A type system for deadlock freedom in the linear π-calculus

Un système de types pour prévenir les interblocages dans le π-calcul linéaire

Luca Padovani – Département d'Informatique – Turin

Processus communicants

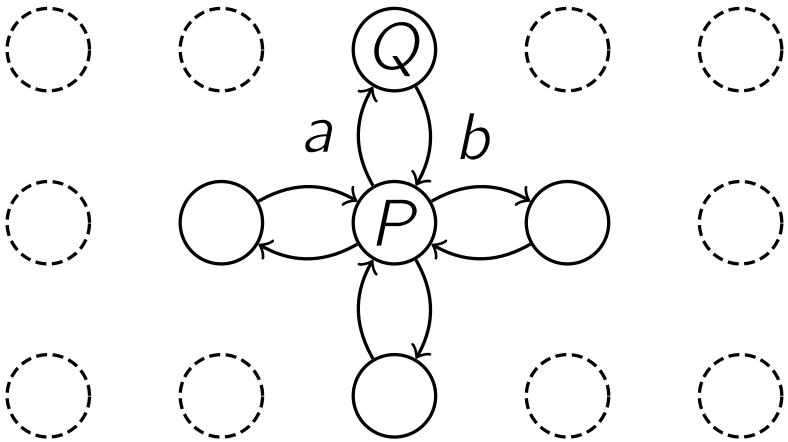
Proprietés

- absence d'erreurs de communication
- **absence d'interblocages**
- vivacité

Méthode

- système de types

Exemple




```
while true do
    send(a, state)
    state′ := receive(b)
    state := update(state, state′)
```

Difficile à traiter par les systèmes de types actuels

- itération
- reseaux cyclique
- plusieurs canaux

Sommaire

Sommaire

Qu'est-ce que c'est le π-calcul linéaire ?

$$id \equiv \lambda x.x$$

$$*id?(x, c).c!\langle x \rangle$$

- Kobayashi, Pierce, Turner, **Linearity and the pi-calculus**, TOPLAS 1999

Qu'est-ce que c'est le π-calcul linéaire ?

$$id \equiv \lambda x.x$$

canal illimité

$$*id?(x, c).c!\langle x \rangle$$

canal linéaire

- Kobayashi, Pierce, Turner, **Linearity and the pi-calculus**, TOPLAS 1999

Pour quoi le π-calcul linéaire ?

"communications on linear channels [...] account for up to 50% of communications in a typical program"

Kobayashi, Pierce, Turner 1999

Pour quoi le π-calcul linéaire ?

> *"communications on linear channels [...] account for up to 50% of communications in a typical program"*
>
> Kobayashi, Pierce, Turner 1999

Canaux "linéarisés" vs canaux linéaires

$a!\langle 3\rangle.a!\langle 4\rangle\ldots$ $\qquad$ $(\nu b)a!\langle 3, b\rangle.(\nu c)b!\langle 4, c\rangle\ldots$

- Kobayashi, **Type systems for concurrent programs**, 2002
- Dardha, Giachino, Sangiorgi, **Session types revisited**, 2012

Formulation du problème

Théorème (Kobayashi, Pierce, Turner, 1999)

Dans un processus bien typé, chaque canal linéaire est utilisé pour une communication au plus une fois

Formulation du problème

Théorème (Kobayashi, Pierce, Turner, 1999)

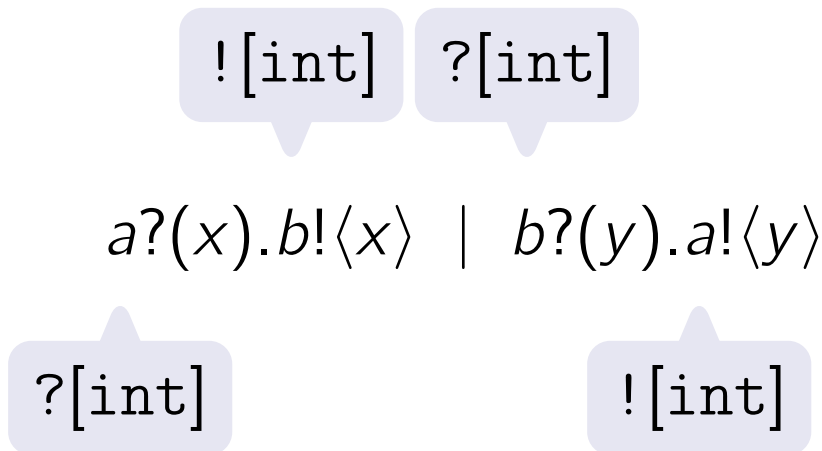
Dans un processus bien typé, chaque canal linéaire est utilisé pour une communication au plus une fois

$$a?(x).b!\langle x\rangle \mid b?(y).a!\langle y\rangle$$

Formulation du problème

Théorème (Kobayashi, Pierce, Turner, 1999)

Dans un processus bien typé, chaque canal linéaire est utilisé pour une communication au plus une fois

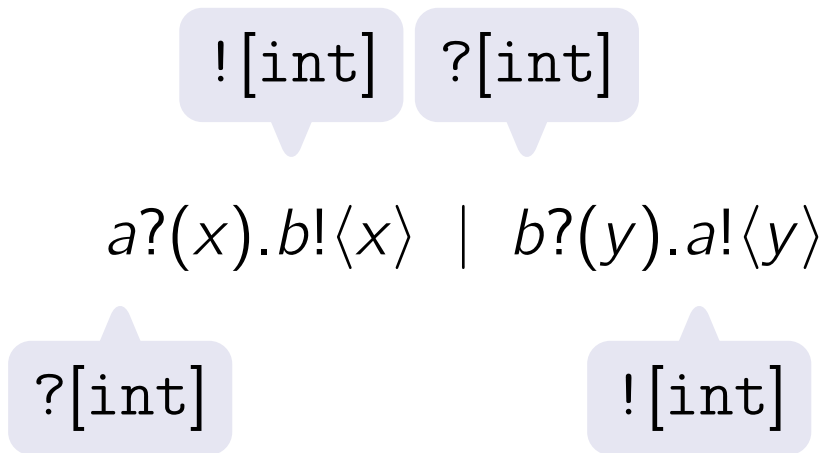


- ☺ le processus est bien typé (a et b sont linéaires)
- ☹ aucune communication possible

Formulation du problème

Théorème (Kobayashi, Pierce, Turner, 1999)

Dans un processus bien typé, chaque canal linéaire est utilisé pour une communication ~~au plus une fois~~ **exactement une fois**



- ☺ le processus est bien typé (a et b sont linéaires)
- ☹ aucune communication possible

Sommaire

Stratégie

1. associer à chaque canal un numéro
   $\Rightarrow$ le numéro fait partie du type du canal

2. verifier que les canaux soient utilisés en ordre
   $\Rightarrow$ selon le numéro associé

$$a\ \ ?(x).b\ !\langle x\rangle\ \ |\ \ b\ ?(y).a\ \ !\langle y\rangle$$

Stratégie

1. associer à chaque canal un numéro
   $\Rightarrow$ le numéro fait partie du type du canal

2. verifier que les canaux soient utilisés en ordre
   $\Rightarrow$ selon le numéro associé

$$a^m?(x).b^n!\langle x\rangle \;\mid\; b^n?(y).a^m!\langle y\rangle$$

Stratégie

1. associer à chaque canal un numéro
   $\Rightarrow$ le numéro fait partie du type du canal

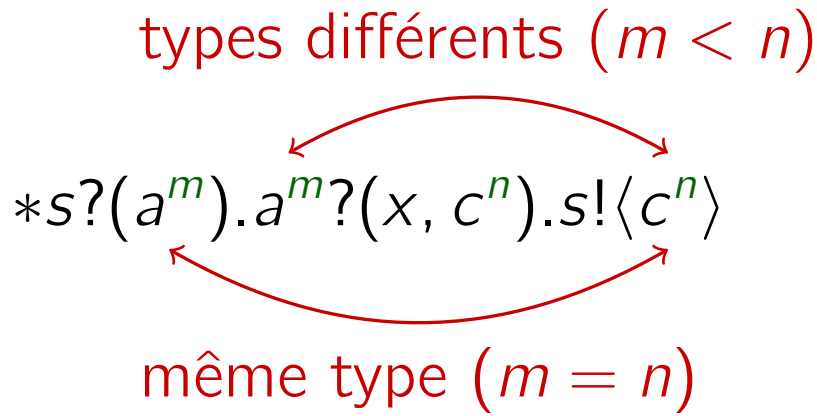
2. verifier que les canaux soient utilisés en ordre
   $\Rightarrow$ selon le numéro associé

$$a^{m}?(x).b^{n}!\langle x\rangle \;\mid\; b^{n}?(y).a^{m}!\langle y\rangle \qquad (a^{m} < b^{n},\ b^{n} < a^{m})$$

La plupart des processus récursifs sont mal typés

$$*s?(a^m).a^m?(x, c^n).s!\langle c^n \rangle$$

La plupart des processus récursifs sont mal typés



Renoncer à la linearité

$\mu\alpha.?[\texttt{int}].\alpha$

$$*s?(a).a?(x).s!\langle a\rangle$$

$\mu\alpha.?[\texttt{int}].\alpha$

Usages

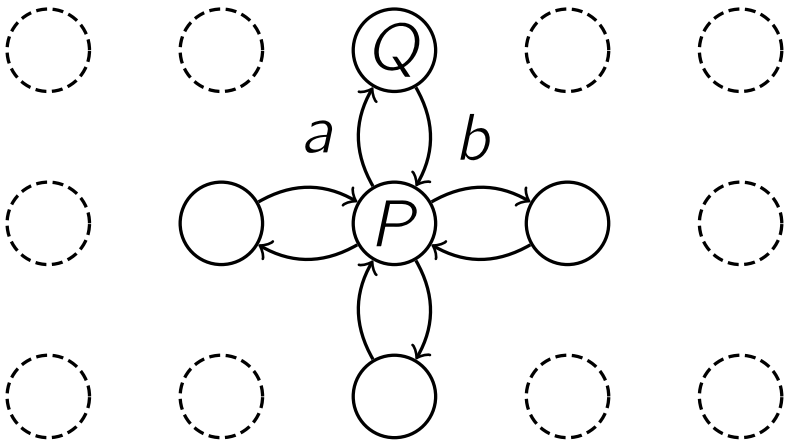
- Kobayashi, *Information & Computation*, 2002
- Kobayashi, *Acta Informatica*, 2005
- Kobayashi, CONCUR 2006

Types de session

- Dezani *et al.* CONCUR 2008 + MSCS (à paraître)
- Padovani, PLACES 2013
- Vasconcelos et Vieira, COORDINATION 2013
- ...

Renoncer à la linearité

- ☺ processus récursifs bien typés. . .
- ☹ . . .s'ils utilisent seulement **un** canal




```
while
   send
   state′ :=      ve(b)
   state := update(state, state′)
```

Mal typé!

Tous les processus recursifs qui utilisent deux (ou plus) canaux differents sont mal typés

Typage avec polymorphisme

$$a?(b^{7}).b?(x).c^{8}!\langle x\rangle \qquad a?(b^{7},c^{8}).b?(x).c!\langle x\rangle$$

Typage avec polymorphisme

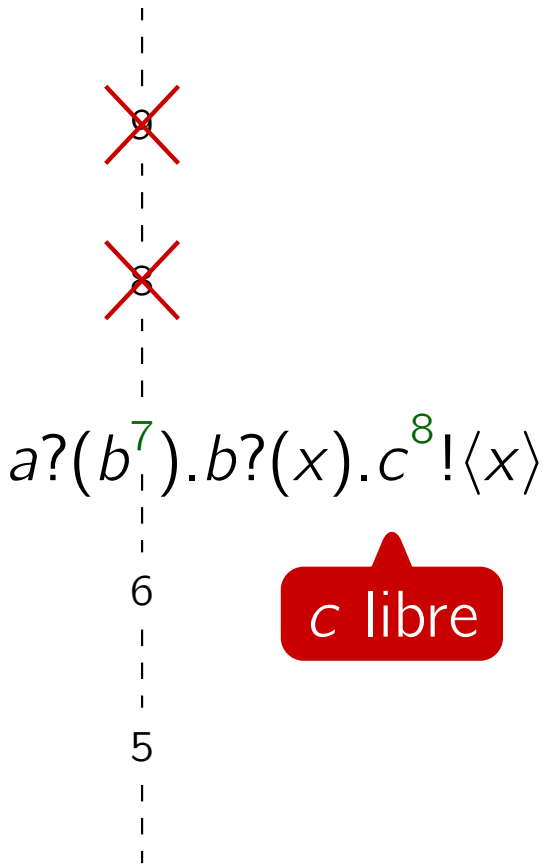
$a?(b^{7}).b?(x).c^{8}!\langle x\rangle$

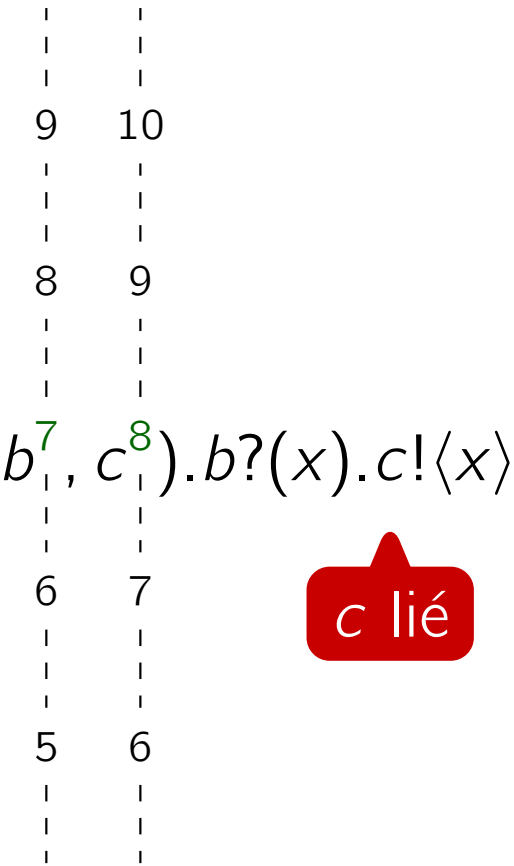
c libre

$a?(b^{7}, c^{8}).b?(x).c!\langle x\rangle$

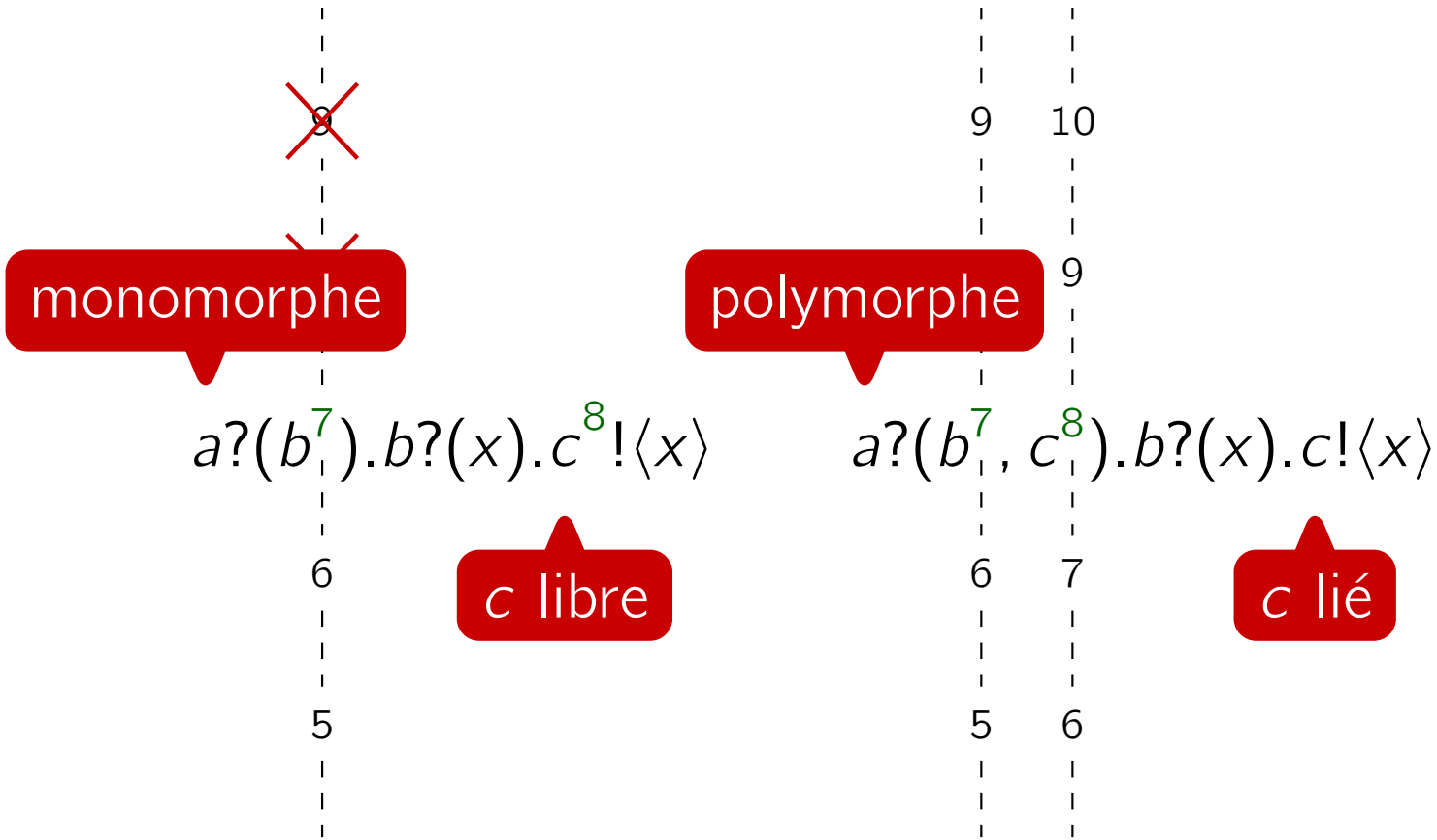
c lié

Typage avec polymorphisme





Typage avec polymorphisme



Quand est-ce qu'un canal est polymorphe ?

Il faut regarder la continuation après la réception sur le canal

$$a?(x).P$$

Quand est-ce qu'un canal est polymorphe ?

Il faut regarder la continuation après la réception sur le canal

$a?(x).P$ quelque canal linéaire libre $\Rightarrow$ a monomorphe

Quand est-ce qu'un canal est polymorphe ?

Il faut regarder la continuation après la réception sur le canal

$a?(x).P$ aucun canal linéaire libre $\Rightarrow$ a polymorphe

Quand est-ce qu'un canal est polymorphe ?

Il faut regarder la continuation après la réception sur le canal

$a?(x).P$ aucun canal linéaire libre $\Rightarrow$ a polymorphe

Typage du π-calcul linéaire (Kobayashi, Pierce, Sangiorgi, 1999)

$$\frac{\vdash P \qquad \text{aucun canal linéaire libre en } P}{\vdash *a?(x).P}$$

Quand est-ce qu'un canal est polymorphe ?

Il faut regarder la continuation après la réception sur le canal

$a?(x).P$ aucun canal linéaire libre $\Rightarrow$ a polymorphe

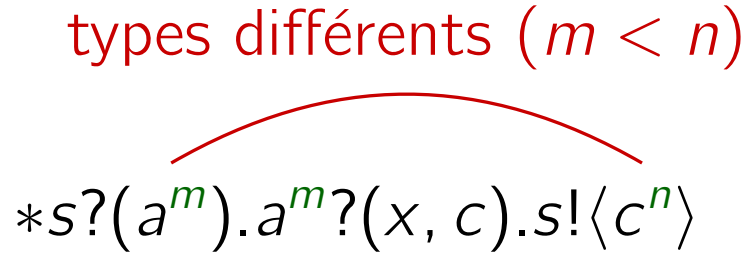
Typage du π-calcul linéaire (Kobayashi, Pierce, Sangiorgi, 1999)

$$\frac{\vdash P \qquad \text{aucun canal linéaire libre en } P}{\vdash *a?(x).P}$$

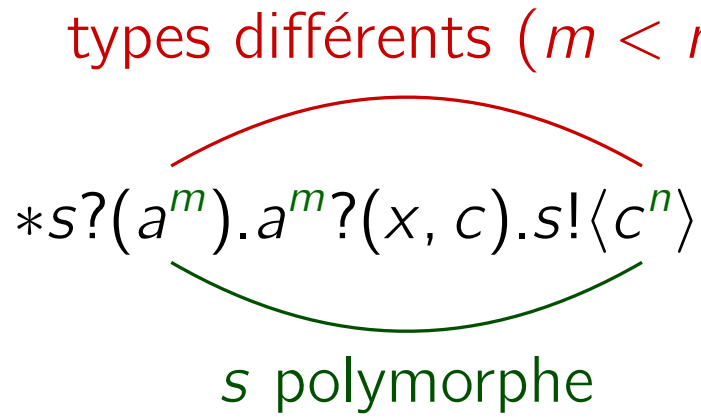
Conclusion

Tous les canaux utilisés par processus repliqués sont polymorphes

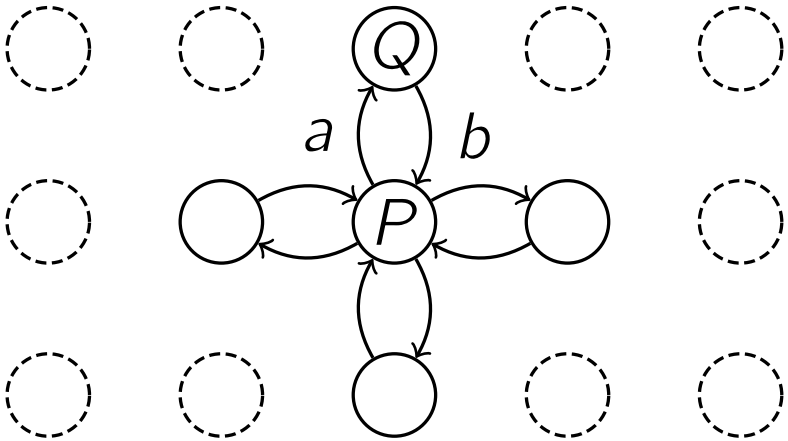
Le récepteur récursif



Le récepteur récursif



Exemple




```
while
   send
   state′ :=  ve(b)
   state := update(state, state′)
```

Bien typé !

Les résultats

Théorème (preservation du typage)

Si P est bien typé et $P \to Q$, alors Q est bien typé aussi

Théorème (absence d'interblocages)

Si P est bien typé et $P \not\to$, alors P n'a pas des canaux linéaires

Sommaire

Expressivité

☹ Problème ouvert

- caracterisation des processus bien typés

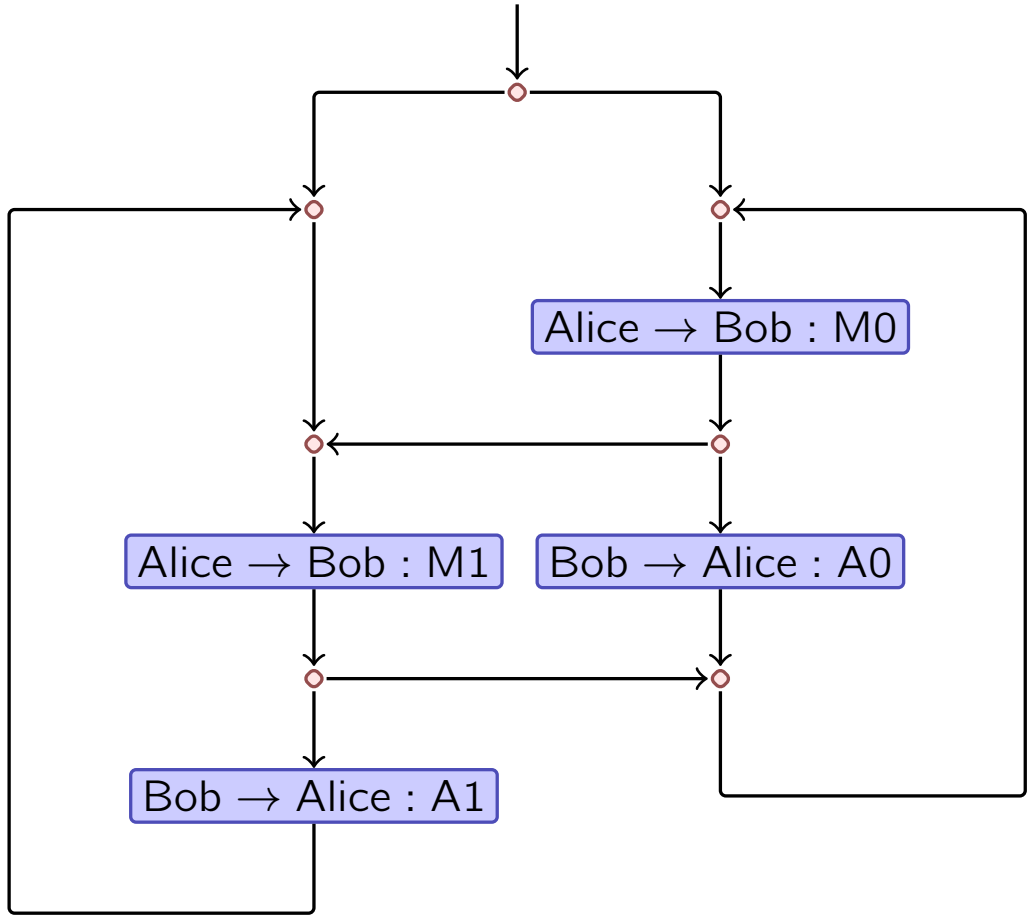
Expressivité

☹ Problème ouvert

- caracterisation des processus bien typés

☺ Types “globaux”

- Honda, Yoshida, Carbone, POPL 2008
- Deniélou, Yoshida, ESOP 2012



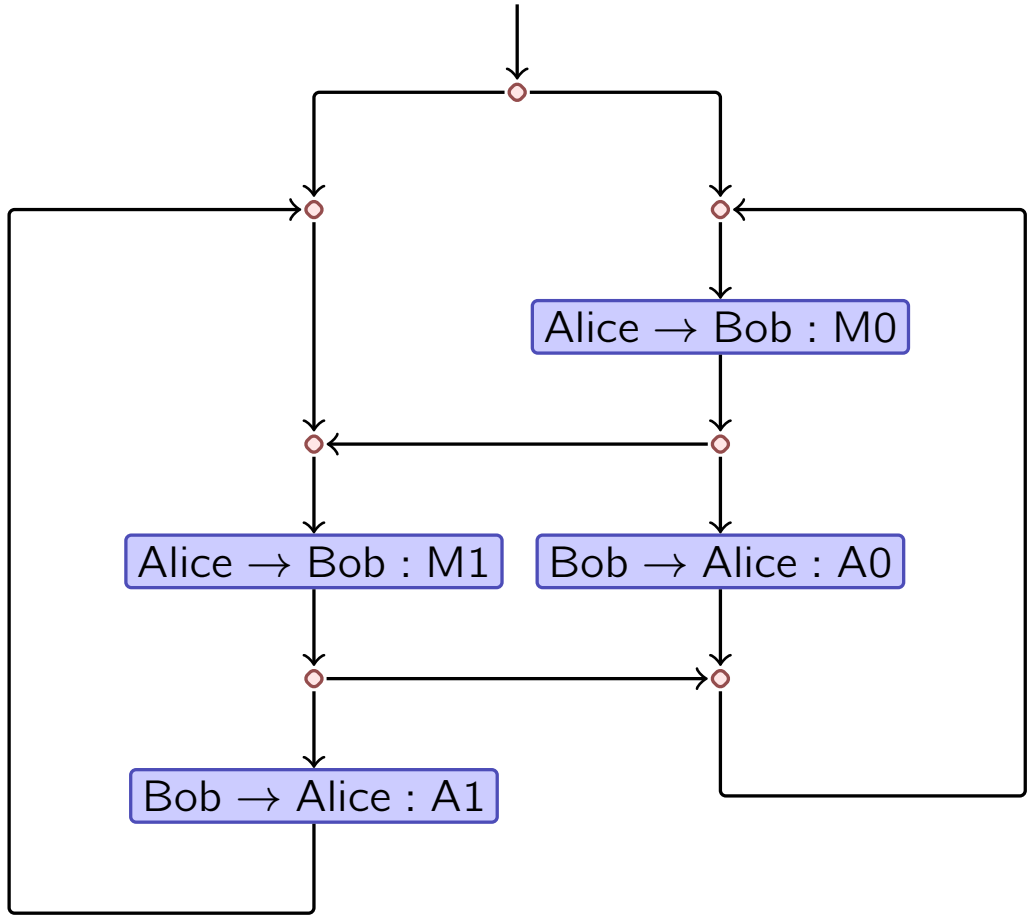
Expressivité

☹ Problème ouvert

- caracterisation des processus bien typés

☺ Types “globaux”

- Honda, Yoshida, Carbone, POPL 2008
- Deniélou, Yoshida, ESOP 2012



Théorème

Si un protocole d’interaction entre deux (ou plusieurs) participants peut être décrit par un type global, alors il est réalisable par un processus bien typé

De l'absence d'interblocages à la vivacité

Théorème (absence d'interblocages)

Si P est bien typé et $P \not\rightarrow$, alors P n'a pas des canaux linéaires

De l'absence d'interblocages à la vivacité

Théorème (absence d'interblocages)

Si P est bien typé et $P \not\rightarrow$, alors P n'a pas des canaux linéaires

Il faut renforcer le système de types

- assurer l'**arrêt** du processus
  Kobayashi, Sangiorgi, TOPLAS 2010
- limiter la **mobilité** des canaux
  Kobayashi, *Information & Computation* 2002
  Padovani, CSL-LICS 2014 (à paraître)

De l'absence d'interblocages à la vivacité

Théorème (absence d'interblocages)

Si P est bien typé et $P \not\rightarrow$, alors P n'a pas des canaux linéaires

Il faut renforcer le système de types

- assurer l'**arrêt** du processus
  Kobayashi, Sangiorgi, TOPLAS 2010
- limiter la **mobilité** des canaux
  Kobayashi, *Information & Computation* 2002
  Padovani, CSL-LICS 2014 (à paraître)

Théorème (vivacité)

Si P est bien typé et il a une émission/réception en attente sur un canal linéaire, alors tôt ou tard cette operation se termine

Inférence (en cours, avec Tzu-Chun Chen et Andrea Tosatto)

☹ Les types sont très détaillés

$$*bob?(x^0, y^2).\\ (\nu a^3 b^5)\big(x^0?(\bar{x}^1).(\bar{x}^1!\langle a^3\rangle \mid y^2?(\bar{y}^3).(\bar{y}^3!\langle b^5\rangle \mid bob!\langle a^3, b^5\rangle))\big)$$

☺ Il y a un algorithme d'inférence

1. les numéros deviennent des variables entières
2. les regles de typage produisent des contraintes linéaires
3. si les contraintes ont une solution, le processus est bien typé

Sommaire

En conclusion

Système de types

- pour prévenir interblocages
- dans processus communicants
- sur canaux linéaires

Comparaison avec les systèmes de types actuels

- plus simple
- plus expressif

Bibliographie

- Padovani, **Deadlock and lock freedom in the linear π-calculus**, CSL-LICS 2014 (à paraître)

`http://www.di.unito.it/~padovani`