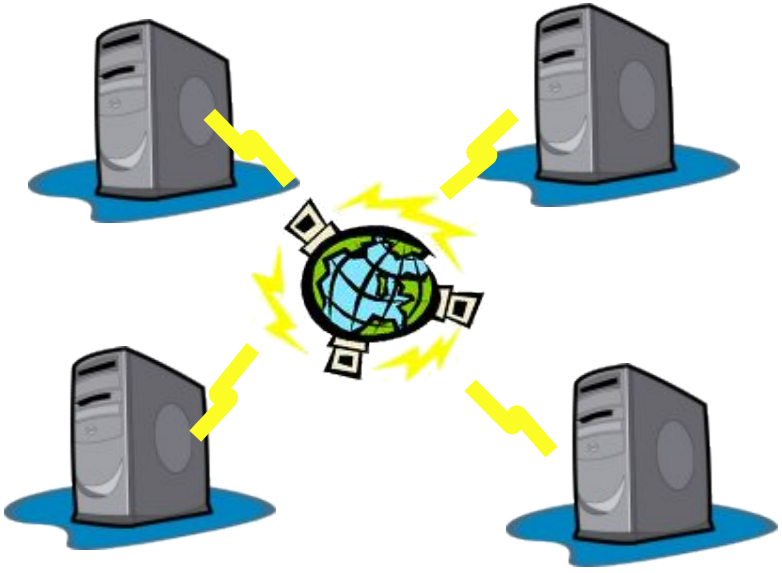
Analyse statique d'un calcul d'acteurs par interprétation abstraite

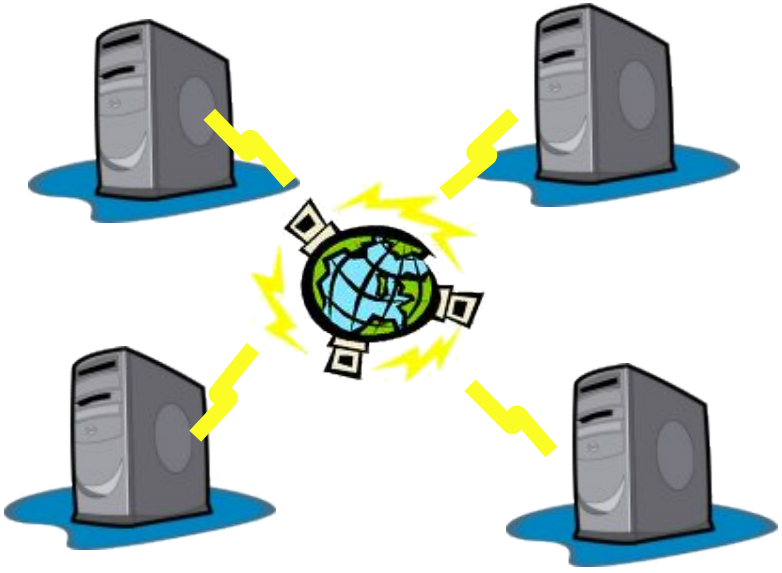
Pierre-Loïc Garoche
sous la direction de Patrick Sallé

IRIT, INPT, Université de Toulouse, France

10 juin 2008

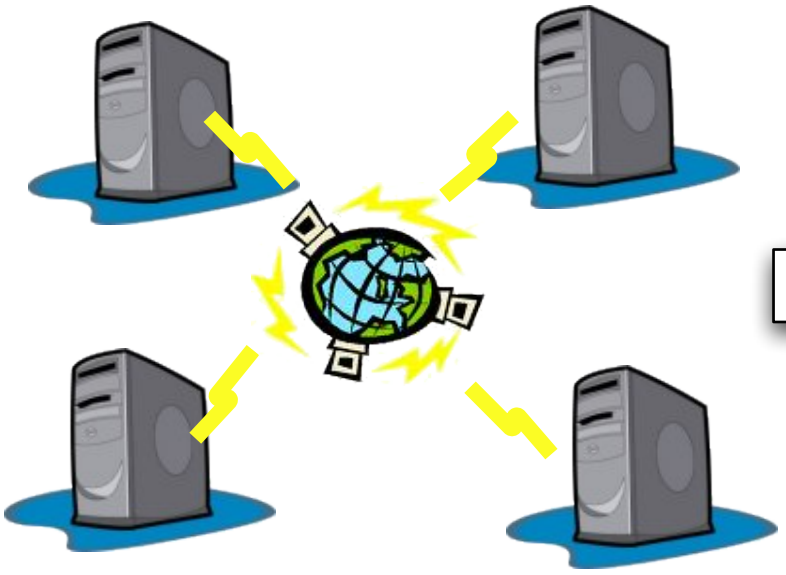


Systèmes concurrents et communiquants

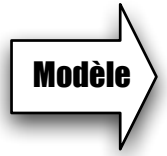


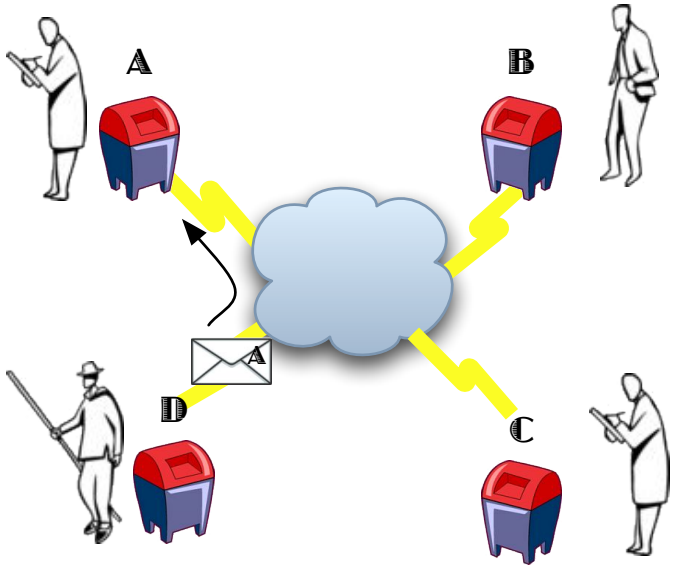
Systèmes concurrents et communiquants

Calculs de processus (π, Join, ...)



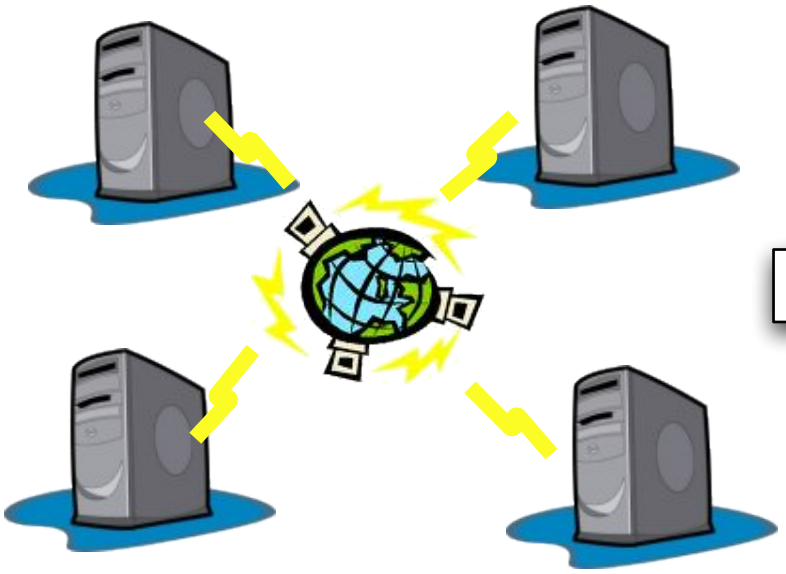
Systèmes concurrents
et communiquants



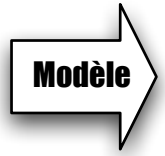


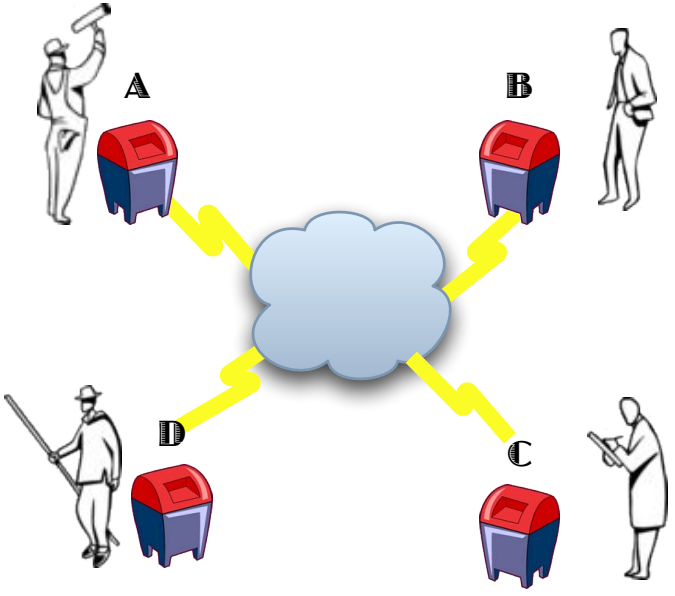
Le modèle des acteurs

Calculs de processus (π, Join, ...)



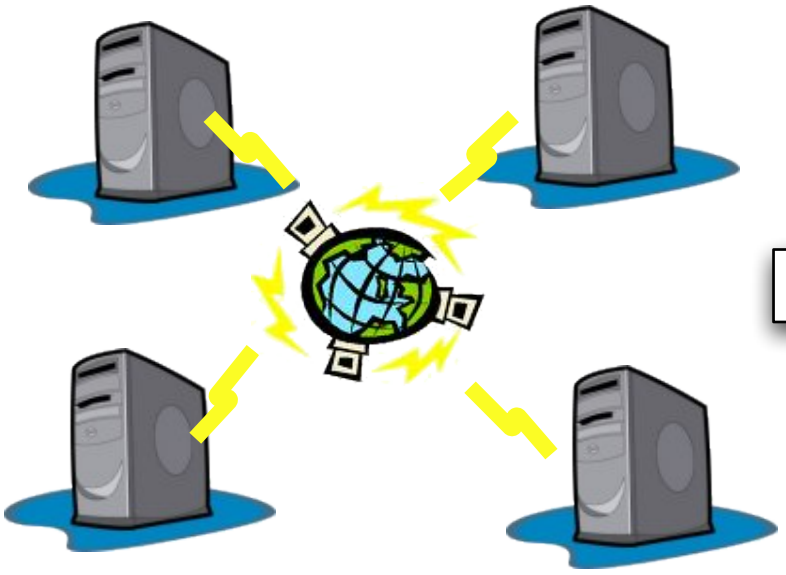
Systèmes concurrents
et communiquants



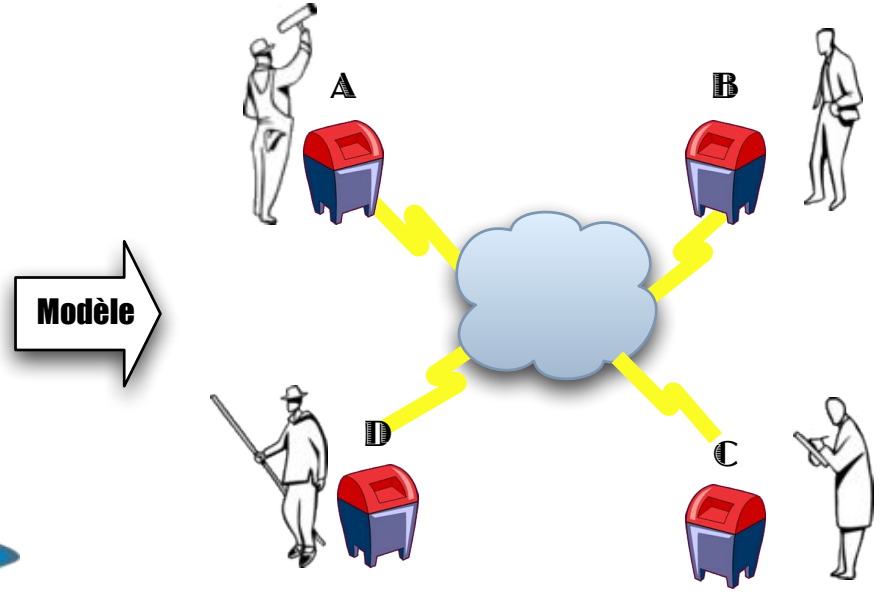


Le modèle des acteurs

Calculs de processus (π, Join, ...)



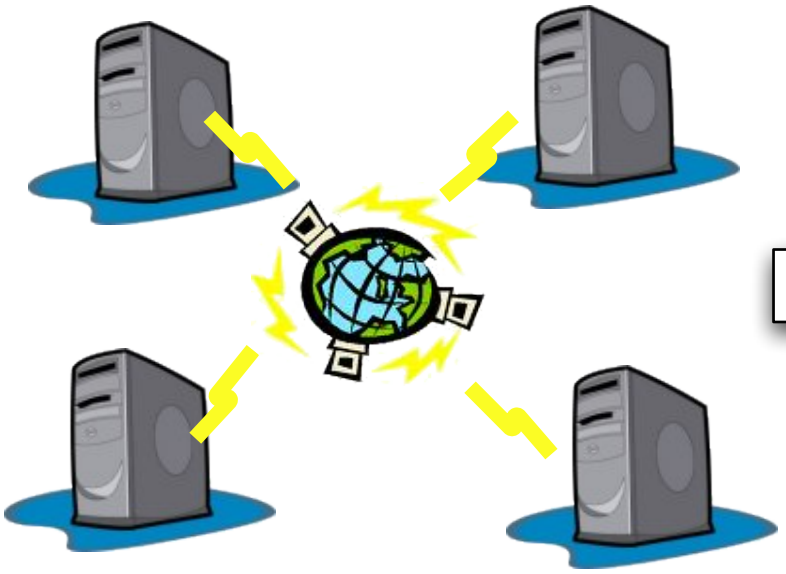
Systèmes concurrents
et communiquants



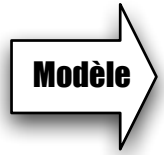
Le modèle des acteurs

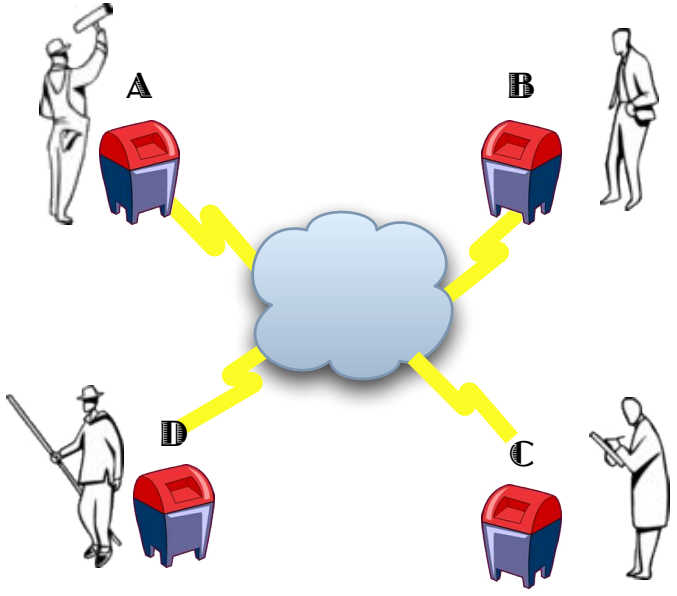
Calculs de processus (π, Join, ...)

Calcul d'Acteurs Primitifs (CAP)



Systèmes concurrents
et communiquants





Le modèle des acteurs

Calculs de processus (π, Join, ...) Calcul d'Acteurs Primitifs (CAP)

Travaux de thèse :

- Proposition d'un cadre d'analyse de CAP par interprétation abstraite
  - extension des travaux de Feret sur le π
- Définition de domaines abstraits pour vérifier les propriétés :
  1. linéarité (adresses vues comme des ressources)
  2. absence des messages orphelins
- Réalisation d'un prototype PACSA : un interprète abstrait pour CAP

Aujourd'hui : analyse liée à l'absence de messages orphelins

Plan de la présentation

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

CAP : un calcul d'acteurs primitifs (Colaço '96)

Syntaxe et sémantique basées sur le π-calcul asynchrone et les objets primitifs d'Abadi & Cardelli

- Opérateurs usuels du π-calcul
  - composition parallèle $||$
  - restriction d'un nom (adresse) $\nu a, C$
- 2 entités :
  - l'acteur $a \rhd [\underbrace{m_1(\tilde{x_1}) = \zeta(e,s)\, C_1 \cdots m_n(\tilde{x_n}) = \zeta(e,s)\, C_n}_{\text{comportement}}]$

    $a \rhd x$
  - le message $a \lhd m(\tilde{t})$
- Opérateur ζ (objets primitifs d'Abadi & Cardelli)

$$a \rhd [\underbrace{m(x) = \zeta(e,s)\, C \cdots}]$$

La sémantique de CAP

Définie par une relation de congruence, des règles de compatibilité et une règle de transition :

$$\frac{B = [m_i(\widetilde{x_i}) = \zeta(e_i, s_i)\, C_i{}^{i=1,\ldots,n}] \quad \begin{cases} m = m_k, \\ |\widetilde{y}| = |\widetilde{x_k}|, \\ k \in [\![1; n]\!] \end{cases}}{a \rhd B \;||\; a \lhd m(\widetilde{y}) \rightarrow C_k[e_k \leftarrow a, s_k \leftarrow B, \widetilde{x_k} \leftarrow \widetilde{y}]}$$

- Réception atomique
- Mécanisme de passage de comportement :
  - \+ facilité pour décrire des systèmes complexes
  - – rend les analyses difficiles (arité, orphelins. . .)
- Hypothèse de linéarité

Exemple : la cellule linéaire

$$\begin{array}{rl} \nu a, b, c & a \rhd [put(x) = \zeta(e, s)(\\ & \qquad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \;||\; y \lhd rep(x))])] \\ || & a \lhd put(b) \;||\; a \lhd get(c) \end{array}$$

Exemple : la cellule linéaire

$$\begin{array}{rl} \nu a, b, c & a \rhd [put(x) = \zeta(e, s)(\\ & \qquad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \;||\; y \lhd rep(x))])] \\ || & a \lhd put(b) \;||\; a \lhd get(c) \end{array}$$

l'acteur sur a reçoit le message put

Exemple : la cellule linéaire

$$\begin{array}{lll} \nu a, b, c & a \rhd [put(x) = \zeta(e, s)(& \\ & \qquad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \;||\; y \lhd rep(x))])] \\ \;||\; & a \lhd put(b) \;||\; a \lhd get(c) & \end{array}$$

l'acteur sur a reçoit le message put

$$\begin{array}{lll} \nu a, b, c & a \rhd [get(y) = \zeta(e', s')(& \\ & \qquad e' \rhd [put(x) = \zeta(e, s)(& \\ & \qquad\qquad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \;||\; y \lhd rep(x))])] & \\ & \qquad ||\; y \lhd rep(b))] & \\ \;||\; & a \lhd get(c) & \end{array}$$

Exemple : la cellule linéaire

$$\begin{array}{rl} \nu a, b, c & a \rhd [put(x) = \zeta(e, s)(\\ & \qquad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \mid\mid y \lhd rep(x))])] \\ \mid\mid & a \lhd put(b) \mid\mid a \lhd get(c) \end{array}$$

l'acteur sur a reçoit le message *put*

$$\begin{array}{rl} \nu a, b, c & a \rhd [get(y) = \zeta(e', s')(\\ & \qquad e' \rhd [put(x) = \zeta(e, s)(\\ & \qquad\qquad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \mid\mid y \lhd rep(x))])] \\ & \qquad \mid\mid y \lhd rep(b))] \\ \mid\mid & a \lhd get(c) \end{array}$$

puis le message *get*

Exemple : la cellule linéaire

$$\begin{array}{lll} \nu a, b, c & a \rhd [put(x) = \zeta(e, s)(& \\ & \quad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \;||\; y \lhd rep(x))])] \\ \;||\; & a \lhd put(b) \;||\; a \lhd get(c) & \end{array}$$

l'acteur sur a reçoit le message *put*

$$\begin{array}{lll} \nu a, b, c & a \rhd [get(y) = \zeta(e', s')(& \\ & \quad e' \rhd [put(x) = \zeta(e, s)(& \\ & \quad\quad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \;||\; y \lhd rep(x))])] & \\ & \quad ||\; y \lhd rep(b))] & \\ \;||\; & a \lhd get(c) & \end{array}$$

puis le message *get*

$$\begin{array}{lll} \nu a, b, c & a \rhd [put(x) = \zeta(e, s)(& \\ & \quad e \rhd [get(y) = \zeta(e', s')(e' \rhd s \;||\; y \lhd rep(x))])] \\ \;||\; & c \lhd rep(b) & \end{array}$$

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

Messages orphelins : définitions

Interfaces futures

Ensemble des étiquettes de message acceptés par les comportements futurs

$$Future(a, C) \triangleq \{ m \mid \exists C' : C \rightarrow^{\star} C' \ \wedge \ C' \equiv \nu a. \ldots \| a \rhd [m(\ldots) = \ldots] \}$$

Messages orphelins

Messages non consommables dans tous les futurs

$$Orphan(m, a, C) \triangleq m \notin Future(a, C) \ \wedge \ C \equiv \nu a. \ldots \| a \lhd m(\ldots)$$

Configuration sans orphelin

$$OrphanFree(C) \triangleq \forall a, m, C' : C \rightarrow^{\star} C' \ \Rightarrow \ \neg Orphan(m, a, C')$$

Exemples basés sur la cellule linéaire

Cellule linéaire avec deux messages : création d'un orphelin

$$C_0 = LC \;||\; (\nu x, a \lhd put(x) || a \lhd get(a))$$

Unique trace possible : $C_0 \rightarrow C_1 \rightarrow C_2$
Interfaces : $\forall C$, t.q. $C_0 \rightarrow^\star C, Future(a, C) = \{put, get\}$
Messages : $\{rep\}$ pour a dans C_2 $\qquad rep \notin Future(a, C_2)$

Cellule linéaire avec serveur PutGet : configuration sans orphelin

$$LC \;||\; (\nu b, b \rhd [rep(v) = \zeta(e, s)(e \rhd s \;||\; \nu x, a \lhd put(x) || a \lhd get(e))] \;||\; b \lhd rep(b))$$

Interfaces : $\forall C$, t.q. $C_0 \rightarrow^\star C, \begin{cases} Future(a, C) = \{put, get\} \\ Future(b, C) = \{rep\} \end{cases}$
Messages : $\{put, get\}$ pour a et $\{rep\}$ pour b

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

Quelles analyses (historiques) pour CAP ?

- typage simple ['95] : *arité* et orphelins « triviaux »

$$\cdots \parallel a \rhd [m() = \zeta(e, s)(e \rhd s)] \parallel a \lhd m(b) \parallel a \lhd p() \parallel \cdots$$

- analyse de linéarité ['96] : *un acteur* $\leftrightarrow$ *une adresse*

$$\cdots \parallel a \rhd [m() = \ldots] \parallel a \rhd [p() = \ldots] \parallel \cdots$$

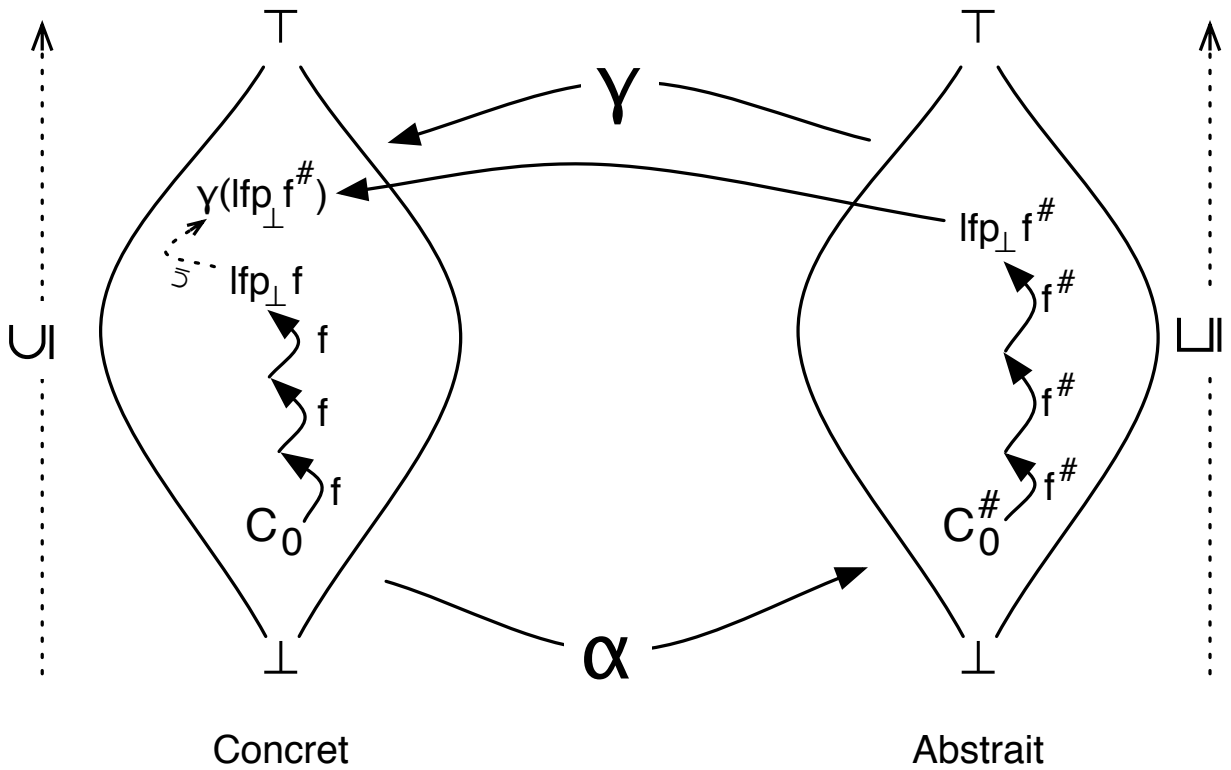
- détection de messages orphelins ['97] :

$$\begin{array}{l} \nu a, \quad a \rhd [\ init(v) = \zeta(e_v, s_v)(\\ \qquad\qquad e_v \rhd [get(c) = \zeta(e_p, s_p)(c \lhd rep(v) \ \parallel \ e_p \rhd s_p)]) \\ \qquad\qquad] \\ \quad \parallel \quad a \lhd init(v_1) \ \parallel \ a \lhd init(v_2) \ \parallel \ \cdots \end{array}$$

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

Interprétation abstraite – Un cadre pour l'analyse de calculs de processus

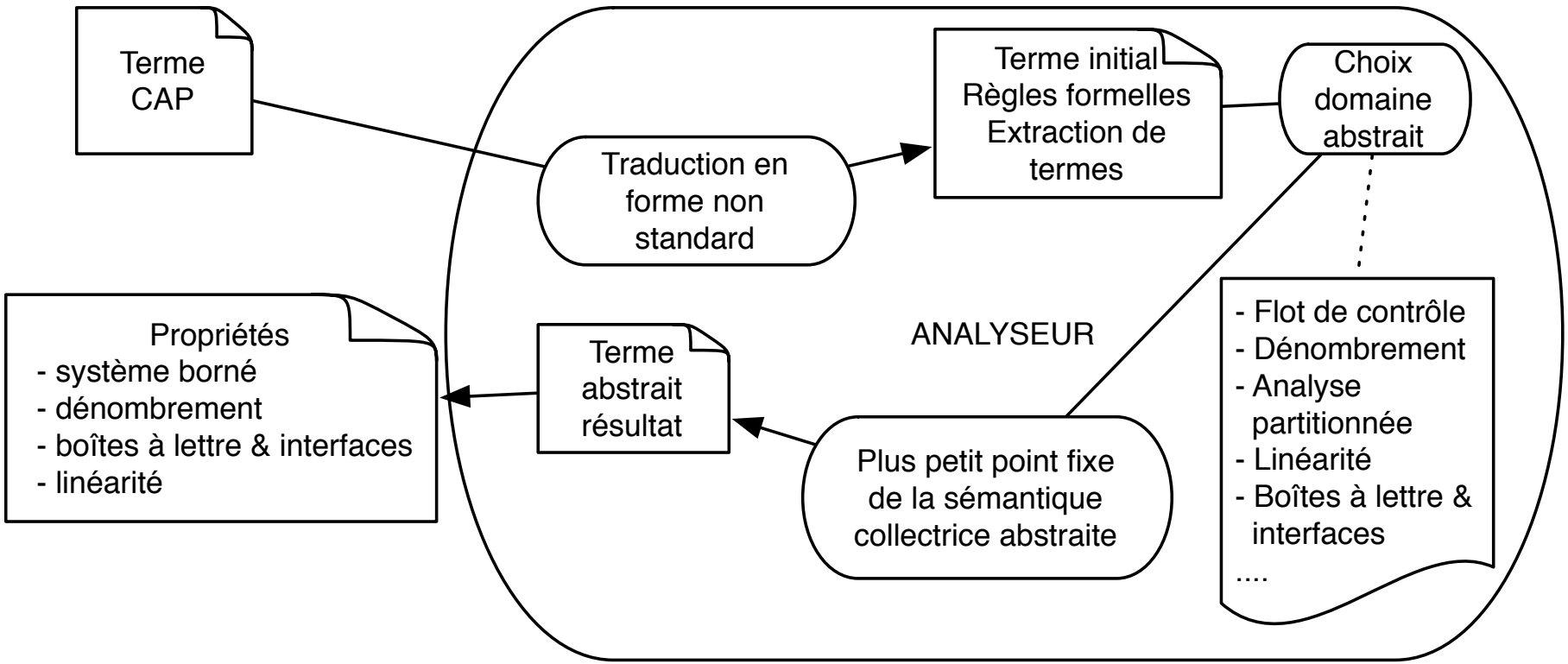
Sémantique collectrice : $\left\{(u, C) | C_0 \xrightarrow{u} C\right\}$



Sémantique collectrice exprimée comme point fixe : $lfp_{\perp} f$

$$f(X) \triangleq (\epsilon, C_0) \cup \left\{(u.\lambda, C') | (u, C) \in x \wedge C \xrightarrow{\lambda} C'\right\}$$

Démarche globale (à la Feret)



1 CAP, un calcul d'acteurs primitifs
- Syntaxe – Sémantique
- Caractériser les messages orphelins

2 Analyse statique de CAP
- Interprétation abstraite
- Encoder CAP
- Domaines abstraits pour la détection des messages orphelins
- Vérifier l'absence de messages orphelins

3 Travaux connexes & Conclusion
- Comparaison
- Bilan & perspectives

Encoder CAP

Idée : expliciter l'α-conversion par l'historique des transitions

But : différentier les instances récursives des adresses et des processus

Terme CAP $\qquad\qquad$ Ensemble de processus

$$a \rhd^{pp_1} x \quad \Longrightarrow \quad \left(pp_1, id_1, \left[\begin{array}{ccc} a & \rightarrow & pp_2, id_2 \\ x & \rightarrow & pp_3, id_3 \end{array}\right]\right)$$

Terme CAP :

$$\nu a^{\alpha},\ b^{\beta},\\ a \rhd^{1} [put^{2}(x) = \zeta(e,s)(e \rhd^{3} [get^{4}(y) = \zeta(e',s')(e' \rhd^{5} s \parallel y \lhd^{6} rep(x))])]\\ \parallel a \lhd^{7} put(b) \parallel a \lhd^{8} get(b)$$

Terme encodé :

$$\left\{ \begin{array}{ll} (1, \epsilon, \left[\begin{array}{ccc} a & \mapsto & \alpha, \epsilon \end{array}\right]) & (2, \epsilon, [\,]) \\ \left(7, \epsilon, \left[\begin{array}{ccc} a & \mapsto & \alpha, \epsilon \\ b & \mapsto & \beta, \epsilon \end{array}\right]\right) & \left(8, \epsilon, \left[\begin{array}{ccc} a & \mapsto & \alpha, \epsilon \\ b & \mapsto & \beta, \epsilon \end{array}\right]\right) \end{array} \right\}$$

Sémantique non standard

Parties actives (processus) :

acteurs, messages et branches des comportements

Sémantique : Deux règles

- un acteur défini syntaxiquement reçoit un message

$$a \rhd [\ldots] \;||\; a \lhd m(\ldots)$$

- un acteur avec un comportement défini dynamiquement reçoit un message

$$a \rhd x \;||\; a \lhd m(\ldots)$$

Expression de la fonctionnalité d'ordre supérieur par des processus de *définition* (comme en Join-calcul).

Communication avec un acteur statique ou dynamique

- Règle statique

$$\frac{\mathit{static\_actor}}{a \rhd [m_i(\tilde{x_i}) = \zeta(e,s)\,C_i]} \qquad \frac{\mathit{message}}{a' \lhd m(\tilde{y})}$$

- Règle dynamique

$$\frac{\mathit{behavior}}{m_i(\tilde{x_i}) = \zeta(e,s)\,C_i} \qquad \frac{\mathit{dynamic\_actor}}{a \rhd b} \qquad \frac{\mathit{message}}{a' \lhd m(\tilde{y})}$$

Communication avec un acteur statique ou dynamique

- Règle statique

$$\frac{\mathit{static\_actor} \qquad \mathit{message}}{a \rhd [m_i(\tilde{x_i}) = \zeta(e,s)\,C_i] \qquad a' \lhd m(\tilde{y})}$$

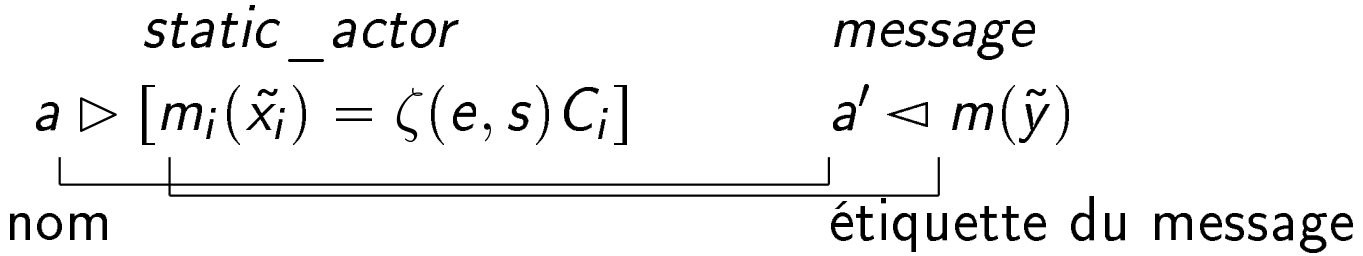
- Règle dynamique

$$\frac{\mathit{behavior} \quad \mathit{dynamic\_actor} \qquad \mathit{message}}{m_i(\tilde{x_i}) = \zeta(e,s)\,C_i \qquad a \rhd b \qquad a' \lhd m(\tilde{y})}$$

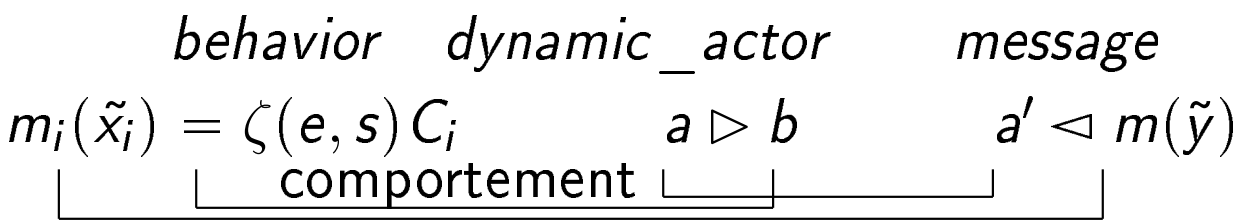
Vérifier les contraintes de compatibilité

Communication avec un acteur statique ou dynamique

- Règle statique



- Règle dynamique



Vérifier les contraintes de compatibilité

Communication avec un acteur statique ou dynamique

- Règle statique

$$\frac{\mathit{static\_actor} \qquad \mathit{message}}{a \rhd [m_i(\tilde{x_i}) = \zeta(e, s)\, C_i] \qquad a' \lhd m(\tilde{y})}$$

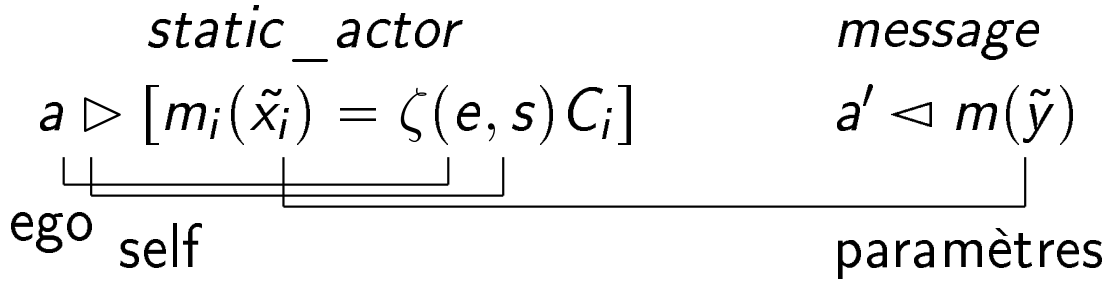
- Règle dynamique

$$\frac{\mathit{behavior} \quad \mathit{dynamic\_actor} \qquad \mathit{message}}{m_i(\tilde{x_i}) = \zeta(e, s)\, C_i \qquad a \rhd b \qquad a' \lhd m(\tilde{y})}$$

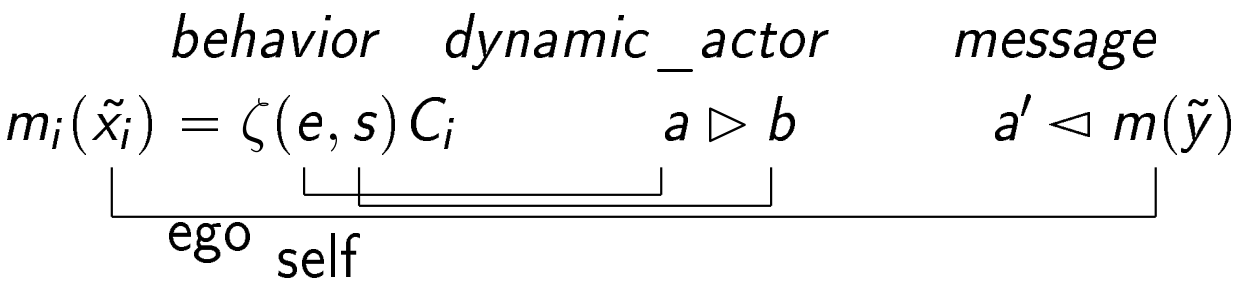
Calculer le passage de valeur

Communication avec un acteur statique ou dynamique

- Règle statique



- Règle dynamique



Calculer le passage de valeur

Communication avec un acteur statique ou dynamique

- Règle statique

$$\frac{\mathit{static\_actor} \qquad \mathit{message}}{a \rhd [m_i(\tilde{x_i}) = \zeta(e,s)\mathbf{C_i}] \qquad a' \lhd m(\tilde{y})}$$

- Règle dynamique

$$\frac{\mathit{behavior} \quad \mathit{dynamic\_actor} \qquad \mathit{message}}{m_i(\tilde{x_i}) = \zeta(e,s)\mathbf{C_i} \qquad a \rhd b \qquad a' \lhd m(\tilde{y})}$$

Lancer les nouveaux processus ainsi que les définitions.

Communication avec un acteur statique ou dynamique

- Règle statique

$$\frac{\mathit{static\_actor} \qquad\qquad \mathit{message}}{a \rhd [m_i(\tilde{x_i}) = \zeta(e,s)\,C_i] \qquad a' \lhd m(\tilde{y}) \qquad C_i'}$$

- Règle dynamique

$$\frac{\mathit{behavior} \quad \mathit{dynamic\_actor} \qquad \mathit{message}}{m_i(\tilde{x_i}) = \zeta(e,s)\,C_i \qquad a \rhd b \qquad a' \lhd m(\tilde{y}) \qquad C_i'}$$

Lancer les nouveaux processus ainsi que les définitions.

Communication avec un acteur statique ou dynamique

- Règle statique

$$\frac{\mathit{static\_actor} \qquad\qquad \mathit{message}}{a \rhd [m_i(\tilde{x_i}) = \zeta(e,s)\,C_i] \qquad a' \lhd m(\tilde{y}) \qquad C_i'}$$

- Règle dynamique

$$\frac{\mathit{behavior} \quad \mathit{dynamic\_actor} \qquad \mathit{message}}{m_i(\tilde{x_i}) = \zeta(e,s)\,C_i \qquad a \rhd b \qquad a' \lhd m(\tilde{y}) \qquad C_i'}$$

Supprimer les processus interagissants

Communication avec un acteur statique ou dynamique

- Règle statique

$$\frac{a \rhd [m_i(\tilde{x_i}) = \zeta(e, s)\, C_i] \qquad a' \lhd m(\tilde{y})}{C_i'}$$

static_actor *message*

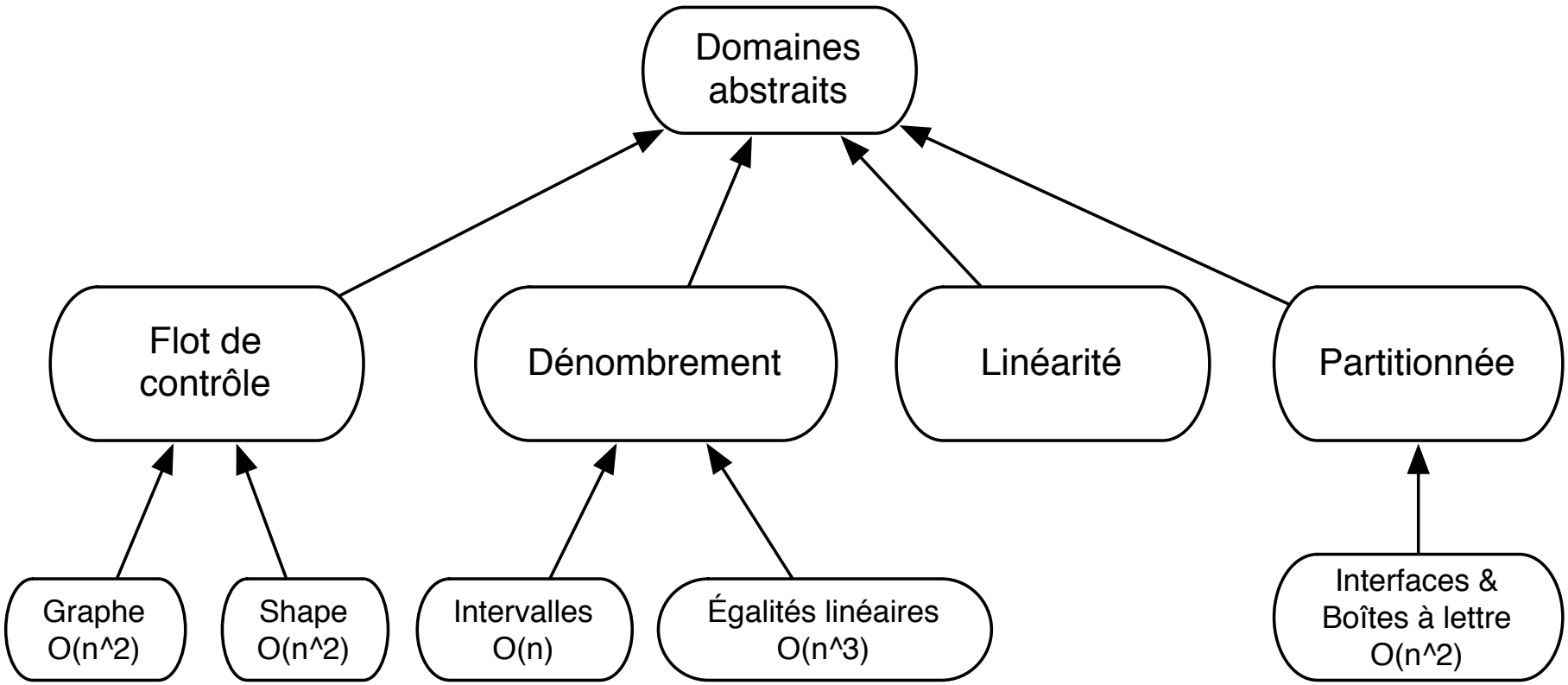
- Règle dynamique

$$\frac{m_i(\tilde{x_i}) = \zeta(e, s)\, C_i \qquad a \rhd b \qquad a' \lhd m(\tilde{y})}{C_i'}$$

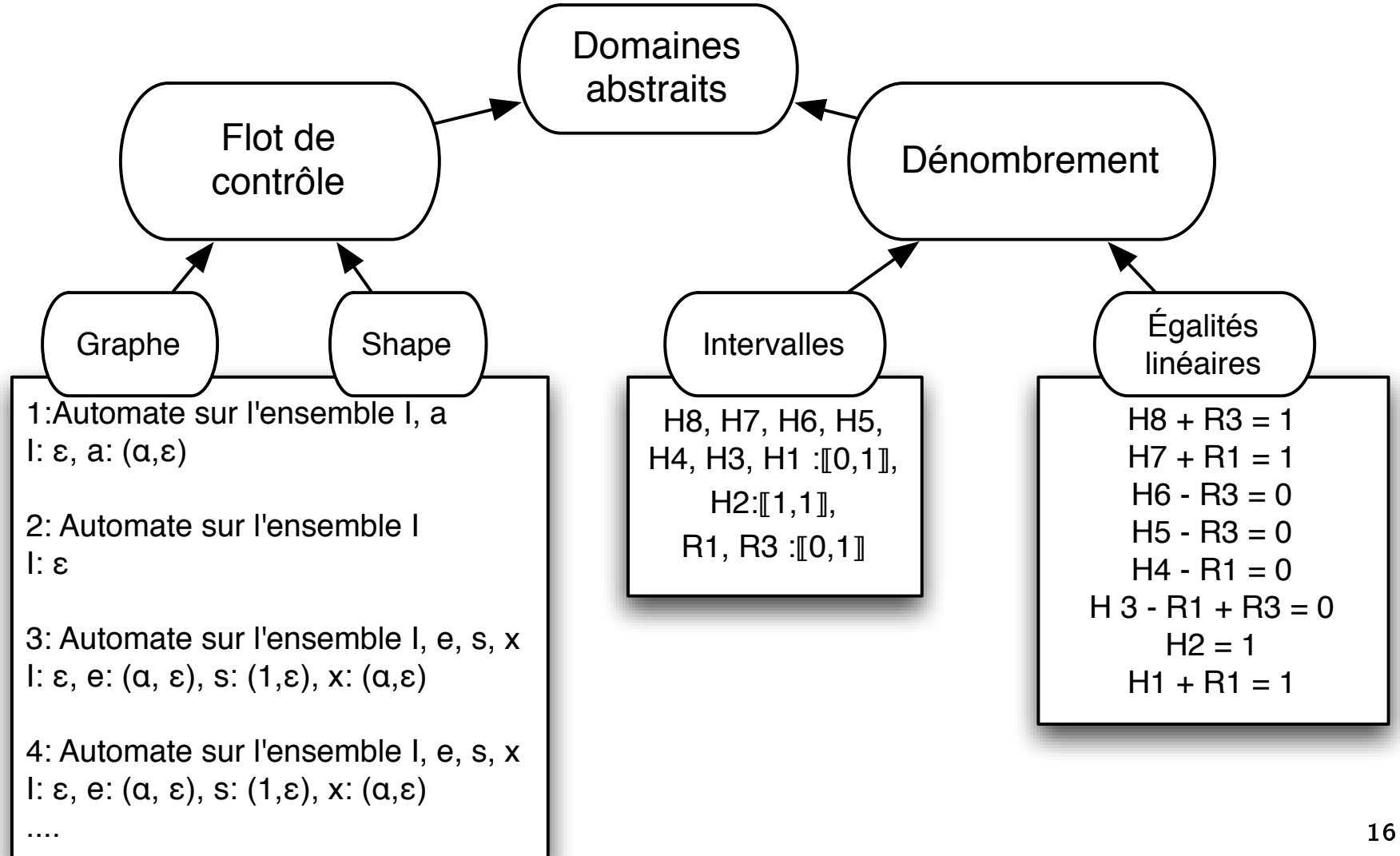
behavior *dynamic_actor* *message*

Supprimer les processus interagissants

Abstraire les propriétés des configurations atteignables

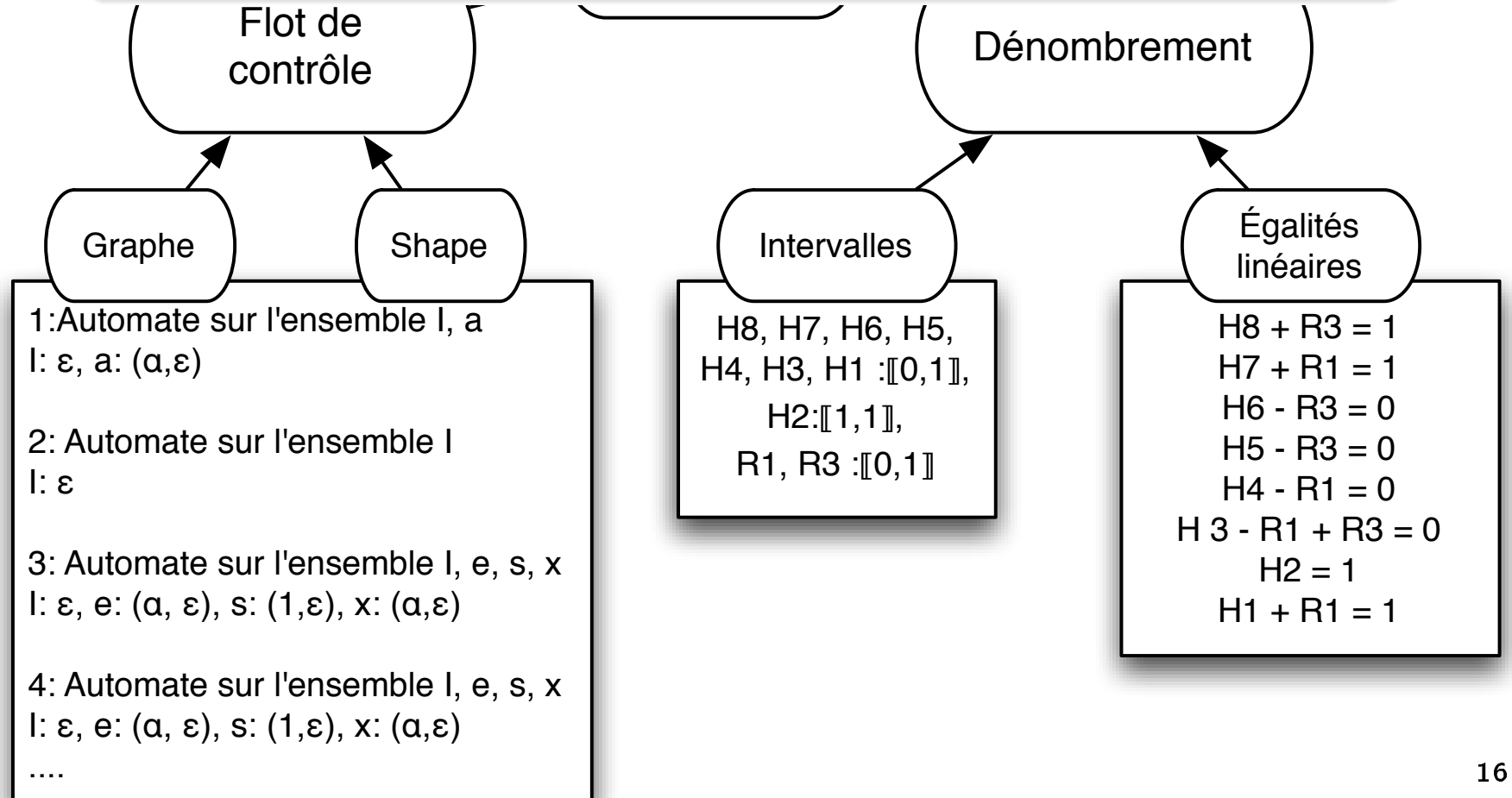


Point fixe de la sémantique collectrice abstraite pour la cellule linéaire



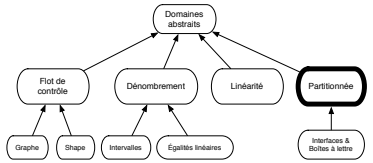
Point fixe de la sémantique collectrice abstraite pour la cellule linéaire

$$\nu a^{\alpha},\ b^{\beta},\ a\triangleright^{1}[put^{2}(x)=\zeta(e,s)(e\triangleright^{3}[get^{4}(y)=\zeta(e',s')(e'\triangleright^{5}s\ ||\ y\triangleleft^{6}rep(x))])]\ ||\ a\triangleleft^{7}put(b)\ ||\ a\triangleleft^{8}get(b)$$

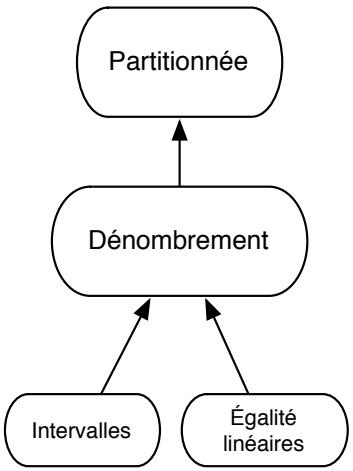


1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

Analyse partitionnée



- Propriétés partitionnées par lieur d'adresse (ν)
  (union des propriétés des instances récursives)
- Transition sur une adresse donnée $\Longrightarrow$ dans une partition
- Nouveaux processus
  - dans la partition active, si leur adresse est l'adresse courante
  - dans les autres partitions sinon



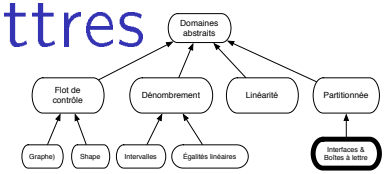
$$\alpha : \begin{cases} H8, H7, H6, H3, H1 : [\![0,1]\!], H5, H4, H2 : [\![0,0]\!] \\ H8 + R3 = 1 \\ H7 + R1 = 1 \\ H6 - R3 = 0 \\ H3 - R1 + R3 = 0 \\ H1 + R1 = 1 \end{cases}$$

$$\beta : \begin{cases} H8, H7, H6, H4, H3, H2, H1 : [\![0,0]\!], H5 : [\![0,1]\!] \\ H5 - R3 = 0 \end{cases}$$

$$\nu a^{\alpha}, b^{\beta}, a \triangleright^{1} [put^{2}(x) = \zeta(e,s)(e \triangleright^{3} [get^{4}(y) = \zeta(e',s')(e' \triangleright^{5} s \;||\; y \triangleleft^{6} rep(x))])] \;||\; a \triangleleft^{7} put(b) \;||\; a \triangleleft^{8} get(b)$$

Approximation des interfaces et des boîtes aux lettres

sous l'analyse partitionnée

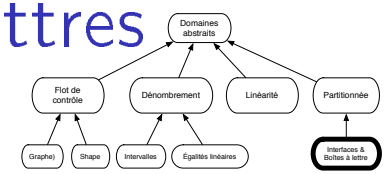


- un domaine abstrait pour représenter la **causalité** : séquences d'interfaces associées à une adresse
- représente les changements locaux de messages (cons./prod.)

Graphe construit sur les points de programme d'acteurs

Approximation des interfaces et des boîtes aux lettres

sous l'analyse partitionnée



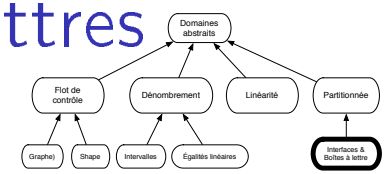
- un domaine abstrait pour représenter la **causalité** : séquences d'interfaces associées à une adresse
- représente les changements locaux de messages (cons./prod.)

Graphe construit sur les points de programme d'acteurs :

- nœud = acteur installé sur l'adresse ou adresse sans acteur

Approximation des interfaces et des boîtes aux lettres

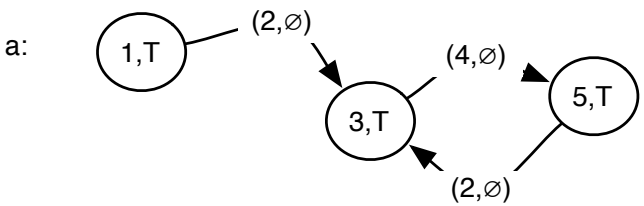
sous l'analyse partitionnée



- un domaine abstrait pour représenter la **causalité** : séquences d'interfaces associées à une adresse
- représente les changements locaux de messages (cons./prod.)

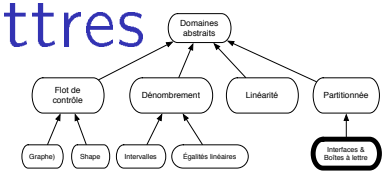
Graphe construit sur les points de programme d'acteurs :

- nœud = acteur installé sur l'adresse ou adresse sans acteur
- arc = réception d'un message (causalité – cons./prod. locale)



Approximation des interfaces et des boîtes aux lettres

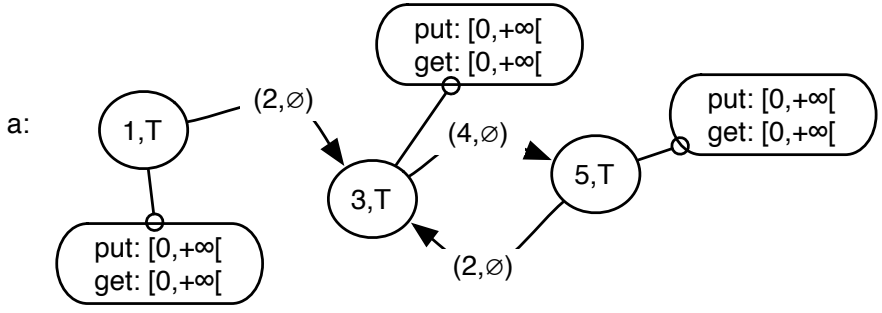
sous l'analyse partitionnée



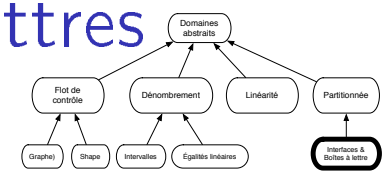
- un domaine abstrait pour représenter la **causalité** : séquences d'interfaces associées à une adresse
- représente les changements locaux de messages (cons./prod.)

Graphe construit sur les points de programme d'acteurs :

- nœud = acteur installé sur l'adresse ou adresse sans acteur
- arc = réception d'un message (causalité – cons./prod. locale)
- boîtes aux lettres locales abstraites associées aux nœuds



Approximation des interfaces et des boîtes aux lettres

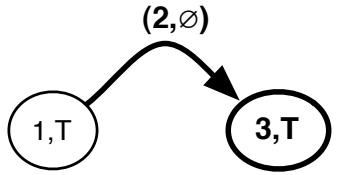


Installer un acteur et/ou envoyer des messages dans la partition active

Transition avec un acteur sur a, défini au point de programme p.
Causalité : le nœud (p, T) est le meilleur ancêtre pour les processus lancés.

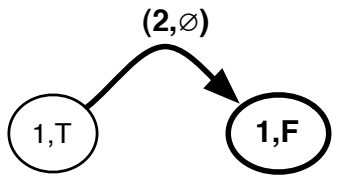
- un acteur sur a défini au point de programme p' est lancé
  $\Longrightarrow$ ajout d'un arc de (p, T) à (p', T)

$$a \rhd^1 [m^2() = \zeta(e, s)(e \rhd^3 s)]$$



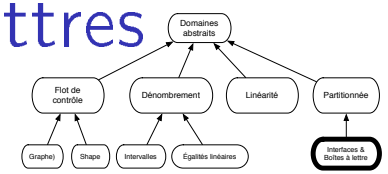
- pas d'acteur lancé sur a : création d'un nœud *mort* pour p
  $\Longrightarrow$ ajout d'un arc de (p, T) à (p, F)

$$a \rhd^1 [m^2() = \zeta(e, s)(0)]$$



Approximation des interfaces et des boîtes aux lettres

Lancer des processus hors de la partition active (1/2)

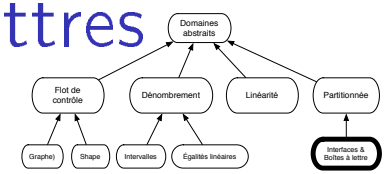


Calcul d'une transition sur une adresse a

$\Longrightarrow$ production possible de processus sur une adresse différente b.

Problème : qui est l'ancêtre maintenant ? (causalité)

Approximation des interfaces et des boîtes aux lettres



Lancer des processus hors de la partition active (1/2)

Calcul d'une transition sur une adresse a

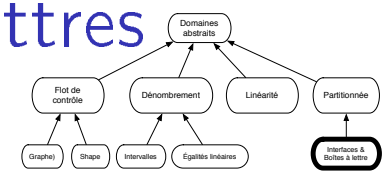
$\Longrightarrow$ production possible de processus sur une adresse différente b.

Problème : qui est l'ancêtre maintenant ? (causalité)

Nécessité de sur-approximer cet ancêtre :

- une première approche : tous les nœuds du graphe

Approximation des interfaces et des boîtes aux lettres



Lancer des processus hors de la partition active (1/2)

Calcul d'une transition sur une adresse a

$\Longrightarrow$ production possible de processus sur une adresse différente b.

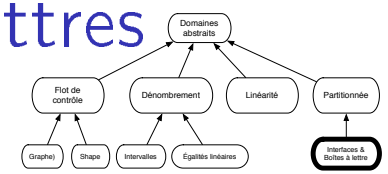
Problème : qui est l'ancêtre maintenant ? (causalité)

Nécessité de sur-approximer cet ancêtre :

- une première approche : tous les nœuds du graphe
- une seconde : utiliser les informations du flot de contrôle

  $\Longrightarrow$ les marqueurs des processus interagissant lors de la transition

Approximation des interfaces et des boîtes aux lettres



Lancer des processus hors de la partition active (1/2)

Calcul d'une transition sur une adresse a
$\Longrightarrow$ production possible de processus sur une adresse différente b.

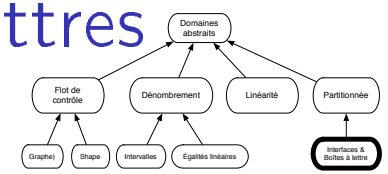
Problème : qui est l'ancêtre maintenant ? (causalité)

Nécessité de sur-approximer cet ancêtre :

- une première approche : tous les nœuds du graphe
- une seconde : utiliser les informations du flot de contrôle
  $\Longrightarrow$ les marqueurs des processus interagissant lors de la transition

Lancements des processus produits dans la partition non active pour chaque ancêtre possible.

Approximation des interfaces et des boîtes aux lettres

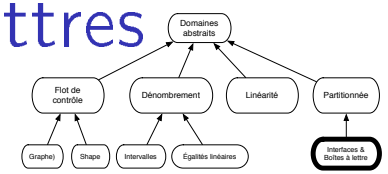


Lancer des processus hors de la partition active (2/2)

Pour chaque partition et chaque ancêtre possible associé p_a :

- si un acteur sur p est lancé
  $\Longrightarrow$ ajout d'un arc de (p_a, F) à (p, T)

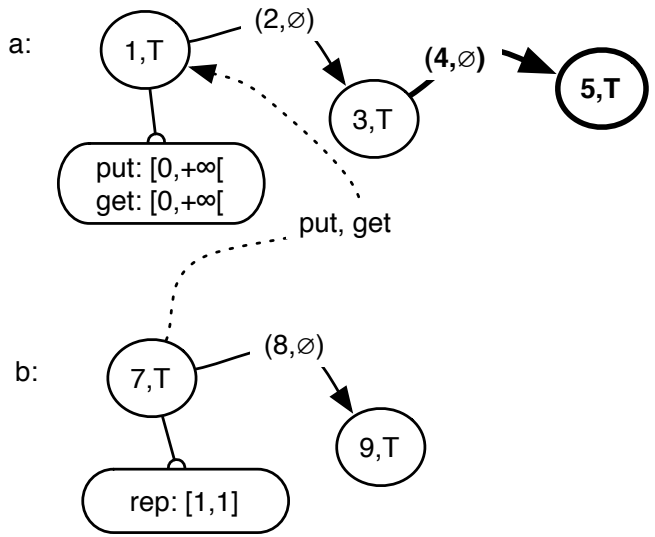
Approximation des interfaces et des boîtes aux lettres



Lancer des processus hors de la partition active (2/2)

Pour chaque partition et chaque ancêtre possible associé p_a :

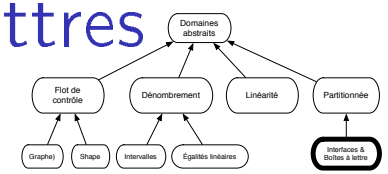
- si un acteur sur p est lancé
  $\Longrightarrow$ ajout d'un arc de (p_a, F) à (p, T)
- si aucun acteur n'est lancé
  Les messages lancés sont accessibles pour les successeurs de l'ancêtre : mise à jour de leur boîte aux lettres locale abstraite



3 reçoit le message get

création d'un message *rep* sur l'adresse b

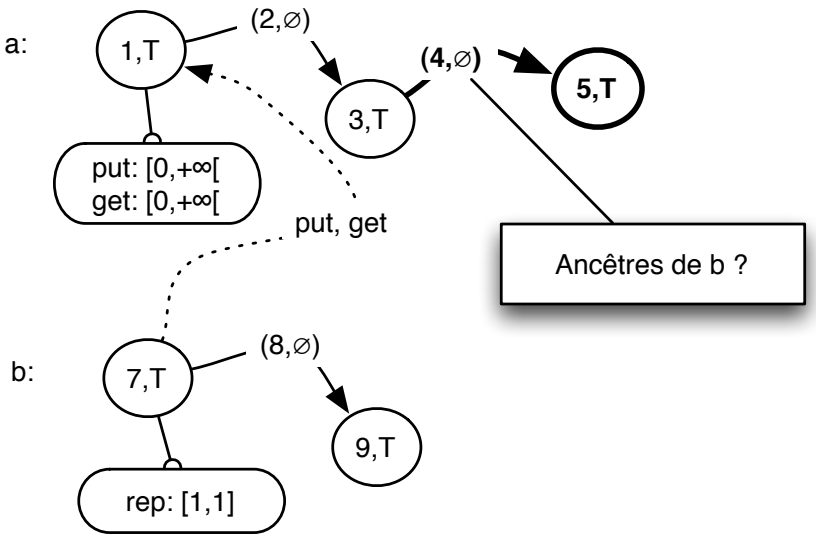
Approximation des interfaces et des boîtes aux lettres



Lancer des processus hors de la partition active (2/2)

Pour chaque partition et chaque ancêtre possible associé p_a :

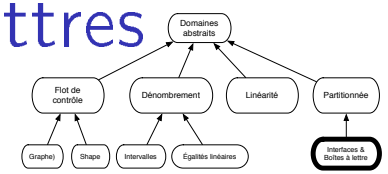
- si un acteur sur p est lancé
  $\Longrightarrow$ ajout d'un arc de (p_a, F) à (p, T)
- si aucun acteur n'est lancé
  Les messages lancés sont accessibles pour les successeurs de l'ancêtre : mise à jour de leur boîte aux lettres locale abstraite



3 reçoit le message get

création d'un message *rep* sur l'adresse b

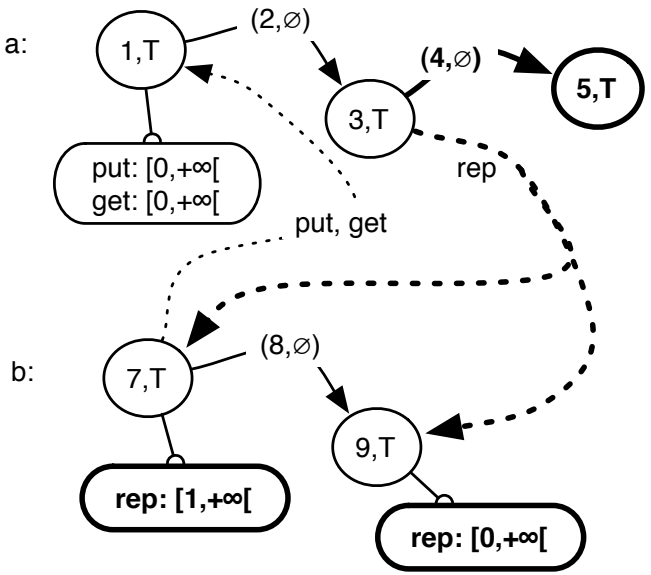
Approximation des interfaces et des boîtes aux lettres



Lancer des processus hors de la partition active (2/2)

Pour chaque partition et chaque ancêtre possible associé p_a :

- si un acteur sur p est lancé
  $\Longrightarrow$ ajout d'un arc de (p_a, F) à (p, T)
- si aucun acteur n'est lancé
  Les messages lancés sont accessibles pour les successeurs de l'ancêtre : mise à jour de leur boîte aux lettres locale abstraite



3 reçoit le message get

création d'un message *rep* sur l'adresse b

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

Garantir l'absence d'orphelins

Objectif

Dans chaque état atteignable, pour chaque adresse,

$$\text{Messages} \subseteq \text{Interfaces Futures}$$

Garantir l'absence d'orphelins

Objectif

Dans chaque état atteignable, pour chaque adresse,

$$\text{Messages} \subseteq \text{Interfaces Futures}$$

En utilisant l'élément abstrait obtenu :

1. identifier l'ensemble des chemins maximaux (interfaces futures) à partir d'un état
2. identifier les étiquettes de messages à chaque état

Vérifier que chaque étiquette est comprise dans chaque chemin maximal

Caractériser (correctement) les chemins maximaux

Correction de l'analyse

Est-ce que l'approche est correcte avec la sur-approximation due

- au flot de contrôle,
- au calcul de l'ancêtre (installation hors de la partition active) ?

Un chemin maximal peut-il être incorrect ?

Caractériser (correctement) les chemins maximaux

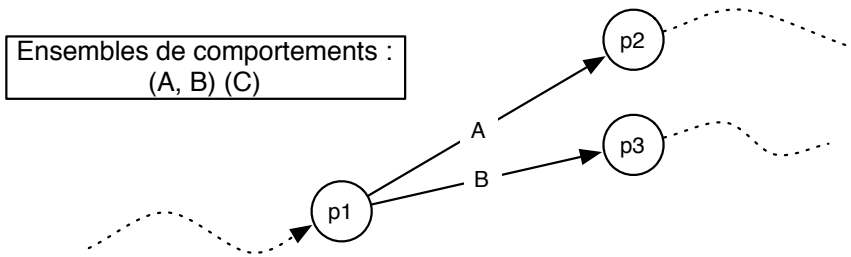
Correction de l'analyse

Est-ce que l'approche est correcte avec la sur-approximation due

- au flot de contrôle,
- au calcul de l'ancêtre (installation hors de la partition active) ?

Un chemin maximal peut-il être incorrect ?

- Acteur défini syntaxiquement



Caractériser (correctement) les chemins maximaux

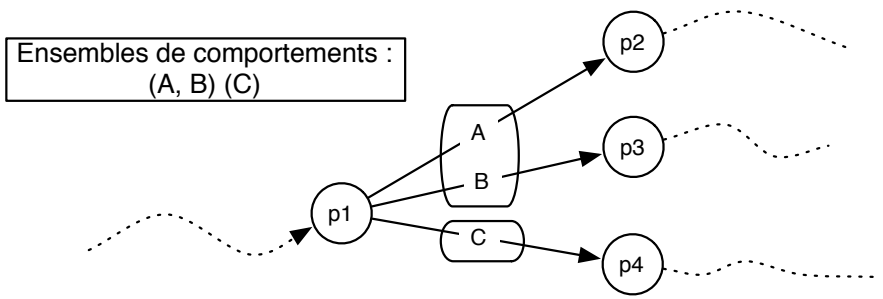
Correction de l'analyse

Est-ce que l'approche est correcte avec la sur-approximation due

- au flot de contrôle,
- au calcul de l'ancêtre (installation hors de la partition active) ?

Un chemin maximal peut-il être incorrect ?

- Acteur défini syntaxiquement
- Acteur défini dynamiquement



Il faut traiter les acteurs dynamiques avec soin : considérer chaque ensemble de comportements !

Caractériser (correctement) les chemins maximaux

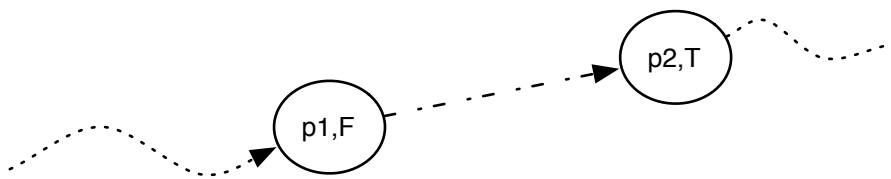
Correction de l'analyse

Est-ce que l'approche est correcte avec la sur-approximation due

- au flot de contrôle,
- au calcul de l'ancêtre (installation hors de la partition active) ?

Un chemin maximal peut-il être incorrect ?

- Acteur défini syntaxiquement
- Acteur défini dynamiquement
- Nœud mort



Arcs sortants : installations d'acteurs hors de la partition active
$\Longrightarrow$ pas de garantie d'installation effective d'un acteur

Caractériser (correctement) les chemins maximaux

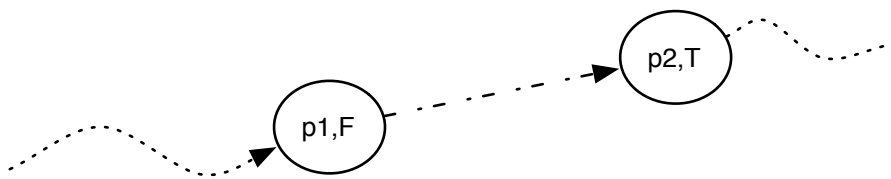
Correction de l'analyse

Est-ce que l'approche est correcte avec la sur-approximation due

- au flot de contrôle,
- au calcul de l'ancêtre (installation hors de la partition active) ?

Un chemin maximal peut-il être incorrect ?

- Acteur défini syntaxiquement
- Acteur défini dynamiquement
- Nœud mort



Arcs sortants : installations d'acteurs hors de la partition active

$\Longrightarrow$ pas de garantie d'installation effective d'un acteur

$\Longrightarrow$ nœuds morts **terminaux** pour les chemins maximaux

Garantir l'absence d'orphelins (rappel)

Objectif

Dans chaque état atteignable, pour chaque adresse,

$$\text{Messages} \subseteq \text{Interfaces Futures}$$

En utilisant l'élément abstrait obtenu :

1. identifier l'ensemble des chemins maximaux (interfaces futures) à partir d'un état
2. identifier les étiquettes de messages à chaque état

Vérifier que chaque étiquette est comprise dans chaque chemin maximal

Caractériser les messages à chaque nœud

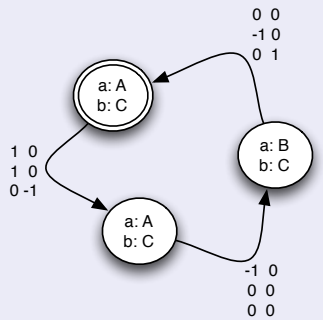
1. Calcul explicite des boîtes aux lettres abstraites

Expression des boîtes comme plus petit point fixe d'une fonction :

- des arcs entrants et des boîtes aux lettres associées
- des boîtes aux lettres abstraites locales

2. Tester les occurrences des étiquettes de messages

Domaine abstrait : représentation du système par un VASS
$\Longrightarrow$ test d'occurrence d'étiquette de message à un nœud donné (lemme de Higman)



Aller plus loin : garantir la consommation des messages

Propriété de vivacité $\Longrightarrow$ ajout d'une contrainte d'équité

un message qui peut être infiniment consommé le sera

Absence d'orphelin + Équité + Absence de blocage

$\Longrightarrow$ Garantir la consommation des messages

Aller plus loin : garantir la consommation des messages

Propriété de vivacité $\Longrightarrow$ ajout d'une contrainte d'équité

un message qui peut être infiniment consommé le sera

Absence d'orphelin + Équité + Absence de blocage
$\Longrightarrow$ Garantir la consommation des messages

Garantir l'absence de blocage

- non blocage pour un acteur statique $\Longrightarrow$ occurrence strictement positive d'une étiquette de branche de comportement

Aller plus loin : garantir la consommation des messages

Propriété de vivacité $\Longrightarrow$ ajout d'une contrainte d'équité

un message qui peut être infiniment consommé le sera

Absence d'orphelin + Équité + Absence de blocage
$\Longrightarrow$ Garantir la consommation des messages

Garantir l'absence de blocage

- non blocage pour un acteur statique $\Longrightarrow$ occurrence strictement positive d'une étiquette de branche de comportement
- pour un acteur dynamique : *idem*, par ensemble de comportements

$\Longrightarrow$ représentation explicite des boîtes aux lettres par point fixe (borne inférieure)

Aller plus loin : garantir la consommation des messages

Propriété de vivacité $\Longrightarrow$ ajout d'une contrainte d'équité

un message qui peut être infiniment consommé le sera

Absence d'orphelin + Équité + Absence de blocage
$\Longrightarrow$ Garantir la consommation des messages

Garantir l'absence de blocage

- non blocage pour un acteur statique $\Longrightarrow$ occurrence strictement positive d'une étiquette de branche de comportement
- pour un acteur dynamique : *idem*, par ensemble de comportements

$\Longrightarrow$ représentation explicite des boîtes aux lettres par point fixe (borne inférieure)

- graphe sans création d'acteur hors de la partition active (arcs sortant d'un nœud mort)

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

Comparaison avec les analyses précédentes

Analyses par inférence de types comportementaux de Colaço, Colin *et al.*

- Propriétés

	Typage	IA
Arité	✓	✓
Linéarité	pas de preuve	✓
Orphelins	restriction à des termes sans passage de comportement	✓

Comparaison avec les analyses précédentes

Analyses par inférence de types comportementaux de Colaço, Colin *et al.*

- Propriétés

	Typage	IA
Arité	✓	✓
Linéarité	pas de preuve	✓
Orphelins	restriction à des termes sans passage de comportement	✓

- Traitement des spécificités du calcul
  - Asynchronicité – message étiquettés
  - Passage de comportement
  - Dénombrer

  } difficile dans les informations de type

Comparaison avec les analyses précédentes

Analyses par inférence de types comportementaux de Colaço, Colin *et al.*

- Propriétés

	Typage	IA
Arité	✓	✓
Linéarité	pas de preuve	✓
Orphelins	restriction à des termes sans passage de comportement	✓

- Traitement des spécificités du calcul
  - Asynchronicité – message étiquettés
  - Passage de comportement
  - Dénombrer

  } difficile dans les informations de type
- Comparaison méthodologique
  - Typage : définition guidée par la syntaxe, compositionalité
  - IA : modularité (séparation des propriétés), combinaison de domaines, correction par construction

1. CAP, un calcul d'acteurs primitifs
   - Syntaxe – Sémantique
   - Caractériser les messages orphelins
2. Analyse statique de CAP
   - Interprétation abstraite
   - Encoder CAP
   - Domaines abstraits pour la détection des messages orphelins
   - Vérifier l'absence de messages orphelins
3. Travaux connexes & Conclusion
   - Comparaison
   - Bilan & perspectives

Conclusion

Bilan

- Un cadre générique pour l'analyse de CAP qui traite le passage de comportement
- De nouveaux domaines abstraits spécifiques, pour vérifier l'absence de messages orphelins et la linéarité des termes
- Basé sur l'interprétation abstraite : correction par construction, outils pour combiner les abstractions
- Intégralité des domaines abstraits implantée dans un prototype

Perspectives

- Appliquer nos analyses sur d'autres calculs/langages (programmation par Aspects, Erlang, systèmes multi-agents, services Web, . . .)
- Appliquer les abstractions de la causalité (séquences d'interfaces) et de dénombrement des utilisations des adresses (linéarité) au π-calcul. Comparer avec les travaux de Kobayashi.