Formalisation des interactions asynchrones

Florent Chevrou, Aurélie Hurault, Philippe Quéinnec

IRIT – Équipe ACADIE – Toulouse, France

30 Janvier 2017 - 1 Février 2017 // Kick-off projet PARDI

Problèmatique

Différentes entités sont-elles capables de communiquer entre elles dans un monde asynchrone?

Synchrone

- Rendez-vous entre l'émetteur et le récepteur
- Bloquant

Asynchrone

- L'envoi n'est pas conditionné par la possibilité de réception
- Envoi non-bloquant / réception bloquante
- Transmission non instantanée

Exemple

Scénario

Un étudiant qui a échoué à un examen doit le repasser. Quatre processus sont impliqués :

Superviseur Demande à l'étudiant s'il veut repasser l'examen. Envoie l'ancienne note s'il refuse. Prévient l'enseignant.

Secrétaire Récupère le nom et la note finale de l'étudiant

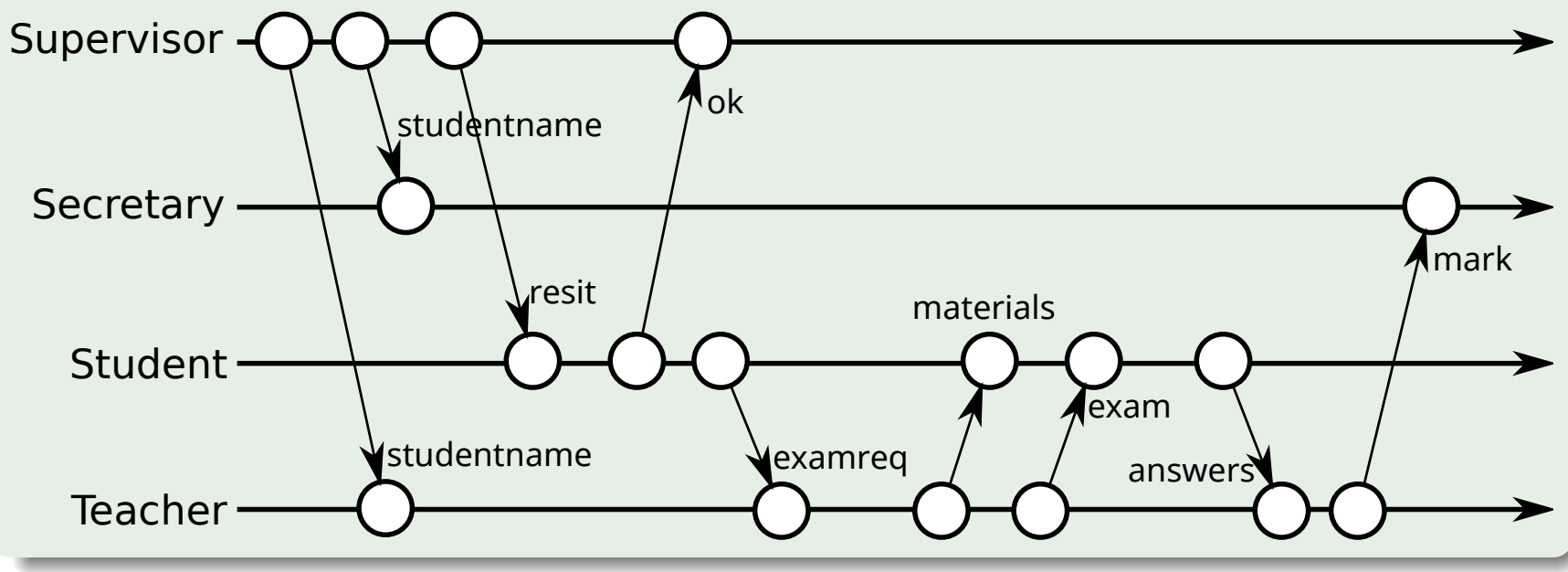
Étudiant Répond au superviseur et contacte l'enseignant pour le nouvel examen

Enseignant Récupère le nom de l'étudiant. Envoie le nouvel examen puis la nouvelle note.

Exécutions acceptables

Cas 1: L'étudiant repasse l'examen

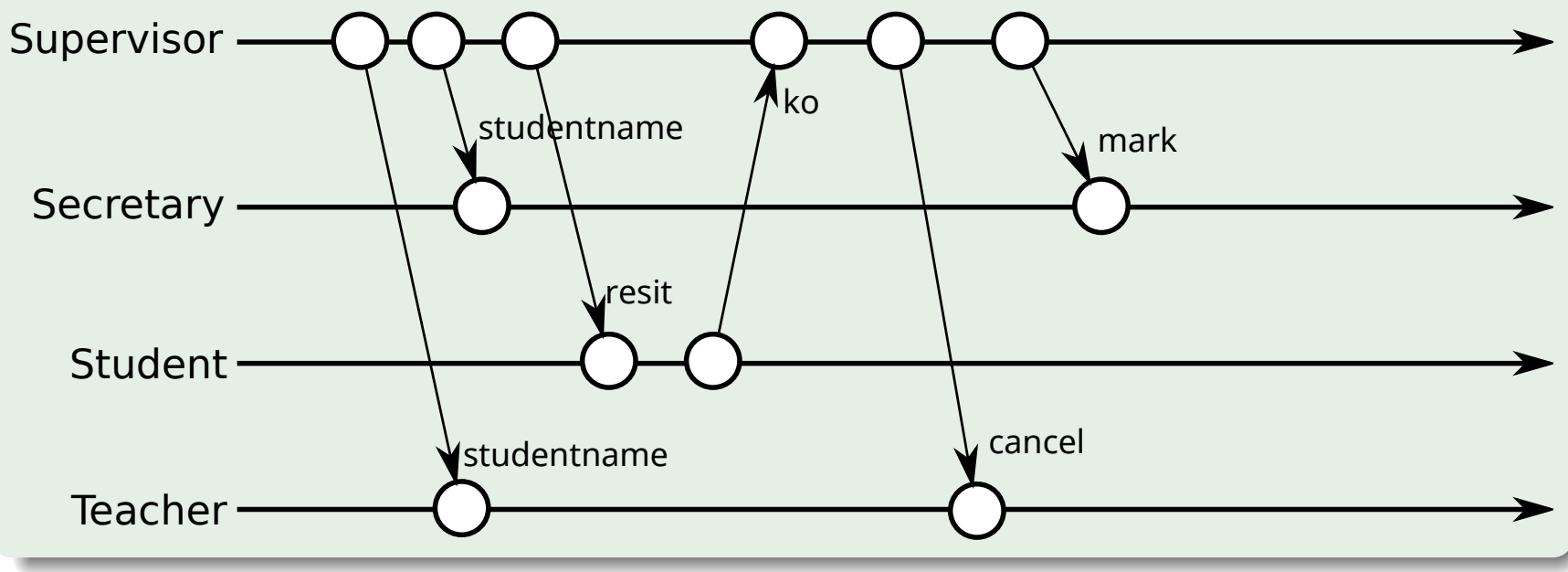
L'étudiant repasse l'examen



Exécutions acceptables

Cas 2: L'étudiant ne repasse pas l'examen

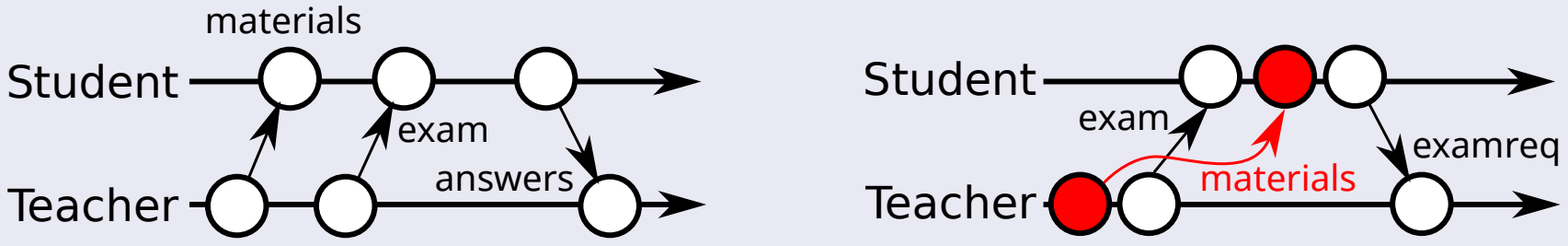
L'étudiant ne repasse pas l'examen



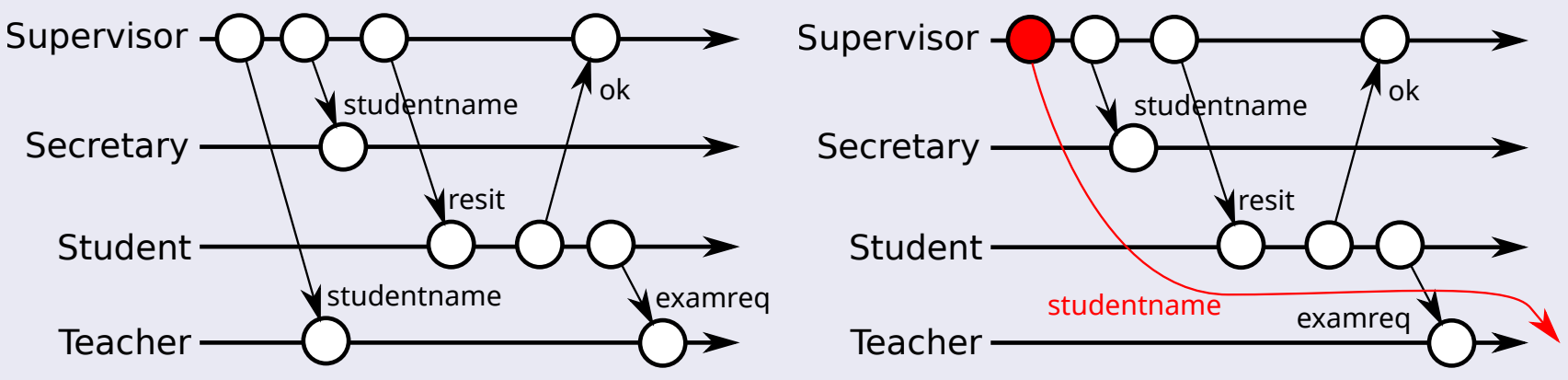
Exécutions problématiques

Problématique d'ordre de délivrance des messages

Violation de l'ordre FIFO



Violation de la causalité



Incompatibilité

Différents critères

Propriétés vérifiées sur le système composé

- Absence d'interblocage
- Terminaison
- Pas de réception inattendue
- Pas de message perpétuellement en attente
- ...

Solution proposée

Propriétés d'ordre de délivrance des messages

- Besoin de garantir des propriétés d'ordre de délivrance des messages pour différents groupes de canaux
- Prise en compte des différents modèles de communication asynchrone

Groupes et ordres

FIFO	materials, exam
Causal	studentname, resit, examreq, cancel, mark
Aucun	ok, ko, answers

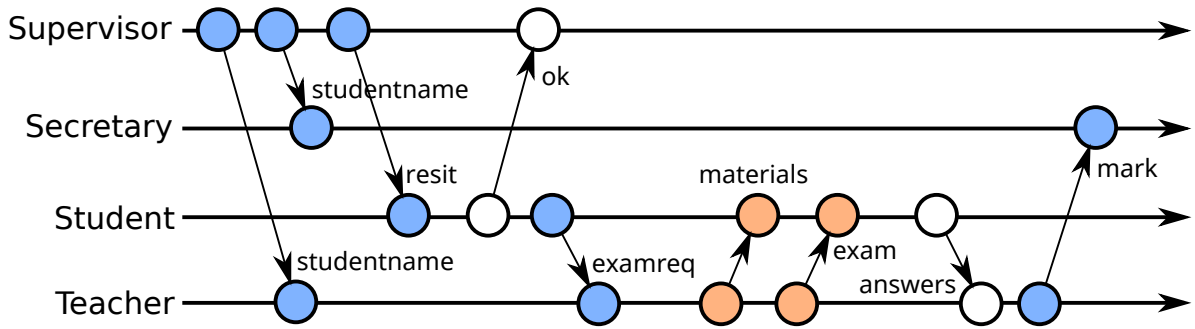
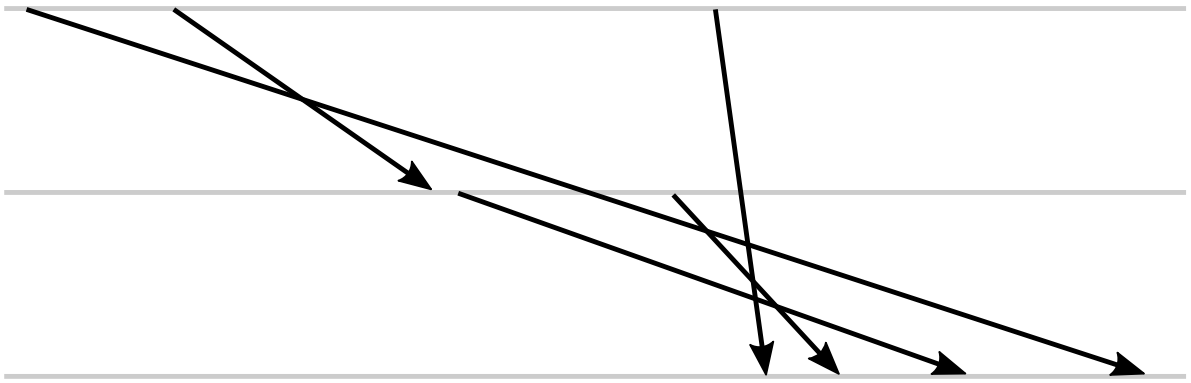
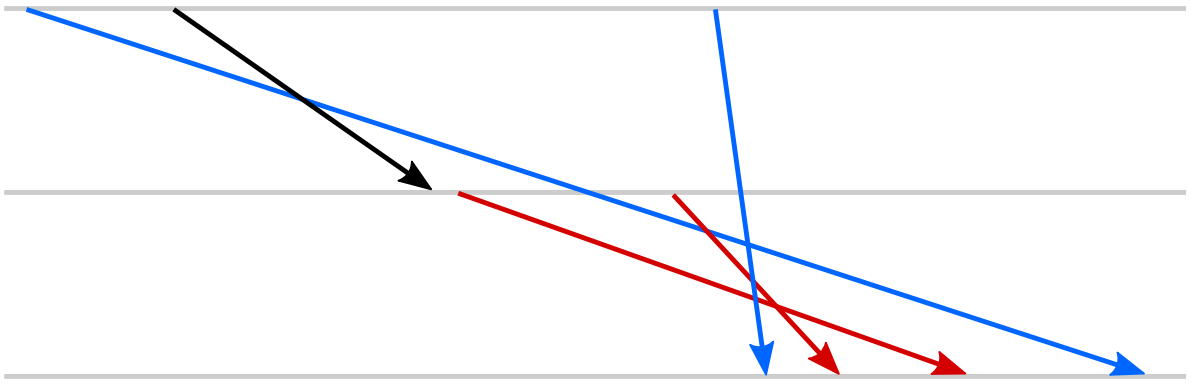


Illustration des modèles de communication



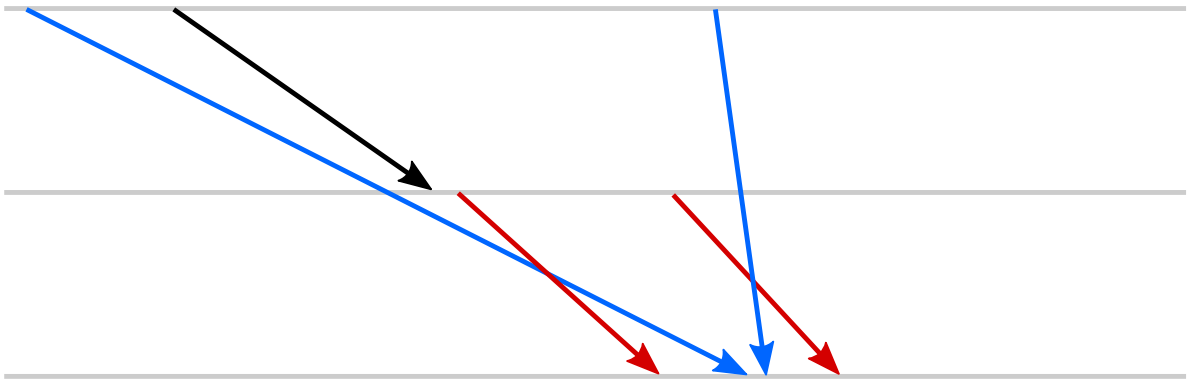
- **Pur async.**
- FIFO 11
- Causal
- FIFO n1
- FIFO nn
- RSC
- Synchrone

Illustration des modèles de communication



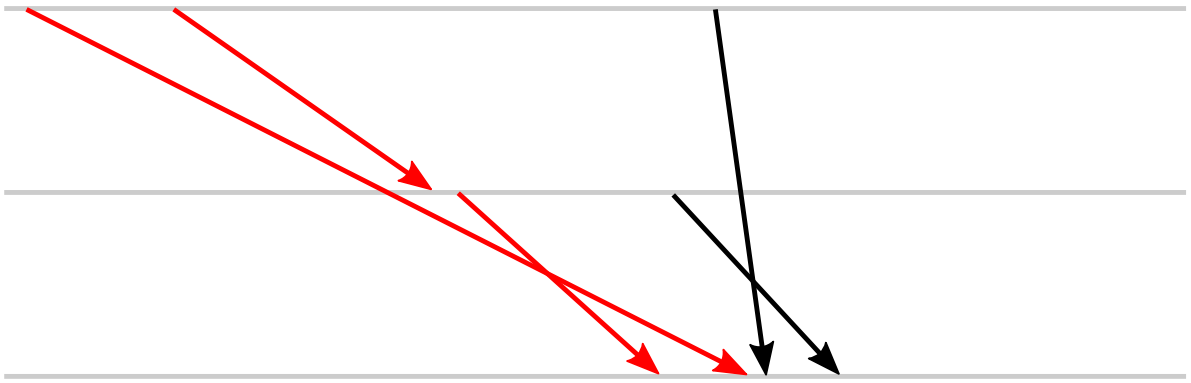
- **Pur async.**
- FIFO 11
- Causal
- FIFO n1
- FIFO nn
- RSC
- Synchrone

Illustration des modèles de communication



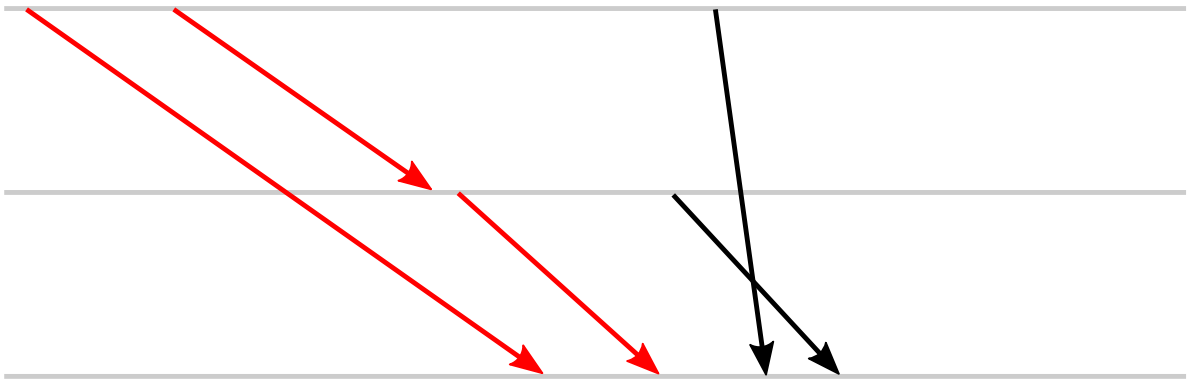
- Pur async.
- **FIFO 11**
- Causal
- FIFO n1
- FIFO nn
- RSC
- Synchrone

Illustration des modèles de communication



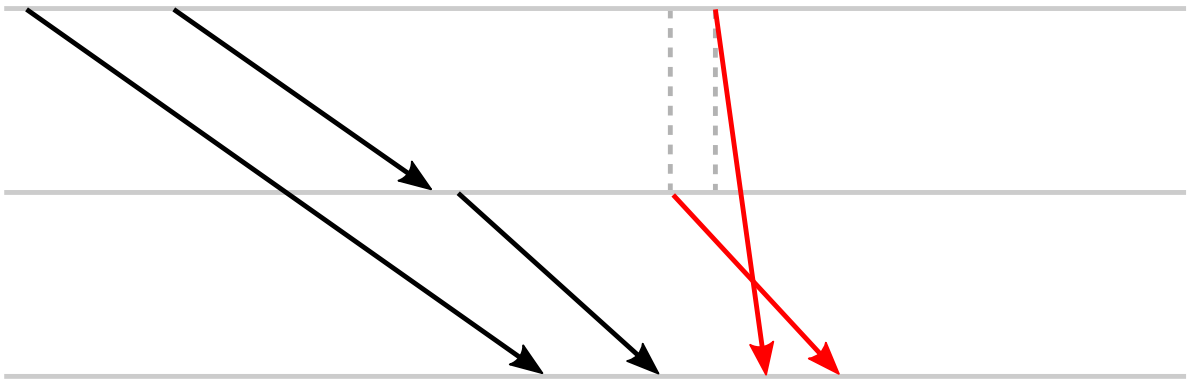
- Pur async.
- **FIFO 11**
- Causal
- FIFO n1
- FIFO nn
- RSC
- Synchrone

Illustration des modèles de communication



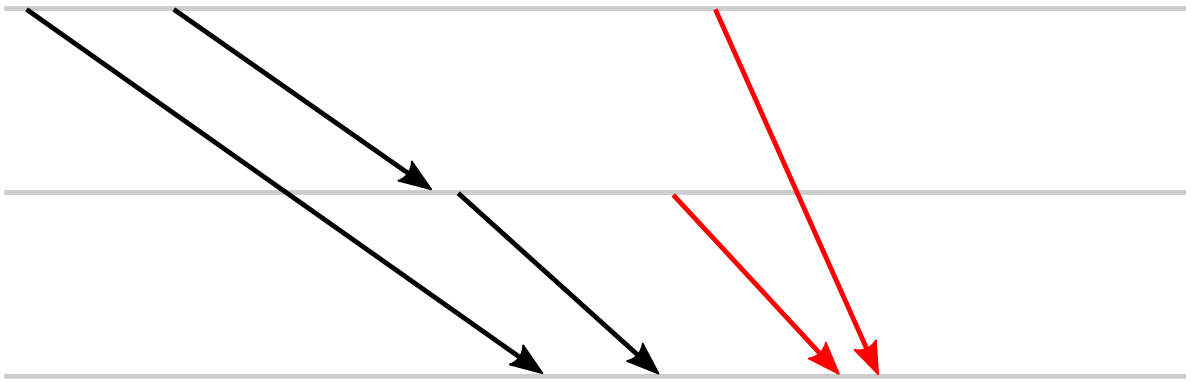
- Pur async.
- FIFO 11
- **Causal**
- FIFO n1
- FIFO nn
- RSC
- Synchrone

Illustration des modèles de communication



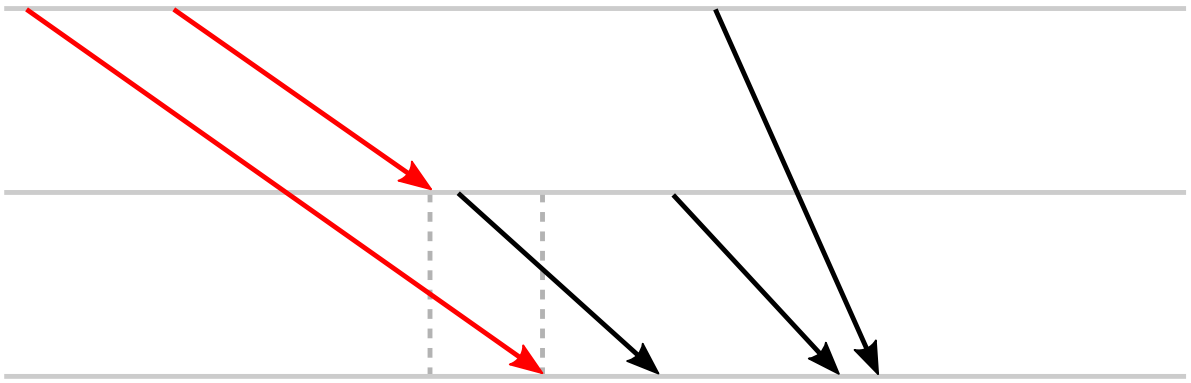
- Pur async.
- FIFO 11
- **Causal**
- FIFO n1
- FIFO nn
- RSC
- Synchrone

Illustration des modèles de communication



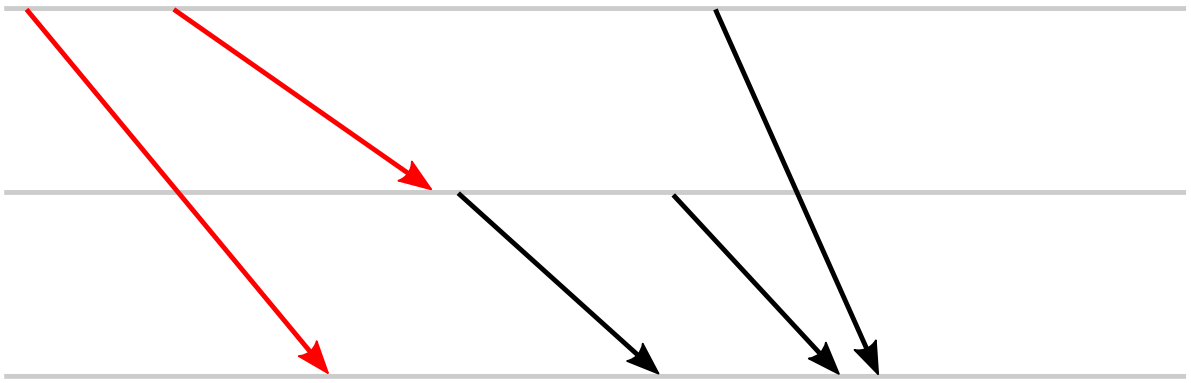
- Pur async.
- FIFO 11
- Causal
- **FIFO n1**
- FIFO nn
- RSC
- Synchrone

Illustration des modèles de communication



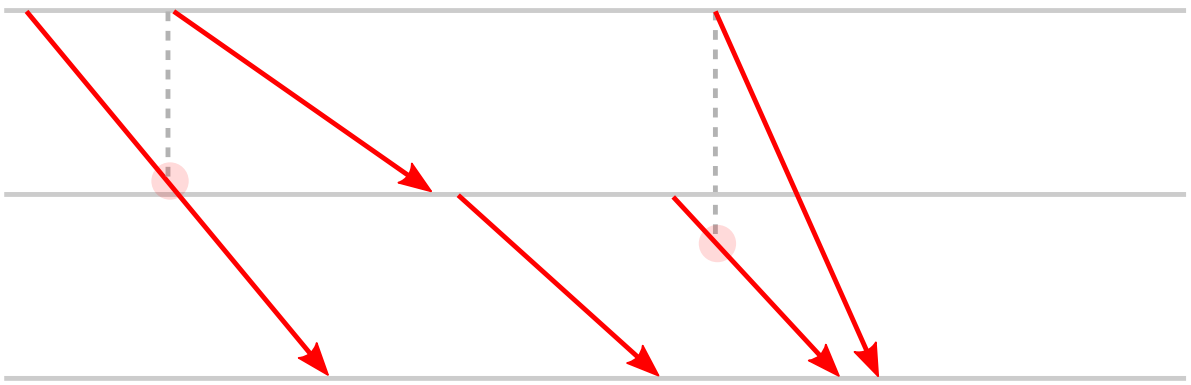
- Pur async.
- FIFO 11
- Causal
- **FIFO n1**
- FIFO nn
- RSC
- Synchrone

Illustration des modèles de communication



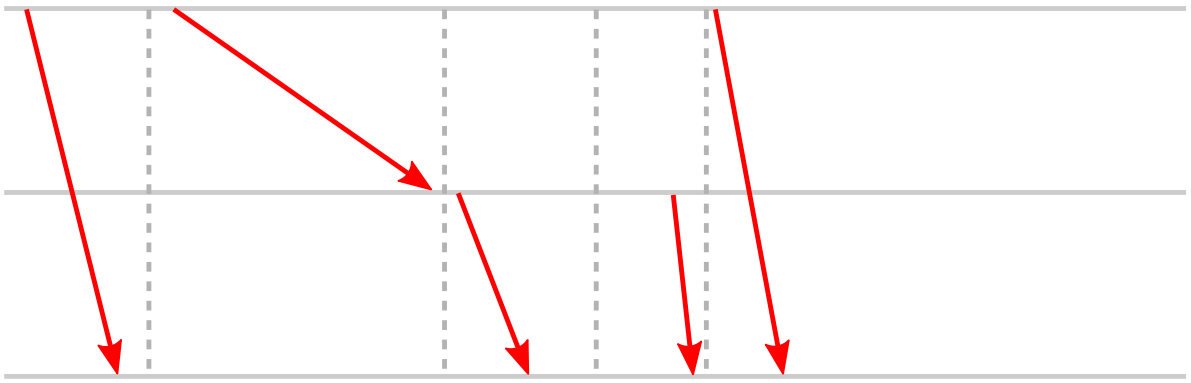
- Pur async.
- FIFO 11
- Causal
- FIFO n1
- **FIFO nn**
- RSC
- Synchrone

Illustration des modèles de communication



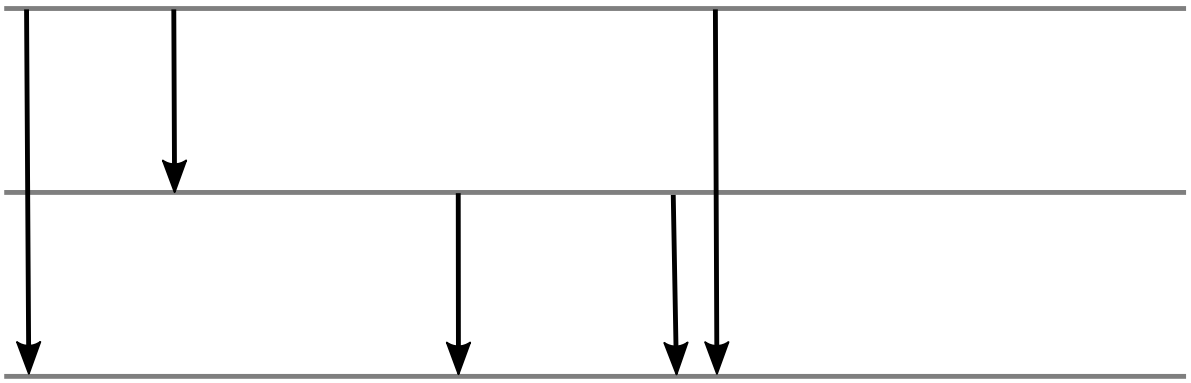
- Pur async.
- FIFO 11
- Causal
- FIFO n1
- **FIFO nn**
- RSC
- Synchrone

Illustration des modèles de communication



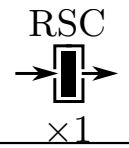
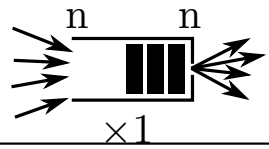
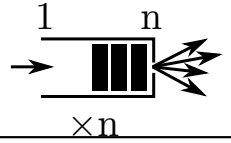
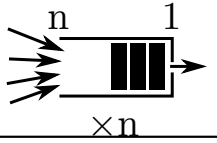
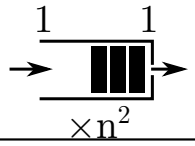
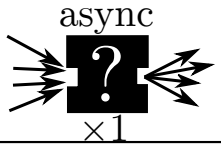
- Pur async.
- FIFO 11
- Causal
- FIFO n1
- FIFO nn
- **RSC**
- Synchrone

Illustration des modèles de communication



- Pur async.
- FIFO 11
- Causal
- FIFO n1
- FIFO nn
- RSC
- **Synchrone**

7 modèles asynchrones

Modèle	Spécification	Intuition	
RSC	Messages délivrés instantanément après leur émission.	Un buffer une place partagé par tous les processus.	
$FIFO_{n-n}$	Messages délivrés dans l'ordre d'émission (ordre global)	Une file unique dans laquelle tous les messages sont déposés et récupérés.	
$FIFO_{1-n}$	Messages d'un même processus délivrés dans l'ordre d'émission.	Une file en sortie de chaque processus où les messages sont déposés.	
$FIFO_{n-1}$	Messages d'un processus donné délivrés dans l'ordre global d'émission.	Une file en entrée de chaque processus où les messages sont instantanément déposés.	
Causal	Messages délivrés en respectant la causalité de leur émission.	Histoire causale ou matrice d'horloges logiques.	
$FIFO_{1-1}$	Messages d'un couple émetteur/récepteur délivrés dans l'ordre d'émission.	Une file entre chaque couple de processus.	
Async	Asynchrone pur. Aucun ordre imposé sur la délivrance des messages.	Un multi-ensemble duquel les messages sont extraits de façon arbitraire.	

Exemples de spécification des modèles de communication

Trois approches pour les spécifications.

Ordre

- Spécification d'un ordre sur la réception des messages en fonction de :
  - leurs ordres d'émission
  - leurs peers d'émission
  - leurs peers de réception
- $FIFO_{1-1}$: deux messages envoyés par le même peer et reçus par le même peer doivent être reçus dans leur ordre d'émission.

Exemples de spécification des modèles de communication

Histoire

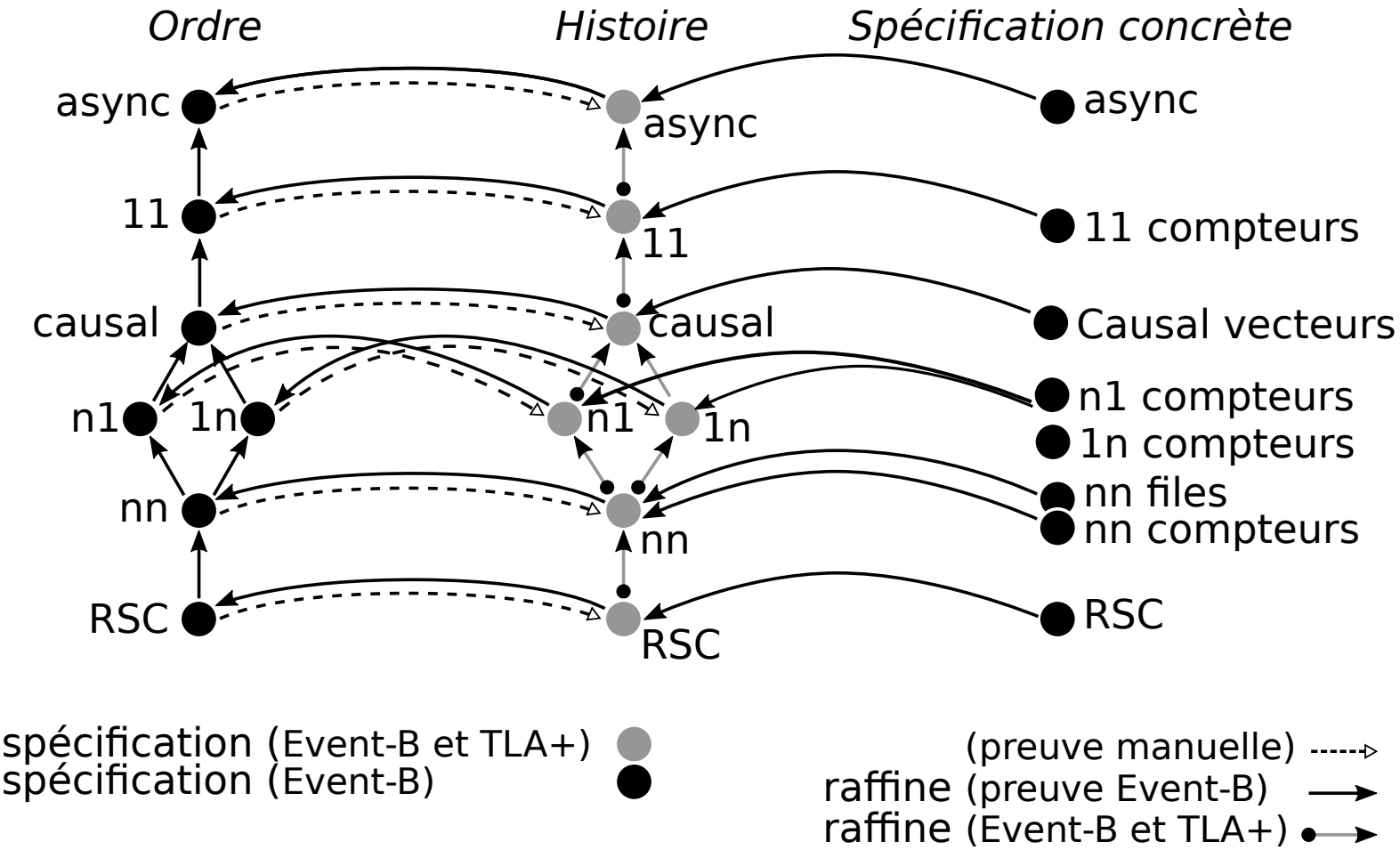
- Spécification unifiée et facilement implémentable
- Une variable *Net* représente l'ensemble des messages en transit
- Les messages en transit transportent l'ensemble de ceux dont ils dépendent (histoire) et leur site d'émission
- La réception d'un message est conditionnée par l'existence, ou non, d'un message en transit qui le bloque
- Pour chaque modèle, la condition de blocage peut dépendre :
  - de l'ordre d'émission : un message est dans l'histoire d'un autre
  - des peers d'émission
  - des peers de réception : peer de réception *intéressé* par les deux messages
- Remarque : dans certains travaux, il y a des destinataires explicites
- $FIFO_{1-1}$: un message en bloque un autre, s'il est dans son histoire (envoyé avant), depuis le même peer et que le site de réception est intéressé par les deux messages (même destinataire).

Exemples de spécification des modèles de communication

Compteur, file, vecteurs, . . .

- Spécification Adhoc, plus réalistes, basées sur des compteurs, files, vecteurs, . . .
- n : nombre de peers
  - 2 compteurs ($FIFO_{n-n}$), $2n$ compteurs ($FIFO_{1-n}$ et $FIFO_{n-1}$), $2n^2$ compteurs ($FIFO_{1-1}$)
  - 1 file ($FIFO_{n-n}$), n files ($FIFO_{1-n}$ et $FIFO_{n-1}$), n^2 files ($FIFO_{1-1}$)
- Compteur : l'autorisation de délivrance d'un message dépend de son rang, de son origine et de sa destination
- File : seuls les messages en tête de files sont autorisés à être délivrés
- $FIFO_{1-1}$: un message ne peut être reçu que si son rang est la valeur du compteur (pour son couple émetteur / récepteur) + 1

Résumé des travaux



Framework

Caractéristiques principales

But : vérifier un système (composé de plusieurs peers) dans la **variété des modèles de com. asynchrone**

Modélisation

- Communication **asynchrone** entre les processus
- Canaux multi-émetteurs / multi-destinataires
- Groupes de canaux avec modèles de communication différents

Méthode

- Spécification des peers et des modèles de com. utilisant des systèmes de transitions
- Vérification de propriétés de compatibilité LTL
- **Model checking**: systèmes à nombre fini d'états
- TLA+

Formalisation

Peers

- Système de transition étiqueté
- Étiquette
  - Événement de com. ($c!/c?$: envoi / réception)
  - Événement interne τ
- Peut être dérivé d'un terme CCS

Modèles de communication

- Système de transition étiqueté
- Étiquette
  - 1 identificateur de processus + 1 événement de com.
  - Événement interne τ

Système

- "produit synchronisé"

FIFO 1-1

MODULE *fifo11*

EXTENDS *Naturals*

VARIABLES *net*, *hg*, *hl*, *hc*, *id*

$Init \triangleq id = 1 \land net = \{\} \land hl = [i \in Peer \mapsto \{\}] \land hc = [i \in Peer \mapsto \{\}] \land hg = \{\}$

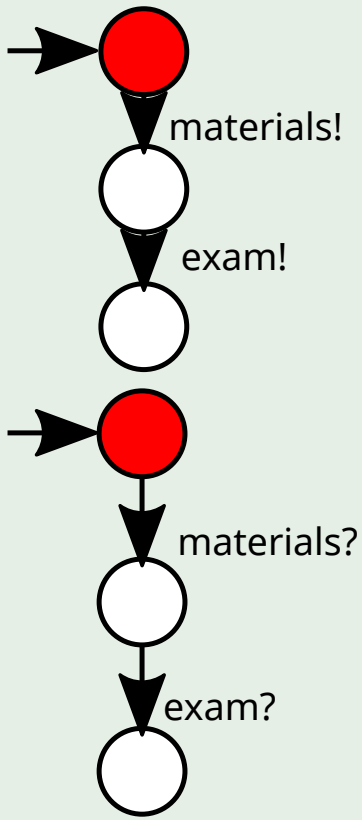
$send(peer, chan) \triangleq$
$\land\ id' = id + 1$
$\land$ LET $m \triangleq \{id\} \times \{chan\} \times \{peer\} \times \{hl[peer]\} \times \{hc[peer]\} \times \{hg\}$ IN
$net' = net \cup m$
$\land\ hl' = [hl$ EXCEPT $![peer] = @ \cup \{id\}]$
$\land\ hc' = [hc$ EXCEPT $![peer] = @ \cup \{id\}]$
$\land\ hg' = hg \cup \{id\}$

$deliveryOk(m, listened) \triangleq \neg(\exists\, m2 \in net :$
$\land\ mp(m) = mp(m2)$ // même peer d'émission
$\land\ mc(m2) \in listened$ // peer de réception intéressé par m2
$\land\ mid(m2) \in mhl(m))$// m2 envoyé avant m

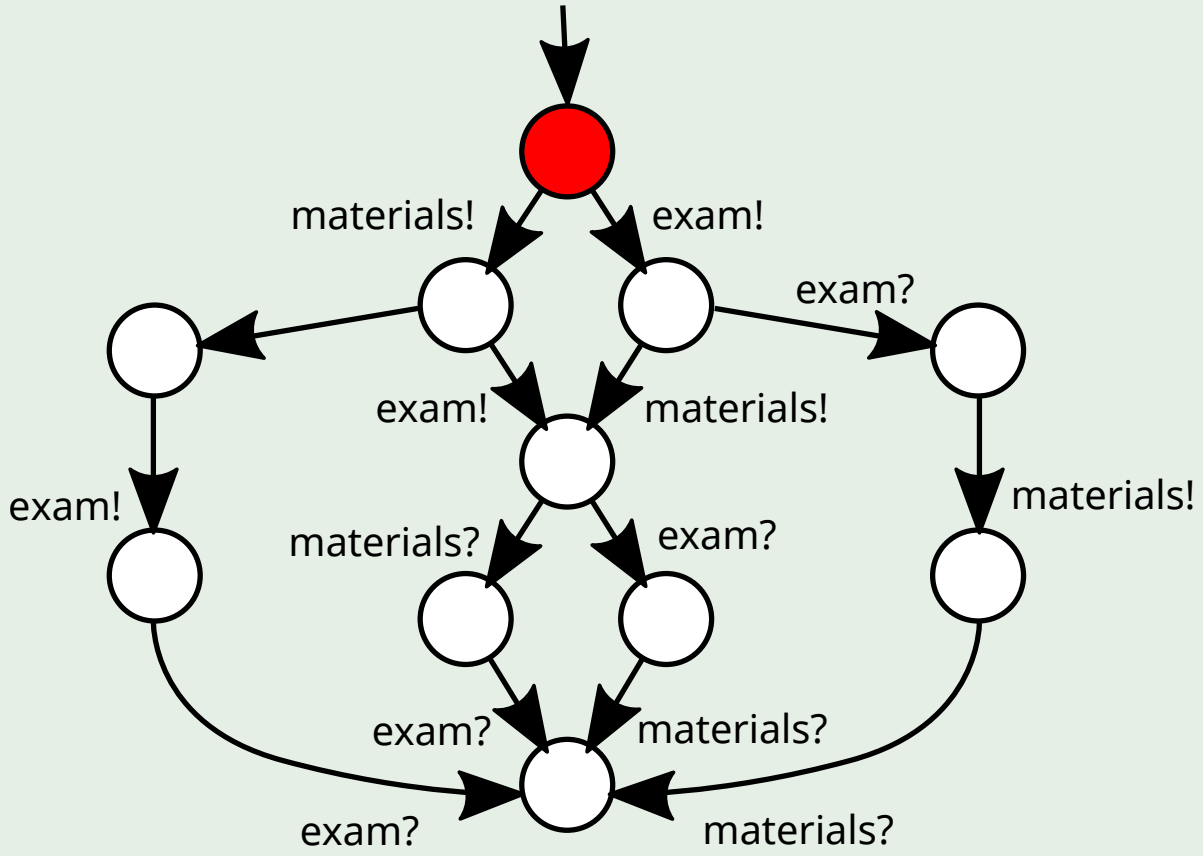
$receive(peer, chan, listened) \triangleq$
$\exists\, m \in net :$
$\land\ chan = mc(m)$
$\land\ deliveryOk(m, listened)$
$\land\ net' = net \setminus \{m\}$
$\land$ UNCHANGED $\langle id, hl, hg \rangle$
$\land\ hc' = [hc$ EXCEPT $![peer] = @ \cup mhc(m) \cup \{mid(m)\}]$

Gestion des réceptions inattendues

Processus

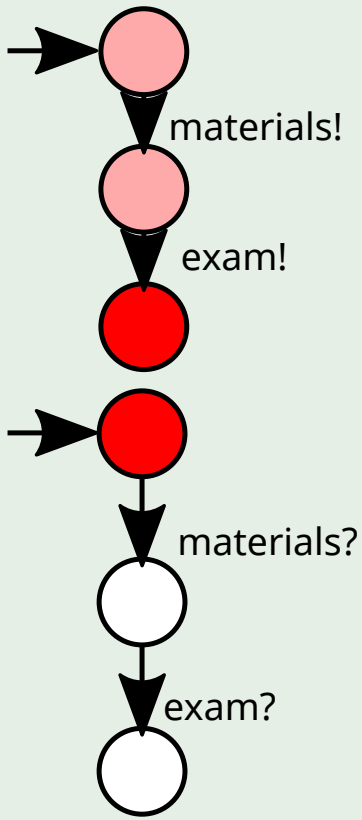


Asynchrone pur

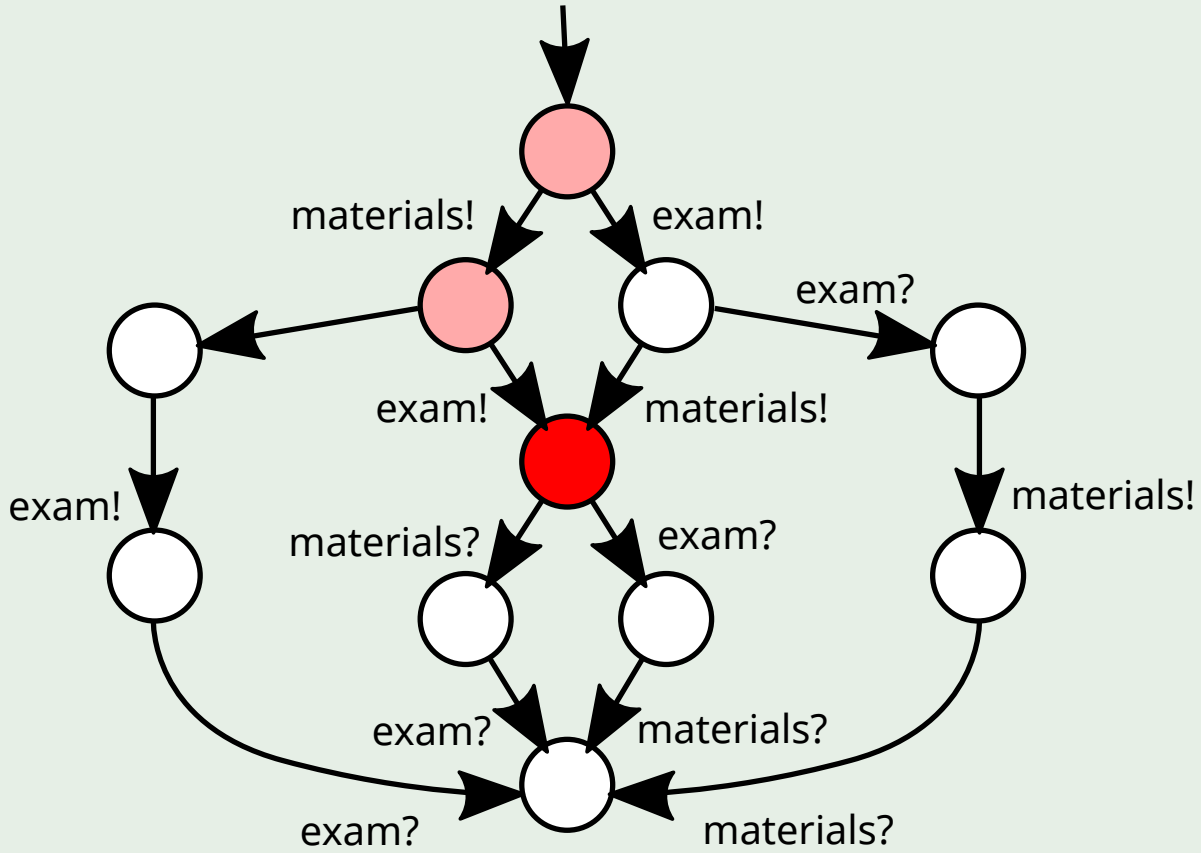


Gestion des réceptions inattendues

Processus

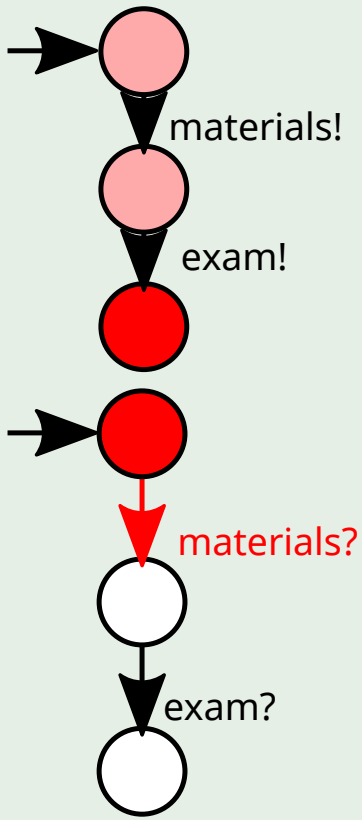


Asynchrone pur

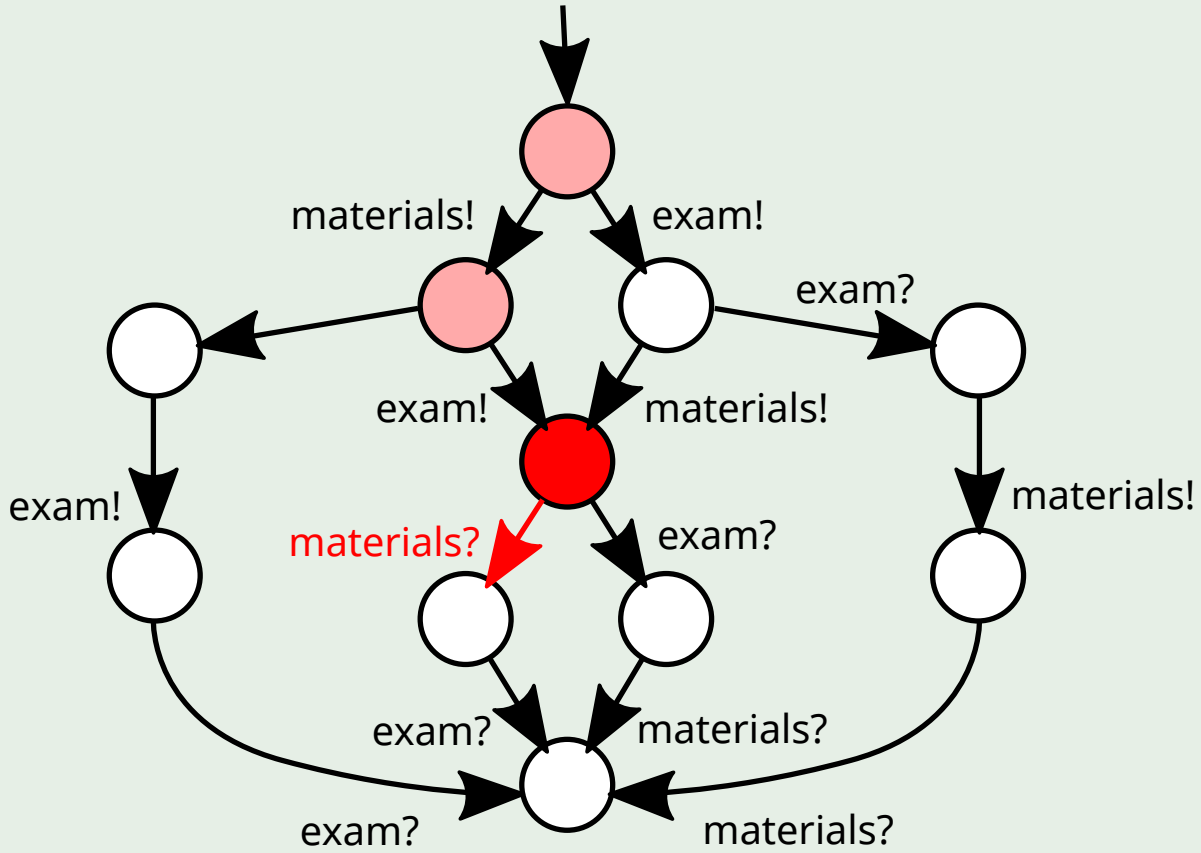


Gestion des réceptions inattendues

Processus

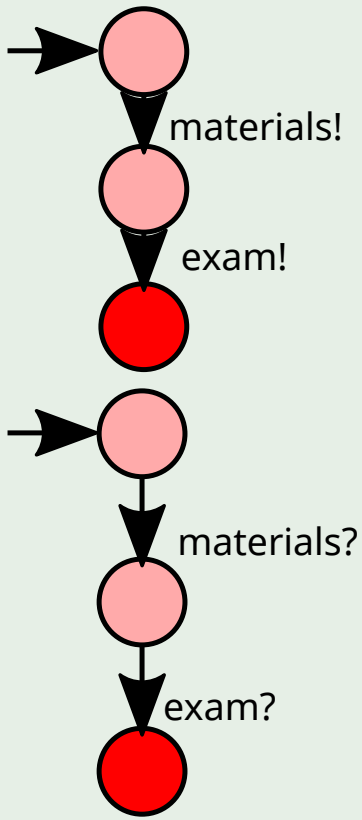


Asynchrone pur

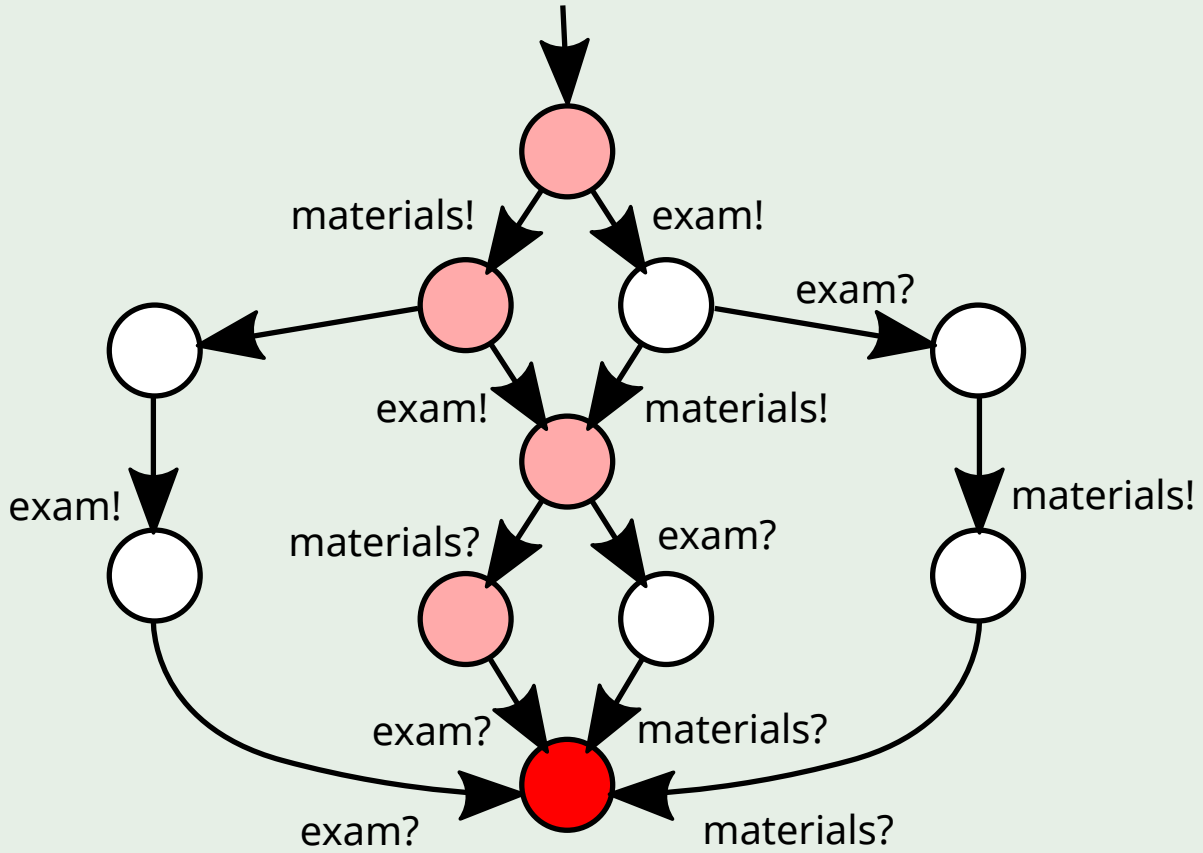


Gestion des réceptions inattendues

Processus

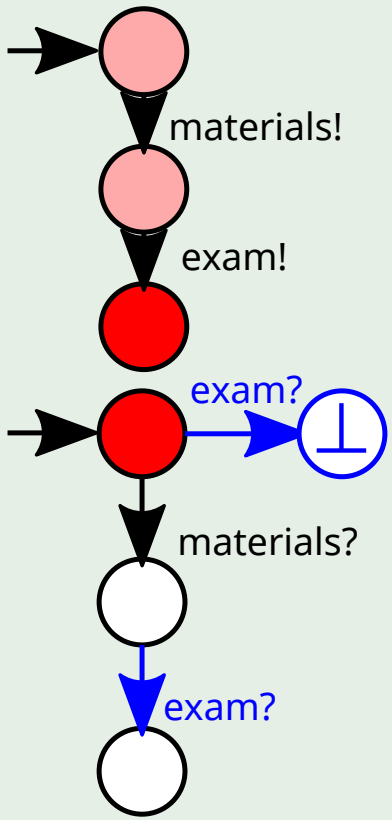


Asynchrone pur

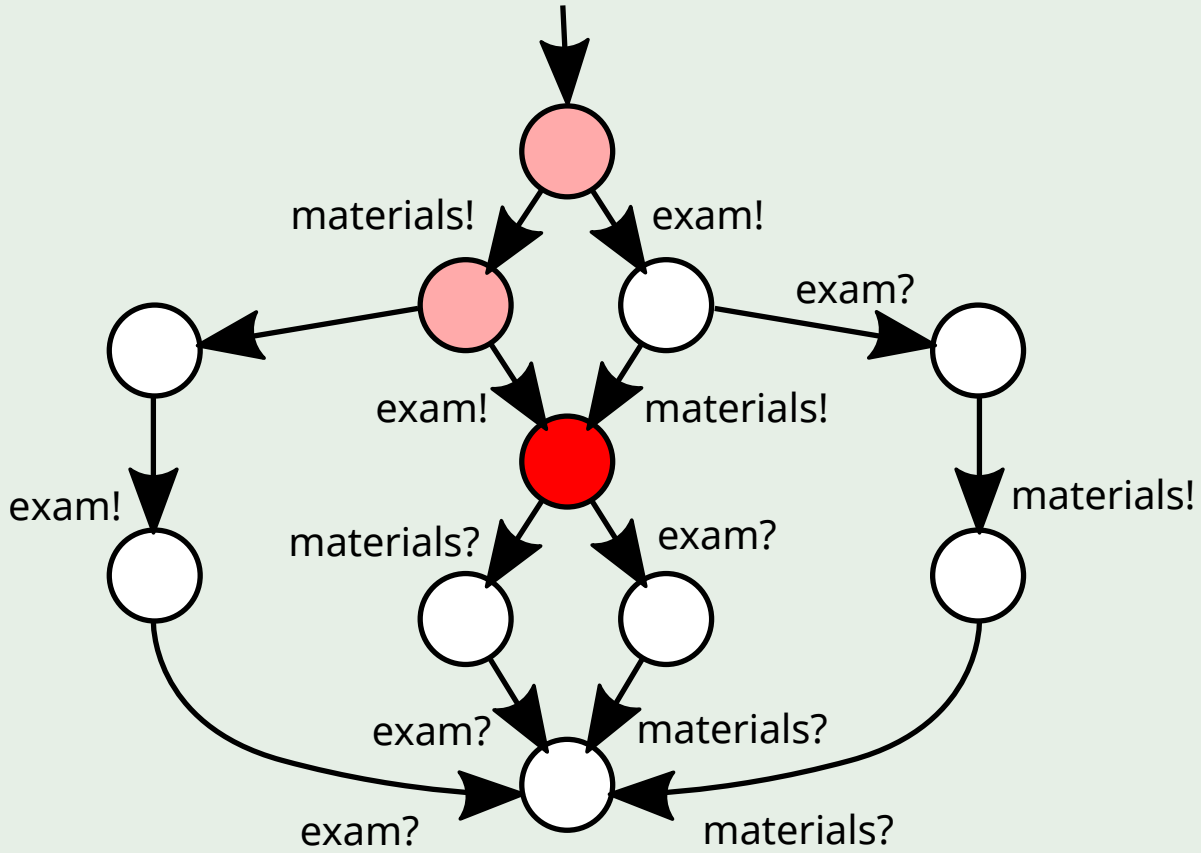


Gestion des réceptions inattendues

Processus

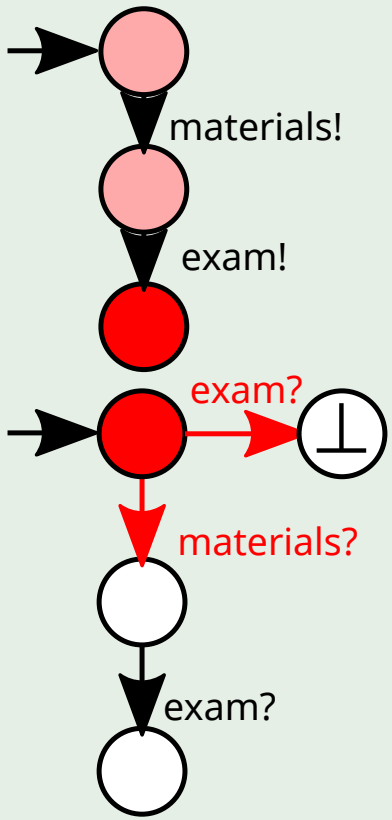


Asynchrone pur

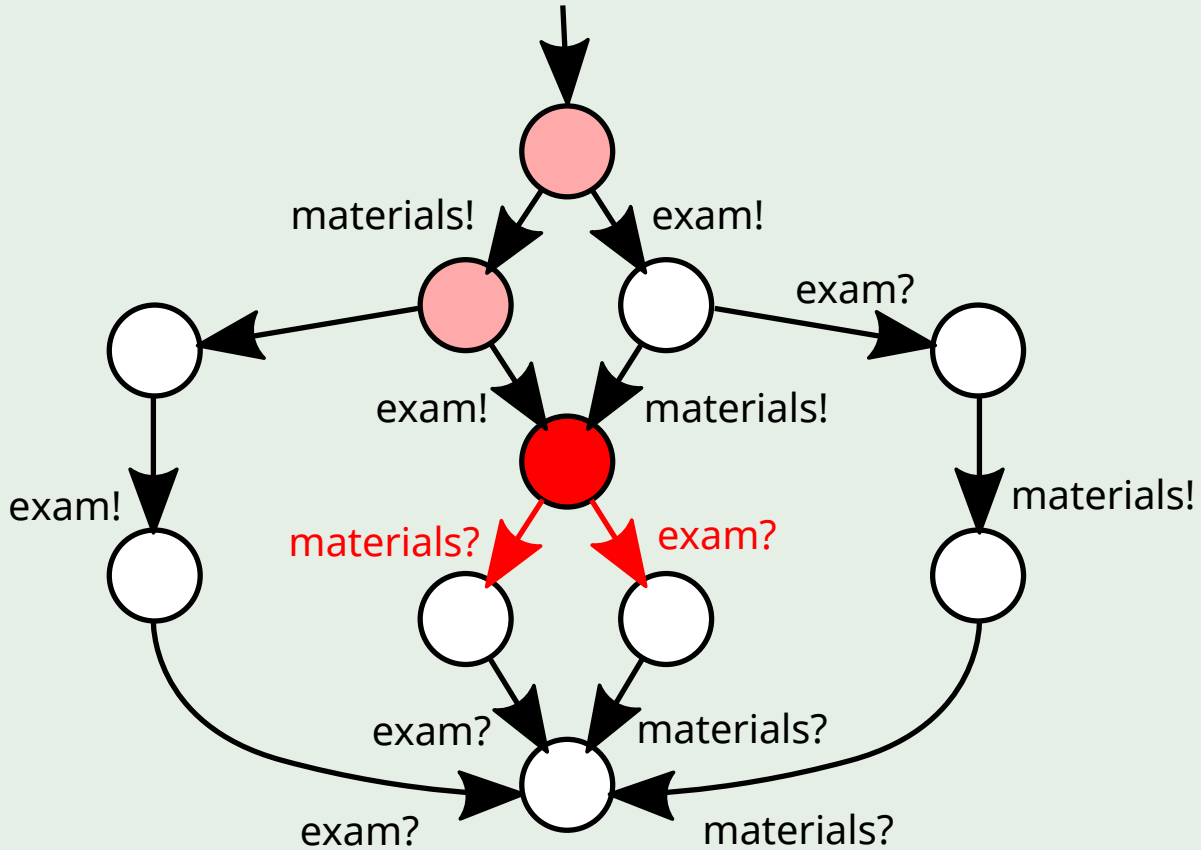


Gestion des réceptions inattendues

Processus

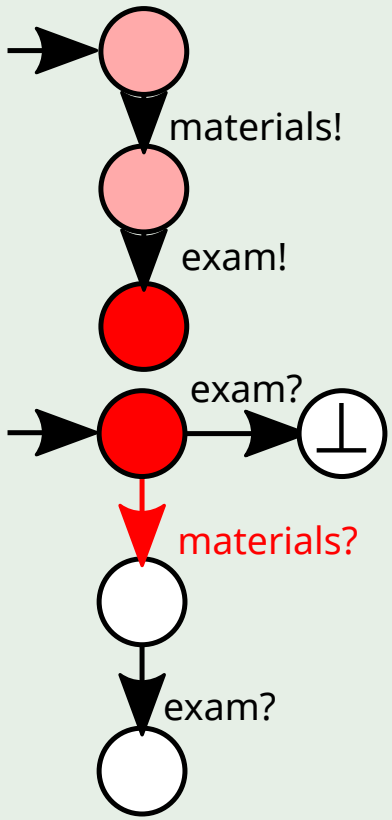


Asynchrone pur

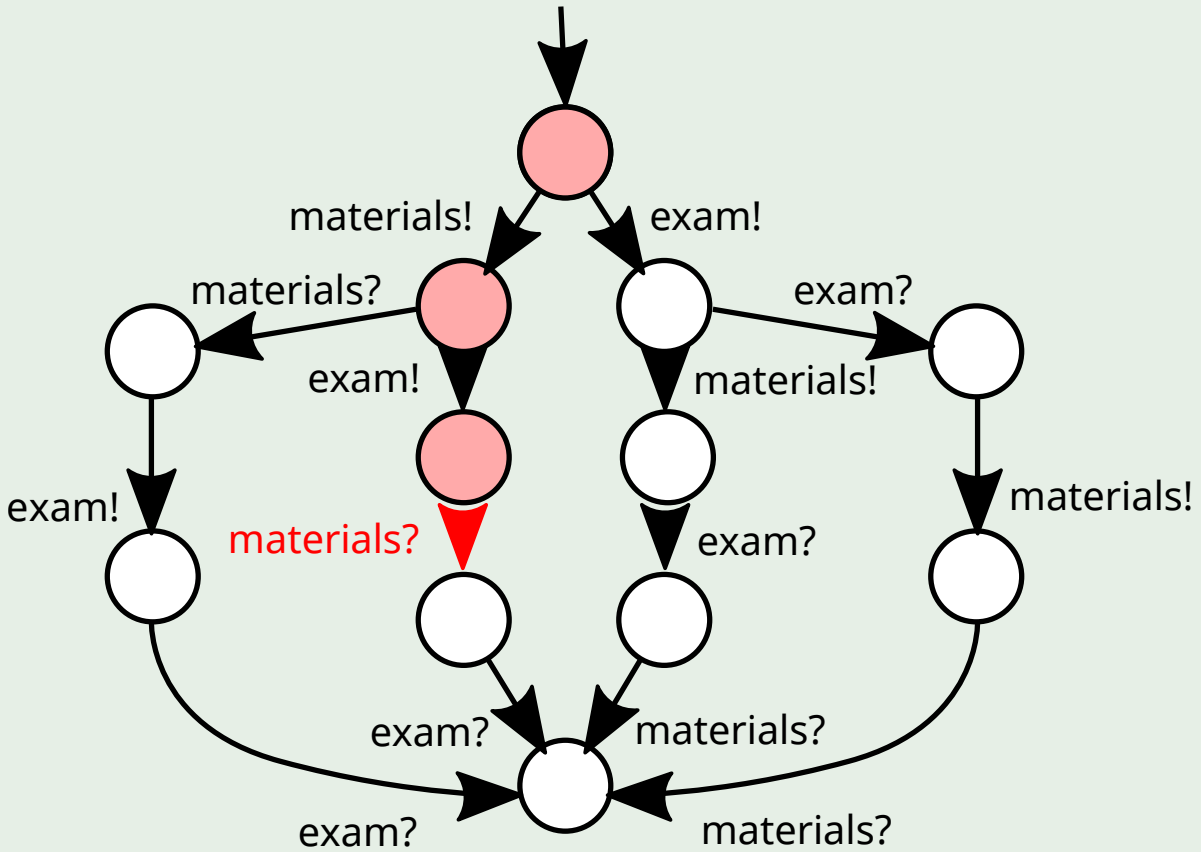


Gestion des réceptions inattendues

Processus



FIFO 1-1



Complétion

Processus complétés et canaux écoutés

Canaux écoutés

Pour un processus dans un état donné, l'ensemble des canaux étiquetant des réceptions futures.

Complétion

- Complétion des états dans lequel une réception est possible
- Nouvelle réception atteignant $\perp$: état d'erreur
- Une transition pour chaque canal écouté

Modèle de communication

- Connaître pour chaque processus les canaux écoutés
- Transition de réception également étiquetée par l'ensemble des canaux écoutés

Correction et complètude

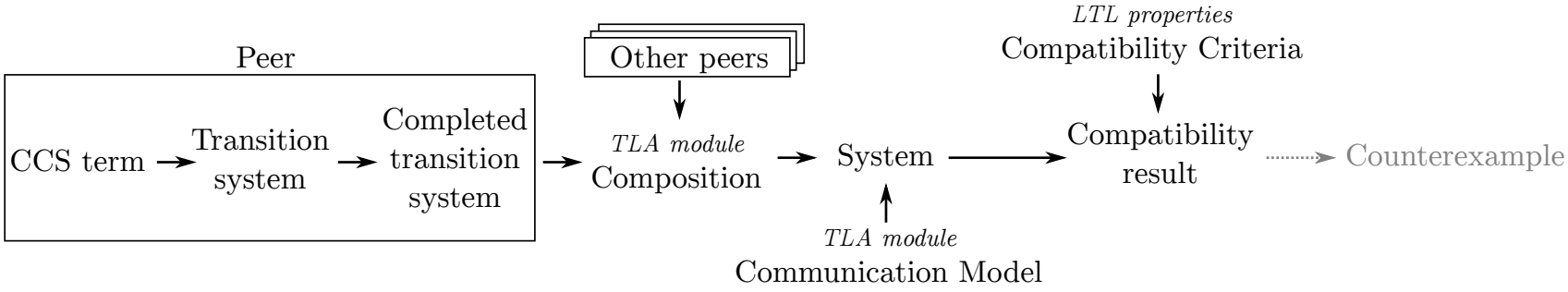
Correction

Toutes les exécutions, générées par le framework pour un système donné et un modèle de communication donné, sont correctes pour ce modèle de communication.
$\Rightarrow$ Preuve (manuelle) nécessitant des propriétés sur les systèmes (décroissance de l'intérêt)

Complètude

Pour un système donné et un modèle de communication donné, le framework génère TOUTES les exécutions correctes.
$\Rightarrow$ Preuve (manuelle) nécessitant des propriétés sur les systèmes (mono-récepteur et tous les messages envoyés sont reçus).

Implantation

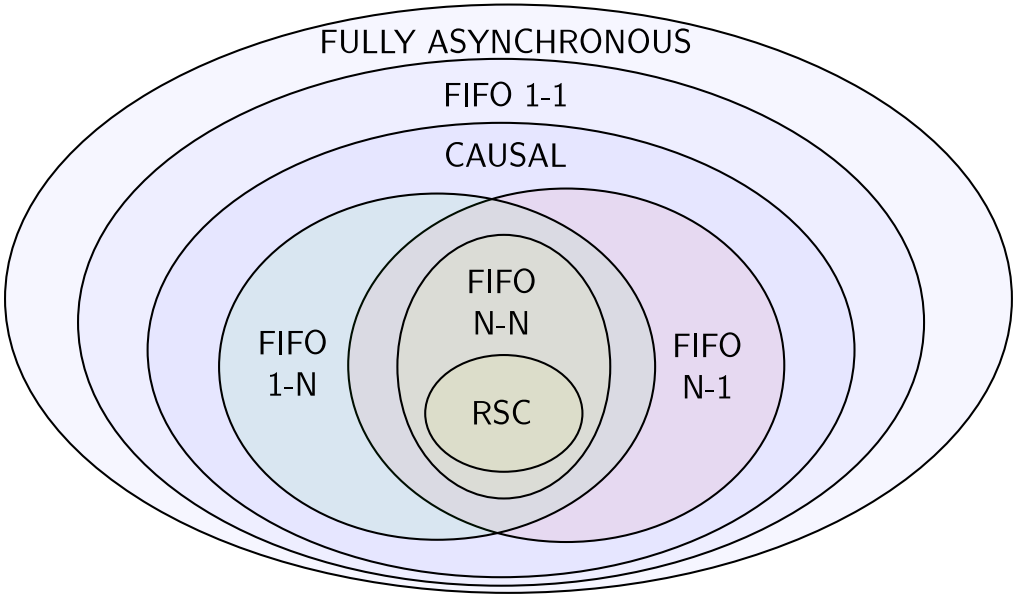


Démonstration

Démonstration.

Hiérarchie des modèles de communication

But: Prouver la hiérarchie de raffinement entre les spécifications, en TLA$^+$, des modèles de communication.



Bénéfice : substituabilité

API des modèles de communication

Actions

- **send**$(sender, channel)$
- **receive**$(receiver, channel, listened)$

Spécification

$$NextSend \triangleq \exists p \in PEER : \exists c \in CHANNEL : send(p, c)$$

$$NextReceive \triangleq \begin{array}{c} \exists p \in PEER : \exists c \in CHANNEL : \\ \exists l \in SUBSET(CHANNEL) : \end{array} receive(p, c, l)$$

$$Next \triangleq NextSend \vee NextReceive$$

$$Spec \triangleq Init \wedge \Box[Next]_{vars}$$

Raffinement en TLA$^+$

Spécification du modèle 1

$Spec_1 \triangleq Init_1 \wedge \Box[Next_1]_{Vars_1}$

Spécification du modèle 2

$Spec_2 \triangleq Init_2 \wedge \Box[Next_2]_{Vars_2}$

Raffinement (safety) en TLA$^+$

Modèle 1 raffine modèle 2 iff $Spec_1 \Rightarrow Spec_2$.

Cas particulier : raffinement des actions

- $\forall p \in PEER : \forall c \in CHANNEL :$
  - $send_1(p, c) \Rightarrow send_2(p, c)$
  - $\forall l \subseteq CHANNEL : receive_1(p, c, l) \Rightarrow receive_2(p, c, l)$
- Équivalent au raffinement de machines en Event-B.

Le TLA$^+$ Proof System repose sur...

Prouveur de logique temporelle (LS4)

$Init_1 \wedge \Box[Next_1]_{Vars_1} \Rightarrow Init_2 \wedge \Box[Next_2]_{Vars_2}$

SMT (CVC3, Z3)

- $Init_1 \Rightarrow Init_2$
- $Next_1 \Rightarrow Next_2$
  - $send_1(p, c) \Rightarrow send_2(p, c)$
    - Invariants de type
    - Invariants communs à tous les modèles
    - Invariants spécifiques à un modèle
  - $receive_1(p, c, l) \Rightarrow receive_2(p, c, l)$
    - Invariants de type
    - Invariants communs à tous les modèles
    - Invariants spécifiques à un modèle

Un mot sur les raffinements Event-B

Raffinements Event-B

- Raffinements faits en Event-B sur des modèles légèrement différents : destinataires explicites
- Pourquoi TLA+ et Event-B? Raison ”historique”
- Tout aurait pu être fait en TLA+ mais assez étonnés par la puissance de Rodin

Modèle de communication applicatif

Modèle de communication applicatif

- Travaux réalisés par un étudiant de master : Nathanaël Sensfelder
- Ordre applicatif au lieu des ordres génériques
- Certains canaux sont plus prioritaires que d'autres
- Un message ne peut être délivré que s'il n'en existe pas un, en transit, plus prioritaire et à destination du même récepteur
- Algorithmes d'inférence de ces priorités pour garantir une propriété LTL quelconque
- Inclus dans le framework
- Peut se combiner avec les modèles précédents

Conclusion

Conclusion

- Formalisation du monde asynchrone dans sa diversité
- Vérification automatique de propriétés LTL
- Preuves des raffinements des modèles

Perspectives PARDI

Modèles de communication

- Diffusion
- Gestion des fautes

Framework

- Gérer *n* peers identiques
- Diffusion
- Gestion des fautes

Preuve

- $FIFO_{n-n} = Causal$
  - *"Les systèmes en échec avec le modèle Causal et le modèle $FIFO_{n-n}$ sont les mêmes."*
  - Caractériser les systèmes et les propriétés pour lesquels c'est vrai
- Modèle minimal