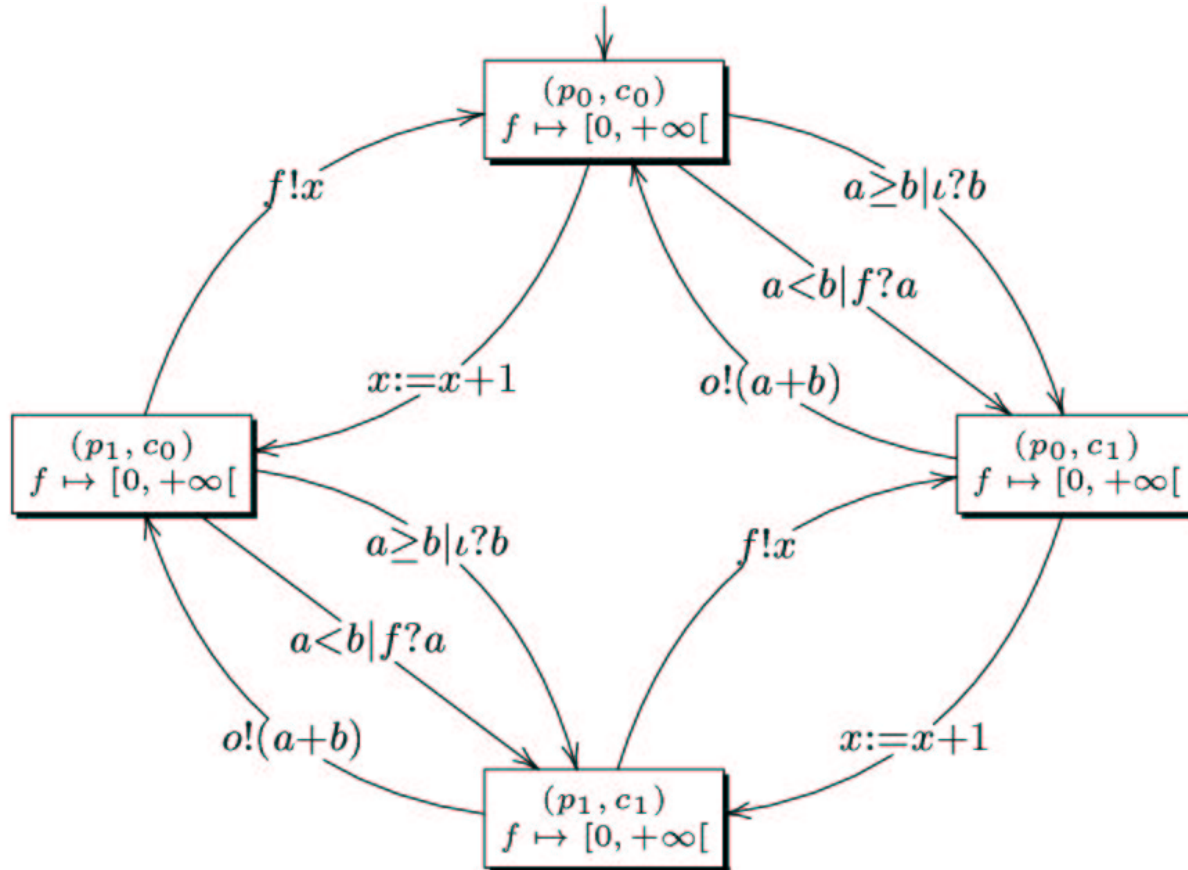
Fusion de composants - Présentation



Graphe abstrait des exécutions (AEG)

- ▶ Représentation finie de la sémantique du réseau
 - ▶ chaque état abstrait résume un ensemble potentiellement infini de configurations
 - ▶ exprimée comme un LTS
- ▶ 2 propriétés clefs :
 - ▶ sûreté (safety)
 - ▶ fidélité (faithfulness)
- ▶ Plusieurs abstractions possibles (taille vs précision)
- ▶ Exemple d'abstraction
 - ▶ pas de prise en compte des valeurs
 - ▶ abstraction des files par $[0; +\infty[$

Exemple - AEG

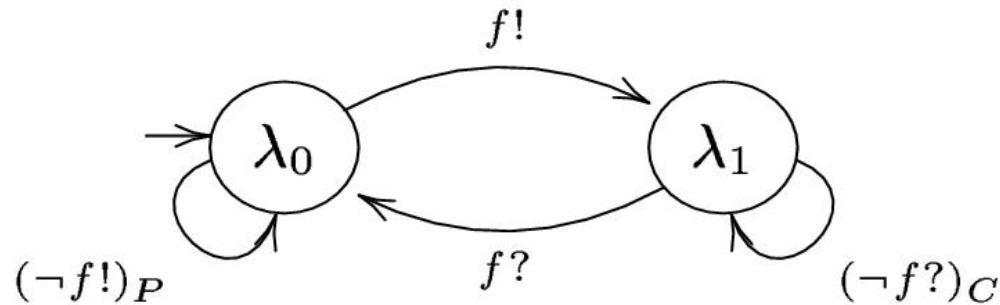


Aspect d'ordonnancement

- ▶ Objectif :
 - ▶ décrire plus finement un assemblage de composant
 - ▶ restreindre le comportement du réseau
 - ▶ imposer des choix d'implémentation
- ▶ Exprimé par des automates
 - ▶ transitions annotées par les actions gardés
 - ▶ avec respect des entrées/sorties, des propriétés d'équité, et de la taille des files
- ▶ Application de l'aspect sur l'AEG par un produit synchronisé

Exemple - Aspects

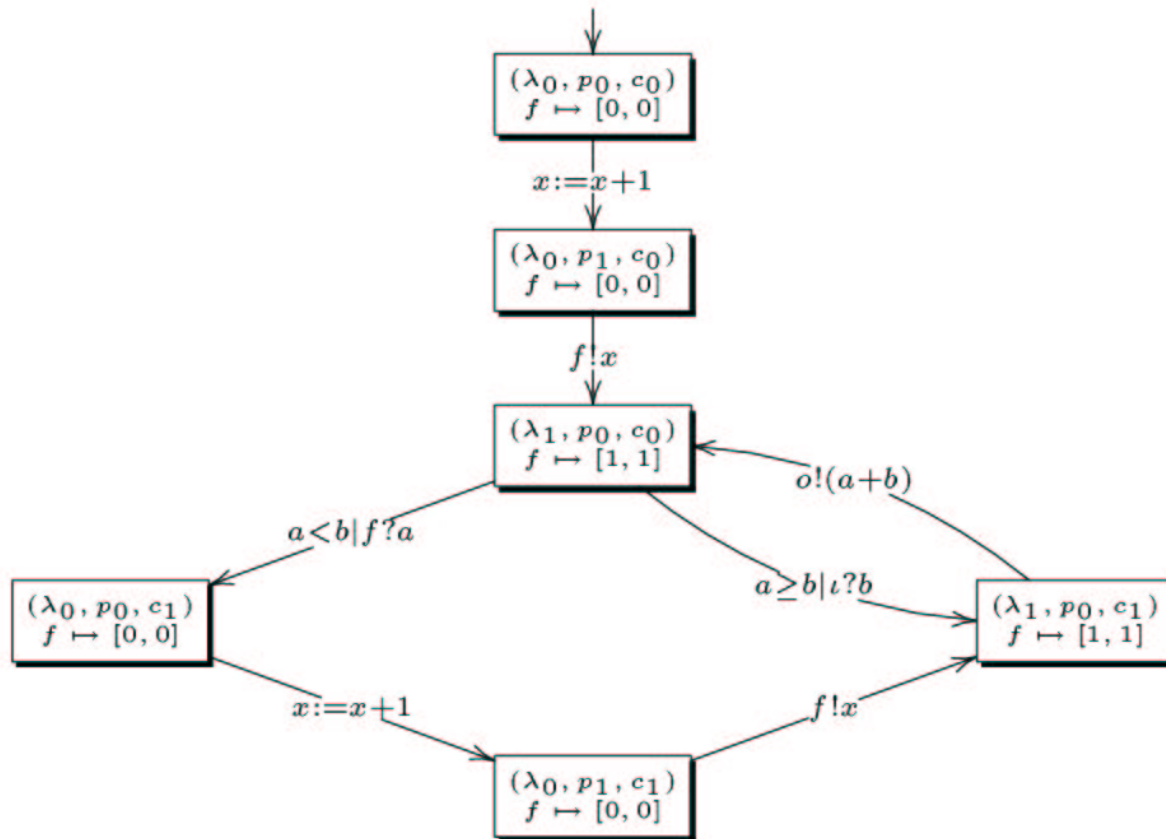
- ▶ Aspect "Filter Fusion" :



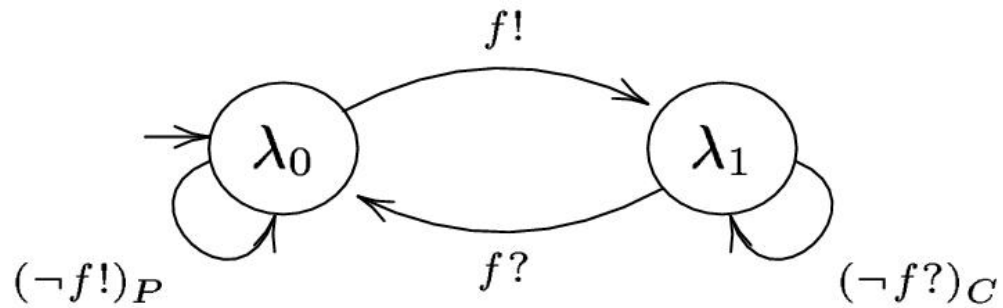
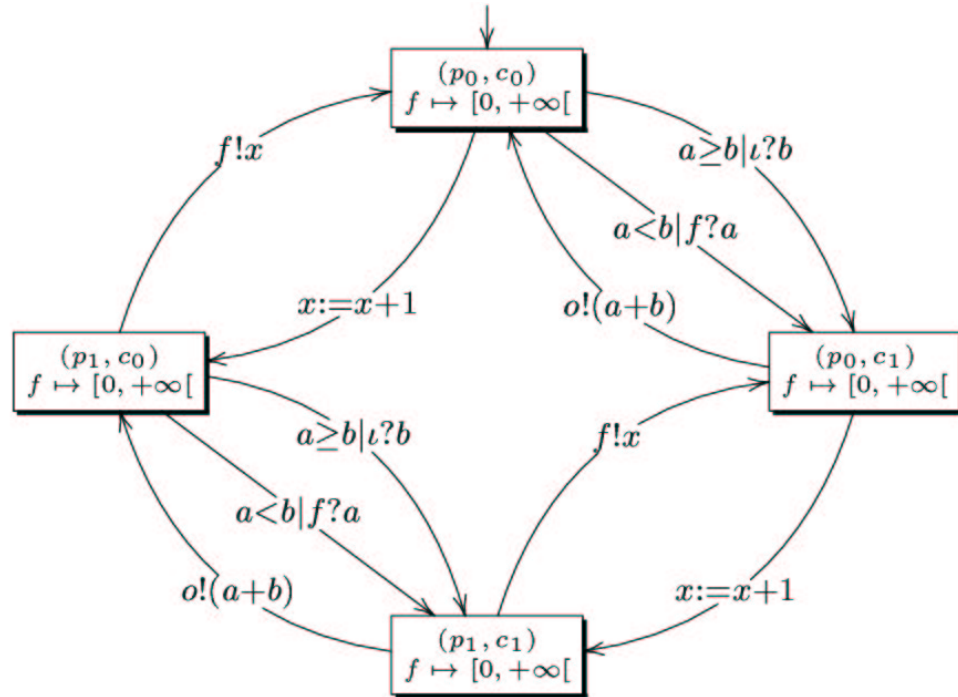
- ▶ exécuter le producteur jusqu'à une écriture
- ▶ puis exécuter le consommateur jusqu'à une lecture
- ▶ puis recommencer

Fusion

- ▶ Produit d'automates entre l'AEG et l'aspect "Filter Fusion", $Res = AEG || FF$
- ▶ Exemple

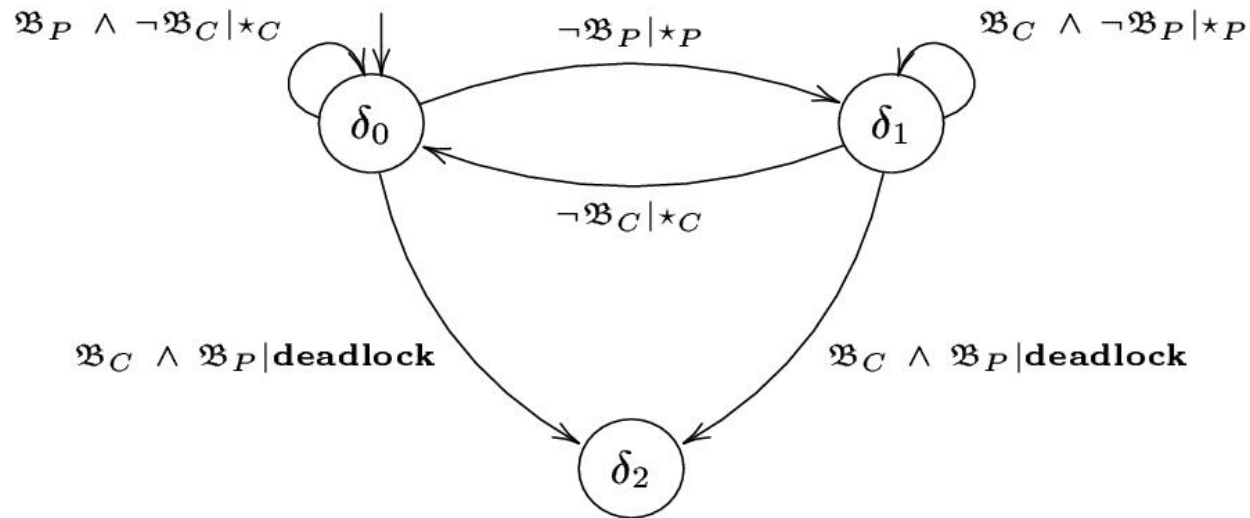


Fusion



Exemple - Aspect round-robin

- ▶ Aspect pour tout séquentialiser



Conclusion sur l'Aspect d'ordonnancement

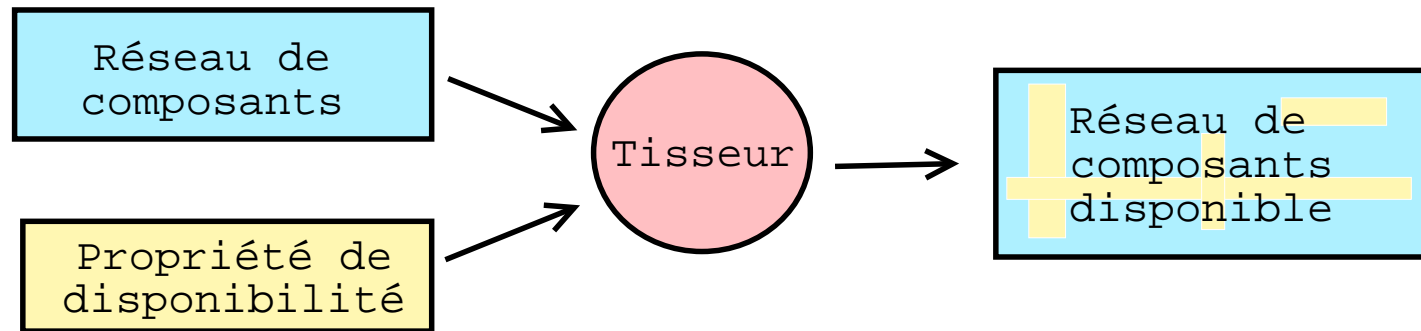
- ▶ Framework générique pour fusionner des composants
 - ▶ basé sur les LTS, le produit synchronisé, l'analyse statique et les transformations de programme
- ▶ Intérêts
 - ▶ Obtenir un assemblage performant de composants
 - ▶ Séparation de l'aspect d'ordonnancement des fonctionnalités du programme
 - ▶ Preuve formelle de la méthode

Aspect de Disponibilité

Aspect de Disponibilité

- ▶ Objectif

- ▶ tisser des propriétés de disponibilité

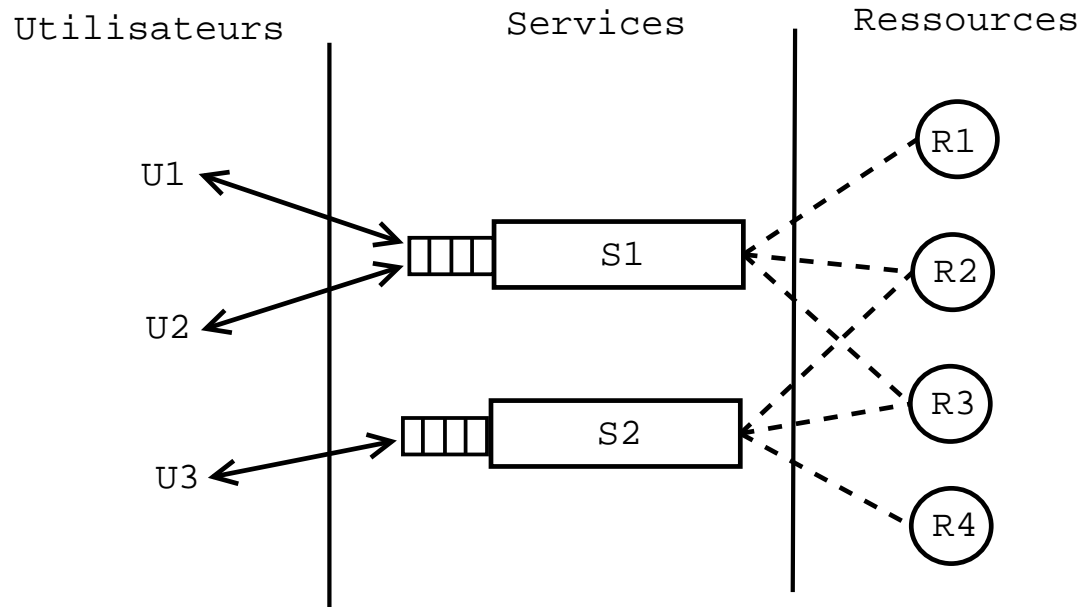


- ▶ Les points étudiés :

- ▶ modèle pour la disponibilité
- ▶ langage d'aspect de disponibilité

Modèle pour la disponibilité

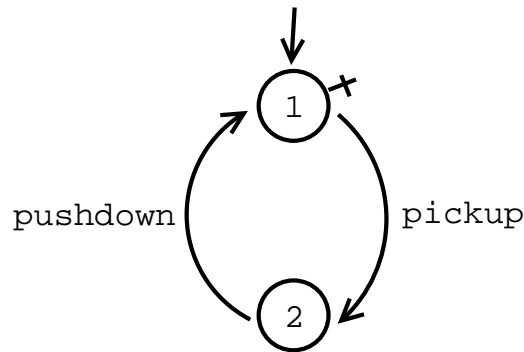
► Modèle en 3 couches



- cohérent avec une architecture client-serveur
- par rapport à Yu&Gligor, possibilité de tisser les services

Automate de disponibilité

- ▶ Définition classique d'un automate
 - ▶ état initial, état final et fonction de transition
 - ▶ différent des automates de Schneider
- ▶ Contraintes sur les traces
 - ▶ pour imposer le contrat utilisateur
 - ▶ les seules traces du service qui sont autorisées sont les traces définies par l'automate
- ▶ Exemple : un *pushdown* après chaque *pickup*



Un premier langage d'aspect

- ▶ Un automate de disponibilité
- ▶ Des contraintes temporelles en relation avec l'automate
 - ▶ durée maximum
 - ▶ hors d'un état terminal
 - ▶ dans un état non terminal
 - ▶ unité de mesure
 - ▶ durée exprimée en secondes
 - ▶ durée exprimée en instructions

Langage d'aspect

- ▶ Liste non exhaustive de paramètres intéressants
 - ▶ état du système
 - ▶ type de la ressource (quelle taxinomie ?)
 - ▶ fréquence et nombre des demandes d'un service
 - ▶ charge de la mémoire et du CPU
 - ▶ mode dégradé/liste noire/...
 - ▶ type des requêtes
 - ▶ priorité du service
 - ▶ dépend du type de la ressource
 - ▶ type des réponses du fournisseur
 - ▶ acceptation + durée avant expiration
 - ▶ refus + redémarrage du service
 - ▶ refus + mise sur liste noire
 - ▶ refus + passage du système en mode dégradé

Conclusion

- ▶ Modèle de composants simple et formel
- ▶ Aspect de synchronisation
 - ▶ assemblage performant des composants
 - ▶ 1 article accepté à APLAS'04 et 1 rapport technique étendu en cours
- ▶ Aspect de disponibilité
 - ▶ modèle pour la disponibilité
 - ▶ un premier langage simple vers un véritable langage plus général
 - ▶ (future work) tissage de cet aspect sur les services