Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Marion Guthmuller

Équipe ~~AlGorille~~ VeriDis

Soutenance de thèse pour l'obtention du grade de

Docteur de l'Université de Lorraine (mention informatique)

29 Juin 2015

UNIVERSITÉ DE LORRAINE

Loria
Laboratoire lorrain de recherche en informatique et ses applications

SIMGRID

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Marion Guthmuller

Équipe ~~AlGorille~~ VeriDis

Soutenance de thèse pour l'obtention du grade de

Docteur de l'Université de Lorraine (mention informatique)

29 Juin 2015

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Marion Guthmuller

Équipe ~~AlGorille~~ VeriDis

Soutenance de thèse pour l'obtention du grade de

Docteur de l'Université de Lorraine (mention informatique)

29 Juin 2015

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Marion Guthmuller

Équipe ~~AlGorille~~ VeriDis

Soutenance de thèse pour l'obtention du grade de

Docteur de l'Université de Lorraine (mention informatique)

29 Juin 2015

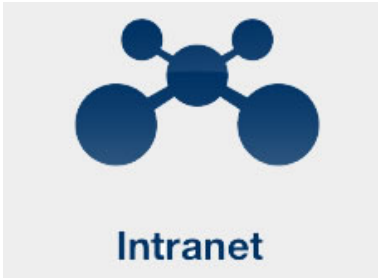
UNIVERSITÉ DE LORRAINE

Loria
Laboratoire lorrain de recherche en informatique et ses applications

SIMGRID

Les systèmes distribués

Les systèmes distribués - Exemples



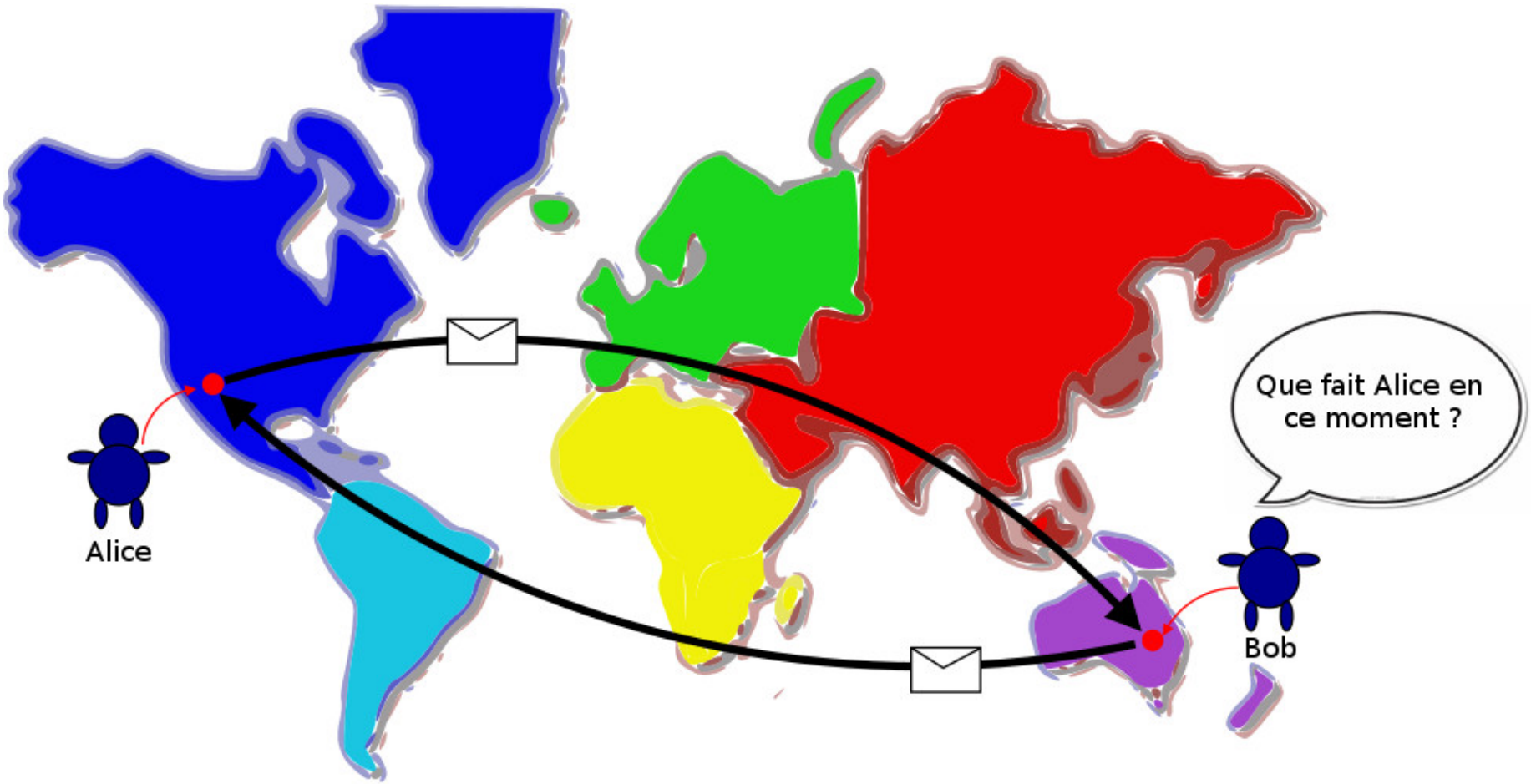
amazon.com

Les systèmes distribués - Caractéristiques

Pas d'horloge globale

Les systèmes distribués - Caractéristiques

Pas de mémoire partagée

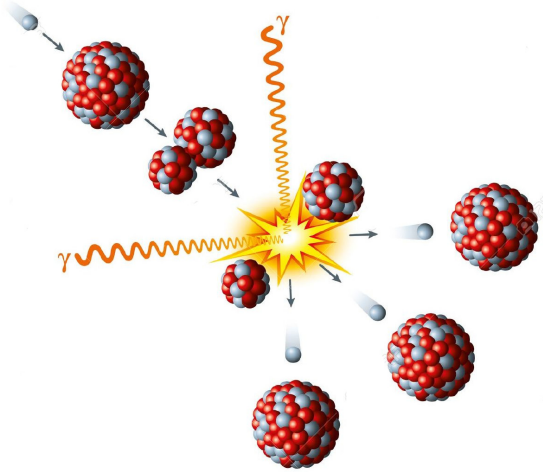


Propriétés de correction

1. « *Le système n'est jamais dans un état invalide* » $\rightsquigarrow$ **Sûreté**

Exemple : Réacteur nucléaire

✓ Refroidissement



✗ Fission incontrôlée

Propriétés de correction

2. « *Le système est toujours finalement dans un état valide* » $\rightsquigarrow$ **Vivacité**

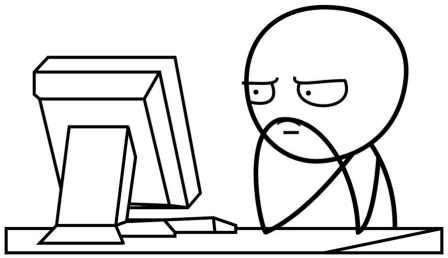
Exemple : Appel à un ascenseur

. . .

✗

Démarche de vérification

Exécution sur plate-forme réelle

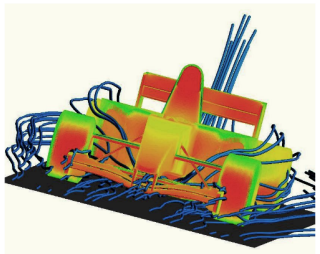


- *Wait and see . . .*
- ☹ Accès à la plate-forme
- ☹ Une exécution parmi d'autres $\rightsquigarrow$ Insuffisant !

Test

- Définition de cas de tests
- ☹ « *Démontre la présence d'erreurs et non l'absence* » (Dijkstra, 1970)

Simulation / Émulation



- Objet d'étude et/ou environnement simulés
- ☹ Validité des modèles
- ☹ Non-exhaustivité des scénarios

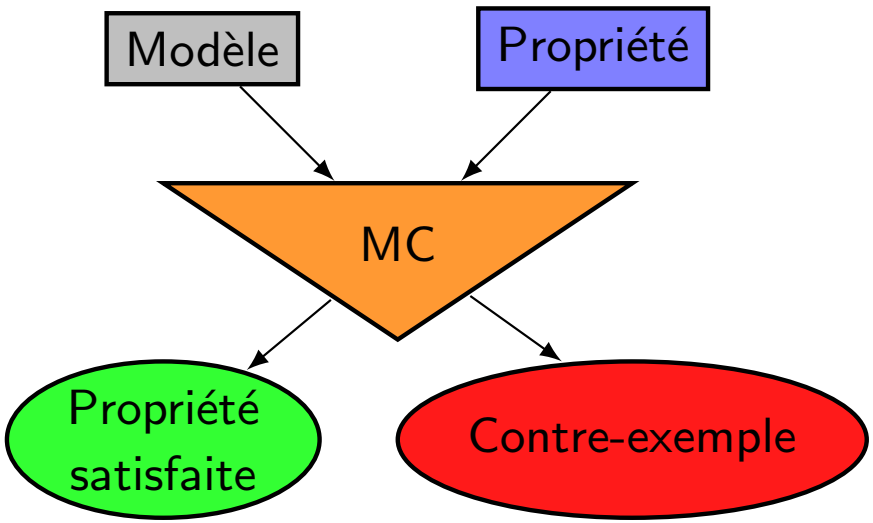
Vérification formelle

Vérification déductive

- Déduction de propriétés prouvées
- Assistant de preuve, preuve automatique, ...
- ☹ Connaissances formelles avancées
- ☹ Pas toujours automatique

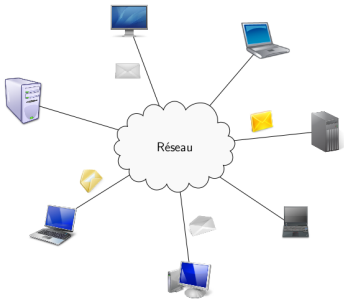
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P \vdash P}\,\textsc{WeakenR}}{P \vdash P, Q}}{\vdash P, P \to Q}\,{\to}\textsc{-Right} \qquad \dfrac{}{P \vdash P}}{(P \to Q) \to P \vdash P, P}\,{\to}\textsc{-Left}}{(P \to Q) \to P \vdash P}\,\textsc{ContractR}}{\vdash ((P \to Q) \to P) \to P}\,{\to}\textsc{-Right}$$

Model checking



- Vérification formelle automatique
- Recherche exhaustive de contre-exemple
- ☹ Vérification d'un modèle du système

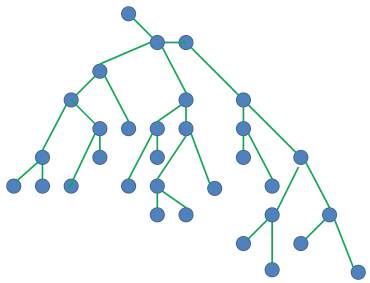
Vérification dynamique formelle



Propriété

Formulation logique

φ



- Exécution systématique de l'application réelle
- Recherche exhaustive d'un contre-exemple
- Explosion combinatoire de l'espace d'états

Propriété satisfaite

Propriété non satisfaite

⇓

Contre-exemple

Échec (Ressources insuffisantes)

⇓

Pas de conclusion possible

Ma thèse (1/2)

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Champ d'étude

- Applications MPI écrites en C, C++ ou Fortran
  - Rigidité de programmation assouplie (passage à l'échelle)
  - Peu d'outils pour la correction
- Exploration exhaustive pour un état initial donné
- Propriétés : propriétés de vivacité
  - Terminaison de programme
  - Propriétés LTL

Intégration dans SimGrid

- Outil de simulation d'applications distribuées
- SimGridMC

Ma thèse (1/2)

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Champ d'étude

- Applications MPI écrites en C, C++ ou Fortran
  - Rigidité de programmation assouplie (passage à l'échelle)
  - Peu d'outils pour la correction
- Exploration exhaustive pour un état initial donné
- Propriétés : propriétés de vivacité
  - Terminaison de programme
  - Propriétés LTL

Intégration dans SimGrid

- Outil de simulation d'applications distribuées
- SimGridMC

Ma thèse (2/2)

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Approche : Adaptation des techniques de *model checking*

- Exploration exhaustive de l'espace d'états
- Recherche de contre-exemple
- Réduction de l'espace d'états

Problématique : Implémentations réelles ⇒ Comportement dynamique inconnu

État de l'art

- Autres types d'applications
- Vérification pour des langages haut niveau
- Limitation sur le type de propriétés vérifiées

Ma thèse (2/2)

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Approche : Adaptation des techniques de *model checking*

- Exploration exhaustive de l'espace d'états
- Recherche de contre-exemple
- Réduction de l'espace d'états

Problématique : Implémentations réelles ⇒ Comportement dynamique inconnu

État de l'art

- Autres types d'applications
- Vérification pour des langages haut niveau
- Limitation sur le type de propriétés vérifiées

Ma thèse (2/2)

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Approche : Adaptation des techniques de *model checking*

- Exploration exhaustive de l'espace d'états
- Recherche de contre-exemple
- Réduction de l'espace d'états

Problématique : Implémentations réelles $\Rightarrow$ Comportement dynamique inconnu

État de l'art

- Autres types d'applications
- Vérification pour des langages haut niveau
- Limitation sur le type de propriétés vérifiées

Plan

Vérification dynamique formelle de propriétés temporelles

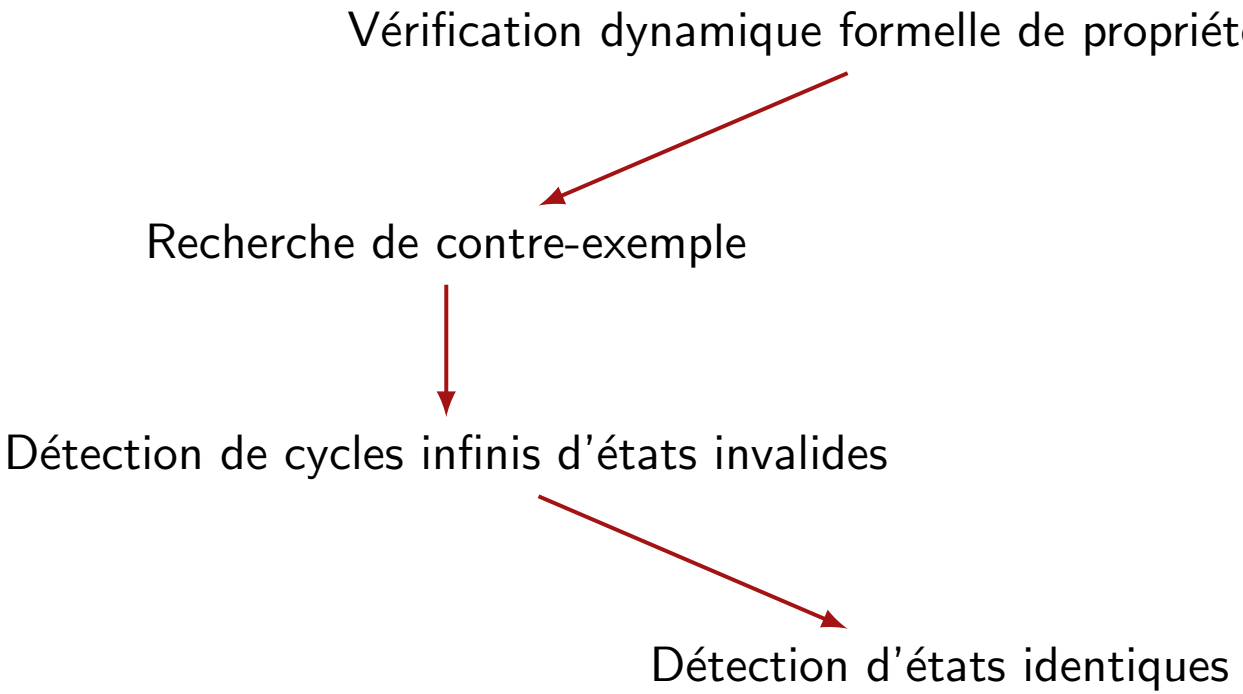
Vérification dynamique formelle de propriétés temporelles

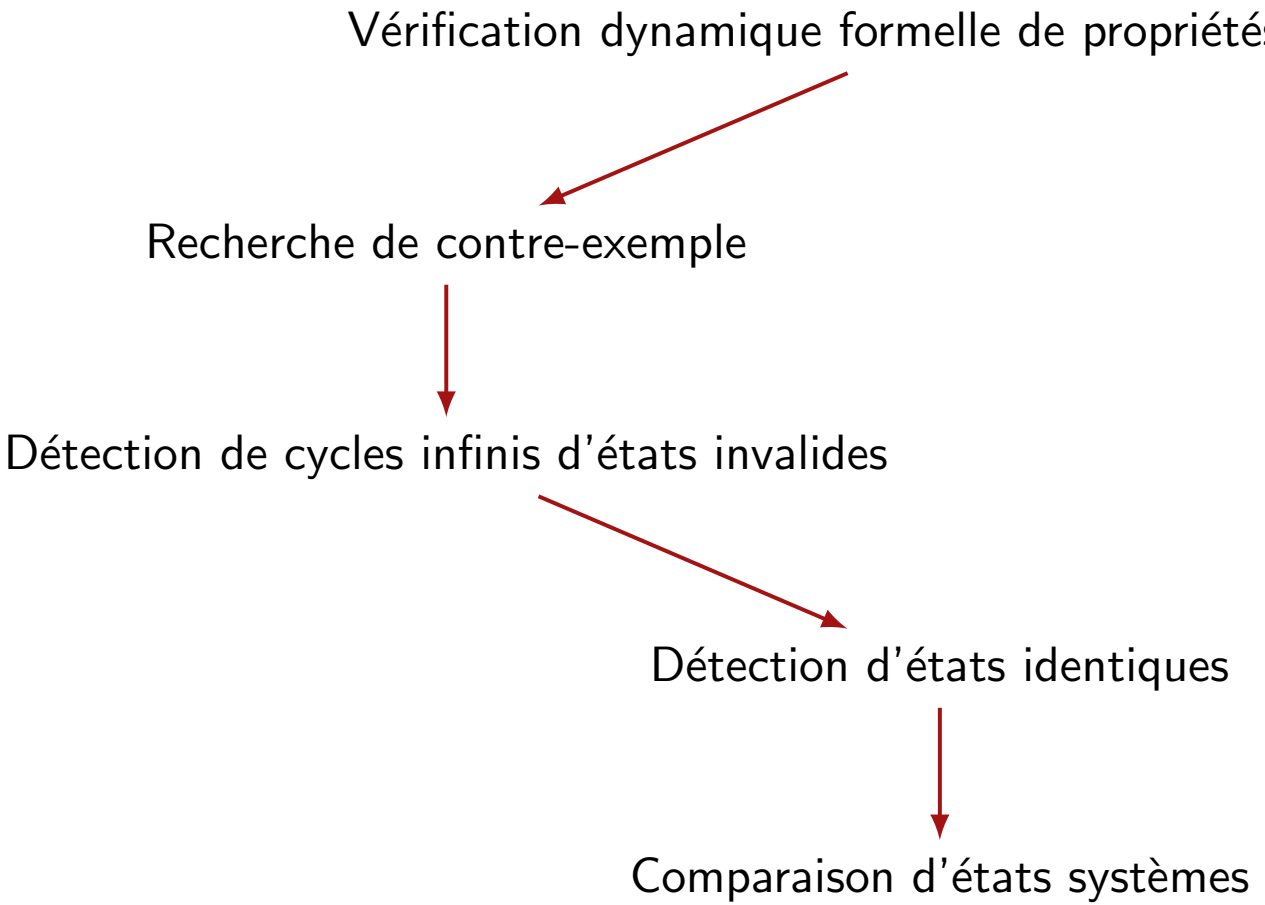
Recherche de contre-exemple

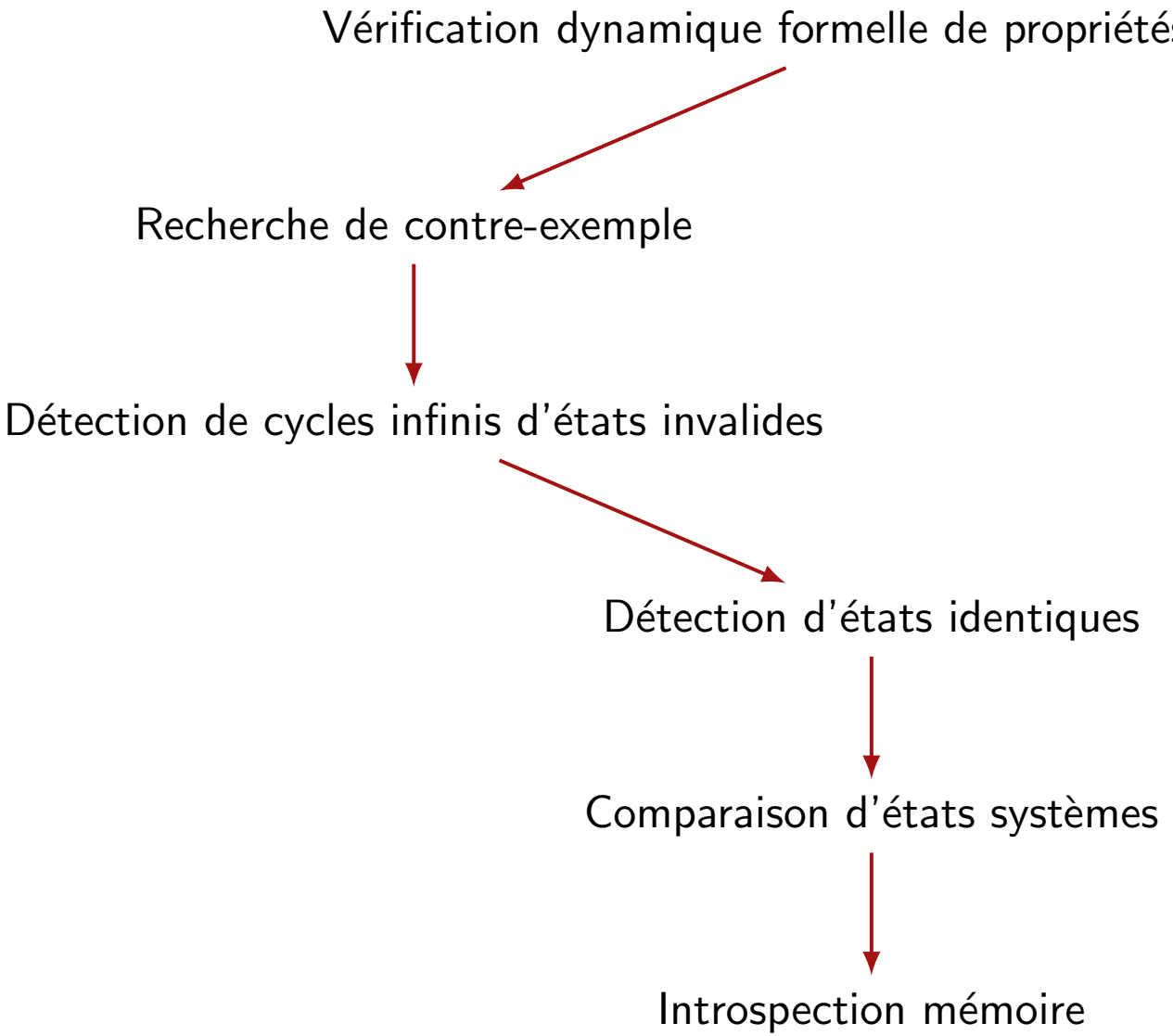
Vérification dynamique formelle de propriétés temporelles

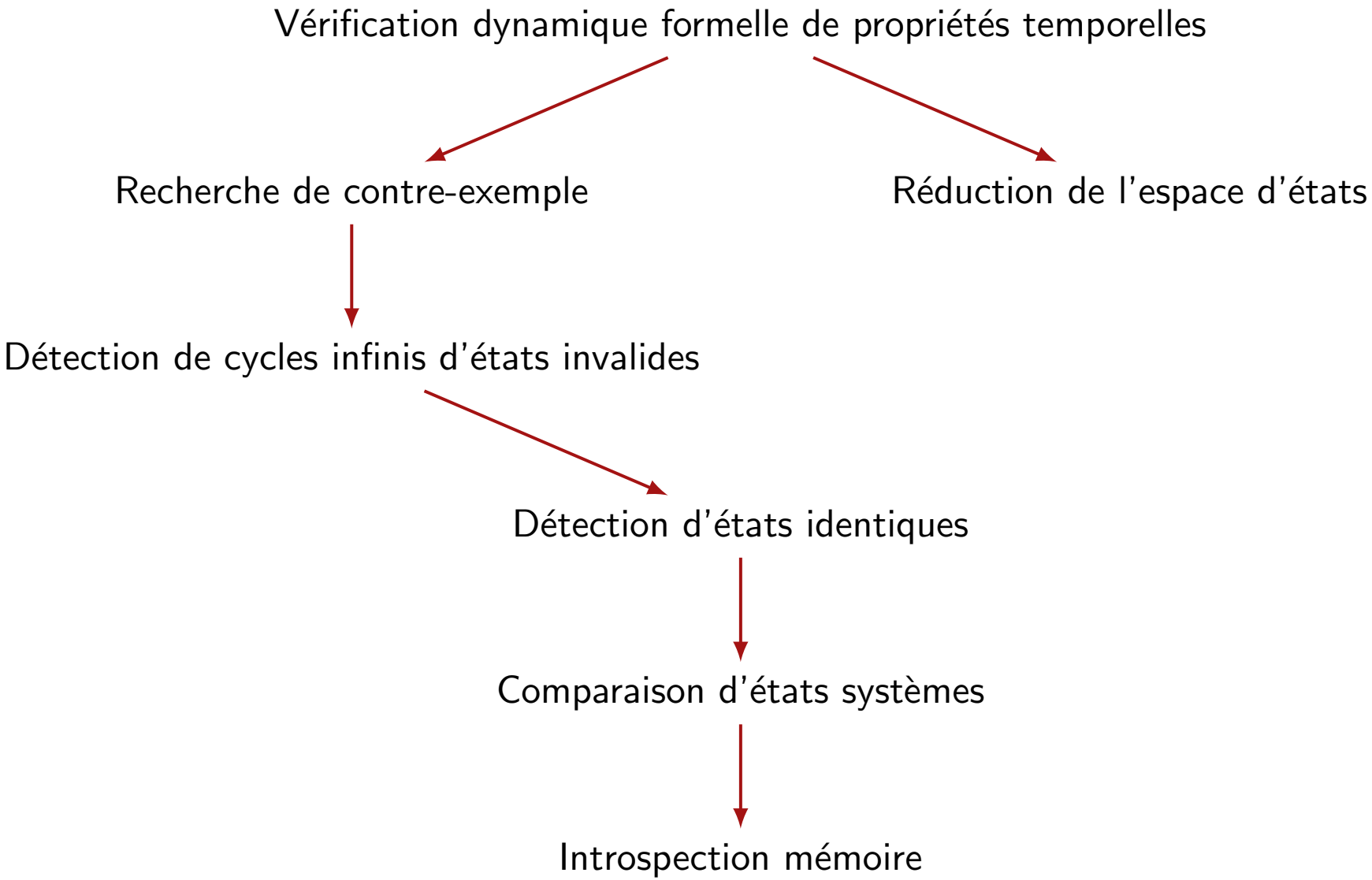
Recherche de contre-exemple

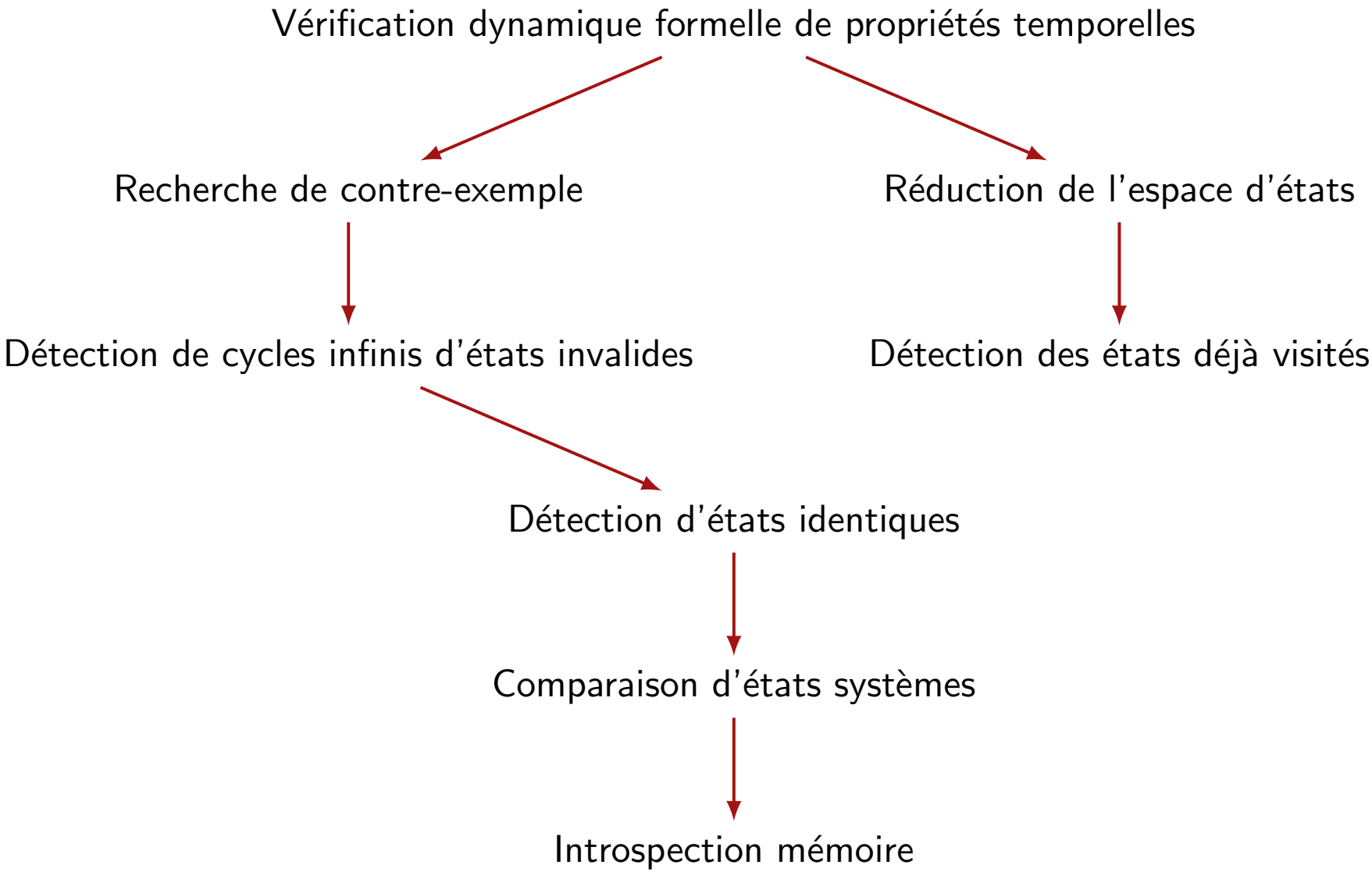
Détection de cycles infinis d'états invalides

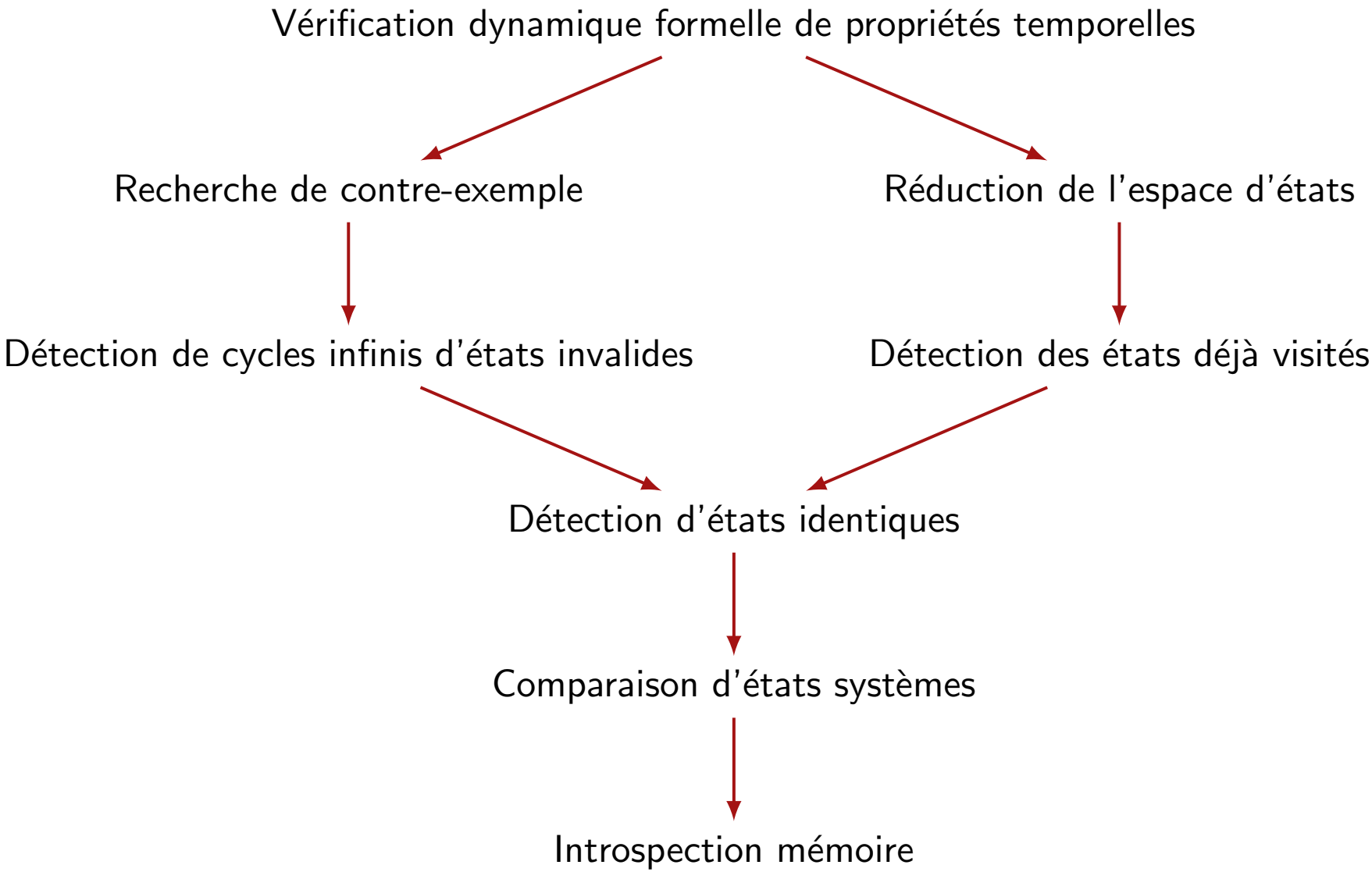

Vérification dynamique formelle de propriétés temporelles
Recherche de contre-exemple
Détection de cycles infinis d'états invalides
Détection d'états identiques

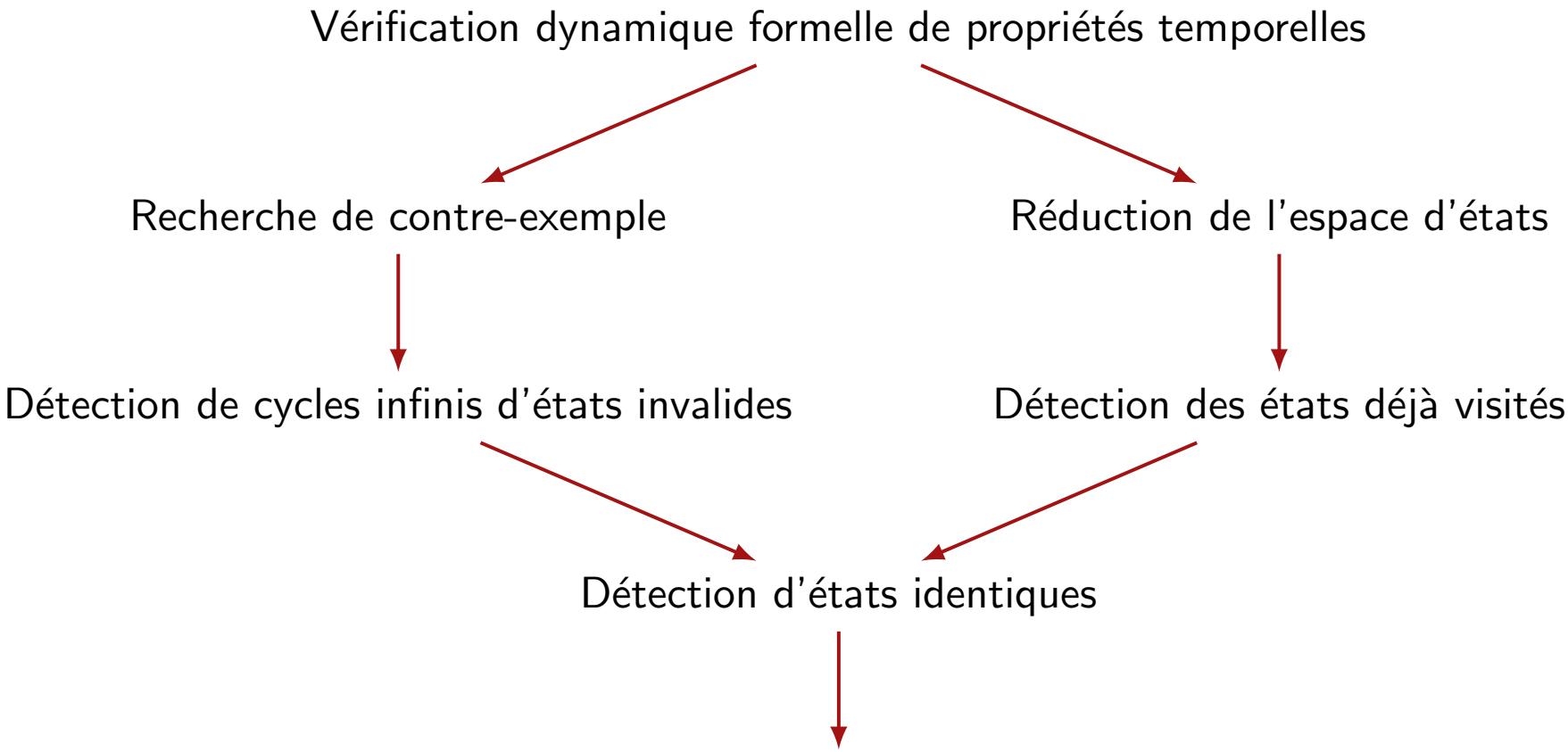

Vérification dynamique formelle de propriétés temporelles
Recherche de contre-exemple
Détection de cycles infinis d'états invalides
Détection d'états identiques
Comparaison d'états systèmes


Vérification dynamique formelle de propriétés temporelles
Recherche de contre-exemple
Détection de cycles infinis d'états invalides
Détection d'états identiques
Comparaison d'états systèmes
Introspection mémoire


Vérification dynamique formelle de propriétés temporelles
Recherche de contre-exemple
Réduction de l'espace d'états
Détection de cycles infinis d'états invalides
Détection d'états identiques
Comparaison d'états systèmes
Introspection mémoire


Vérification dynamique formelle de propriétés temporelles
Recherche de contre-exemple
Réduction de l'espace d'états
Détection de cycles infinis d'états invalides
Détection des états déjà visités
Détection d'états identiques
Comparaison d'états systèmes
Introspection mémoire


Vérification dynamique formelle de propriétés temporelles
Recherche de contre-exemple
Réduction de l'espace d'états
Détection de cycles infinis d'états invalides
Détection des états déjà visités
Détection d'états identiques
Comparaison d'états systèmes
Introspection mémoire



Contribution 1

Analyse sémantique dynamique d'un état système par introspection mémoire

État système

État système $\rightsquigarrow$ Données dynamiques

- Données (des processus) de l'application
- État du réseau

État système

État système $\rightsquigarrow$ Données dynamiques

- Données (des processus) de l'application
  - Variables globales
  - Variables locales
  - Allocations mémoire dynamiques
- État du réseau

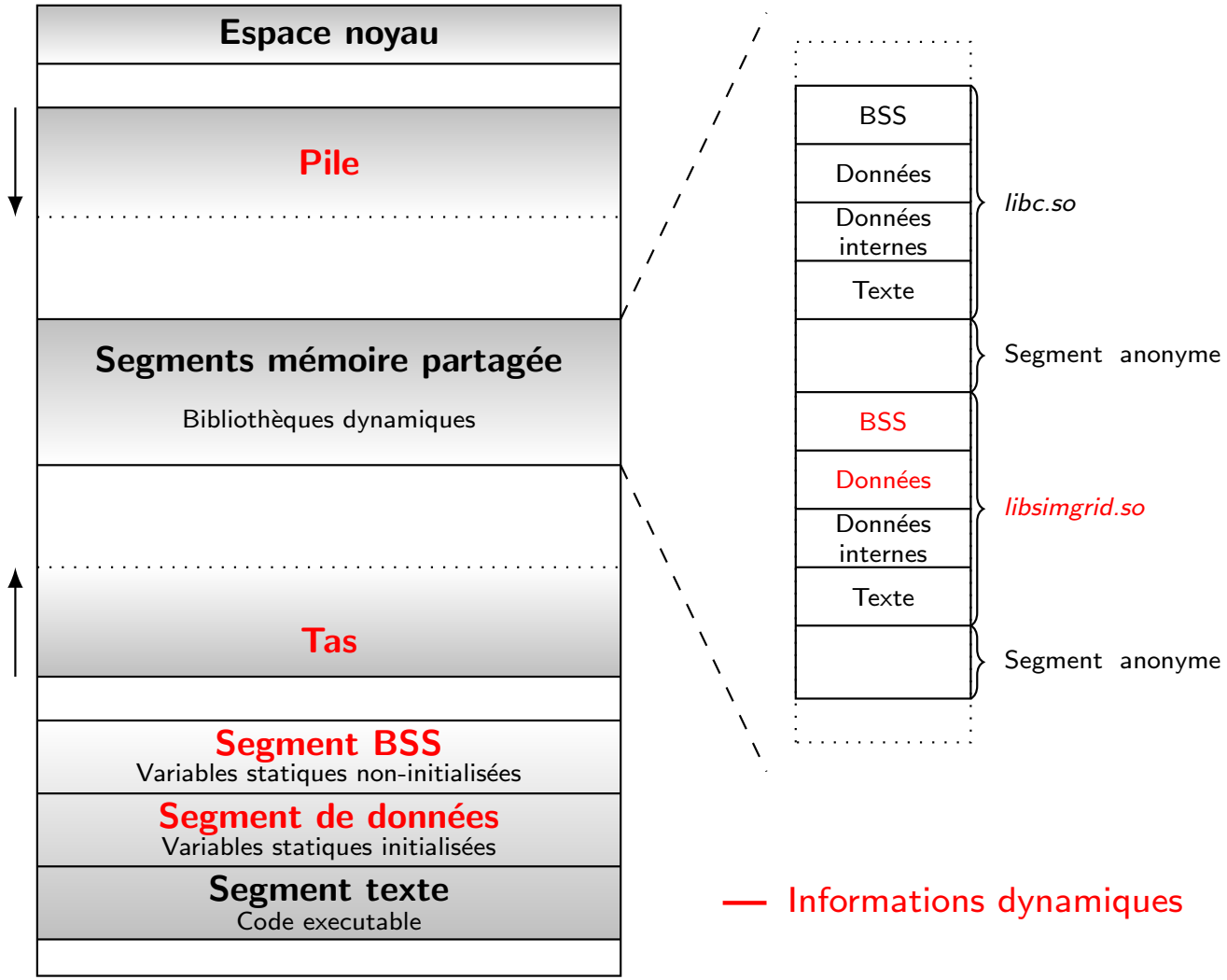
État système

Variables globales

Variables locales

Allocations mémoire dynamiques

État du réseau

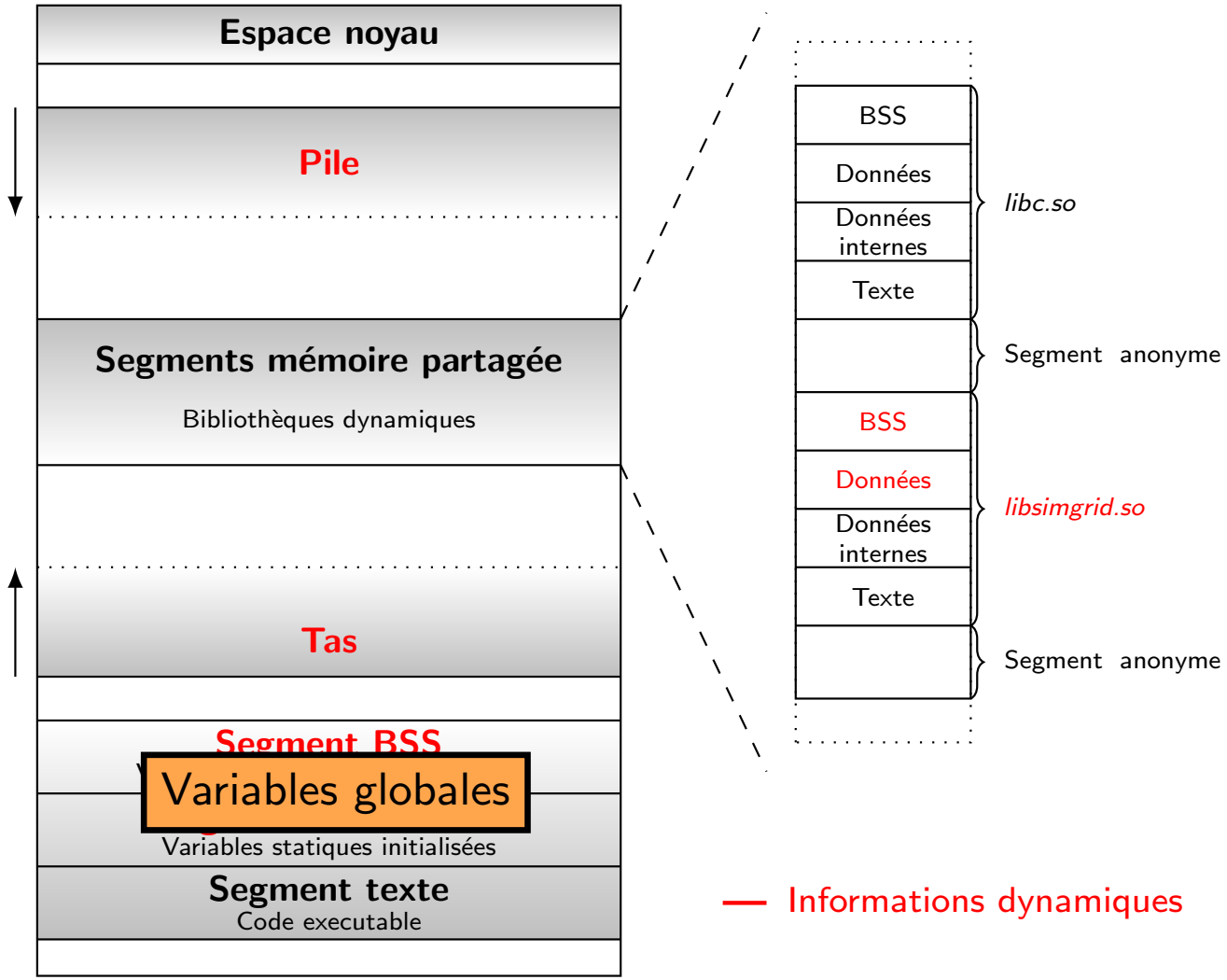


État système

Variables locales

Allocations mémoire dynamiques

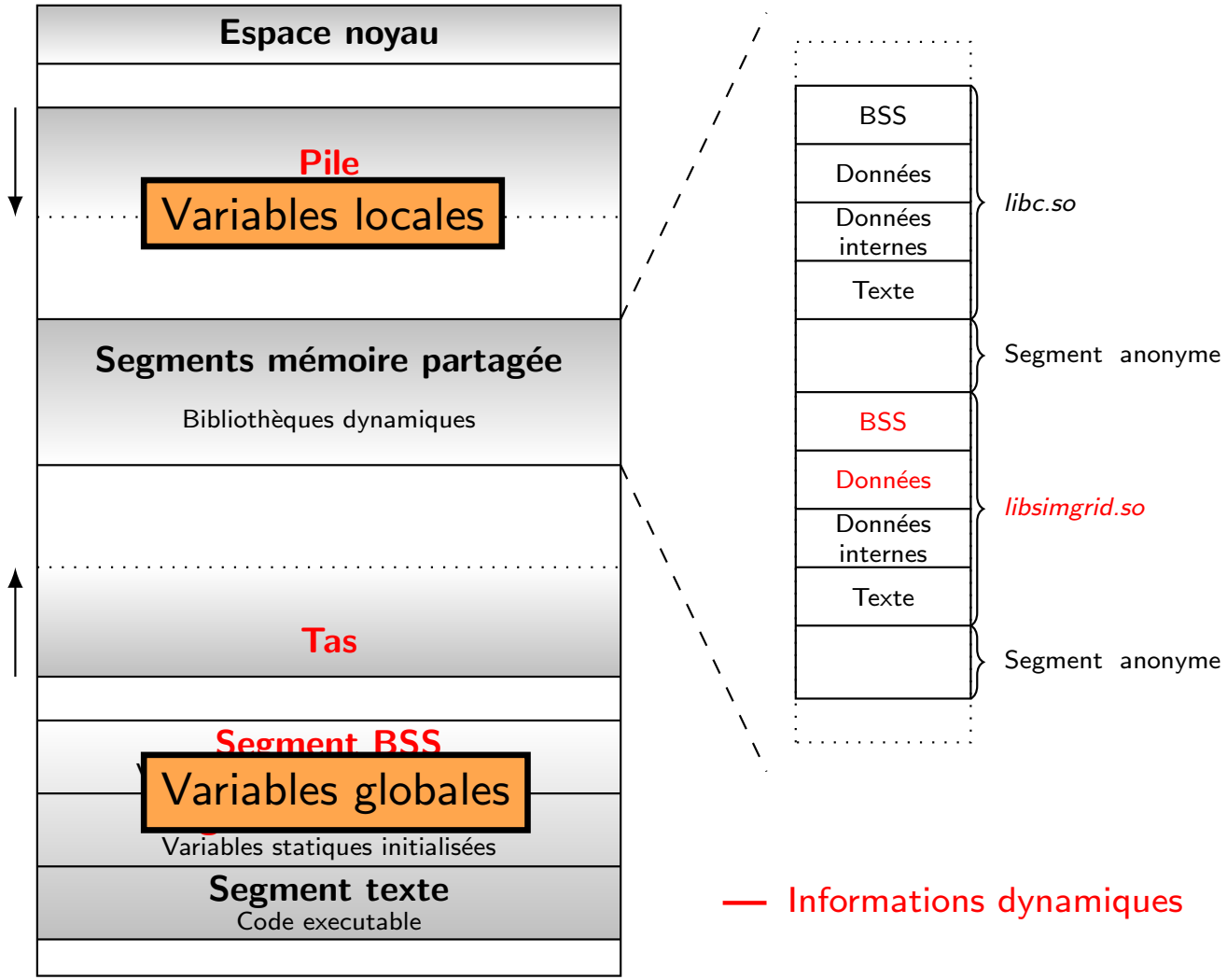
État du réseau



État système

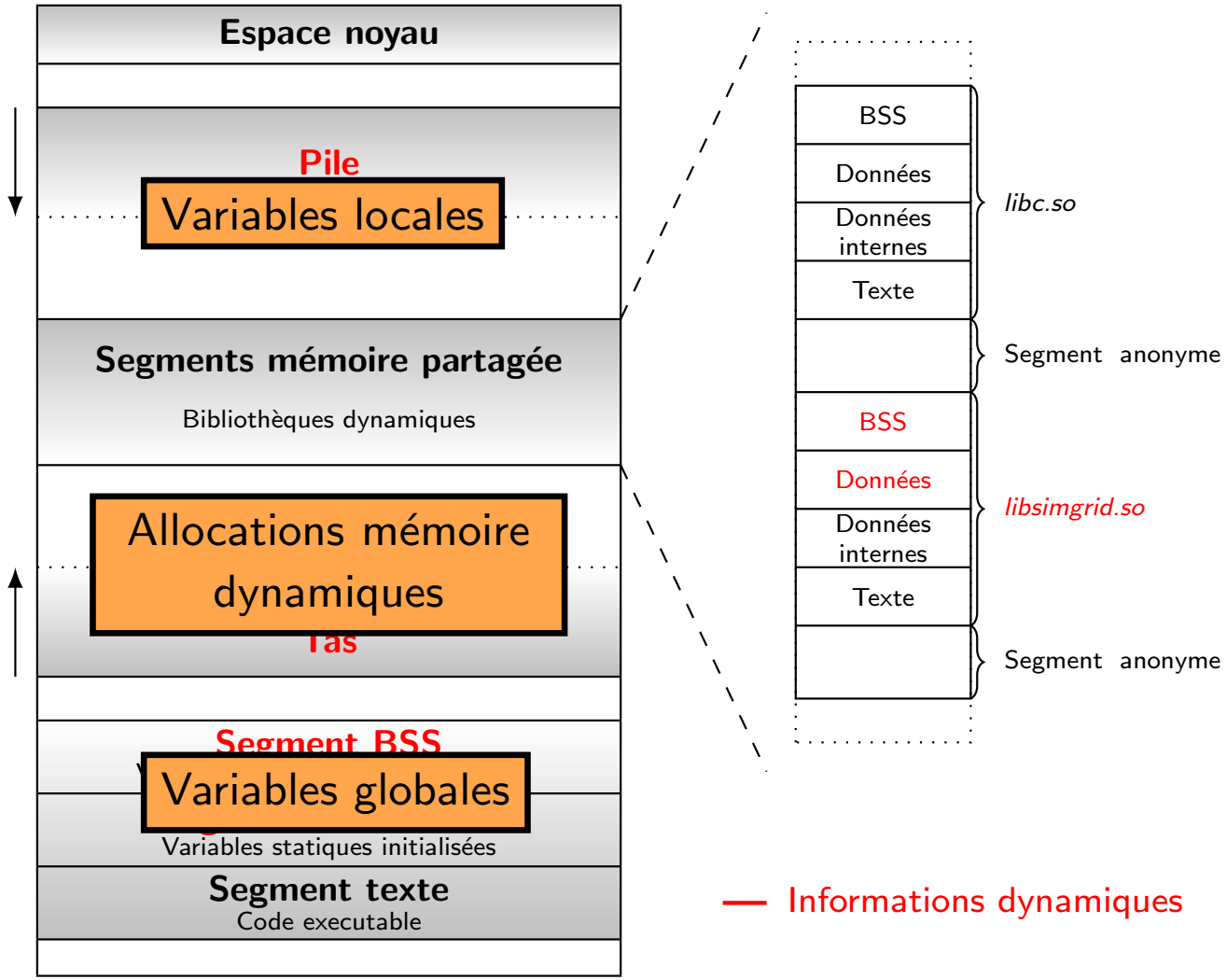
Allocations mémoire dynamiques

État du réseau

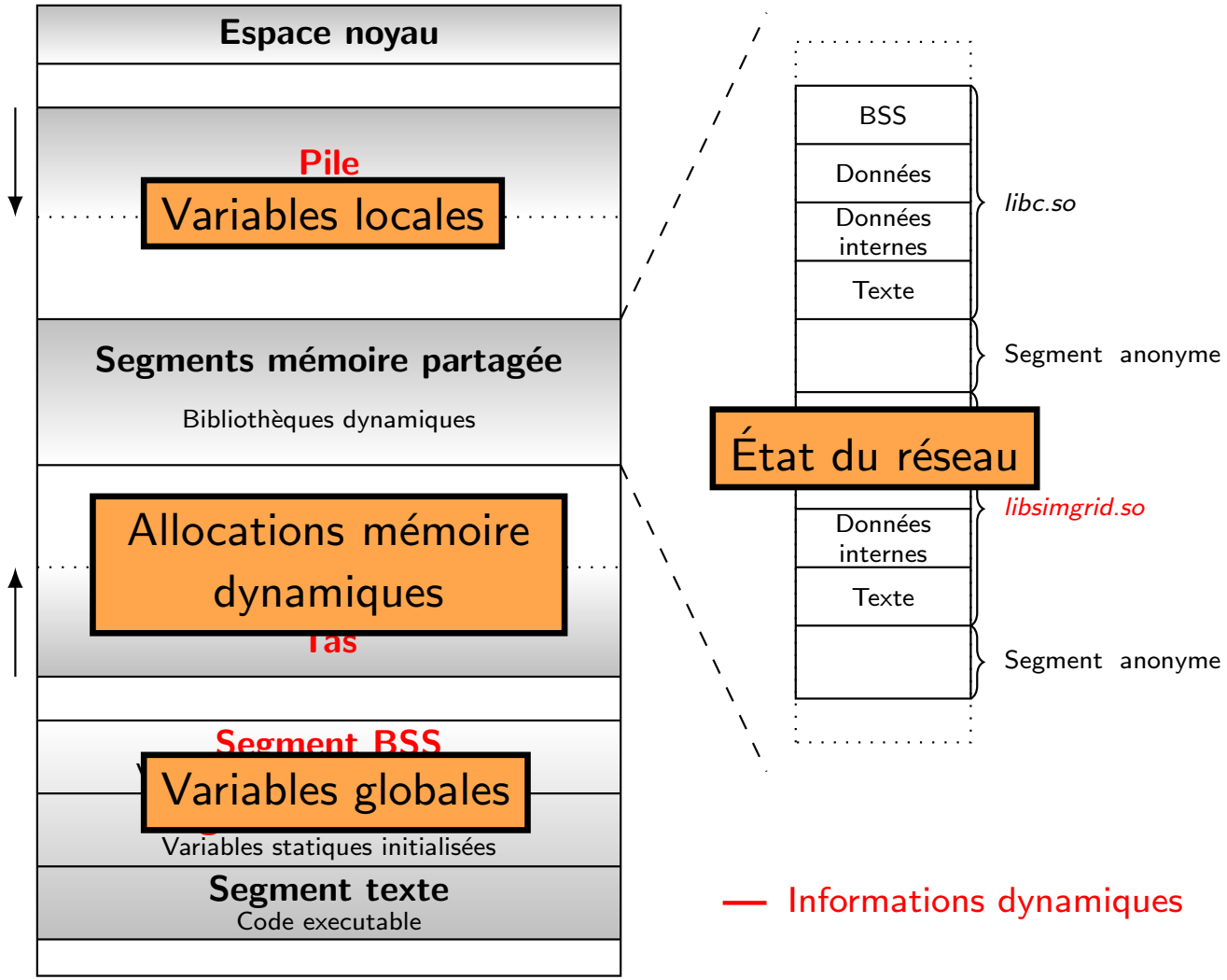


État système

État du réseau



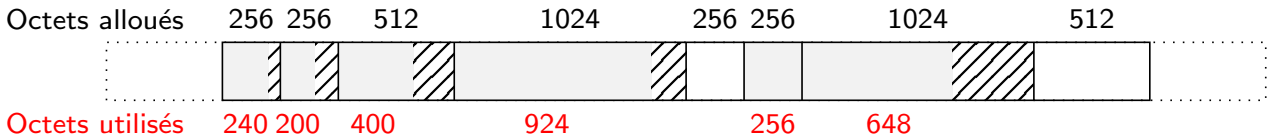
État système



Espace mémoire « creux » - Données persistantes

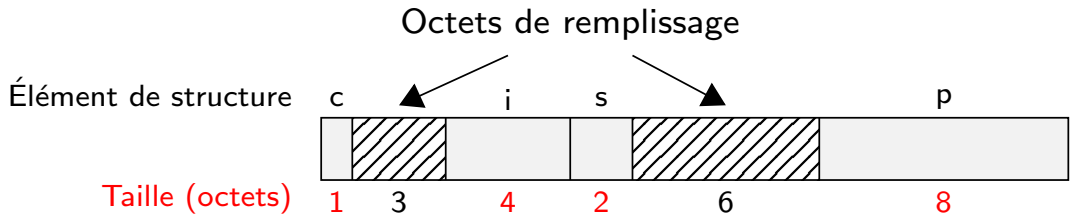
Causes

- *Overprovisioning*



- Octets de remplissage (*Padding bytes*)

```
struct foo {
   char c;
   int i;
   short s;
   void *p;
}
```



Approche initiale : Identification des octets spécifiés

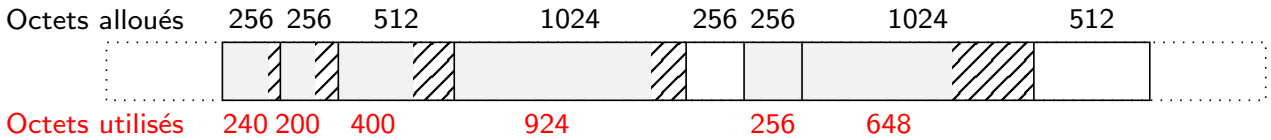
Approche complémentaire : Mise à zéro des octets non spécifiés

- Réimplémentation de malloc
- Ajout d'instructions dans le code assembleur (avec G. Corona)

Espace mémoire « creux » - Données persistantes

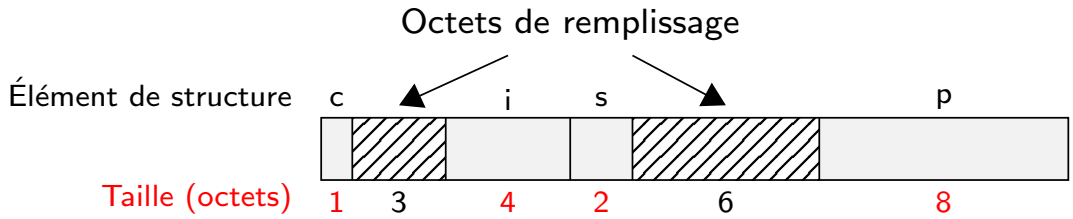
Causes

- *Overprovisioning*



- Octets de remplissage (*Padding bytes*)

```
struct foo {
   char c;
   int i;
   short s;
   void *p;
}
```



Approche initiale : Identification des octets spécifiés

Approche complémentaire : Mise à zéro des octets non spécifiés

- Réimplémentation de malloc
- Ajout d'instructions dans le code assembleur (avec G. Corona)

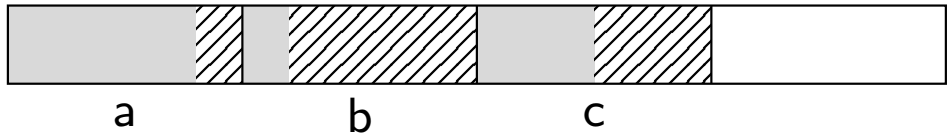
Disposition différente en mémoire

Exemple : 4 allocations (`a`, `b`, `c`, `d`) et 2 libérations (`b`, `c`)

Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

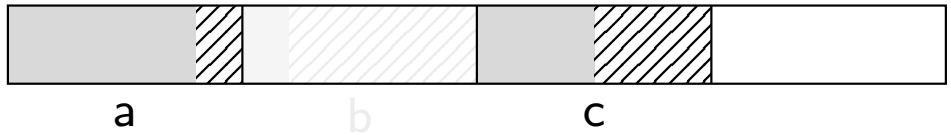
```
a = malloc(48)
b = malloc(10)
c = malloc(32)
```



Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

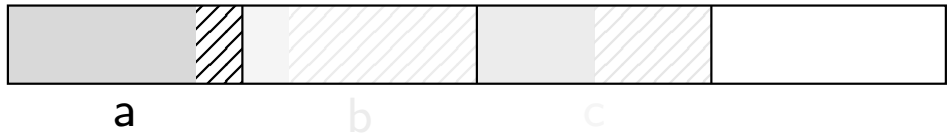
```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(b)
```



Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

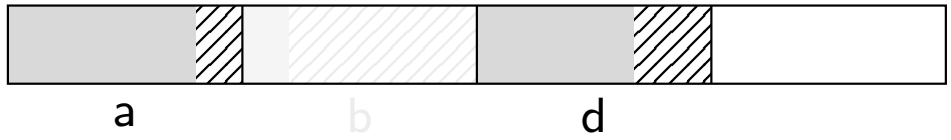
```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(b)
free(c)
```



Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

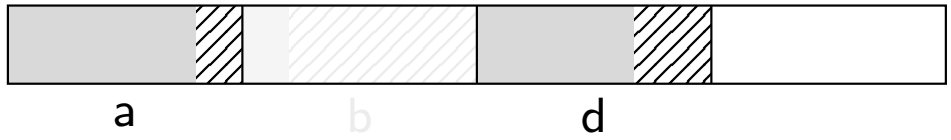
```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(b)
free(c)
d = malloc(40)
```



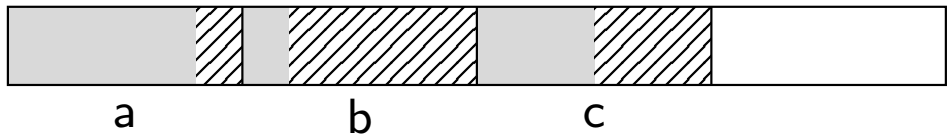
Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(b)
free(c)
d = malloc(40)
```



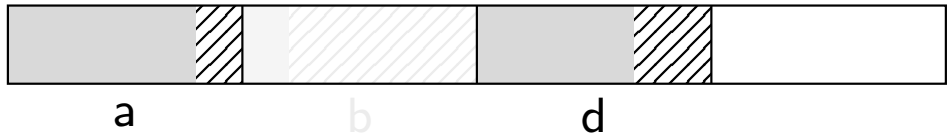

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
```



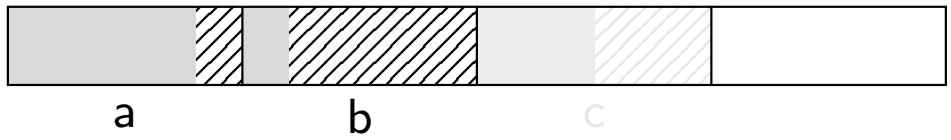
Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(b)
free(c)
d = malloc(40)
```



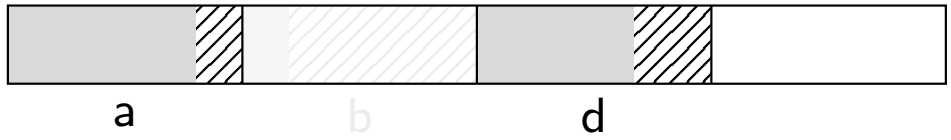

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(c)
```



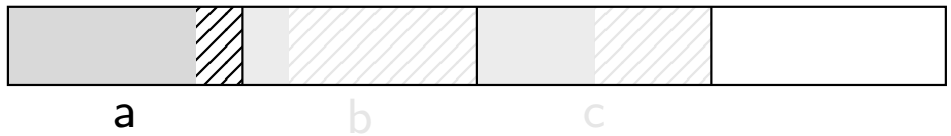
Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(b)
free(c)
d = malloc(40)
```



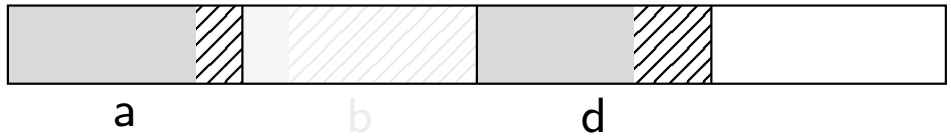

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(c)
free(b)
```



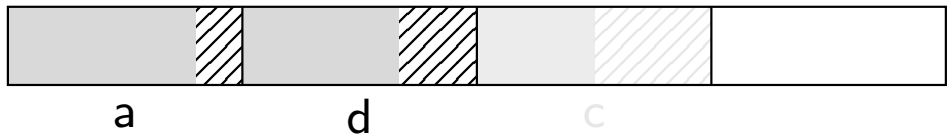
Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(b)
free(c)
d = malloc(40)
```



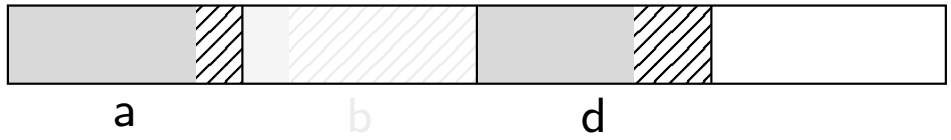

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(c)
free(b)
d = malloc(40)
```



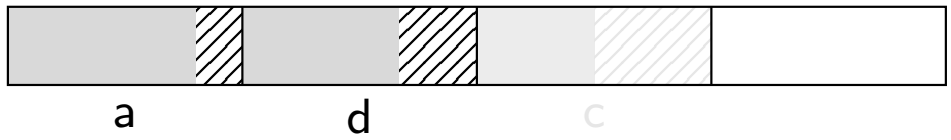
Disposition différente en mémoire

Exemple : 4 allocations (`a, b, c, d`) et 2 libérations (`b, c`)

```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(b)
free(c)
d = malloc(40)
```




```
a = malloc(48)
b = malloc(10)
c = malloc(32)
free(c)
free(b)
d = malloc(40)
```



Différences binaires, sémantique identique

Heuristique de comparaison dynamique

Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

Heuristique de comparaison dynamique

1. Détection des allocations racines

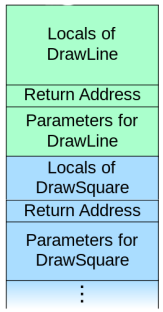
- Variables globales
- Variables locales

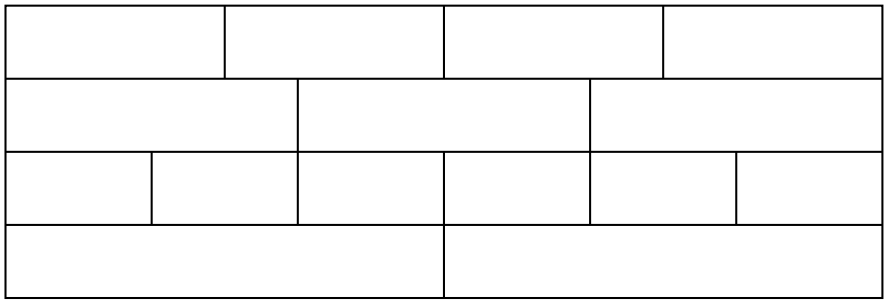
DWARF

```
<1><761>: Abbrev Number: 22 (DW_TAG_subprogram)
    <763>   DW_AT_name        : (indirect string, offset: 0xb335): client
    <769>   DW_AT_prototyped  : 1
    <76a>   DW_AT_type        : <0xc5>
    <76e>   DW_AT_low_pc      : 0x401f90
    <776>   DW_AT_high_pc     : 0x4026cf
    <77e>   DW_AT_frame_base  : 0x16c (location list)
 <2><787>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <788>   DW_AT_name        : (indirect string, offset: 0x577d): argc
    <78e>   DW_AT_type        : <0xc5>
    <792>   DW_AT_location    : 3 byte block: 91 ec 7e  (DW_OP_fbreg: -148)
 <2><796>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <797>   DW_AT_name        : (indirect string, offset: 0x5870): argv
    <79d>   DW_AT_type        : <0x2c1>
    <7a1>   DW_AT_location    : 3 byte block: 91 e0 7e  (DW_OP_fbreg: -160)
 <2><7a5>: Abbrev Number: 24 (DW_TAG_variable)
    <7a6>   DW_AT_name        : (indirect string, offset: 0x109c): pid
    <7ac>   DW_AT_type        : <0xc5>
    <7b0>   DW_AT_location    : 2 byte block: 91 6c   (DW_OP_fbreg: -20)
 <2><7b3>: Abbrev Number: 24 (DW_TAG_variable)
    <7b4>   DW_AT_name        : (indirect string, offset: 0x612): mailbox
    <7ba>   DW_AT_type        : <0x2bb>
    <7be>   DW_AT_location    : 2 byte block: 91 60   (DW_OP_fbreg: -32)
<1><9eb>: Abbrev Number: 32 (DW_TAG_variable)
    <9ec>   DW_AT_name        : global_variable
    <9f1>   DW_AT_type        : <0xc5>
    <9f5>   DW_AT_external    : 1
    <9f6>   DW_AT_location    : 9 byte block: 3 24 57 60 0 0 0 0 0  (DW_OP_addr: 605724)
```

+

libunwind





Heuristique de comparaison dynamique

1. Détection des allocations racines

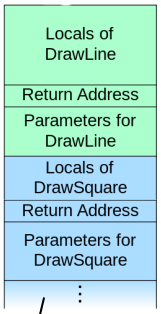
- Variables globales
- Variables locales

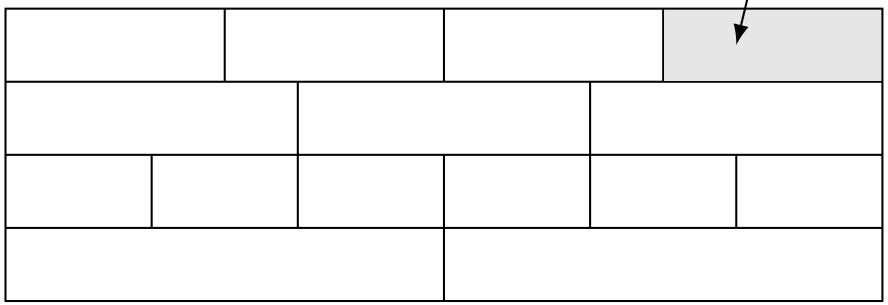
DWARF

```
<1><761>: Abbrev Number: 22 (DW_TAG_subprogram)
    <763>   DW_AT_name        : (indirect string, offset: 0xb335): client
    <769>   DW_AT_prototyped  : 1
    <76a>   DW_AT_type        : <0xc5>
    <76e>   DW_AT_low_pc      : 0x401f90
    <776>   DW_AT_high_pc     : 0x4026cf
    <77e>   DW_AT_frame_base  : 0x16c (location list)
 <2><787>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <788>   DW_AT_name        : (indirect string, offset: 0x577d): argc
    <78e>   DW_AT_type        : <0xc5>
    <792>   DW_AT_location    : 3 byte block: 91 ec 7e  (DW_OP_fbreg: -148)
 <2><796>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <797>   DW_AT_name        : (indirect string, offset: 0x5870): argv
    <79d>   DW_AT_type        : <0x2c1>
    <7a1>   DW_AT_location    : 3 byte block: 91 e0 7e  (DW_OP_fbreg: -160)
 <2><7a5>: Abbrev Number: 24 (DW_TAG_variable)
    <7a6>   DW_AT_name        : (indirect string, offset: 0x109c): pid
    <7ac>   DW_AT_type        : <0xc5>
    <7b0>   DW_AT_location    : 2 byte block: 91 6c   (DW_OP_fbreg: -20)
 <2><7b3>: Abbrev Number: 24 (DW_TAG_variable)
    <7b4>   DW_AT_name        : (indirect string, offset: 0x612): mailbox
    <7ba>   DW_AT_type        : <0x2bb>
    <7be>   DW_AT_location    : 2 byte block: 91 60   (DW_OP_fbreg: -32)
<1><9eb>: Abbrev Number: 32 (DW_TAG_variable)
    <9ec>   DW_AT_name        : global_variable
    <9f1>   DW_AT_type        : <0xc5>
    <9f5>   DW_AT_external    : 1
    <9f6>   DW_AT_location    : 9 byte block: 3 24 57 60 0 0 0 0 0  (DW_OP_addr: 605724)
```

+

libunwind





Heuristique de comparaison dynamique

1. Détection des allocations racines

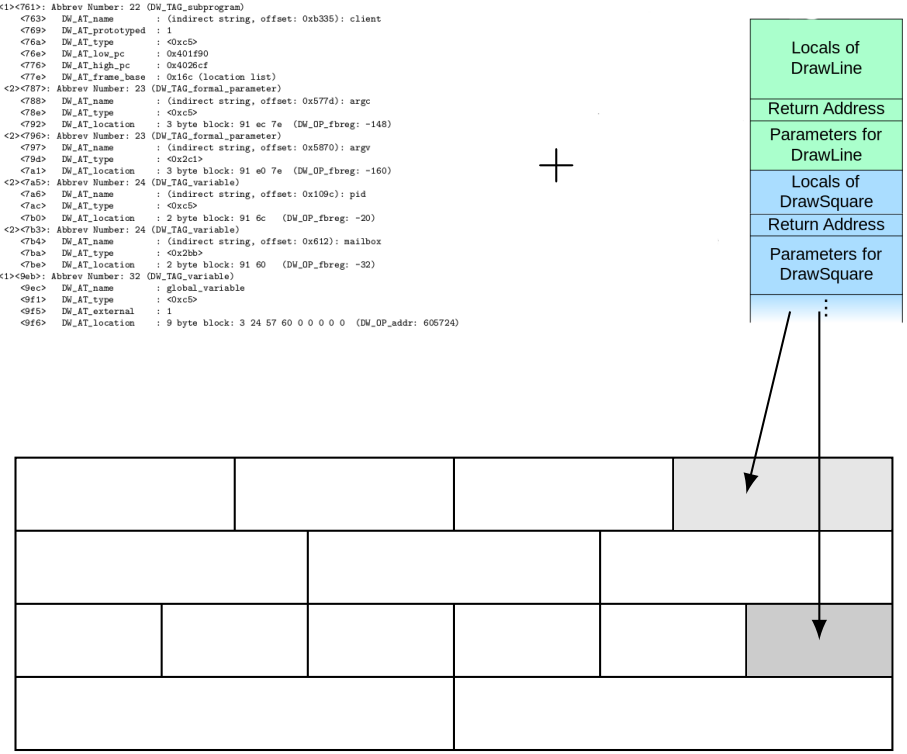
- Variables globales
- Variables locales

DWARF

```
<1><761>: Abbrev Number: 22 (DW_TAG_subprogram)
    <763>   DW_AT_name        : (indirect string, offset: 0xb335): client
    <769>   DW_AT_prototyped  : 1
    <76a>   DW_AT_type        : <0xc5>
    <76e>   DW_AT_low_pc      : 0x401f90
    <776>   DW_AT_high_pc     : 0x4026cf
    <77e>   DW_AT_frame_base  : 0x16c (location list)
 <2><787>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <788>   DW_AT_name        : (indirect string, offset: 0x577d): argc
    <78e>   DW_AT_type        : <0xc5>
    <792>   DW_AT_location    : 3 byte block: 91 ec 7e  (DW_OP_fbreg: -148)
 <2><796>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <797>   DW_AT_name        : (indirect string, offset: 0x5870): argv
    <79d>   DW_AT_type        : <0x2c1>
    <7a1>   DW_AT_location    : 3 byte block: 91 e0 7e  (DW_OP_fbreg: -160)
 <2><7a5>: Abbrev Number: 24 (DW_TAG_variable)
    <7a6>   DW_AT_name        : (indirect string, offset: 0x109c): pid
    <7ac>   DW_AT_type        : <0xc5>
    <7b0>   DW_AT_location    : 2 byte block: 91 6c   (DW_OP_fbreg: -20)
 <2><7b3>: Abbrev Number: 24 (DW_TAG_variable)
    <7b4>   DW_AT_name        : (indirect string, offset: 0x612): mailbox
    <7ba>   DW_AT_type        : <0x2bb>
    <7be>   DW_AT_location    : 2 byte block: 91 60   (DW_OP_fbreg: -32)
<1><9eb>: Abbrev Number: 32 (DW_TAG_variable)
    <9ec>   DW_AT_name        : global_variable
    <9f1>   DW_AT_type        : <0xc5>
    <9f5>   DW_AT_external    : 1
    <9f6>   DW_AT_location    : 9 byte block: 3 24 57 60 0 0 0 0 0  (DW_OP_addr: 605724)
```

+

libunwind



Heuristique de comparaison dynamique

1. Détection des allocations racines

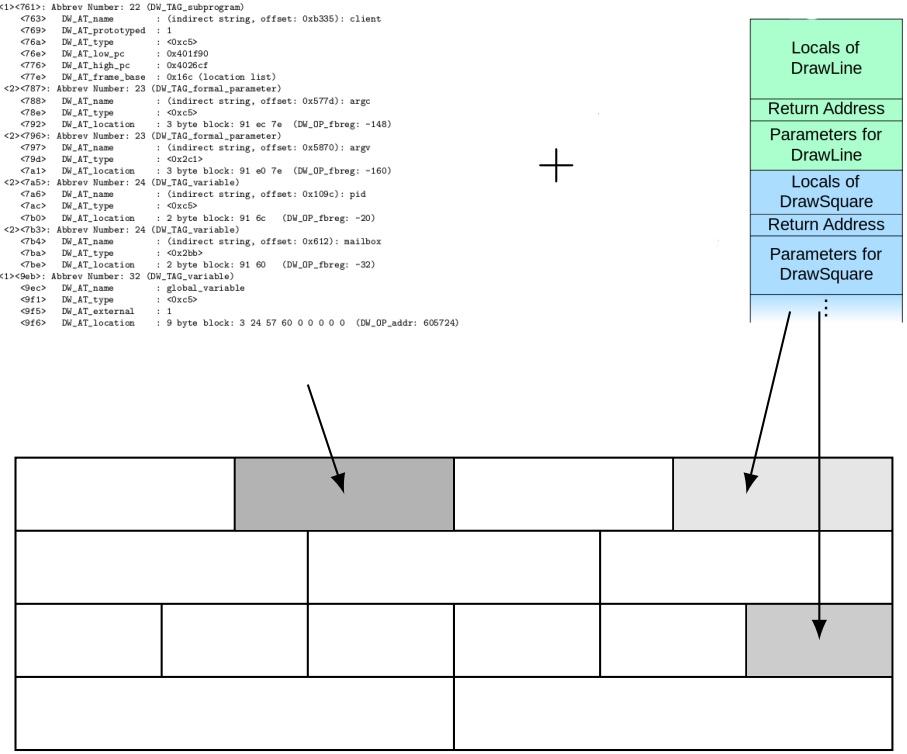
- Variables globales
- Variables locales

DWARF

```
<1><761>: Abbrev Number: 22 (DW_TAG_subprogram)
    <763>   DW_AT_name        : (indirect string, offset: 0xb335): client
    <769>   DW_AT_prototyped  : 1
    <76a>   DW_AT_type        : <0xc5>
    <76e>   DW_AT_low_pc      : 0x401f90
    <776>   DW_AT_high_pc     : 0x4026cf
    <77e>   DW_AT_frame_base  : 0x16c (location list)
 <2><787>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <788>   DW_AT_name        : (indirect string, offset: 0x577d): argc
    <78e>   DW_AT_type        : <0xc5>
    <792>   DW_AT_location    : 3 byte block: 91 ec 7e  (DW_OP_fbreg: -148)
 <2><796>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <797>   DW_AT_name        : (indirect string, offset: 0x5870): argv
    <79d>   DW_AT_type        : <0x2c1>
    <7a1>   DW_AT_location    : 3 byte block: 91 e0 7e  (DW_OP_fbreg: -160)
 <2><7a5>: Abbrev Number: 24 (DW_TAG_variable)
    <7a6>   DW_AT_name        : (indirect string, offset: 0x109c): pid
    <7ac>   DW_AT_type        : <0xc5>
    <7b0>   DW_AT_location    : 2 byte block: 91 6c   (DW_OP_fbreg: -20)
 <2><7b3>: Abbrev Number: 24 (DW_TAG_variable)
    <7b4>   DW_AT_name        : (indirect string, offset: 0x612): mailbox
    <7ba>   DW_AT_type        : <0x2bb>
    <7be>   DW_AT_location    : 2 byte block: 91 60   (DW_OP_fbreg: -32)
<1><9eb>: Abbrev Number: 32 (DW_TAG_variable)
    <9ec>   DW_AT_name        : global_variable
    <9f1>   DW_AT_type        : <0xc5>
    <9f5>   DW_AT_external    : 1
    <9f6>   DW_AT_location    : 9 byte block: 3 24 57 60 0 0 0 0 0  (DW_OP_addr: 605724)
```

+

libunwind



Heuristique de comparaison dynamique

1. Détection des allocations racines

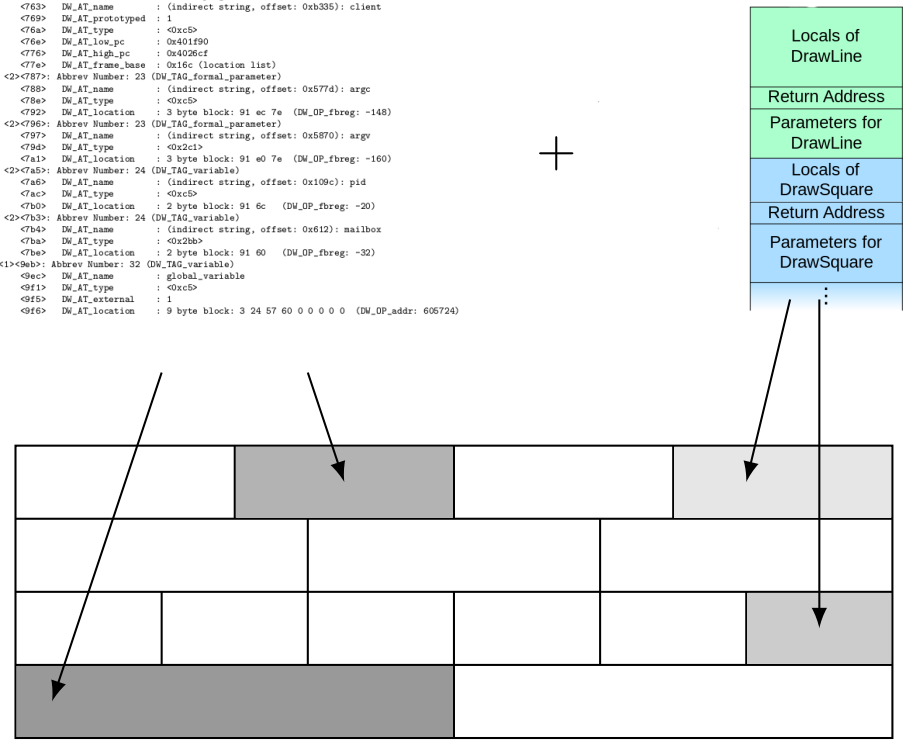
- Variables globales
- Variables locales

DWARF

```
<1><761>: Abbrev Number: 22 (DW_TAG_subprogram)
    <763>   DW_AT_name        : (indirect string, offset: 0xb335): client
    <769>   DW_AT_prototyped  : 1
    <76a>   DW_AT_type        : <0xc5>
    <76e>   DW_AT_low_pc      : 0x401f90
    <776>   DW_AT_high_pc     : 0x4026cf
    <77e>   DW_AT_frame_base  : 0x16c (location list)
 <2><787>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <788>   DW_AT_name        : (indirect string, offset: 0x577d): argc
    <78e>   DW_AT_type        : <0xc5>
    <792>   DW_AT_location    : 3 byte block: 91 ec 7e  (DW_OP_fbreg: -148)
 <2><796>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <797>   DW_AT_name        : (indirect string, offset: 0x5870): argv
    <79d>   DW_AT_type        : <0x2c1>
    <7a1>   DW_AT_location    : 3 byte block: 91 e0 7e  (DW_OP_fbreg: -160)
 <2><7a5>: Abbrev Number: 24 (DW_TAG_variable)
    <7a6>   DW_AT_name        : (indirect string, offset: 0x109c): pid
    <7ac>   DW_AT_type        : <0xc5>
    <7b0>   DW_AT_location    : 2 byte block: 91 6c   (DW_OP_fbreg: -20)
 <2><7b3>: Abbrev Number: 24 (DW_TAG_variable)
    <7b4>   DW_AT_name        : (indirect string, offset: 0x612): mailbox
    <7ba>   DW_AT_type        : <0x2bb>
    <7be>   DW_AT_location    : 2 byte block: 91 60   (DW_OP_fbreg: -32)
<1><9eb>: Abbrev Number: 32 (DW_TAG_variable)
    <9ec>   DW_AT_name        : global_variable
    <9f1>   DW_AT_type        : <0xc5>
    <9f5>   DW_AT_external    : 1
    <9f6>   DW_AT_location    : 9 byte block: 3 24 57 60 0 0 0 0 0  (DW_OP_addr: 605724)
```

+

libunwind



Heuristique de comparaison dynamique

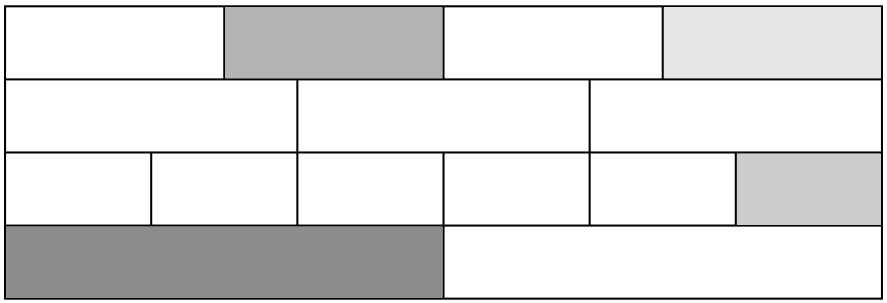
1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

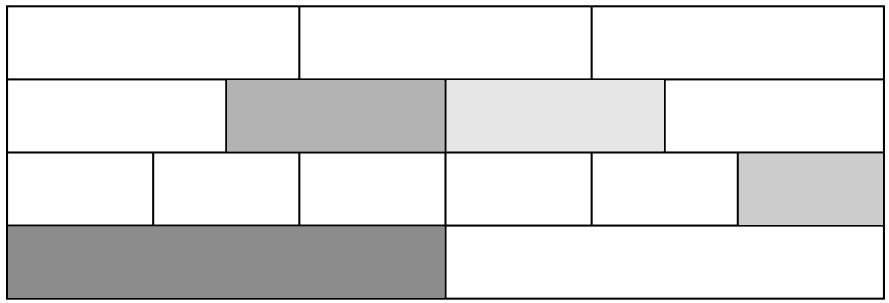
2. **“Mark and sweep”**

Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

2. **“Mark and sweep”**

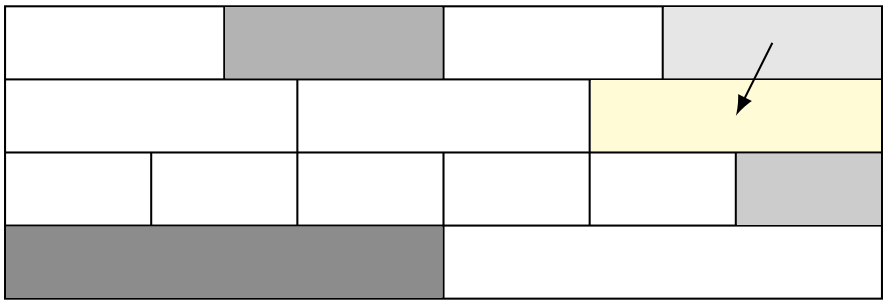


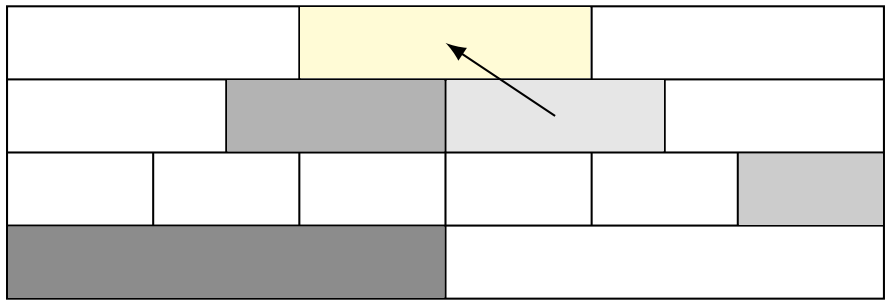


Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

2. **“Mark and sweep”**

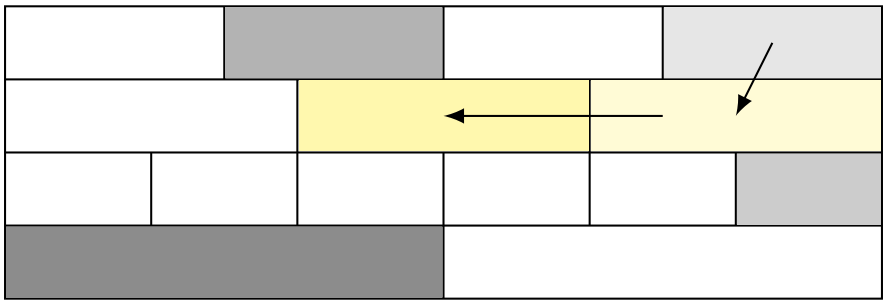


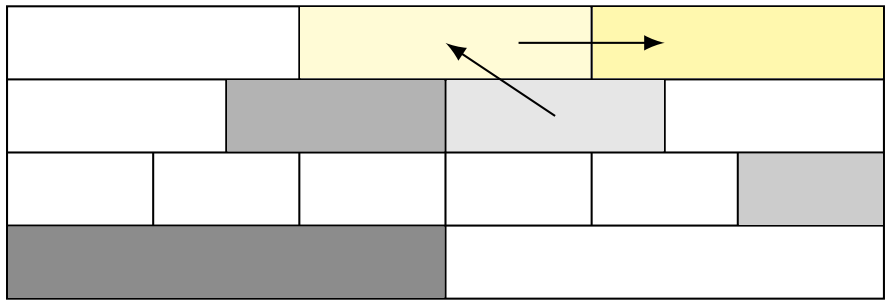


Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

2. **“Mark and sweep”**

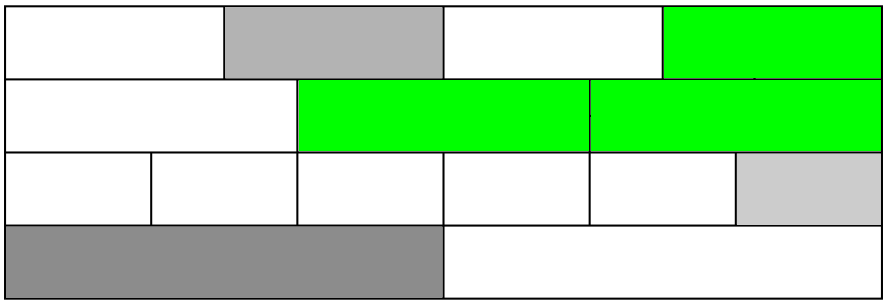


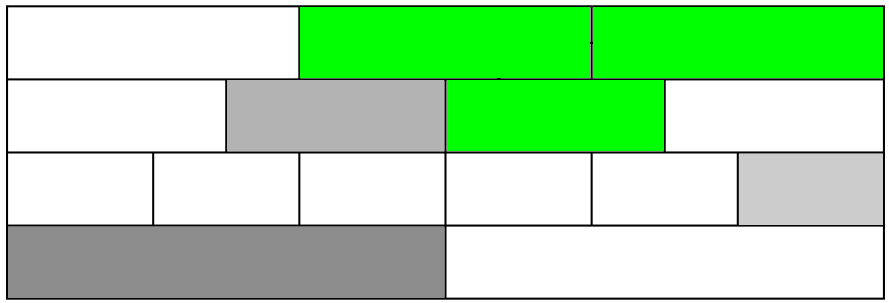


Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

2. **“Mark and sweep”**

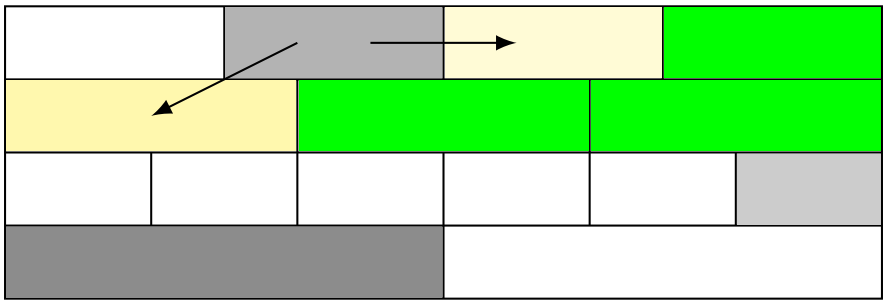


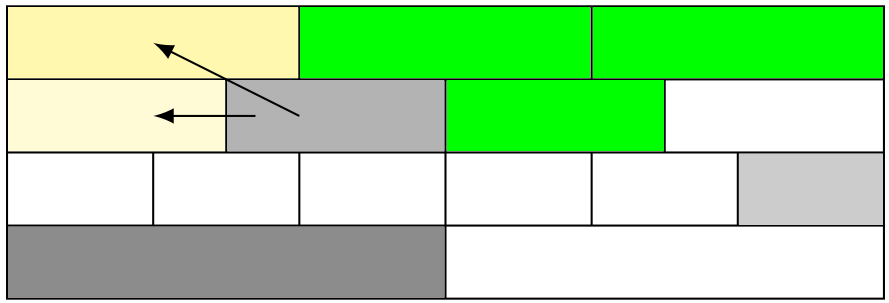


Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

2. **“Mark and sweep”**

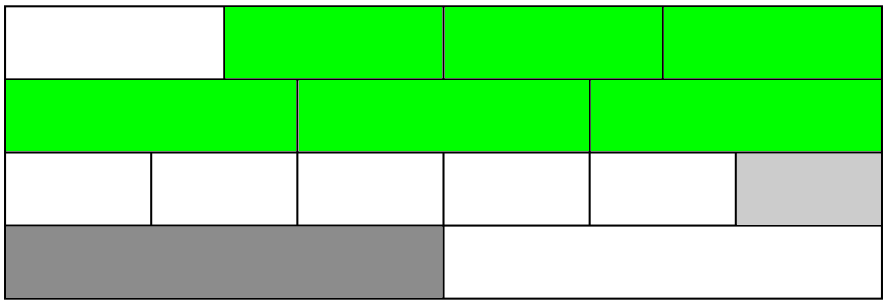


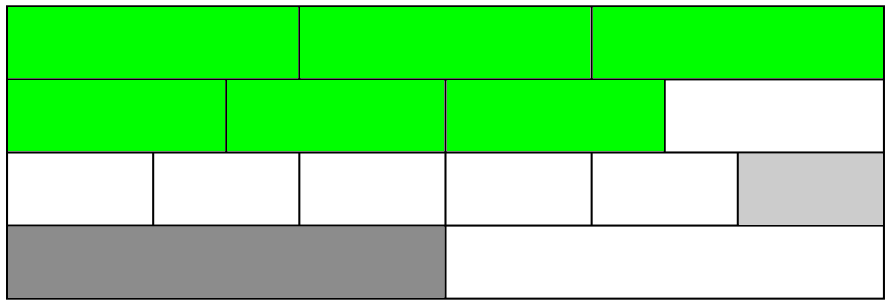


Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

2. **“Mark and sweep”**

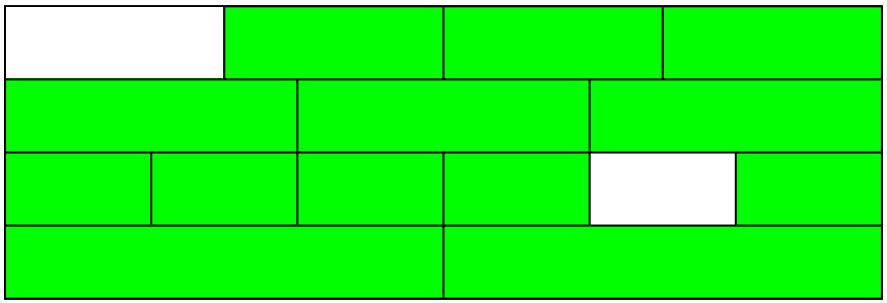


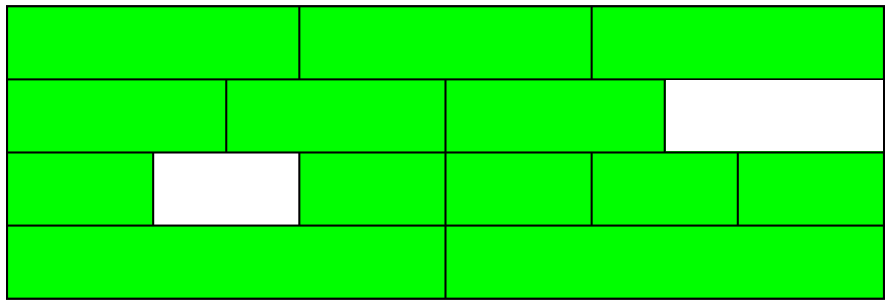


Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

2. **“Mark and sweep”**

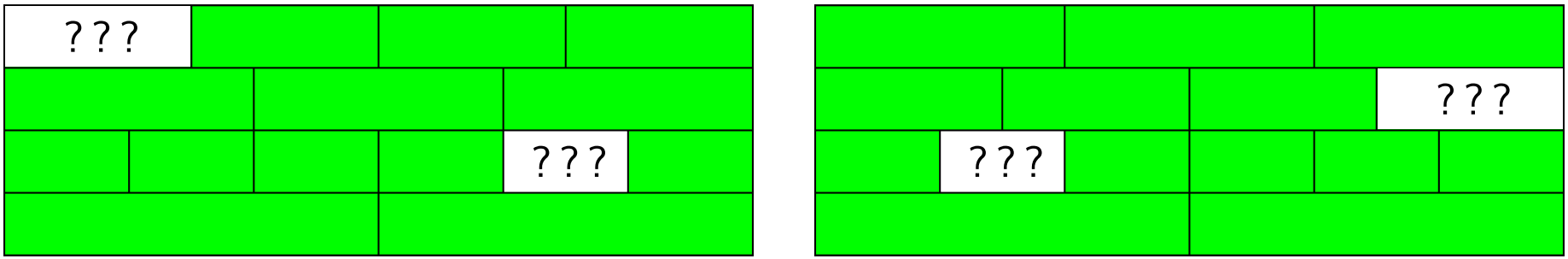




Heuristique de comparaison dynamique

1. **Détection des allocations racines**
   - Variables globales
   - Variables locales

2. **“Mark and sweep”**



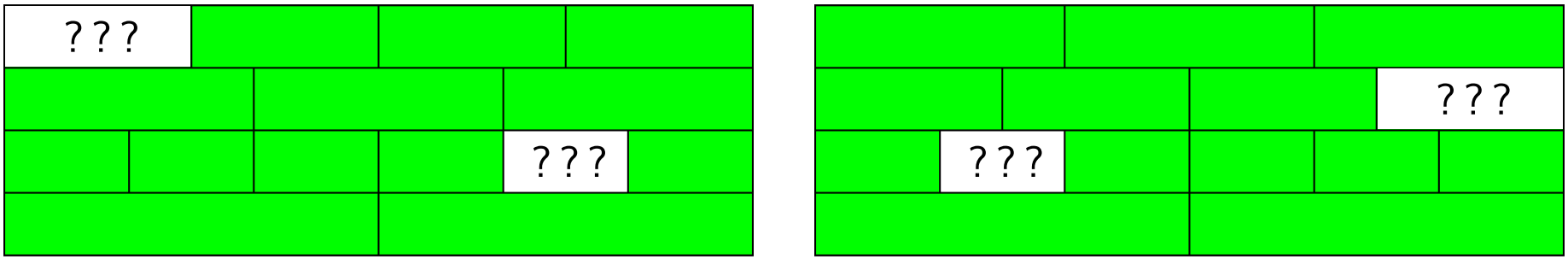
? ? ? : Zones mémoire non vides, non marquées après *mark and sweep*

Heuristique de comparaison dynamique

1. Détection des allocations racines

- Variables globales
- Variables locales

2. "Mark and sweep"



? ? ? : Zones mémoire non vides, non marquées après *mark and sweep*

3. Comparaison binaire des zones non marquées

Détection d'états sémantiquement identiques

Verrou	**Solution pour le tas**	**Solution pour la pile**
Overprovisioning	Mise à 0 des octets supplémentaires	Détection du sommet de pile
Octets de remplissage	Mise à 0 des octets supplémentaires	
Différences binaires	Heuristique de comparaison sémantique	-
Variables non-initialisées	Mise à 0 des octets supplémentaires	
Différences non-pertinentes	Omission de zones explicites	DWARF + libunwind + omission

Évaluation expérimentale

Terminaison de programmes

- Problème indécidable en général
- Détection de cycles de non-progression
- Reproduction de résultats de SV-COMP (2014 - 2015)

Réduction de l'espace d'états

- Exploration exhaustive de tests d'intégration MPICH

Application	# P	Exploration sans réduction			Exploration avec réduction		
		# États	Temps	Consommation mémoire	# États	Temps	Consommation mémoire
bcasttest (C)	3	$>$ 1 million	$>$ 24 h	-	4 823	18 min 31 s	37 Go
groupcreate (C)	6	$>$ 272 millions	$>$ 24 h	-	80 878	16 h 15 min	89.31 Go
inplacef (Fortran)	3	$>$ 182 millions	$>$ 24 h	-	2 941	1 min 07 s	3.87 Go

- ☺ Détection des états déjà visités $\rightsquigarrow$ Réduction de l'espace d'état efficace
- ☹ Consommation mémoire importante

Évaluation expérimentale

Terminaison de programmes

- Problème indécidable en général
- Détection de cycles de non-progression
- Reproduction de résultats de SV-COMP (2014 - 2015)

Réduction de l'espace d'états

- Exploration exhaustive de tests d'intégration MPICH

Application	**# P**	**Exploration sans réduction**			**Exploration avec réduction**		
		# États	**Temps**	**Consommation mémoire**	**# États**	**Temps**	**Consommation mémoire**
bcasttest (C)	3	$>$ 1 million	$>$ 24 h	-	4 823	18 min 31 s	37 Go
groupcreate (C)	6	$>$ 272 millions	$>$ 24 h	-	80 878	16 h 15 min	89.31 Go
inplacef (Fortran)	3	$>$ 182 millions	$>$ 24 h	-	2 941	1 min 07 s	3.87 Go

☺ Détection des états déjà visités $\rightsquigarrow$ Réduction de l'espace d'état efficace

☹ Consommation mémoire importante

Évaluation expérimentale

Terminaison de programmes

- Problème indécidable en général
- Détection de cycles de non-progression
- Reproduction de résultats de SV-COMP (2014 - 2015)

Réduction de l'espace d'états

- Exploration exhaustive de tests d'intégration MPICH

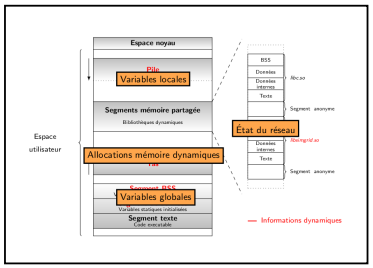
Application	**# P**	**Exploration sans réduction**			**Exploration avec réduction**		
		# États	**Temps**	**Consommation mémoire**	**# États**	**Temps**	**Consommation mémoire**
bcasttest (C)	3	$>$ 1 million	$>$ 24 h	-	4 823	18 min 31 s	37 Go
groupcreate (C)	6	$>$ 272 millions	$>$ 24 h	-	80 878	16 h 15 min	89.31 Go
inplacef (Fortran)	3	$>$ 182 millions	$>$ 24 h	-	2 941	1 min 07 s	3.87 Go

☺ Détection des états déjà visités $\rightsquigarrow$ Réduction de l'espace d'état efficace

☹ Consommation mémoire importante

Compression mémoire

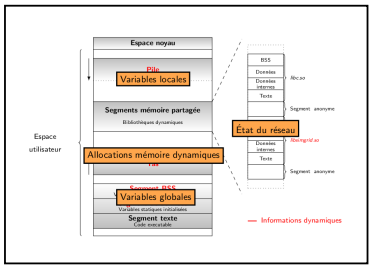
État système s_n



Travail en collaboration avec G. Corona

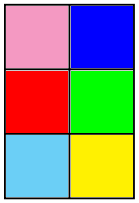
Compression mémoire

État système s_n



$\rightsquigarrow$

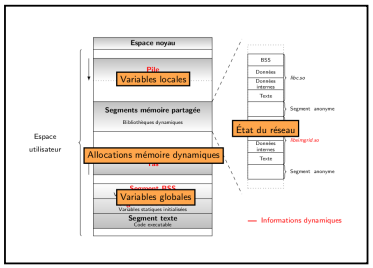
Ensemble de pages mémoires



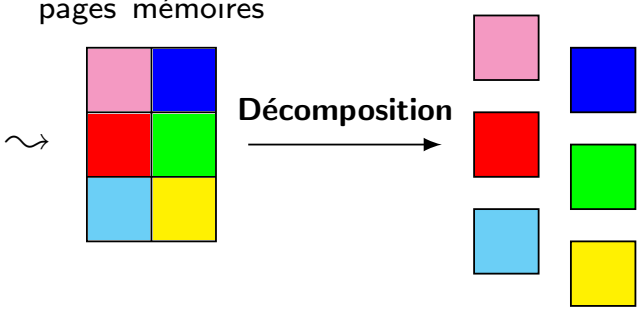
Travail en collaboration avec G. Corona

Compression mémoire

État système s_n



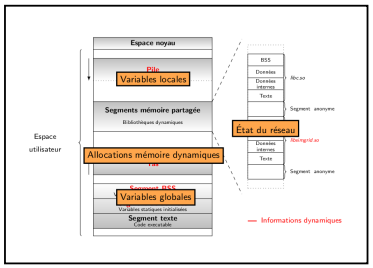
Ensemble de pages mémoires



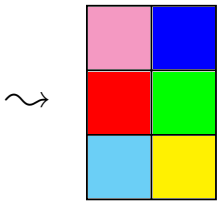
Travail en collaboration avec G. Corona

Compression mémoire

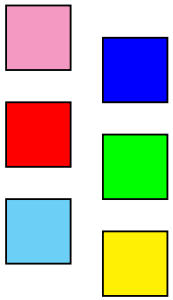
État système s_n



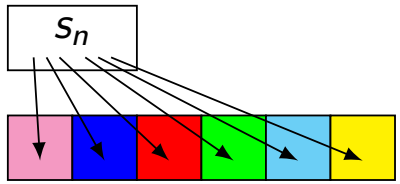
Ensemble de pages mémoires



Décomposition



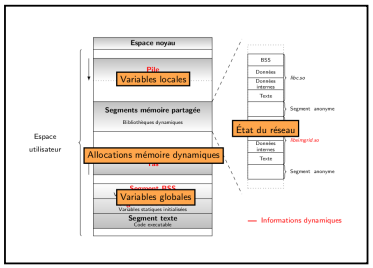
Sauvegarde



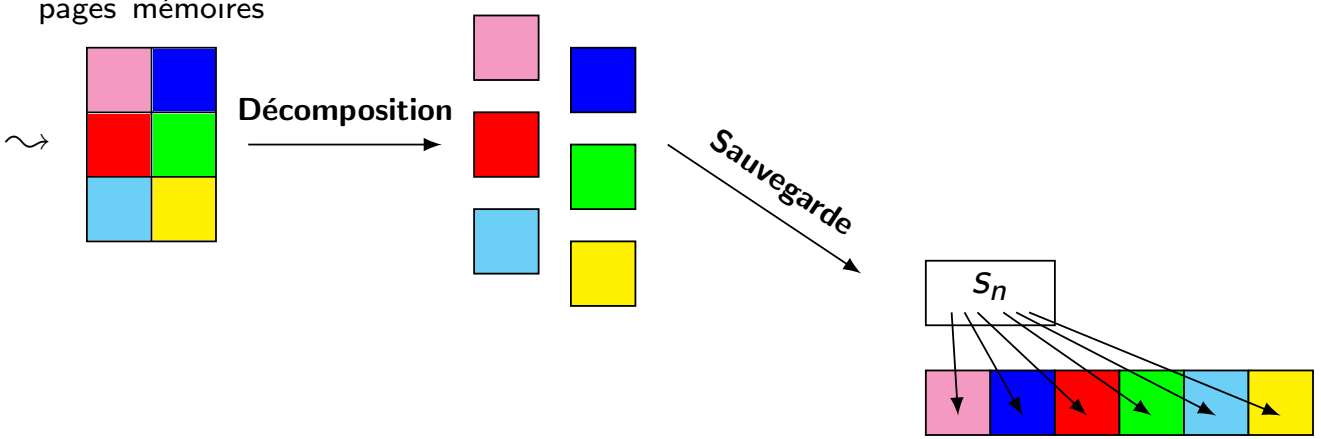
Travail en collaboration avec G. Corona

Compression mémoire

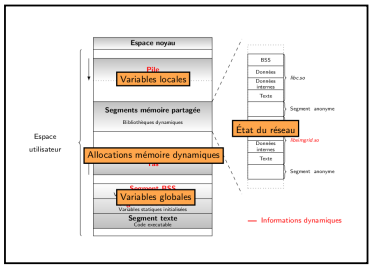
État système s_n



Ensemble de pages mémoires



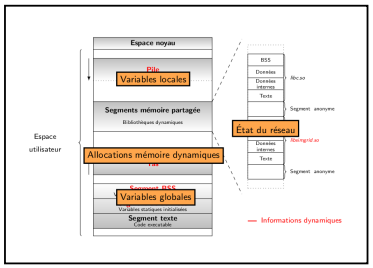
État système s_{n+i}



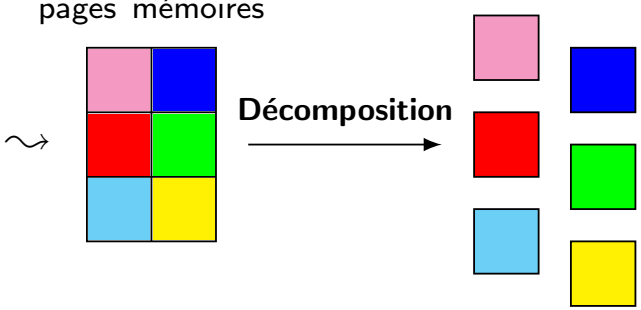
Travail en collaboration avec G. Corona

Compression mémoire

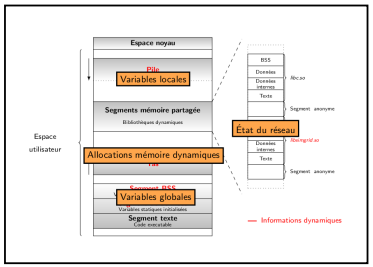
État système s_n



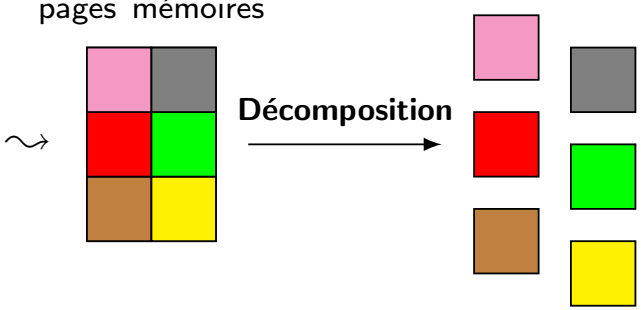
Ensemble de pages mémoires



État système s_{n+i}



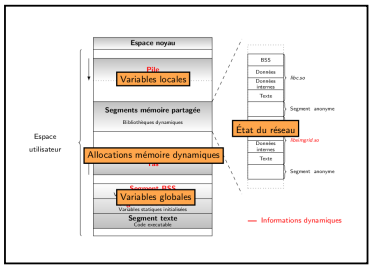
Ensemble de pages mémoires



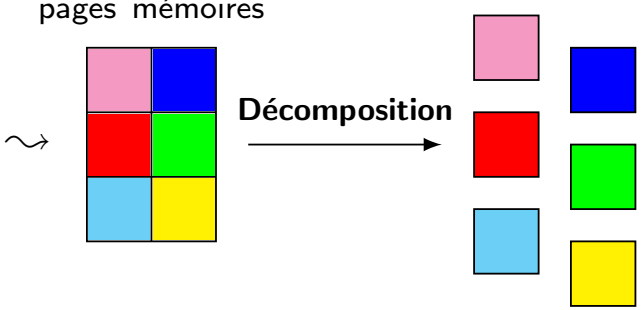
Travail en collaboration avec G. Corona

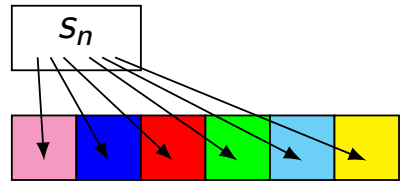
Compression mémoire

État système s_n

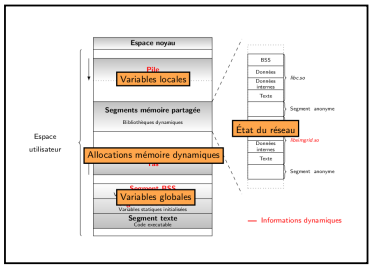


Ensemble de pages mémoires

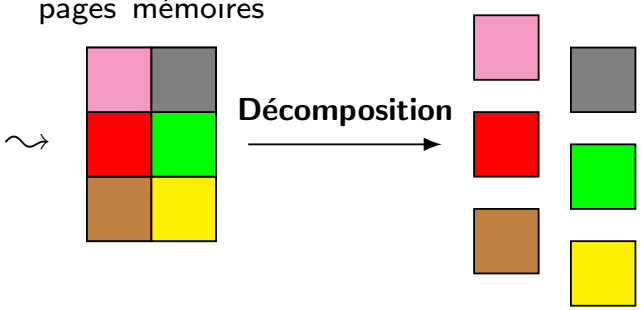


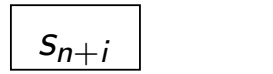


État système s_{n+i}



Ensemble de pages mémoires

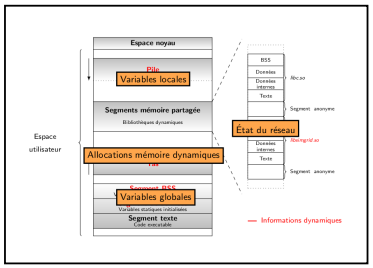




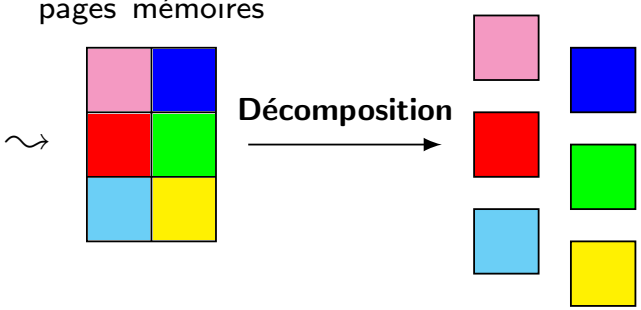
Travail en collaboration avec G. Corona

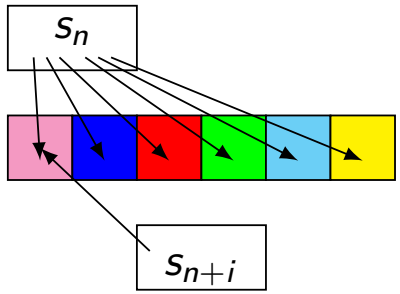
Compression mémoire

État système s_n

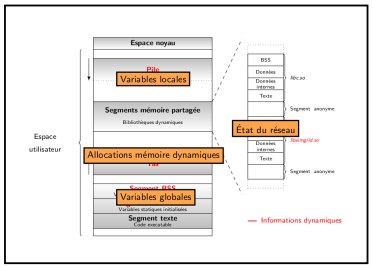


Ensemble de pages mémoires

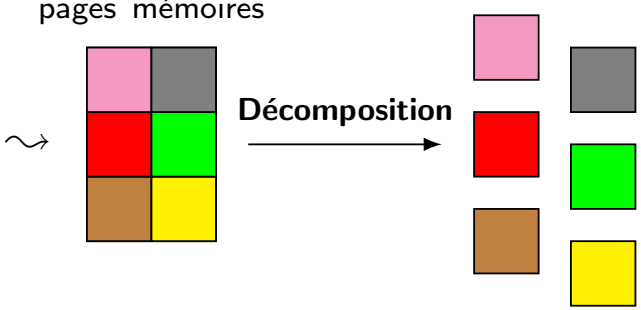




État système s_{n+i}



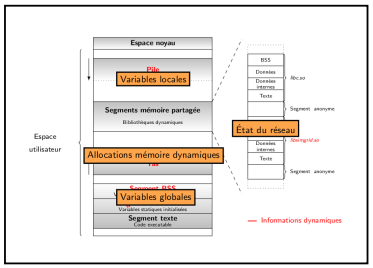
Ensemble de pages mémoires



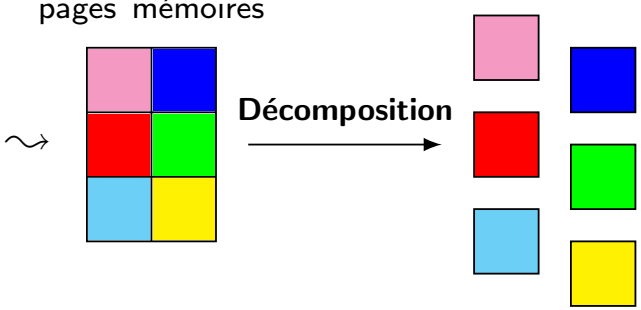
Travail en collaboration avec G. Corona

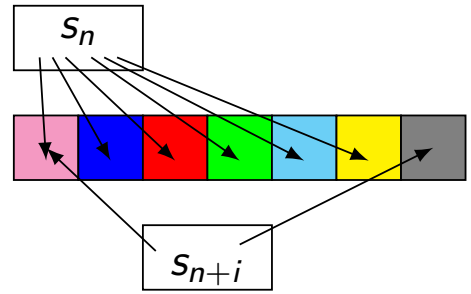
Compression mémoire

État système s_n

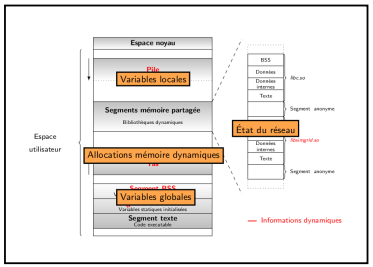


Ensemble de pages mémoires

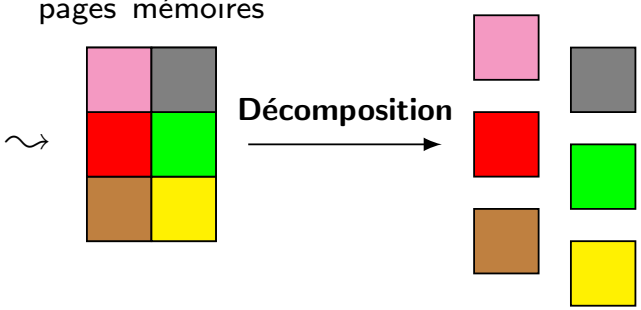




État système s_{n+i}



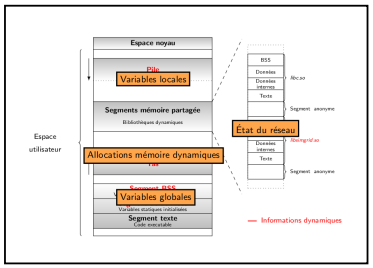
Ensemble de pages mémoires



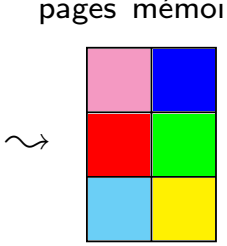
Travail en collaboration avec G. Corona

Compression mémoire

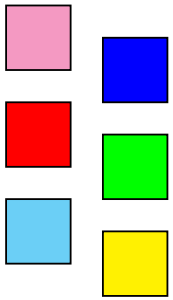
État système s_n



Ensemble de pages mémoires

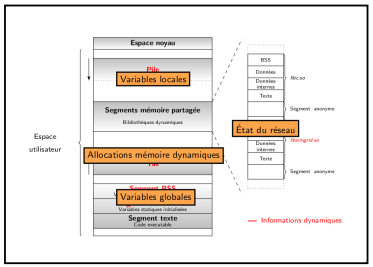


Décomposition

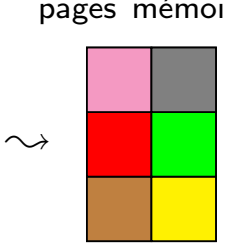


Sauvegarde

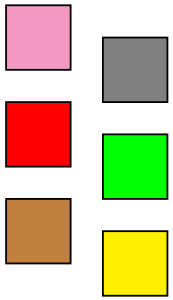
État système s_{n+i}



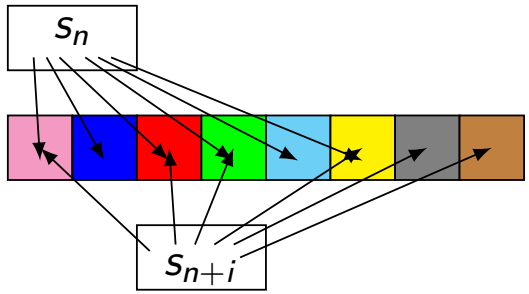
Ensemble de pages mémoires



Décomposition

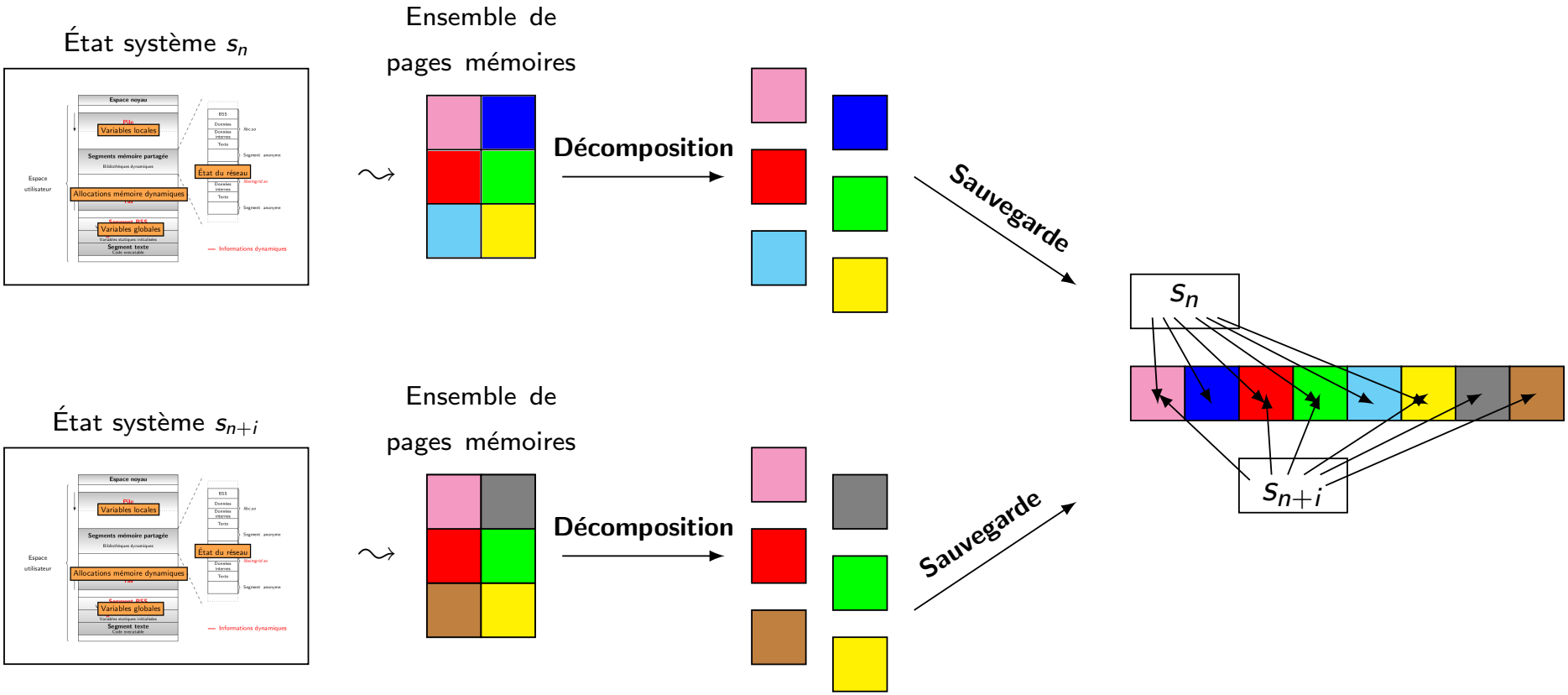


Sauvegarde



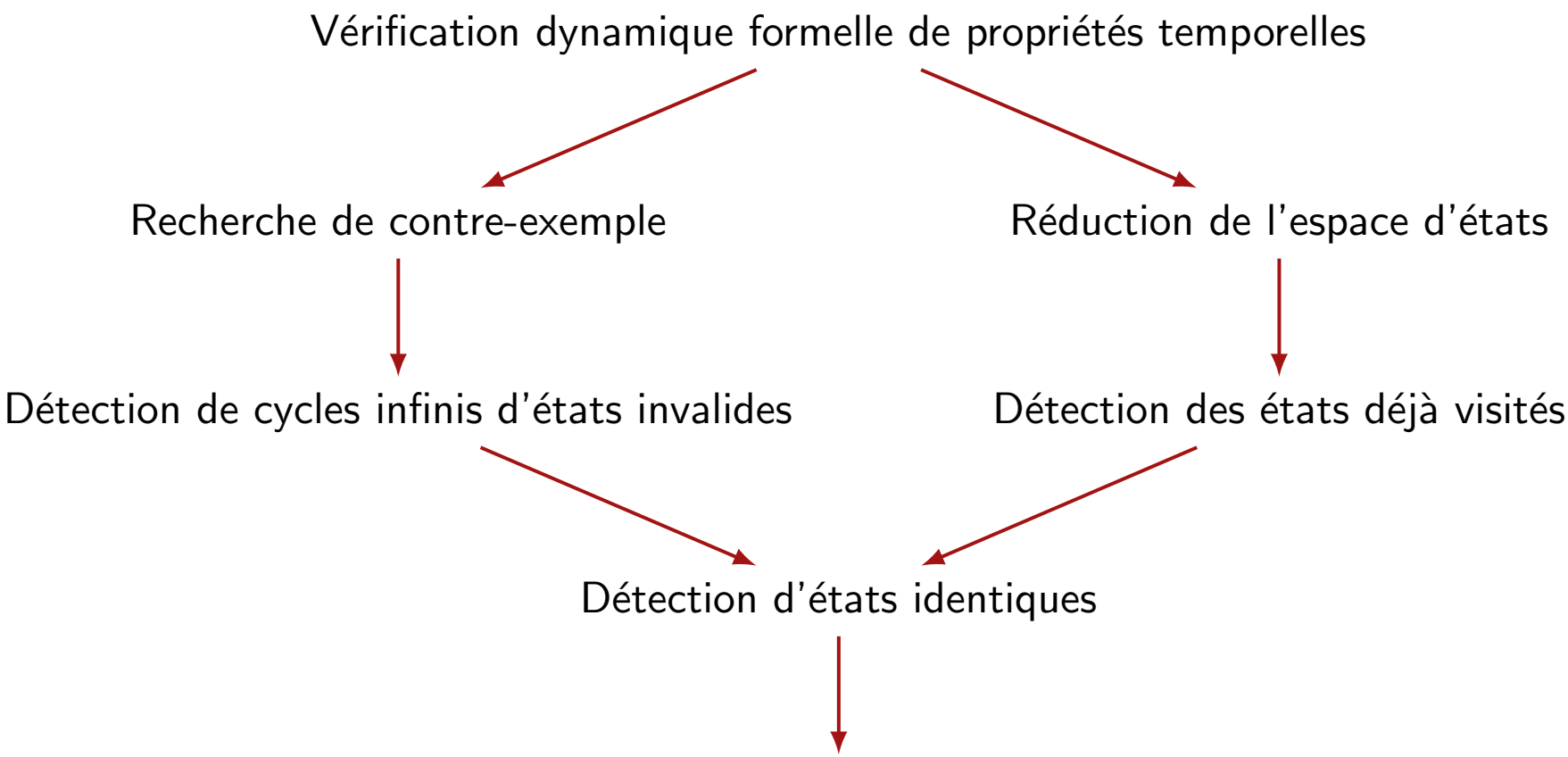
Travail en collaboration avec G. Corona

Compression mémoire



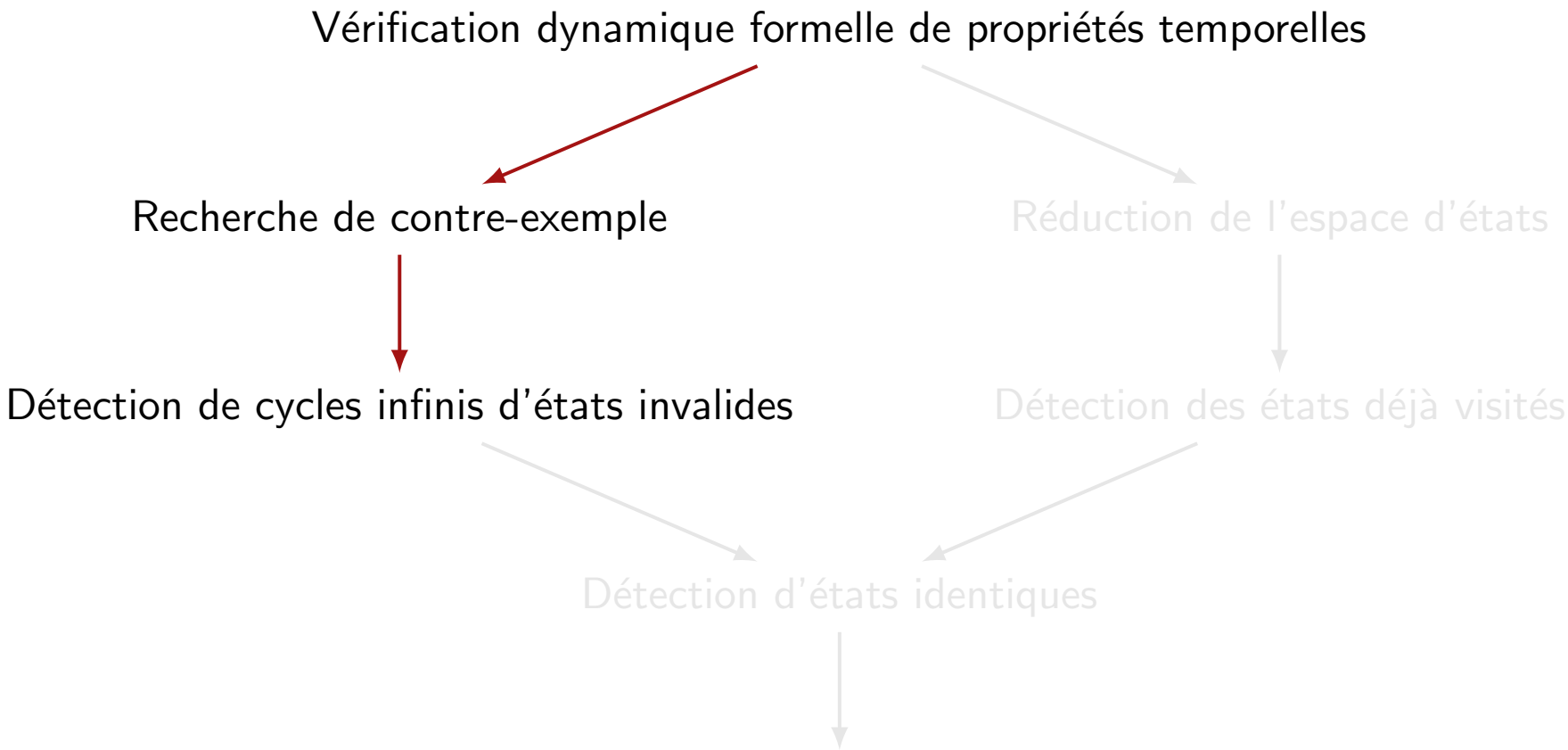
Travail en collaboration avec G. Corona

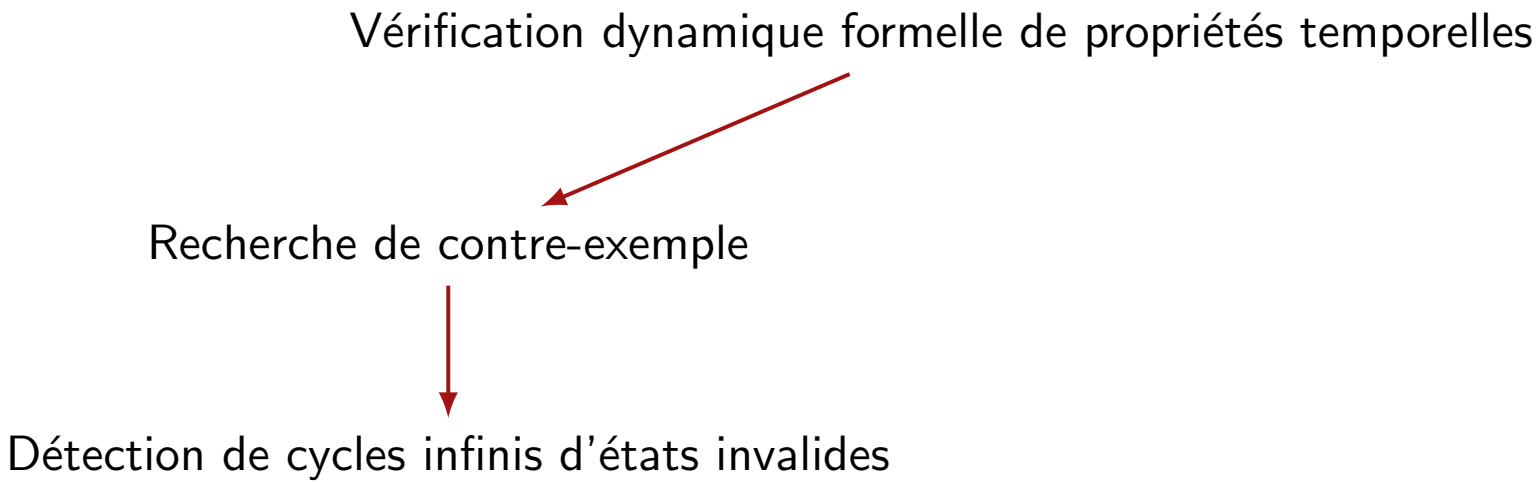
Application	# P	Exploration sans réduction			Exploration avec réduction			Exploration avec réduction et compression		
		# États	Temps	Mémoire	# États	Temps	Mémoire	# États	Temps	Mémoire
bcasttest (C)	3	$>$ 1 million	$>$ 24 h	-	4 823	18 min 31 s	37 Go	4 823	25 min 23 s	1.01 Go
groupcreate (C)	6	$>$ 272 millions	$>$ 24 h	-	80 878	16 h 15 min	89.31 Go	80 878	17 h 35 min	11.47 Go
inplacef (Fortran)	3	$>$ 182 millions	$>$ 24 h	-	2 941	1 min 07 s	3.87 Go	2 941	1 min 15 s	0.73 Go



Contribution 1

Analyse sémantique dynamique
d'un état système par introspection mémoire


Vérification dynamique formelle de propriétés temporelles
Recherche de contre-exemple
Réduction de l'espace d'états
Détection de cycles infinis d'états invalides
Détection des états déjà visités
Détection d'états identiques



Contribution 2

Vérification dynamique formelle de propriétés de vivacité

Vérification formelle de propriétés de vivacité

Propriété de vivacité φ

Vérification formelle de propriétés de vivacité

Propriété de vivacité φ

Contre-exemple infini

Vérification formelle de propriétés de vivacité

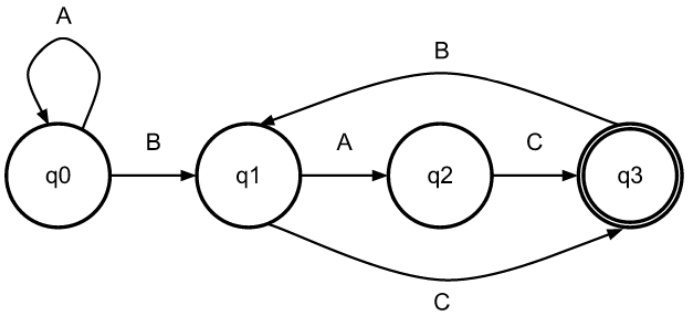
Propriété de vivacité φ

↓

Contre-exemple infini

↓

Automate de Büchi de $\neg\varphi$



Vérification formelle de propriétés de vivacité

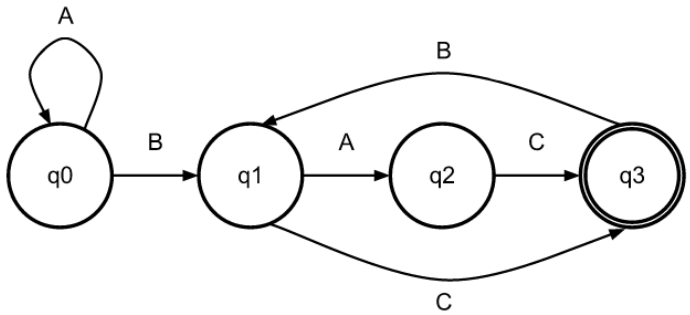
Propriété de vivacité φ

↓

Contre-exemple infini

↓

Automate de Büchi de $\neg\varphi$

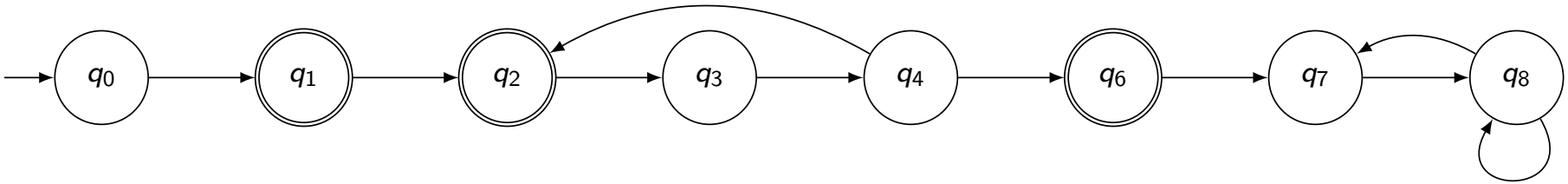


↓

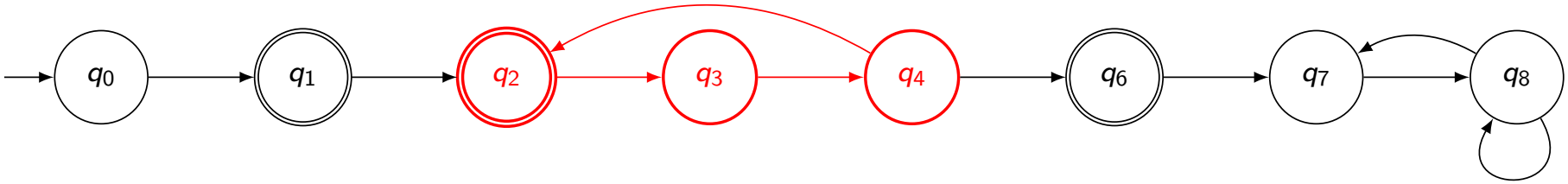
Emptiness check

- Existe-t-il une exécution du système reconnue par AB($\neg\varphi$) ?
  - Exploration du produit cartésien Système $\times$ AB($\neg\varphi$)

Recherche d'un cycle acceptant

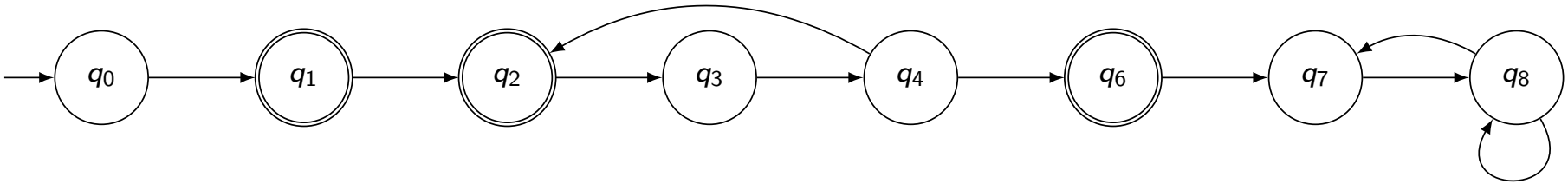


Recherche d'un cycle acceptant



Emptiness check

Recherche d'un cycle acceptant

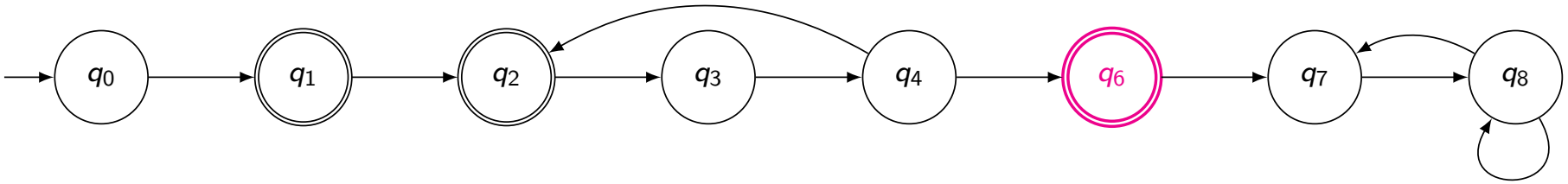


Nested-Depth First Search : Algorithme récursif avec double exploration

- Exploration en profondeur sans recherche de cycle
- Exploration en profondeur avec recherche de cycle

Emptiness check

Recherche d'un cycle acceptant

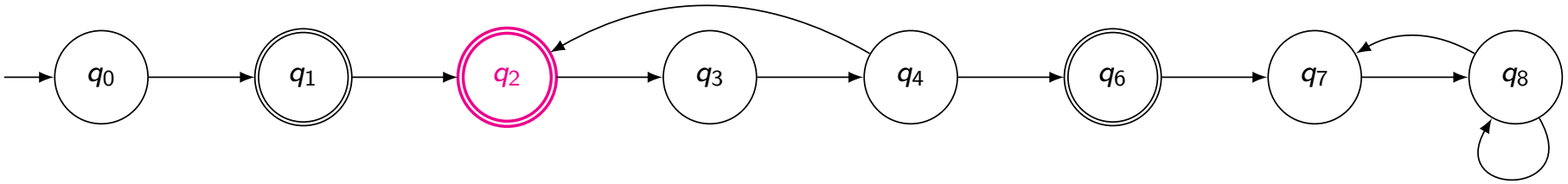


Nested-Depth First Search : Algorithme récursif avec double exploration

- Exploration en profondeur sans recherche de cycle
- Exploration en profondeur avec recherche de cycle ($seed = q_6$)

Emptiness check

Recherche d'un cycle acceptant

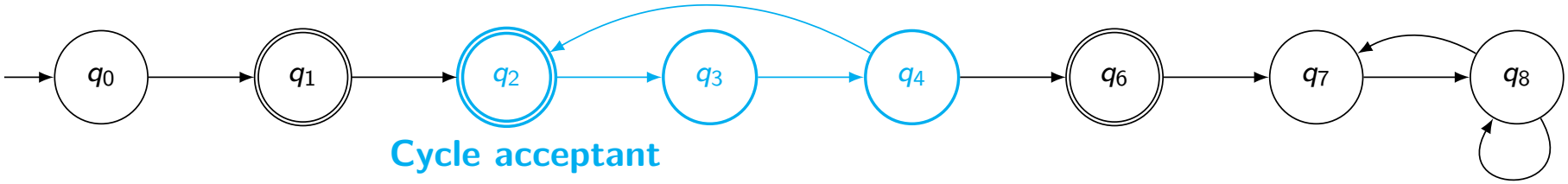


Nested-Depth First Search : Algorithme récursif avec double exploration

- Exploration en profondeur sans recherche de cycle
- Exploration en profondeur avec recherche de cycle ($seed = q_2$)

Emptiness check

Recherche d'un cycle acceptant

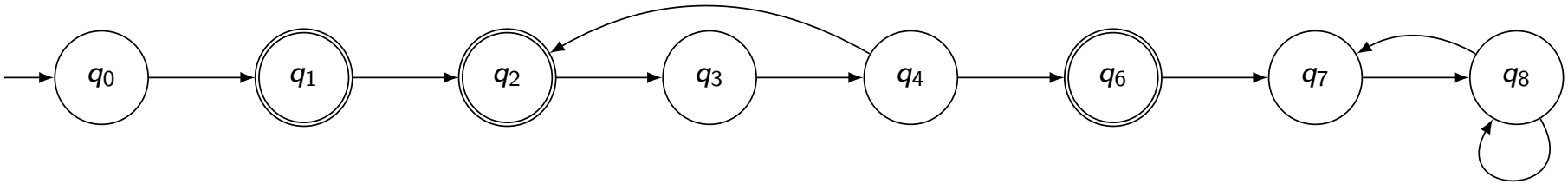


Nested-Depth First Search : Algorithme récursif avec double exploration

- Exploration en profondeur sans recherche de cycle
- Exploration en profondeur avec recherche de cycle

Emptiness check

Recherche d'un cycle acceptant



Nested-Depth First Search : Algorithme récursif avec double exploration

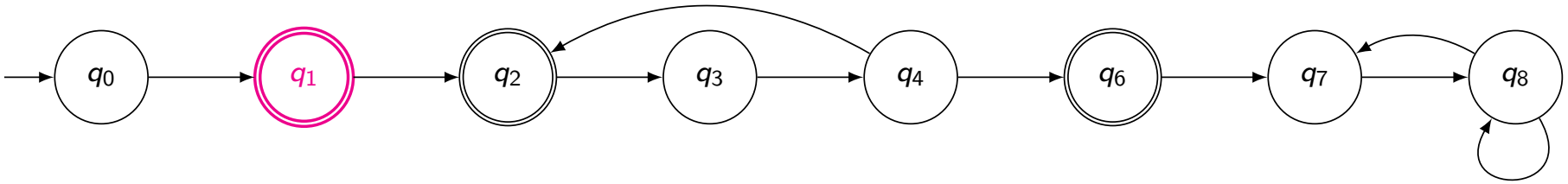
- Exploration en profondeur sans recherche de cycle
- Exploration en profondeur avec recherche de cycle

Modification de l'algorithme d'exploration

- Unique parcours en profondeur, non récursif
- Recherche simultanée de plusieurs cycles acceptants
- Complexité réduite

Emptiness check

Recherche d'un cycle acceptant



Nested-Depth First Search : Algorithme récursif avec double exploration

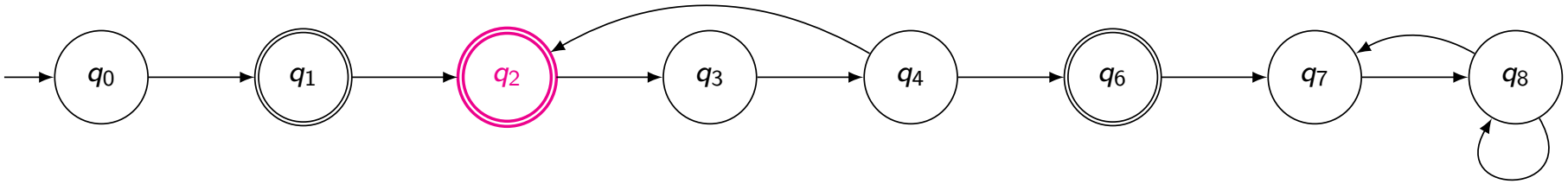
- Exploration en profondeur sans recherche de cycle
- Exploration en profondeur avec recherche de cycle

Modification de l'algorithme d'exploration

- Unique parcours en profondeur, non récursif
- Recherche simultanée de plusieurs cycles acceptants ($seed = q_1$)
- Complexité réduite

Emptiness check

Recherche d'un cycle acceptant



Nested-Depth First Search : Algorithme récursif avec double exploration

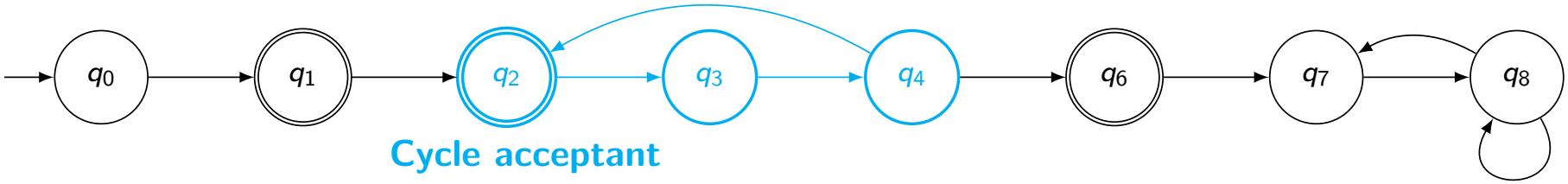
- Exploration en profondeur sans recherche de cycle
- Exploration en profondeur avec recherche de cycle

Modification de l'algorithme d'exploration

- Unique parcours en profondeur, non récursif
- Recherche simultanée de plusieurs cycles acceptants $(seed = \{q_1, q_2\})$
- Complexité réduite

Emptiness check

Recherche d'un cycle acceptant



Nested-Depth First Search : Algorithme récursif avec double exploration

- Exploration en profondeur sans recherche de cycle
- Exploration en profondeur avec recherche de cycle

Modification de l'algorithme d'exploration

- Unique parcours en profondeur, non récursif
- Recherche simultanée de plusieurs cycles acceptants $(seed = \{q_1, q_2\})$
- Complexité réduite
- Validé expérimentalement sur application MPI *ad-hoc*

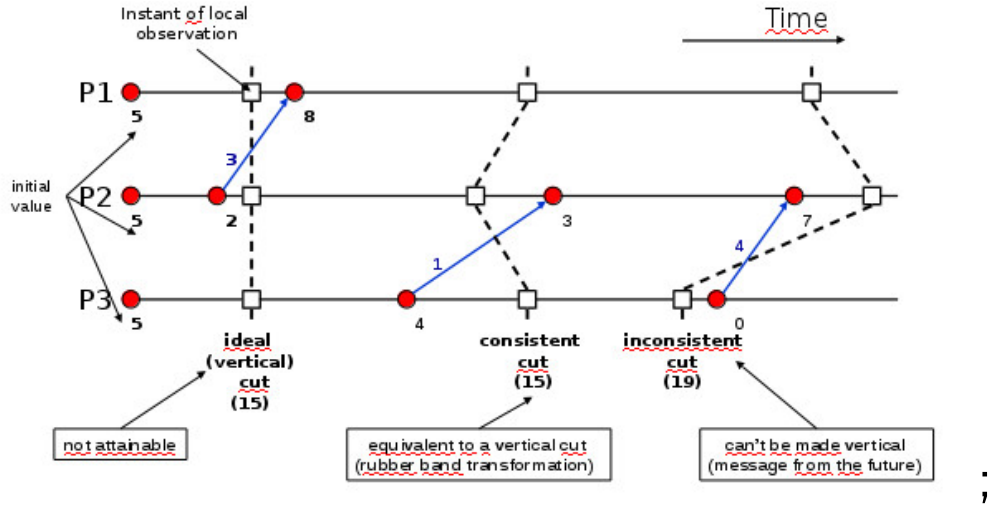
Contribution 3

Vérification dynamique formelle du déterminisme des communications sur des applications distribuées HPC

Motivations

"On Communication Determinism in Parallel HPC Applications",
F. Cappello, A. Guermouche and M. Snir (2010)

- Protocoles de tolérance aux pannes inadaptés aux systèmes distribués



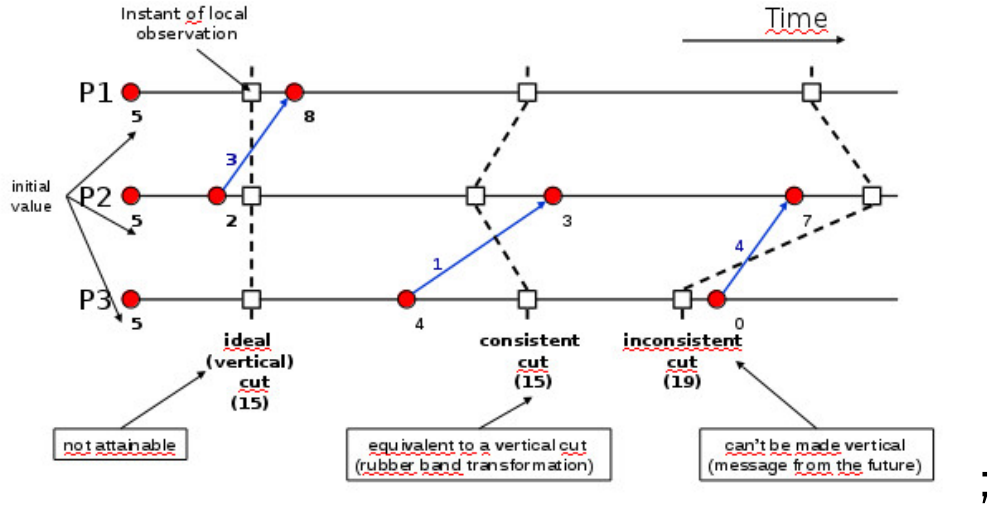
;

- Approche : Exploiter le déterminisme des communications
- Déterminisme ⇒ Ordre identique sur toutes les exécutions
- Étude du déterminisme de 27 applications HPC représentatives

Motivations

"On Communication Determinism in Parallel HPC Applications",
F. Cappello, A. Guermouche and M. Snir (2010)

- Protocoles de tolérance aux pannes inadaptés aux systèmes distribués



;

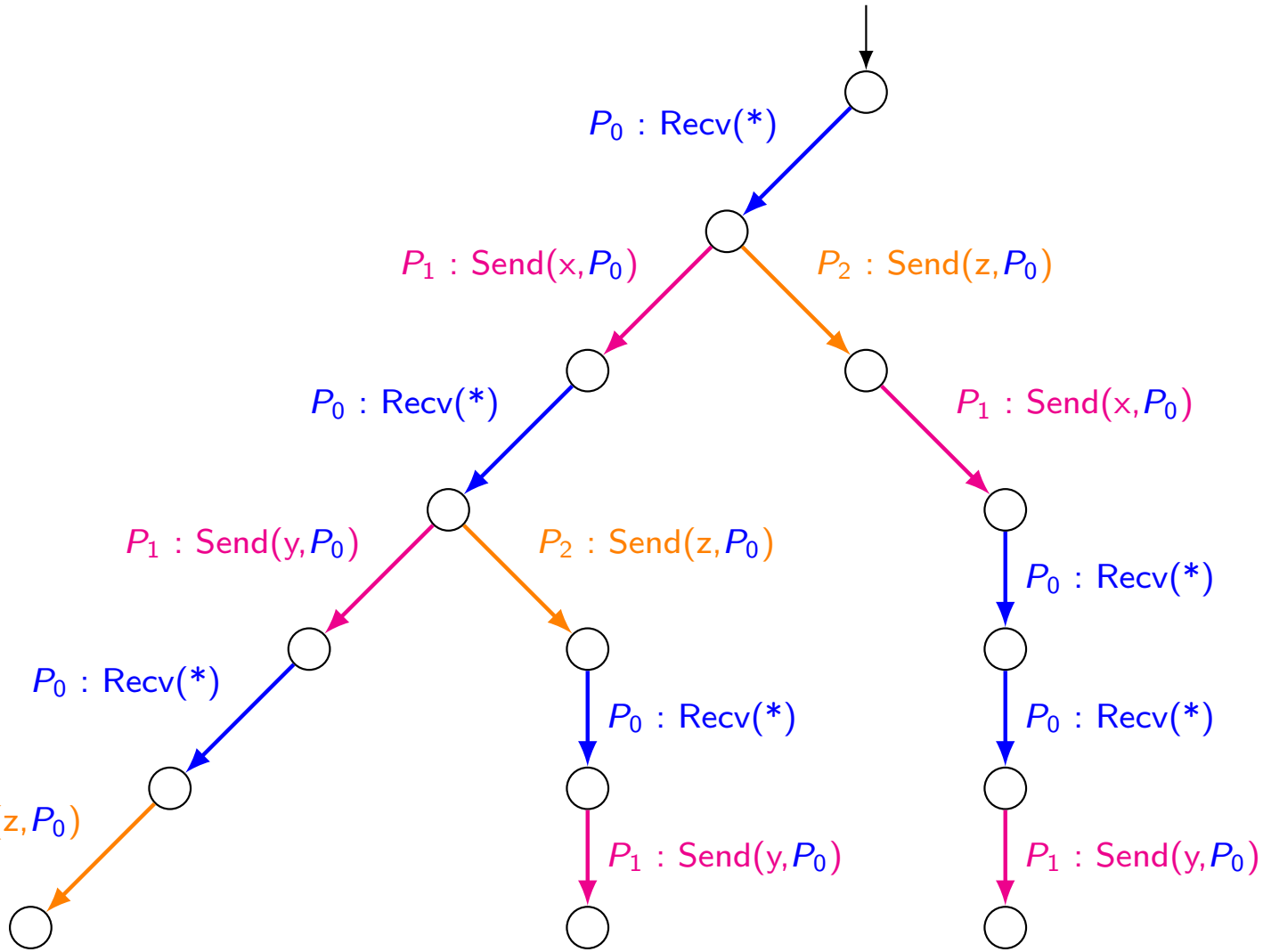
- Approche : Exploiter le déterminisme des communications
- Déterminisme ⇒ Ordre identique sur toutes les exécutions
- Étude **manuelle** du déterminisme de 27 applications HPC représentatives

Déterminisme des émissions

P_0 :
Recv(*)
Recv(*)
Recv(*)

P_1 :
Send(x, P_0)
Send(y, P_0)

P_2 :
Send(z, P_0)

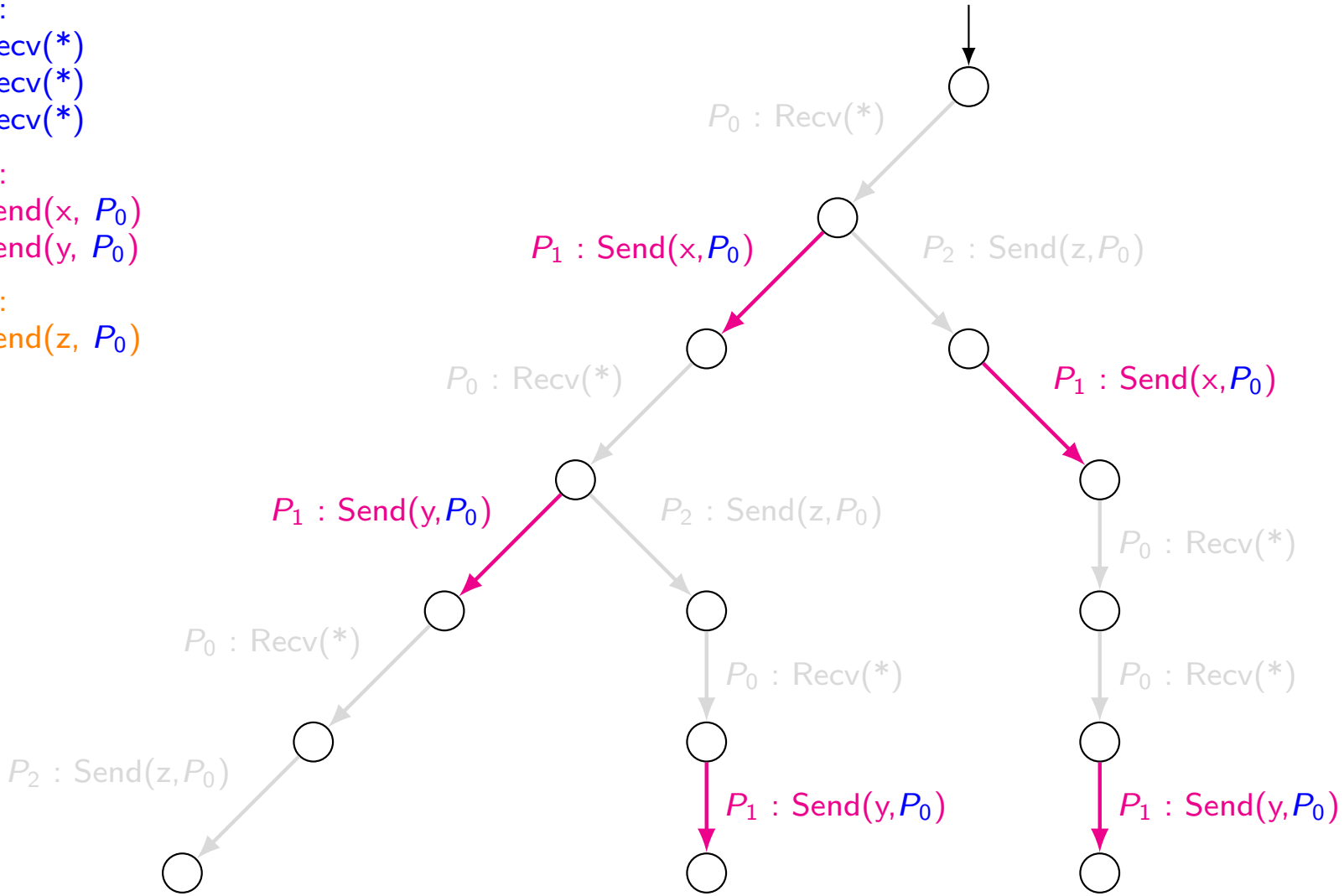


Déterminisme des émissions

P_0 :
Recv(*)
Recv(*)
Recv(*)

P_1 :
Send(x, P_0)
Send(y, P_0)

P_2 :
Send(z, P_0)

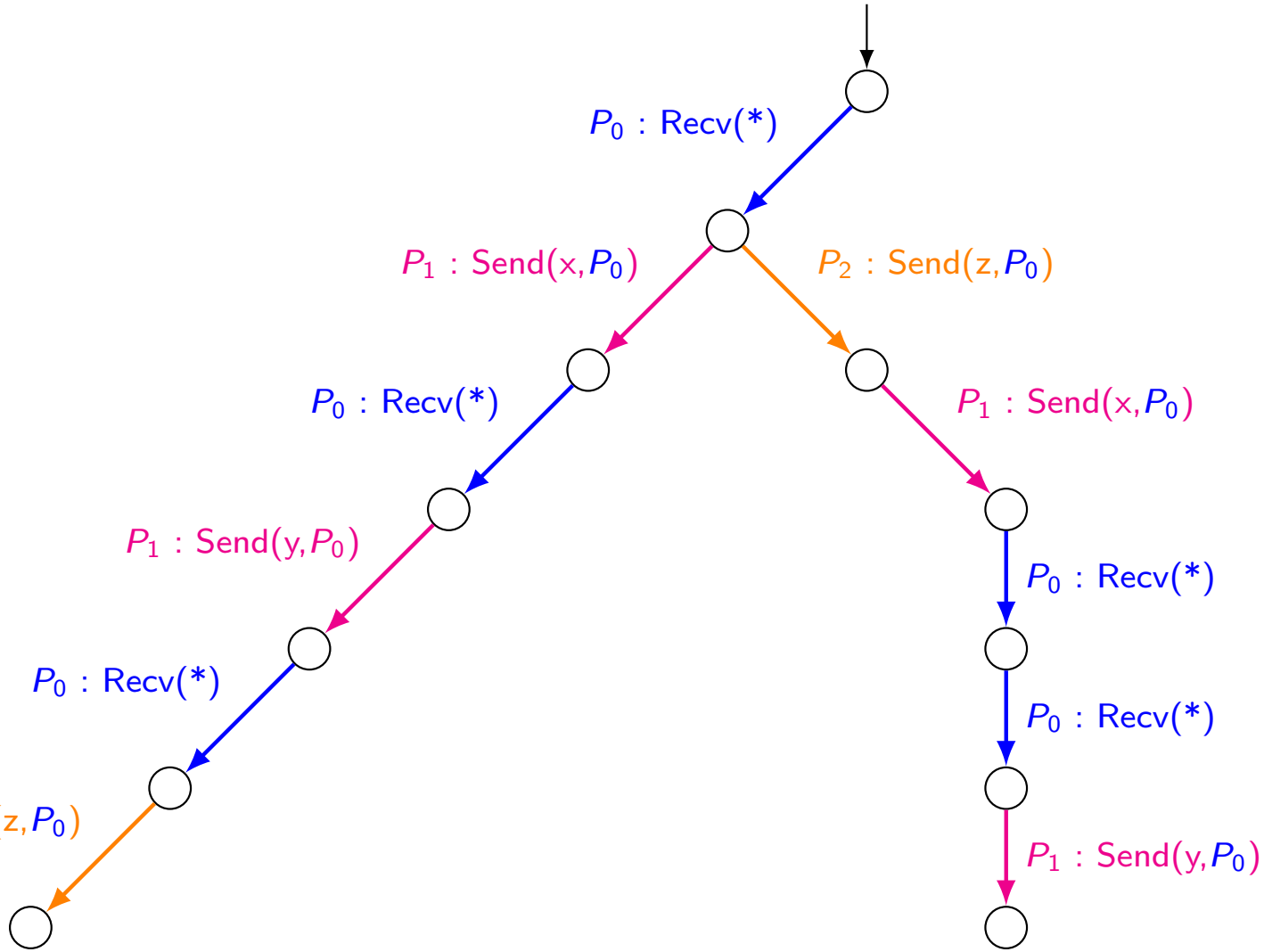


Déterminisme des réceptions

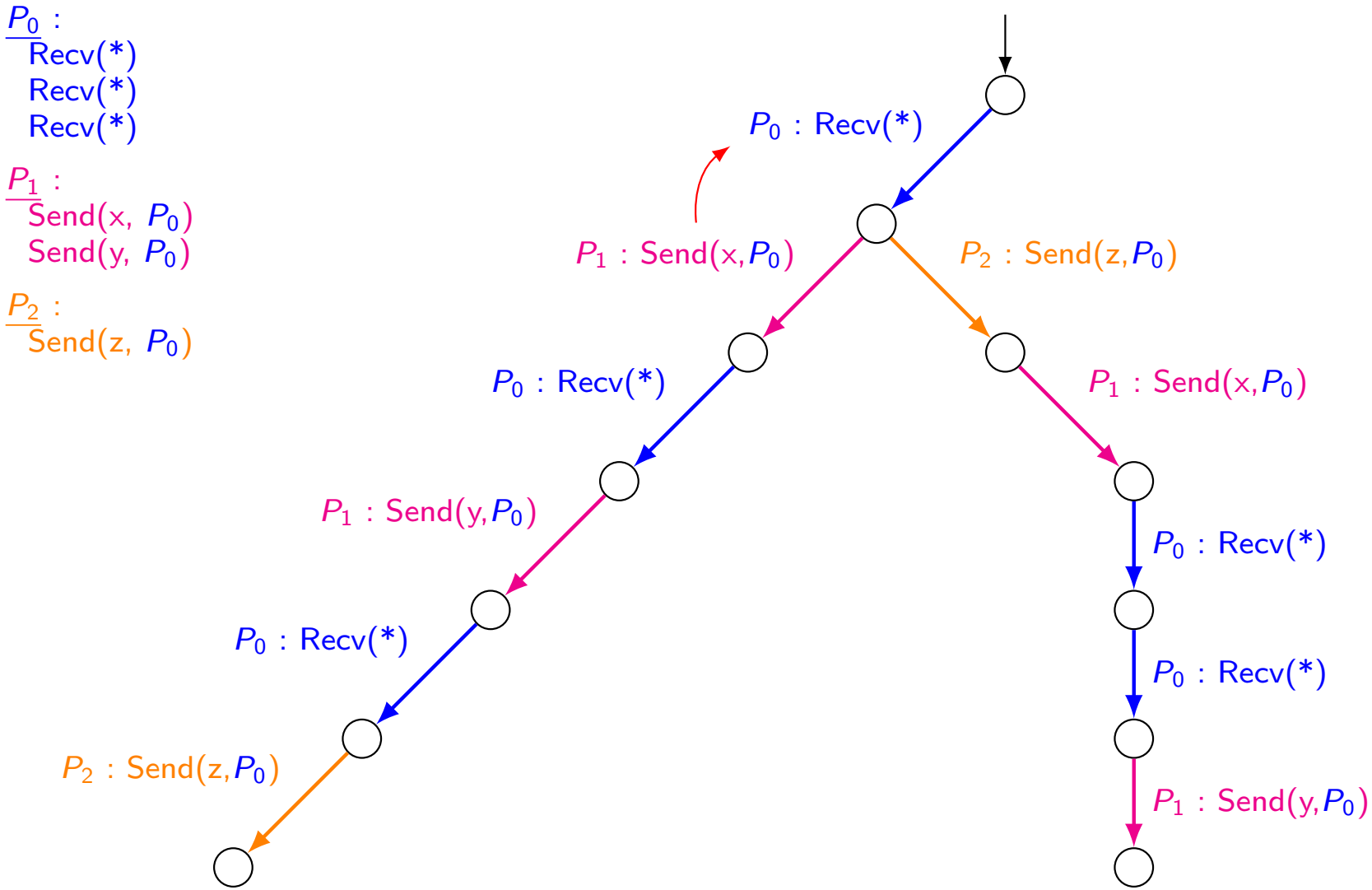
P_0 :
Recv(*)
Recv(*)
Recv(*)

P_1 :
Send(x, P_0)
Send(y, P_0)

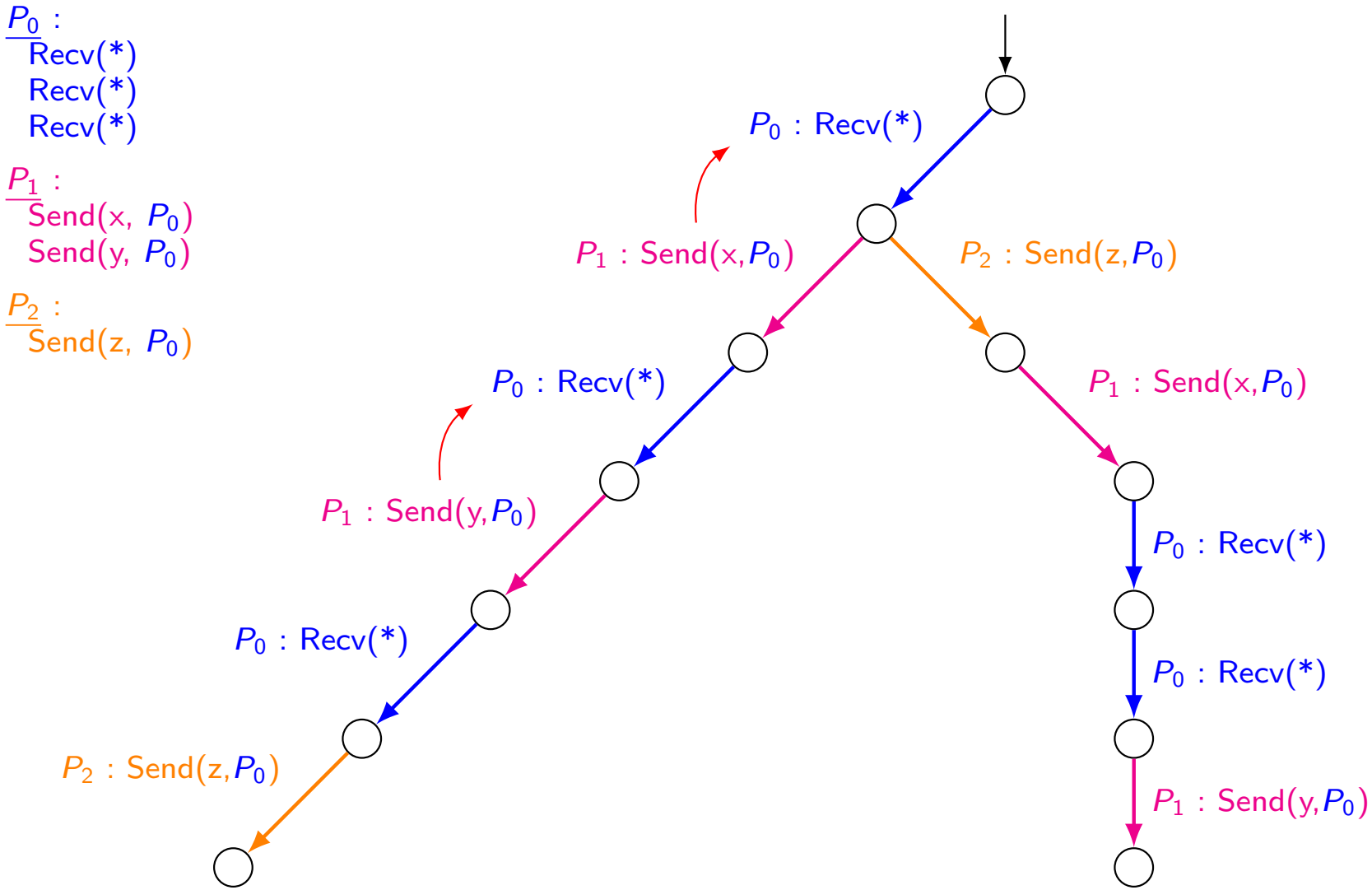
P_2 :
Send(z, P_0)



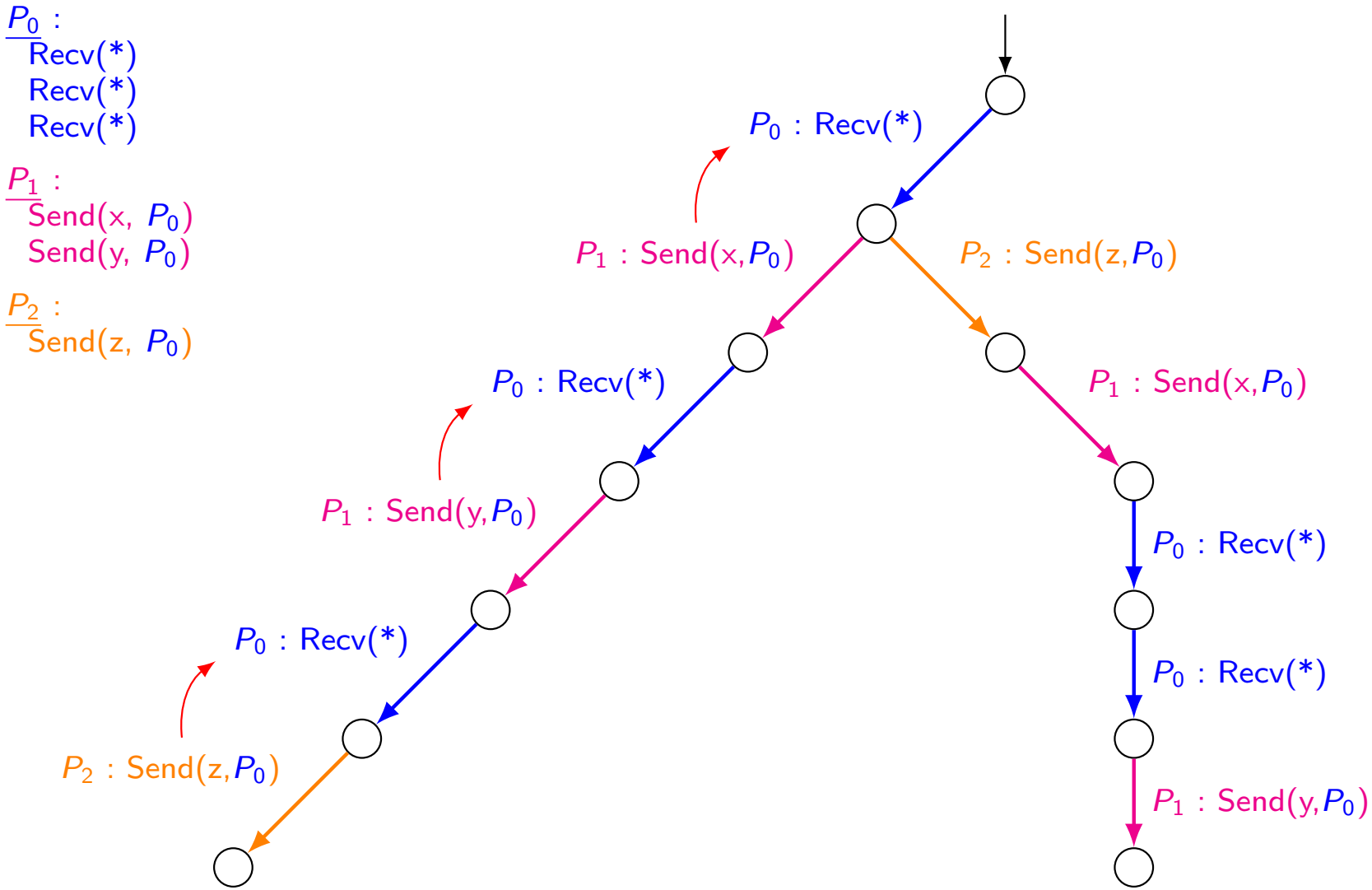
Déterminisme des réceptions



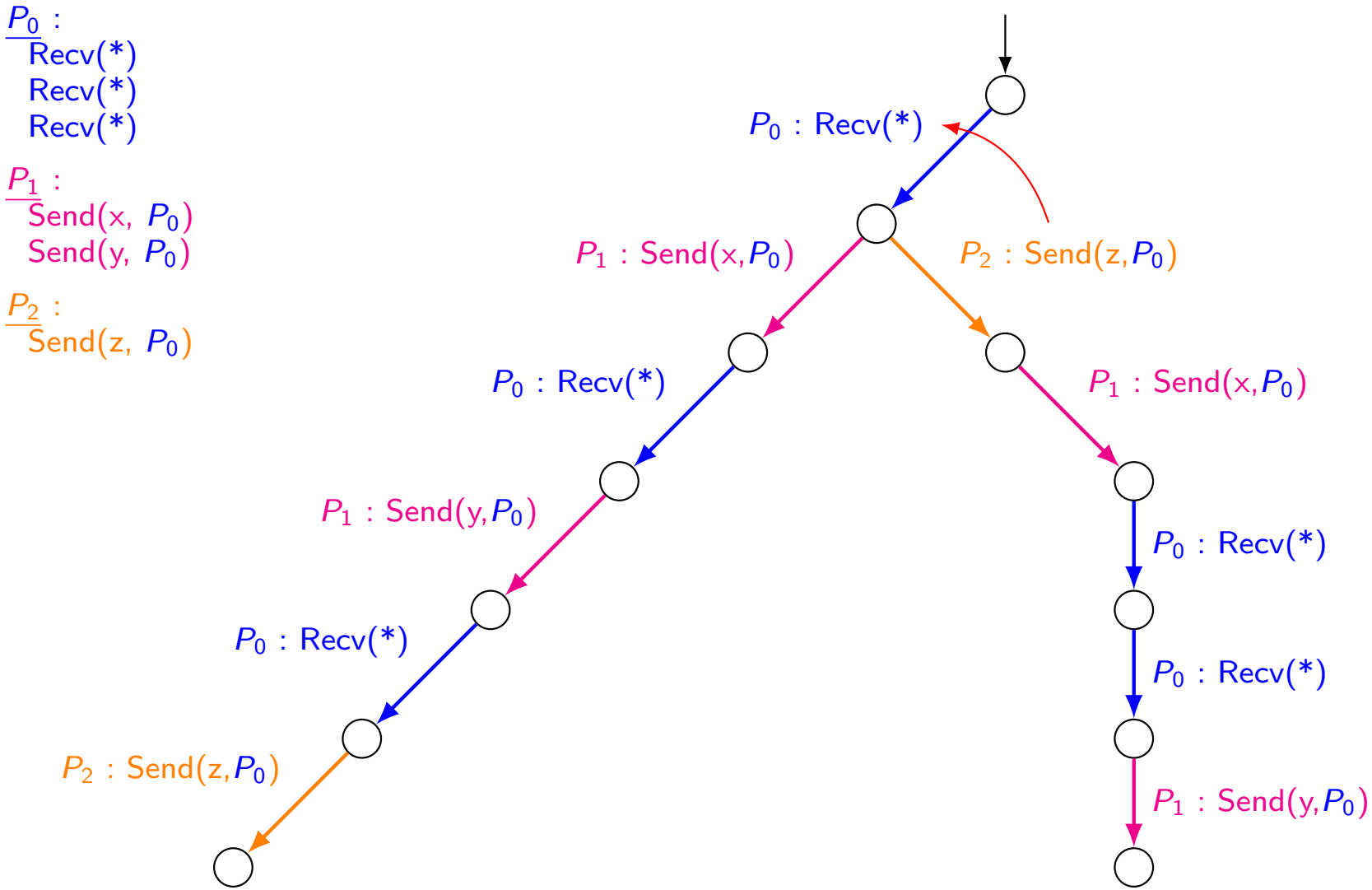
Déterminisme des réceptions



Déterminisme des réceptions



Déterminisme des réceptions

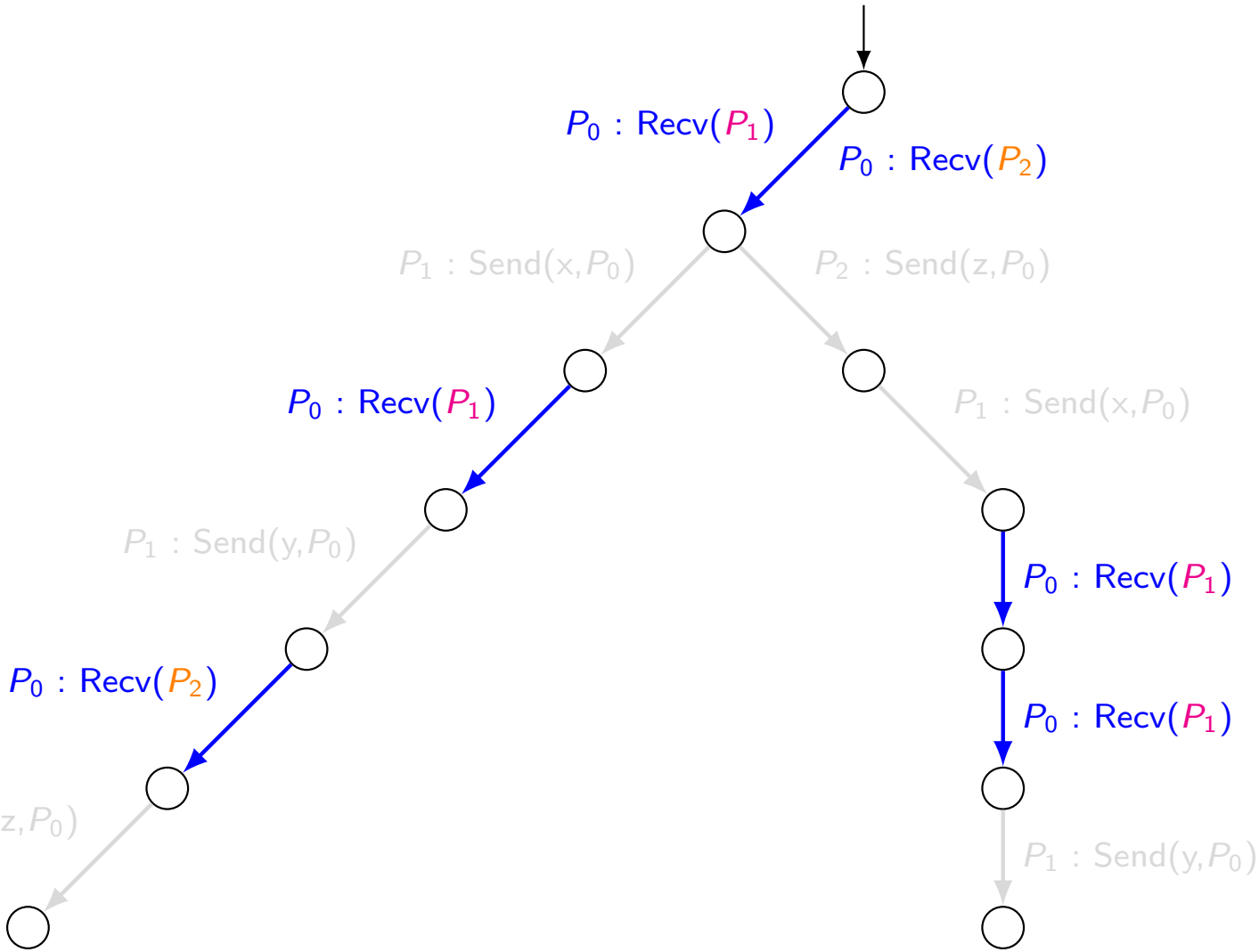


Déterminisme des réceptions

P_0 :
Recv(*)
Recv(*)
Recv(*)

P_1 :
Send(x, P_0)
Send(y, P_0)

P_2 :
Send(z, P_0)

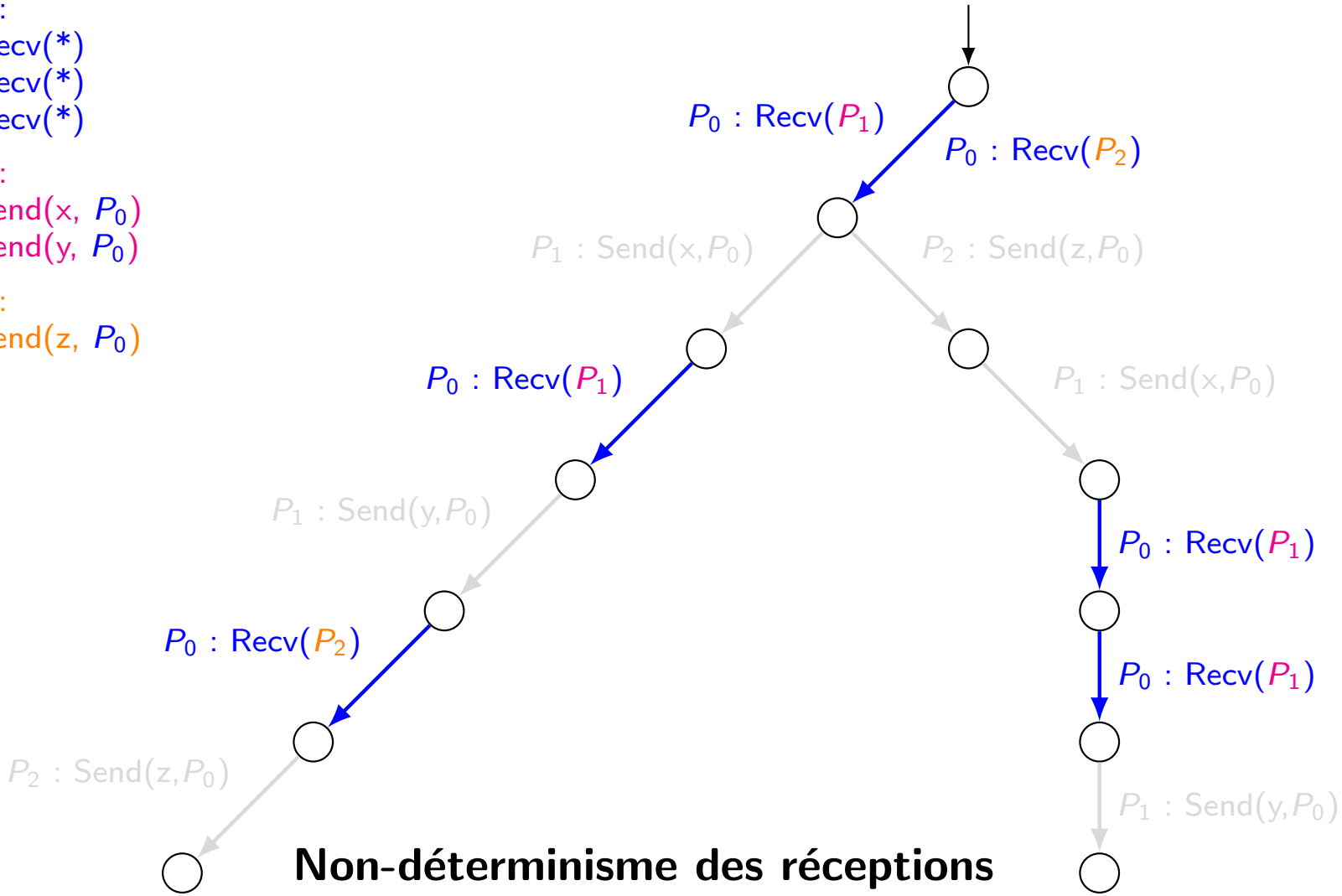


Déterminisme des réceptions

P_0 :
Recv(*)
Recv(*)
Recv(*)

P_1 :
Send(x, P_0)
Send(y, P_0)

P_2 :
Send(z, P_0)



Non-déterminisme des réceptions

Vérification dynamique formelle des déterminismes

« *Il existe un schéma de communications ϕ, tel que, pour toutes les exécutions, ce schéma de communications est respecté.* »

Vérification dynamique formelle des déterminismes

« Il existe un schéma de communications ϕ, tel que, pour toutes les exécutions, ce schéma de communications est respecté. »

1. Découverte de la propriété ϕ

$=$

Sauvegarde du schéma de communications du premier chemin d'exécution

Vérification dynamique formelle des déterminismes

« Il existe un schéma de communications ϕ, tel que, pour toutes les exécutions, ce schéma de communications est respecté. »

1. Découverte de la propriété ϕ

$=$

Sauvegarde du schéma de communications du premier chemin d'exécution

2. Vérification de ϕ sur tous les chemins

$=$

Comparaison avec le schéma de communications de référence

Vérification dynamique formelle des déterminismes

« Il existe un schéma de communications ϕ, tel que, pour toutes les exécutions, ce schéma de communications est respecté. »

1. Découverte de la propriété ϕ

$=$

Sauvegarde du schéma de communications du premier chemin d'exécution

2. Vérification de ϕ sur tous les chemins

$=$

Comparaison avec le schéma de communications de référence

Évaluation expérimentale

- Reproduction de résultats de l'étude manuelle
- Exemples *ad-hoc* avec déterminismes connus

Conclusion

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Conclusion

Vérification dynamique formelle de propriétés temporelles sur des applications distribuées réelles

Mes contributions

Système - **Analyse sémantique d'un état système**
- Identification d'états système sémantiquement identiques
- Réduction de l'espace d'états

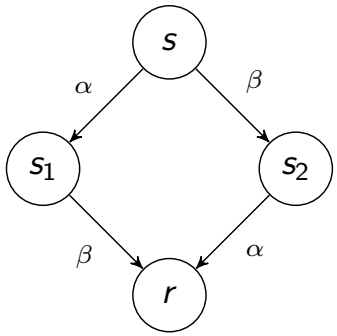
Formel - **Vérification dynamique formelle de propriétés de vivacité**
- Algorithme d'exploration multi-cycles acceptants

HPC - **Vérification du déterminisme des communications**
- Algorithme de vérification *ad-hoc*
- Caractérisation automatique et formelle de déterminismes

Perspectives

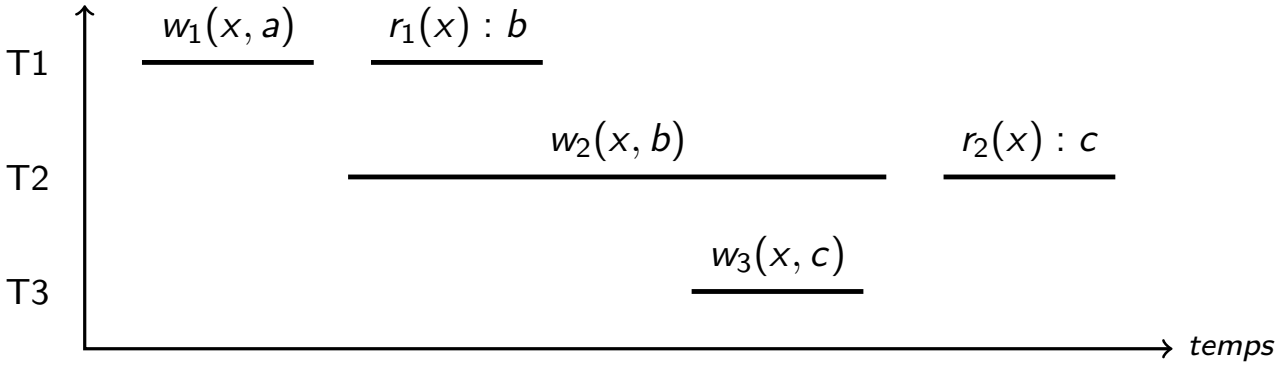
Système

- Autres réductions (combinaison avec DPOR, POE)
- Outil de visualisation (*debugging*, ...)



Formel

- Analyse à la fois dynamique et statique
- Linéarisabilité sur applications *multi-threads*

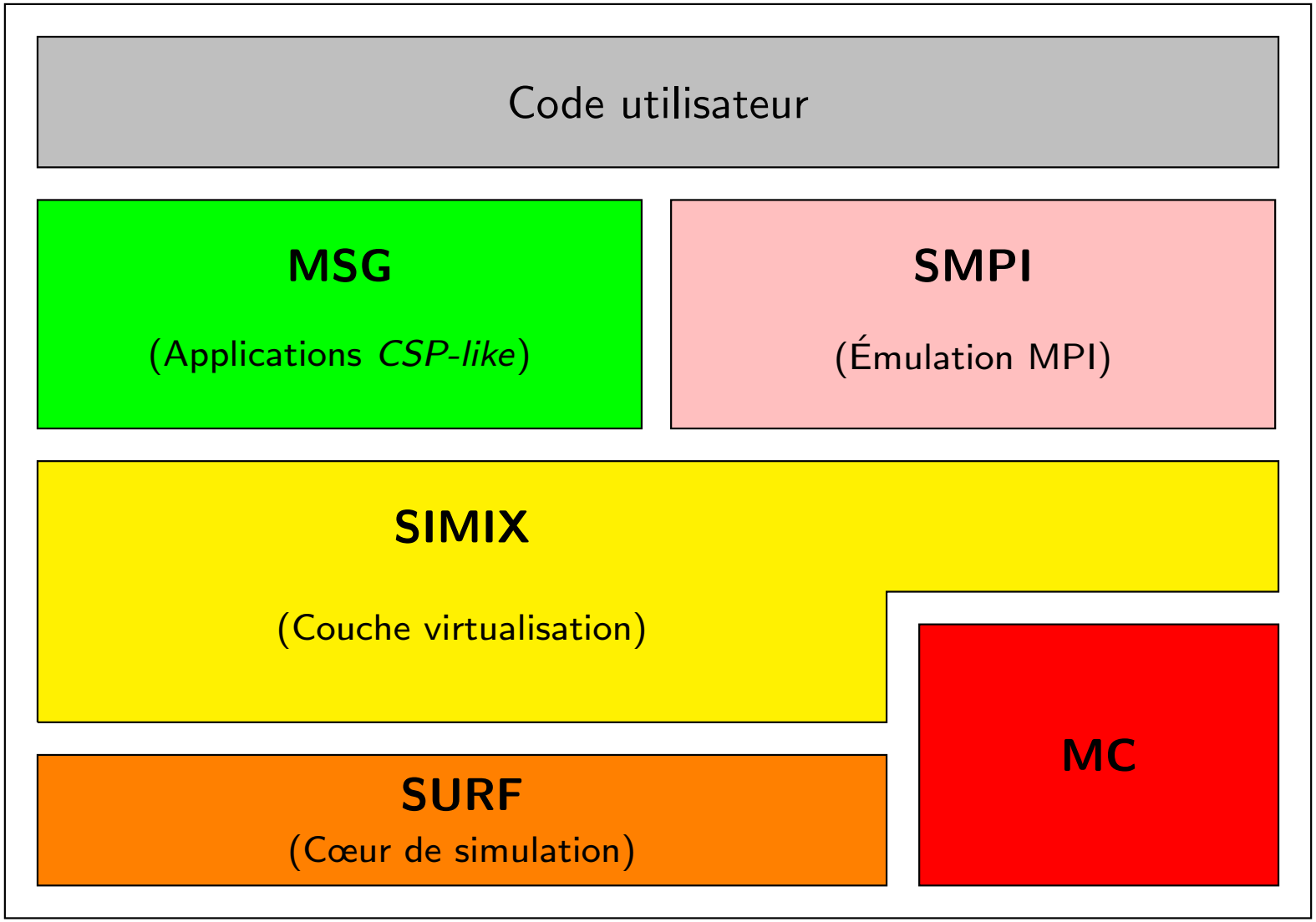


Histoire linéarisable ($w_1(x,a)$ - $w_2(x,b)$ - $r_1(x) : b$ - $w_3(x,c)$ - $r_2(x) : c$)

HPC et distribué

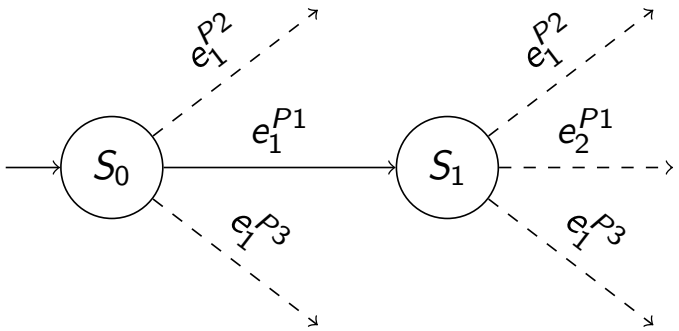
- Consistance à terme (*Eventual consistency*)

SimGrid


Code utilisateur
MSG
(Applications *CSP-like*)
SMPI
(Émulation MPI)
SIMIX
(Couche virtualisation)
MC
SURF
(Cœur de simulation)

Exploration dans SimGridMC

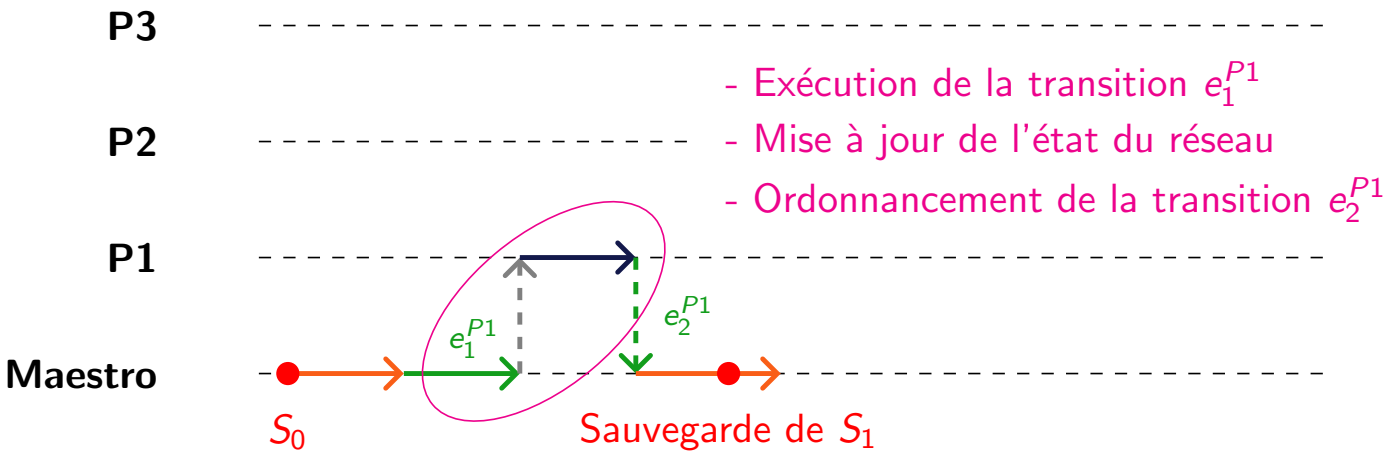
Graphe d'états



Pile d'exploration

- État : S_1
- Transitions à exécuter : $e_2^{P1}, e_1^{P2}, e_1^{P3}$
- Transitions exécutées : $\emptyset$
- Transition courante : $\emptyset$

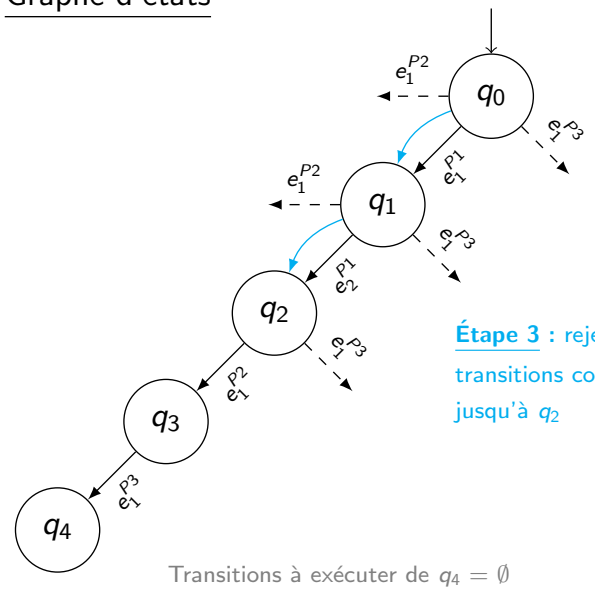
- État : S_0
- Transitions à exécuter : e_1^{P2}, e_1^{P3}
- Transitions exécutées : e_1^{P1}
- Transition courante : e_1^{P1}



MC — Exécution code utilisateur

Backtracking dans SimGridMC

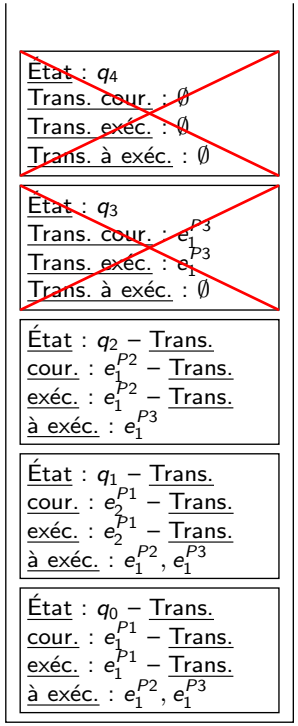
Graphe d'états



Étape 3 : rejeu des transitions courantes jusqu'à q_2

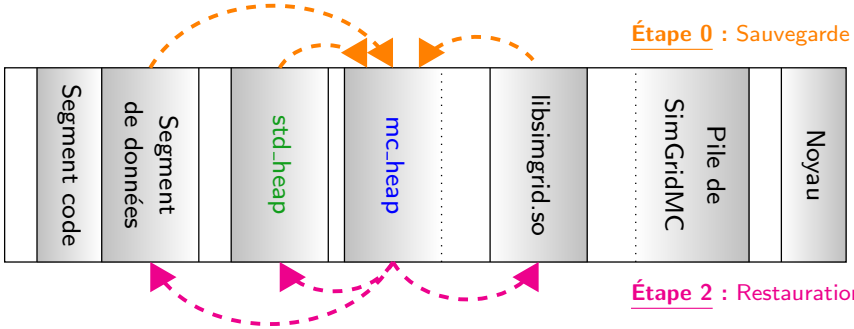
Transitions à exécuter de $q_4 = \emptyset$
⇒ Fin de l'exploration du chemin courant
⇒ *Backtracking*

Pile d'exploration



Étape 1 : suppression dans la pile d'exploration des états dont l'ensemble des transitions à exécuter est vide.

← q_2 est le dernier état en profondeur avec une autre transition à exécuter
⇒ *backtracking* sur q_2



Étape 0 : Sauvegarde de l'état système initial

Étape 2 : Restauration de l'état système initial

DWARF

```
<1><761>: Abbrev Number: 22 (DW_TAG_subprogram)
    <763>   DW_AT_name         : (indirect string, offset: 0xb335): client
    <769>   DW_AT_prototyped   : 1
    <76a>   DW_AT_type         : <0xc5>
    <76e>   DW_AT_low_pc       : 0x401f90
    <776>   DW_AT_high_pc      : 0x4026cf
    <77e>   DW_AT_frame_base   : 0x16c (location list)
 <2><787>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <788>   DW_AT_name         : (indirect string, offset: 0x577d): argc
    <78e>   DW_AT_type         : <0xc5>
    <792>   DW_AT_location     : 3 byte block: 91 ec 7e  (DW_OP_fbreg: -148)
 <2><796>: Abbrev Number: 23 (DW_TAG_formal_parameter)
    <797>   DW_AT_name         : (indirect string, offset: 0x5870): argv
    <79d>   DW_AT_type         : <0x2c1>
    <7a1>   DW_AT_location     : 3 byte block: 91 e0 7e  (DW_OP_fbreg: -160)
 <2><7a5>: Abbrev Number: 24 (DW_TAG_variable)
    <7a6>   DW_AT_name         : (indirect string, offset: 0x109c): pid
    <7ac>   DW_AT_type         : <0xc5>
    <7b0>   DW_AT_location     : 2 byte block: 91 6c   (DW_OP_fbreg: -20)
 <2><7b3>: Abbrev Number: 24 (DW_TAG_variable)
    <7b4>   DW_AT_name         : (indirect string, offset: 0x612): mailbox
    <7ba>   DW_AT_type         : <0x2bb>
    <7be>   DW_AT_location     : 2 byte block: 91 60   (DW_OP_fbreg: -32)
<1><9eb>: Abbrev Number: 32 (DW_TAG_variable)
    <9ec>   DW_AT_name         : global_variable
    <9f1>   DW_AT_type         : <0xc5>
    <9f5>   DW_AT_external     : 1
    <9f6>   DW_AT_location     : 9 byte block: 3 24 57 60 0 0 0 0 0  (DW_OP_addr: 605724)
```

Limites de l'analyse sémantique

- Gestion des pointeurs génériques (`void *`)
- Gestion des unions
- « Pointeurs pendouillants »
- Fuite mémoire

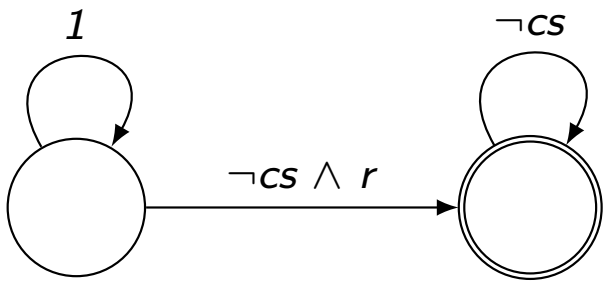
Résultats MPICH

Application (langage)	# P	Sans réduction			Avec réduction			Avec réduction et compression mémoire		
		# États	Temps	Mémoire	# États	Temps	Mémoire	# États	Temps	Mémoire
bcasttest (C)	3	> 1 million	> 24 h	-	4 823	18 min 31 s	37 Go	4 823	25 min 23 s	1.01 Go
bcastzerotype (C)	5	12 135 948	1 h 22 min	0.35 Go	4 734	6 min 35 s	5.83 Go	4 734	6 min 50 s	0.84 Go
	6	> 263 millions	> 24 h	-	56 054	9 h 03 min	62.4 Go	56 054	10 h 02 min	7.16 Go
commcreate1 (C)	4	102 289	44 s	0.35 Go	1 556	1 min 19 s	2.48 Go	1 556	1 min 28 s	0.49 Go
	5	12 710 034	1 h 23 min	0.35 Go	8 359	23 min 22 s	10.56 Go	8 359	25 min	1.48 Go
	6	> 274 millions	> 24 h	-	99 235	23 h 14 min	108.36 Go	99 235	24 h 55 min	14.18 Go
dup (C)	2	907	2 s	0.35 Go	81	2 s	0.45 Go	81	2 s	0.35 Go
	3	138 678	43 s	0.35 Go	405	5 s	0.77 Go	405	7 s	0.36 Go
	4	78 082 843	7 h 36 min	0.35 Go	2 352	1 min 04 s	2.99 Go	2 352	1 min 16 s	0.62 Go
	5	> 276 millions	> 24 h	-	39 263	6 h 32 min	47.63 Go	39 263	7 h 06 min	5.83 Go
groupcreate (C)	4	102 289	31 s	0.35 Go	1 205	40 s	1.86 Go	1 205	44 s	0.44 Go
	5	12 710 034	1 h 22 min	0.35 Go	6 237	11 min	7.62 Go	6 237	11 min 21 s	1.21 Go
	6	> 272 millions	> 24 h	-	80 878	16 h 15 min	89.31 Go	80 878	17 h 35 min	11.47 Go
inplacef (Fortran)	3	> 182 millions	> 24 h	-	2 941	1 min 07 s	3.87 Go	2 941	1 min 15 s	0.73 Go
op_commutative (C)	3	358	2 s	0.35 Go	94	2 s	0.46 Go	94	2 s	0.35 Go
	4	102 289	31 s	0.35 Go	1 545	1 min 16 s	2.47 Go	1 545	1 min 20 s	0.48 Go
	5	12 710 034	1 h 23 min	0.35 Go	10 998	48 min 42 s	14.25 Go	10 998	53 min 29 s	1.79 Go
sendrecv2 (C)	2	> 156 millions	> 24 h	-	1 877	28 s	3.25 Go	1 877	30 s	0.49 Go

Automate de Büchi de $\neg\varphi$

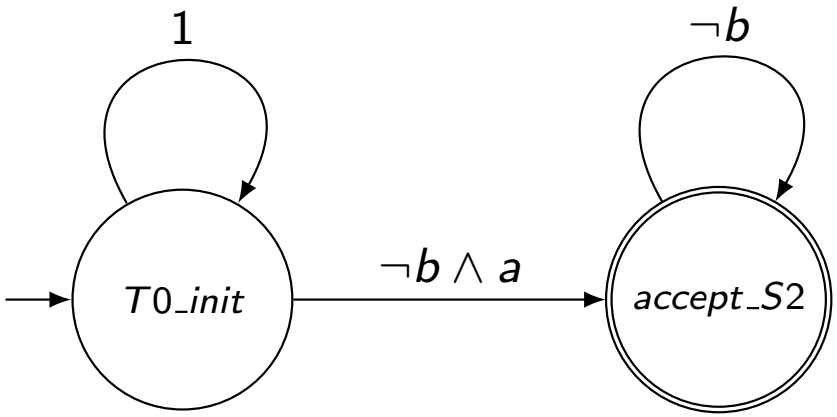
"Any process that asks the critical section will get it"

- r : request
- cs : critical section
- LTL Property : $\Box(r \rightarrow \Diamond cs)$



Description en Promela d'un automate de Büchi

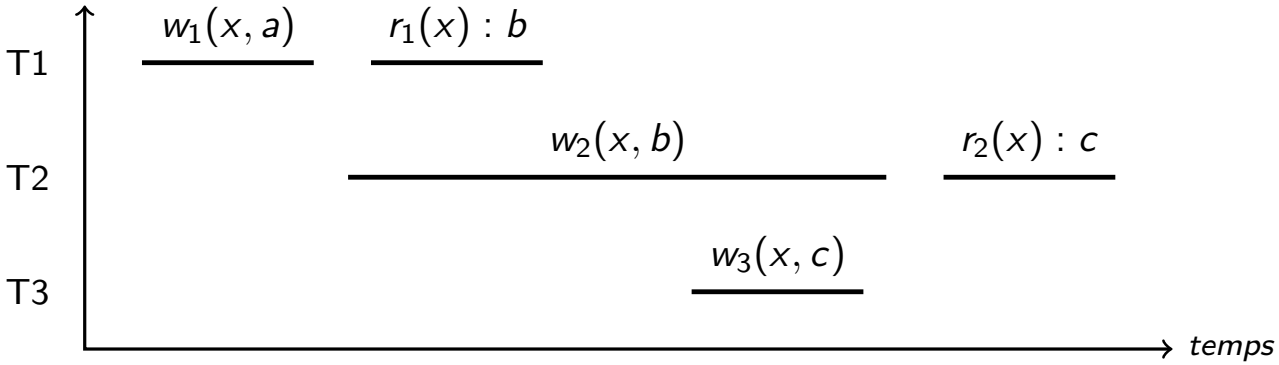
```
never {
  T0_init:
    if
    :: ((!(b)) && (a)) -> goto accept_S2
    :: ((true)) -> goto T0_init
    fi;
  accept_S2:
    if
    :: ((!(b))) -> goto accept_S2
    fi;
}
```



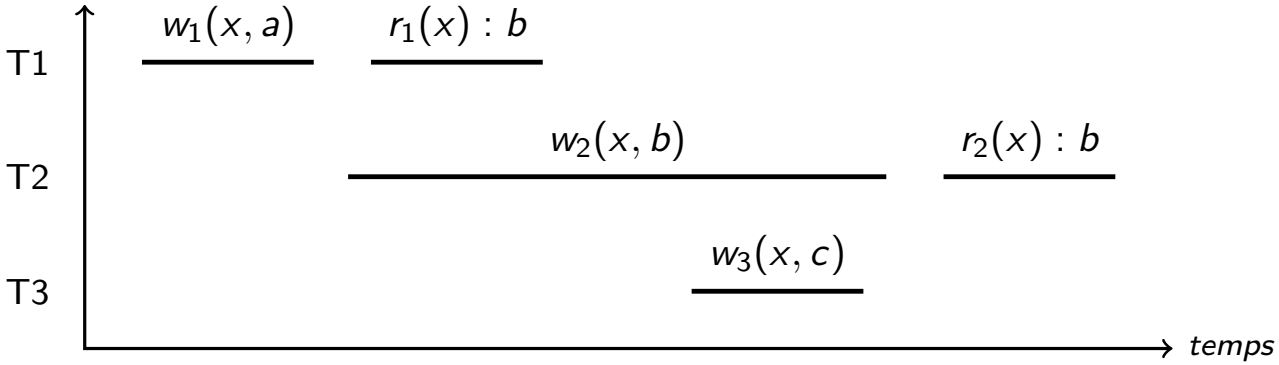
Linéarisabilité

Linéarisabilité : isolation au niveau des opérations

- Séquentialisation des opérations avec respect de la précédence temporelle
- Une lecture renvoie la dernière valeur écrite selon l'ordre établi



Histoire linéarisable ($w_1(x, a)$ - $w_2(x, b)$ - $r_1(x) : b$ - $w_3(x, c)$ - $r_2(x) : c$)



Histoire non linéarisable