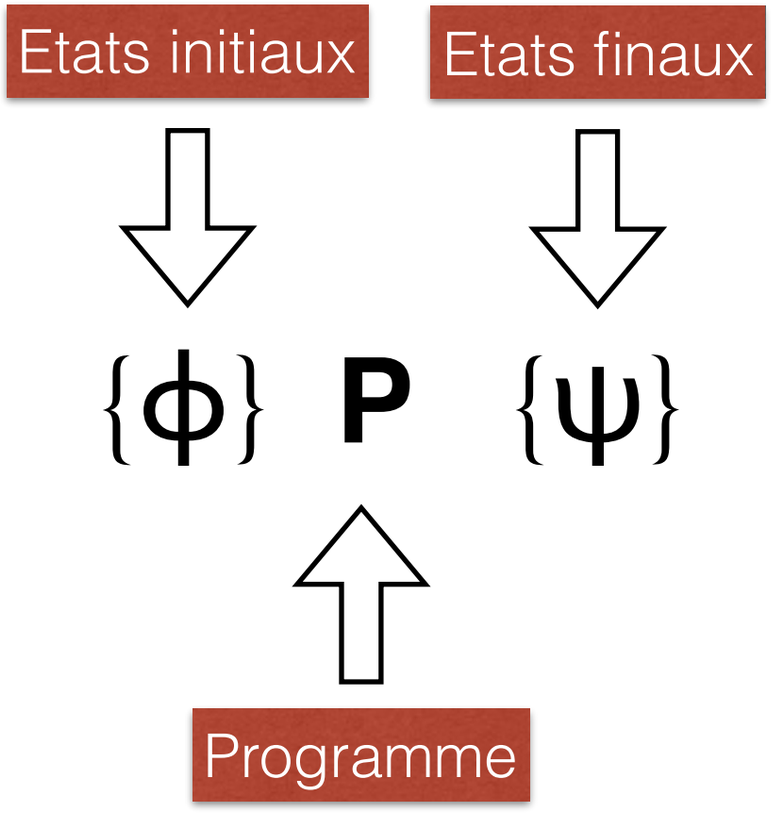
Candidature au concours CNRS 06/01

Radu IOSIF (VERIMAG)
CNRS / Université Grenoble Alpes

Plan de l’exposé

- Travaux effectués pendant la période 2002-2017
  - machines à compteurs plates
  - programmes avec structures de mémoire (listes, arbres, vecteurs)
  - logique de séparation
- Travaux en cours et futurs
  - parallélisme non-borné avec communication par conteneurs
  - automates étendus avec des données paramétrées
  - logique de séparation concurrente

Vérification de programmes



Pour tout état s dans ϕ, si le programme P termine avec s comme état initial, alors il termine dans un état présent dans ψ

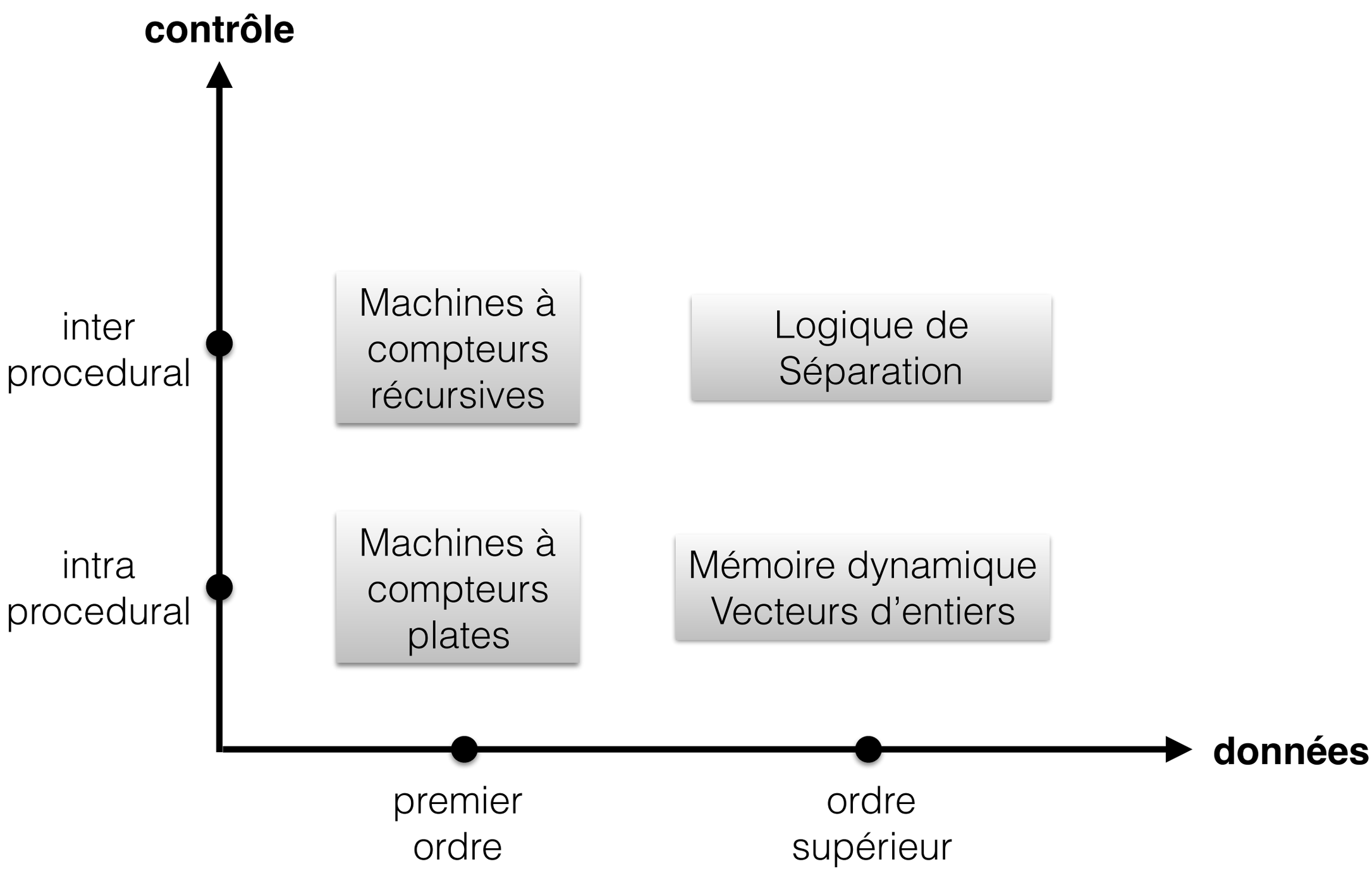
Vérification de programmes

Representations symboliques d'ensembles infinis d'états

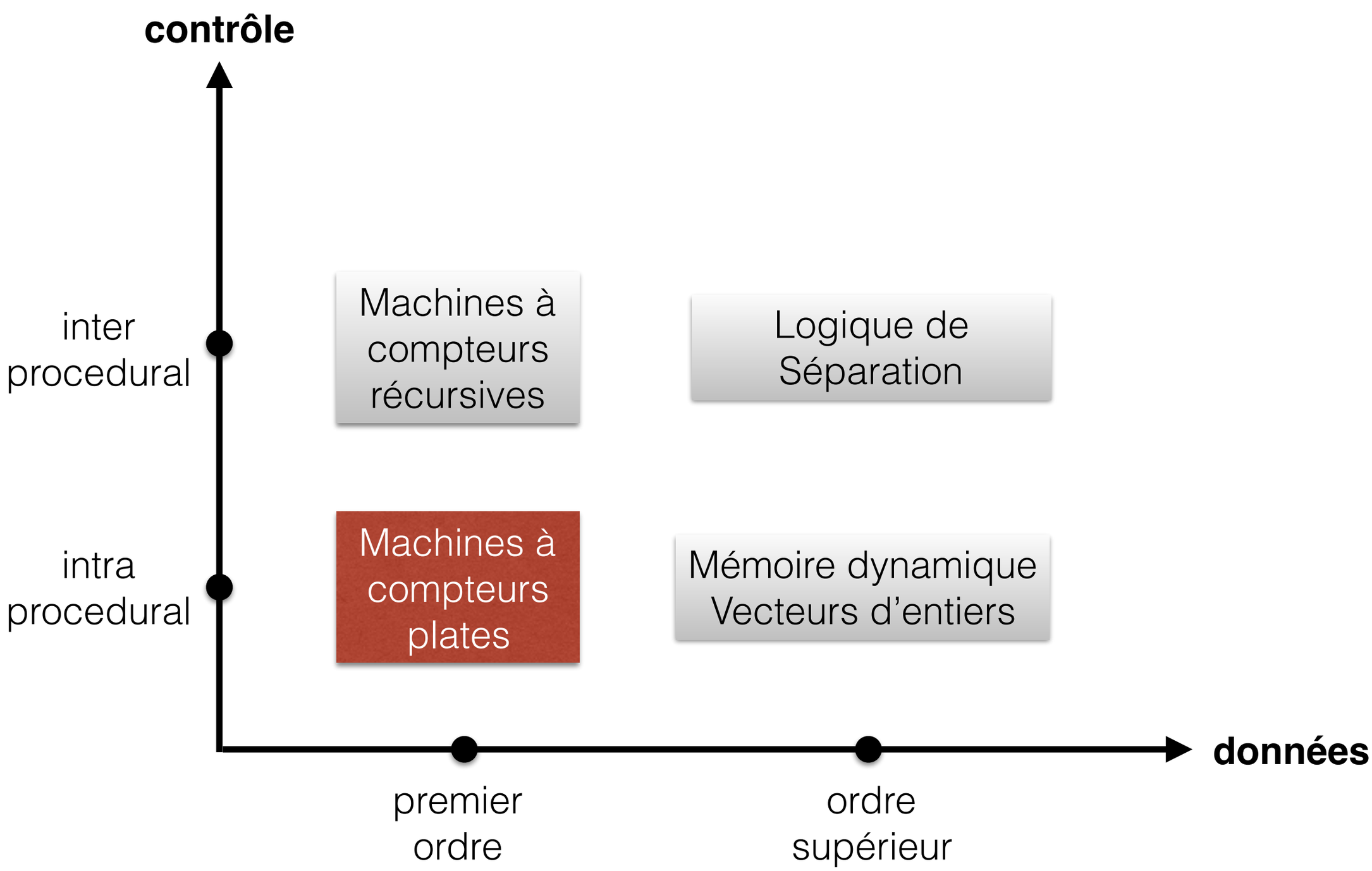
$$\{\phi\} \quad \mathbf{P} \quad \{\psi\}$$

Sémantique du programme

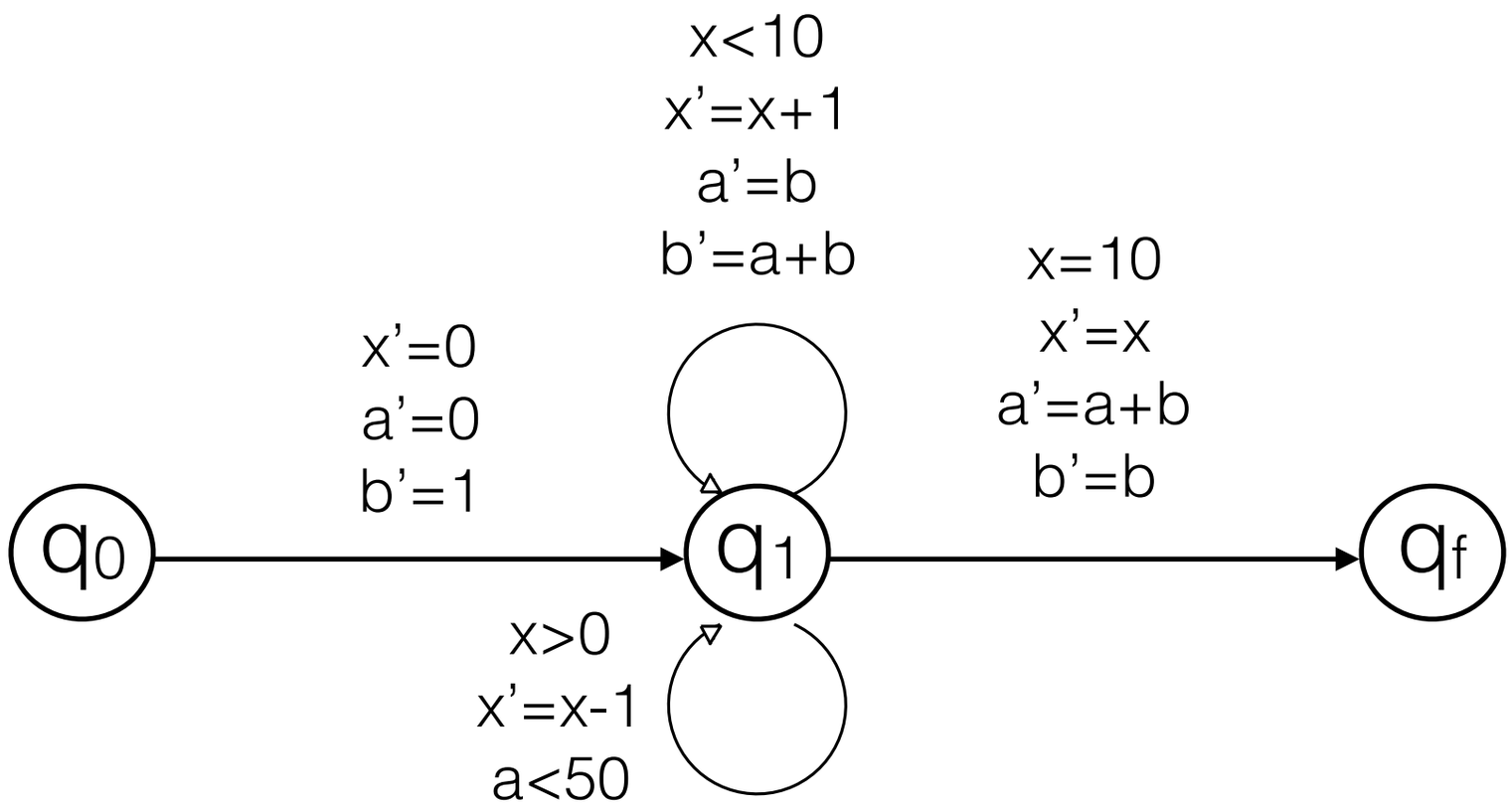
Recapitulatif des résultats (2002-2017)



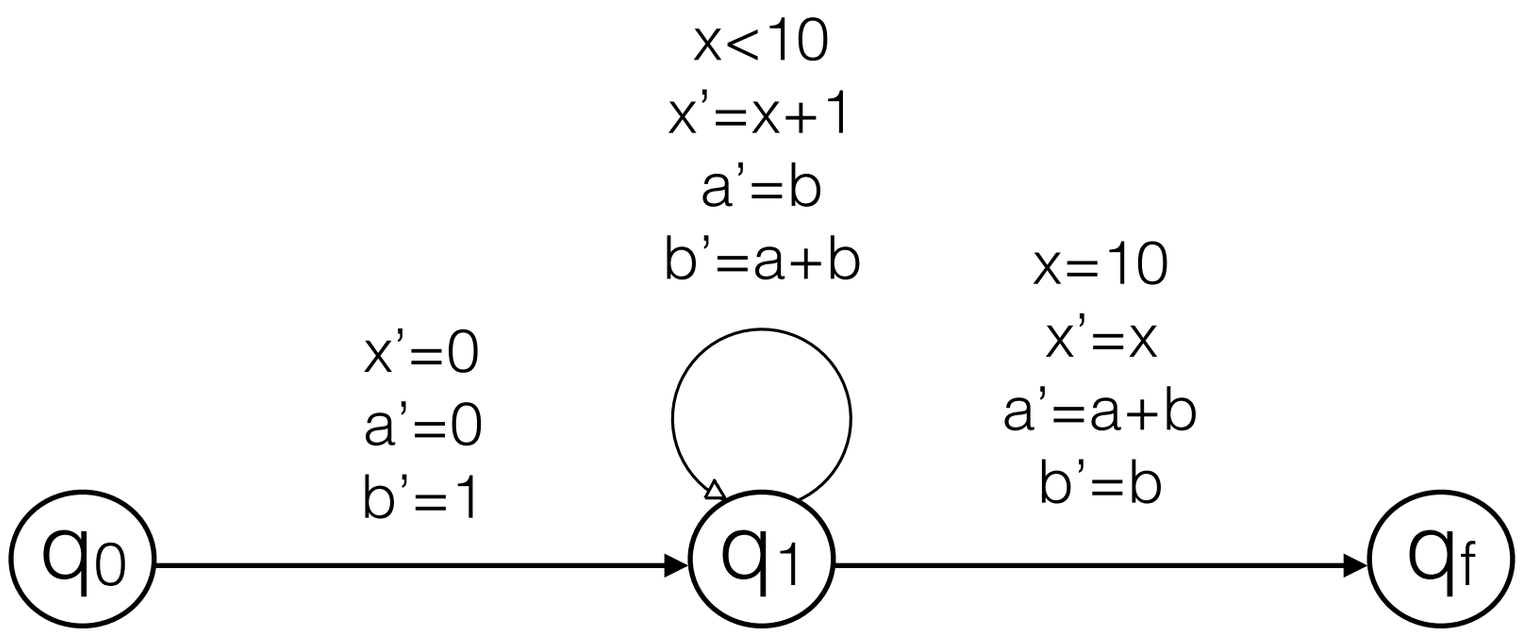
Recapitulatif des résultats (2002-2017)



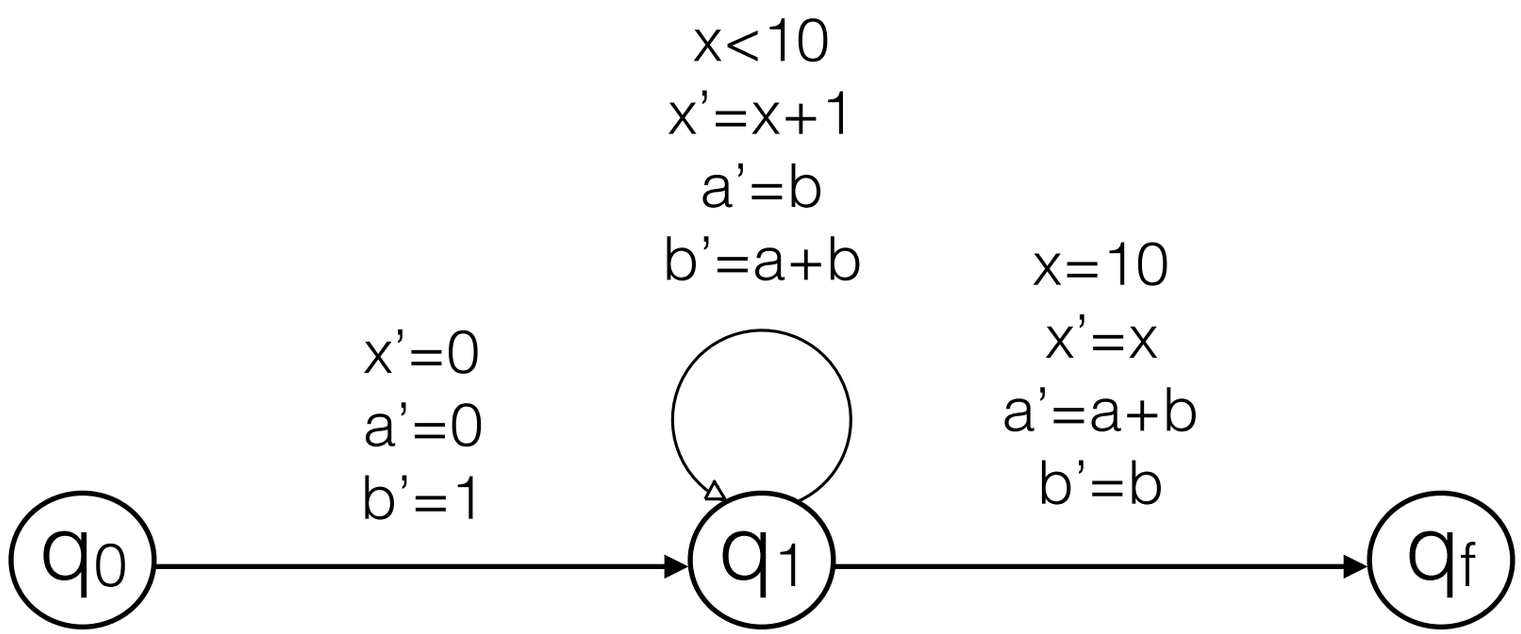
Machines à compteurs plates



Machines à compteurs plates

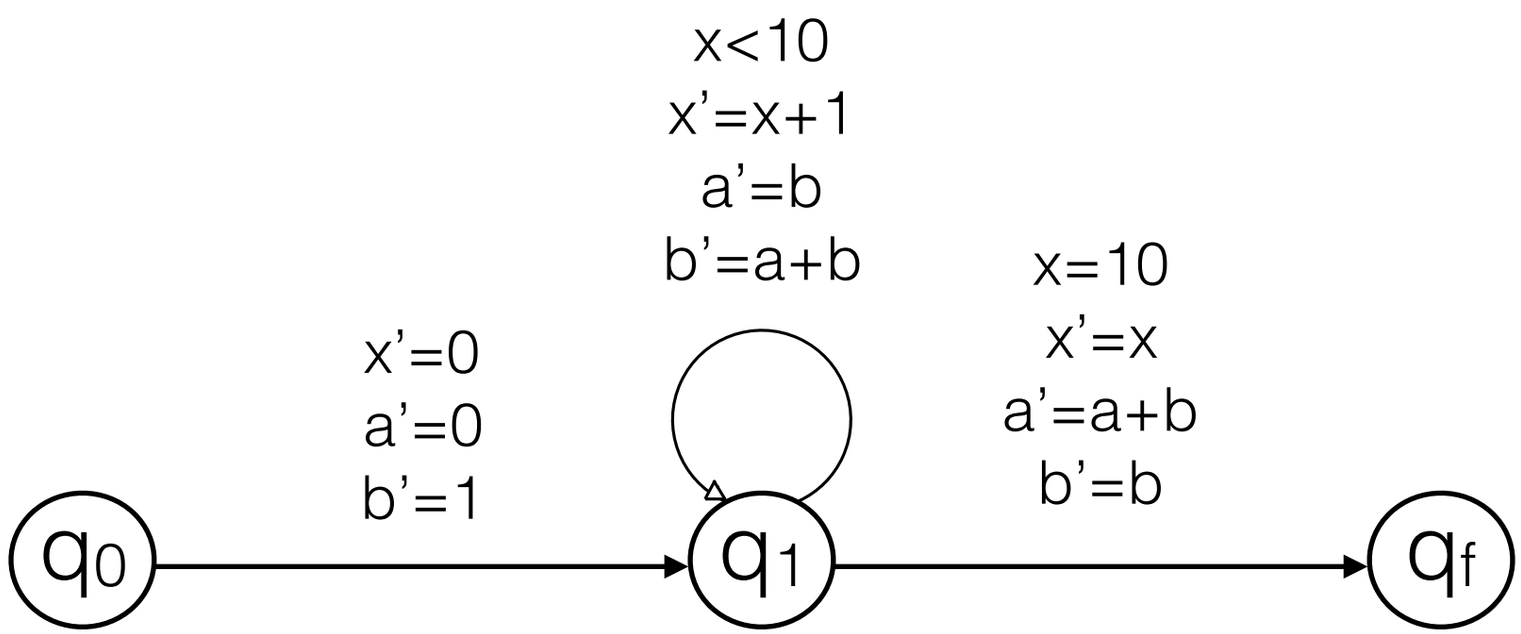


Machines à compteurs plates



Executions décrites par une expression régulière de la forme $\mathbf{w_1^* w_2^* \ldots w_n^*}$

Machines à compteurs plates



Executions décrites par une expression régulière de la forme $\mathbf{w_1^* w_2^* \ldots w_n^*}$

Vérification par **acceleration**: calcul des clôtures transitives des boucles

Machines à compteurs plates

Transformations affines $\phi(\vec{x}) \wedge \vec{x}' = M\vec{x} + \vec{v}$

Monoïdes finis [Boigelot'99, Finkel&Leroux'02]

$$M^0, M^1, \ldots, M^b, \ldots, M^{b+c}$$

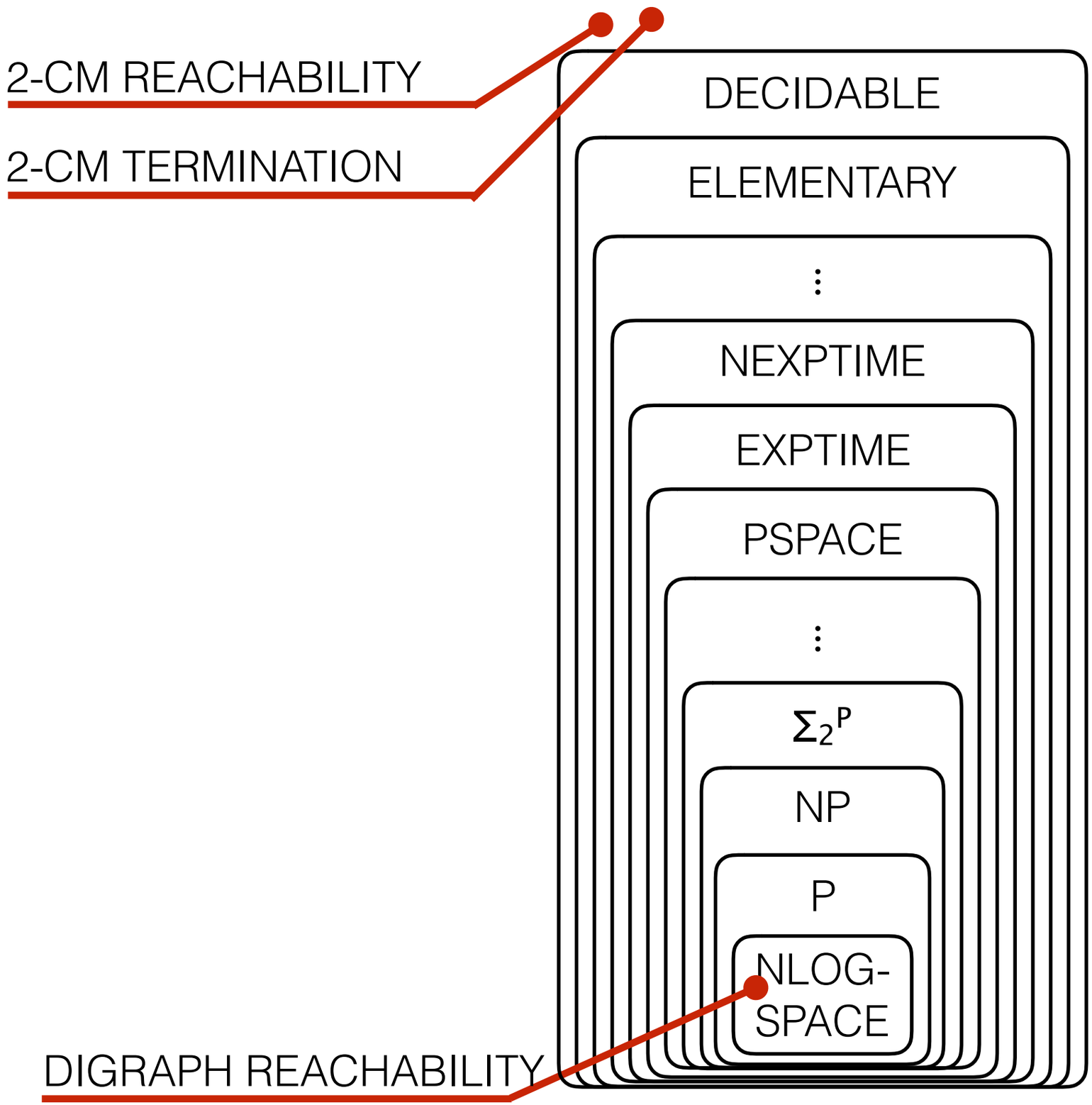
Difference bounds [Comon&Jurski'98]

$$\bigwedge_{ij} x_i - x_j \leq a_{ij} \wedge \bigwedge_{ij} x'_i - x_j \leq b_{ij} \wedge \bigwedge_{ij} x_i - x'_j \leq c_{ij} \wedge \bigwedge_{ij} x'_i - x'_j \leq d_{ij}$$

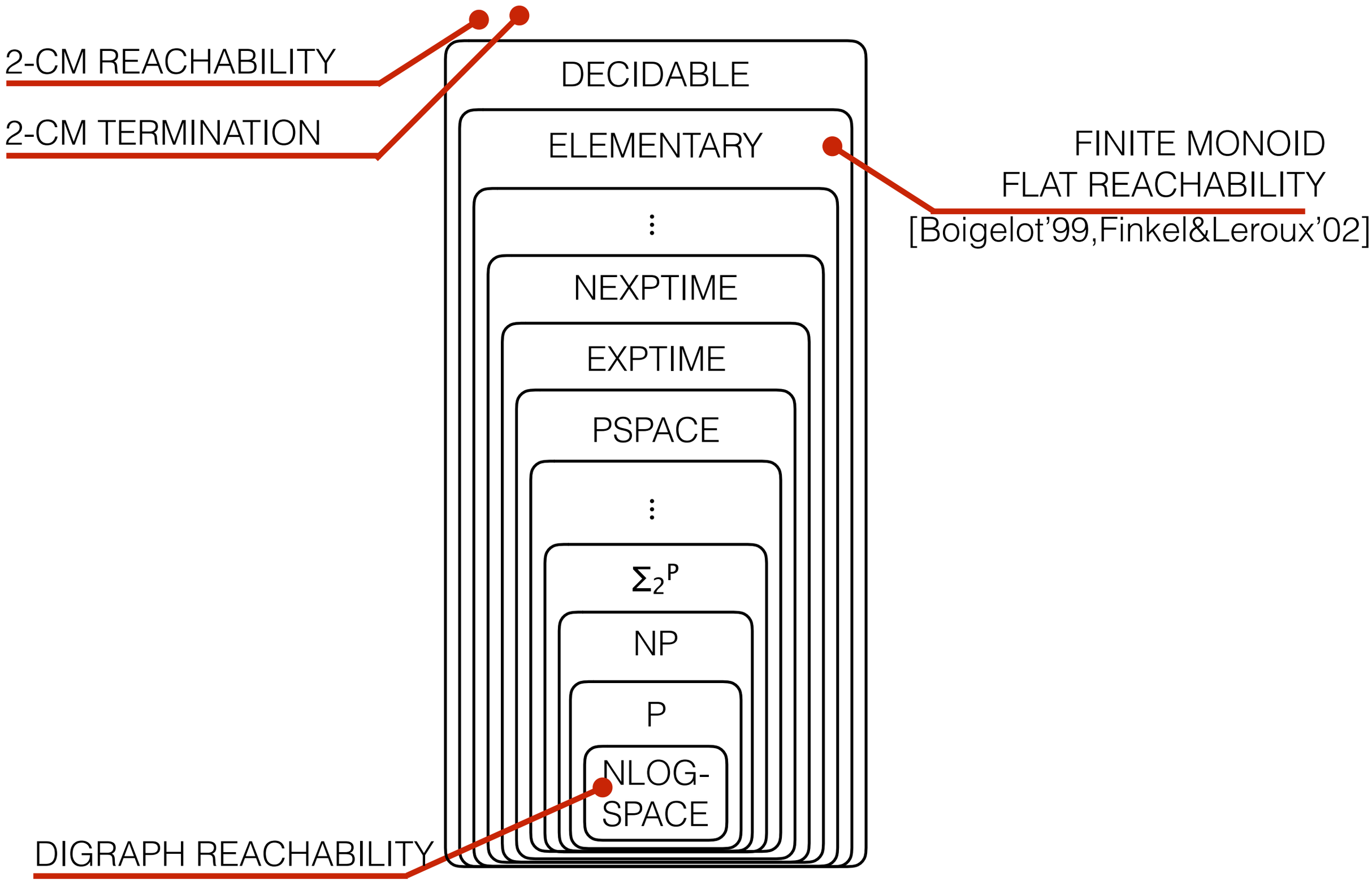
Octogones [Harvey&Stuckey'97, Miné'01]

$$\bigwedge_{ij} \pm x_i \pm x_j \leq a_{ij} \wedge \bigwedge_{ij} \pm x'_i \pm x_j \leq b_{ij} \wedge \bigwedge_{ij} \pm x'_i \pm x'_j \leq c_{ij}$$

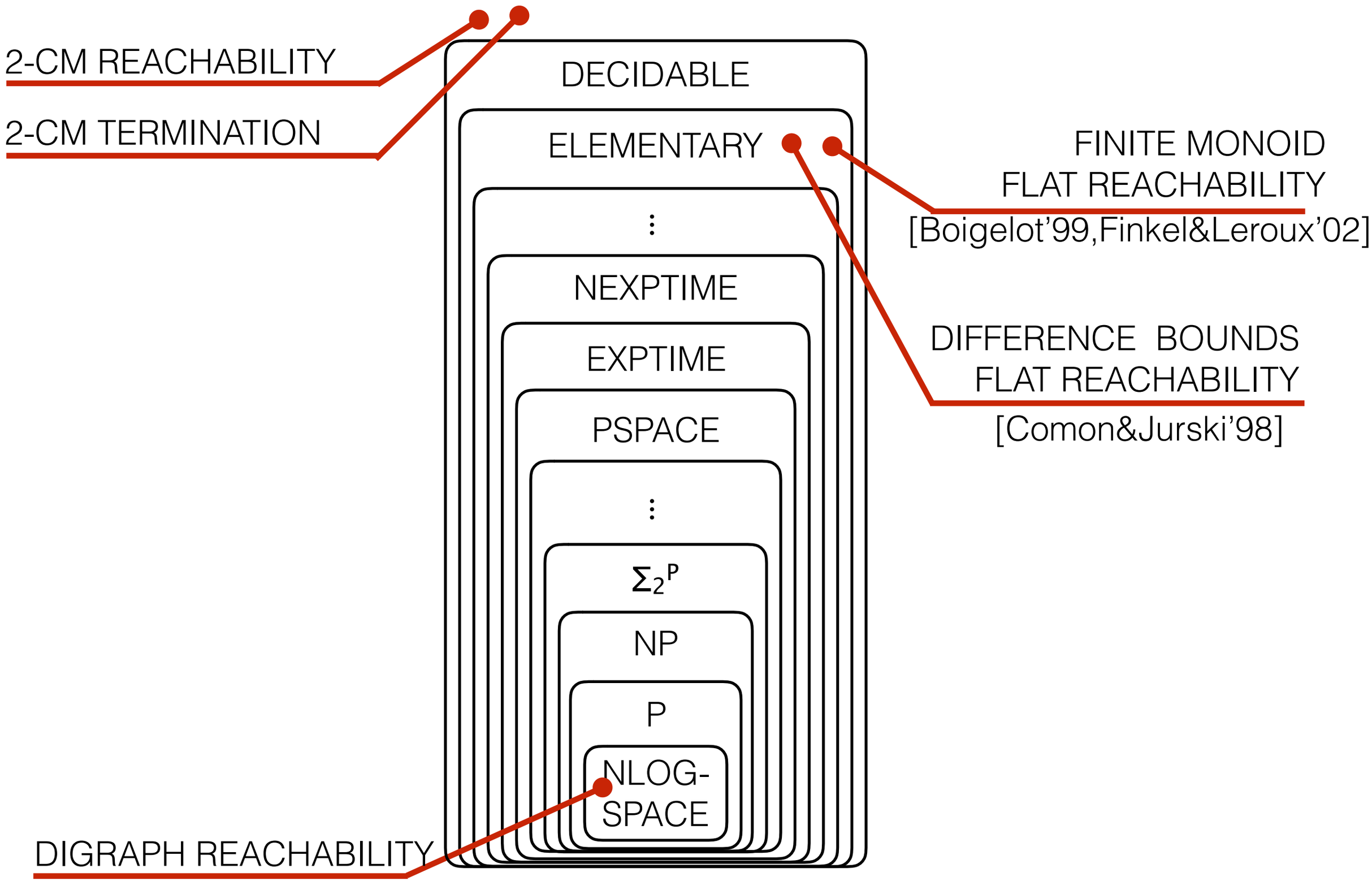
Complexité


2-CM REACHABILITY
2-CM TERMINATION
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ_2^P
NP
P
NLOG-SPACE
DIGRAPH REACHABILITY

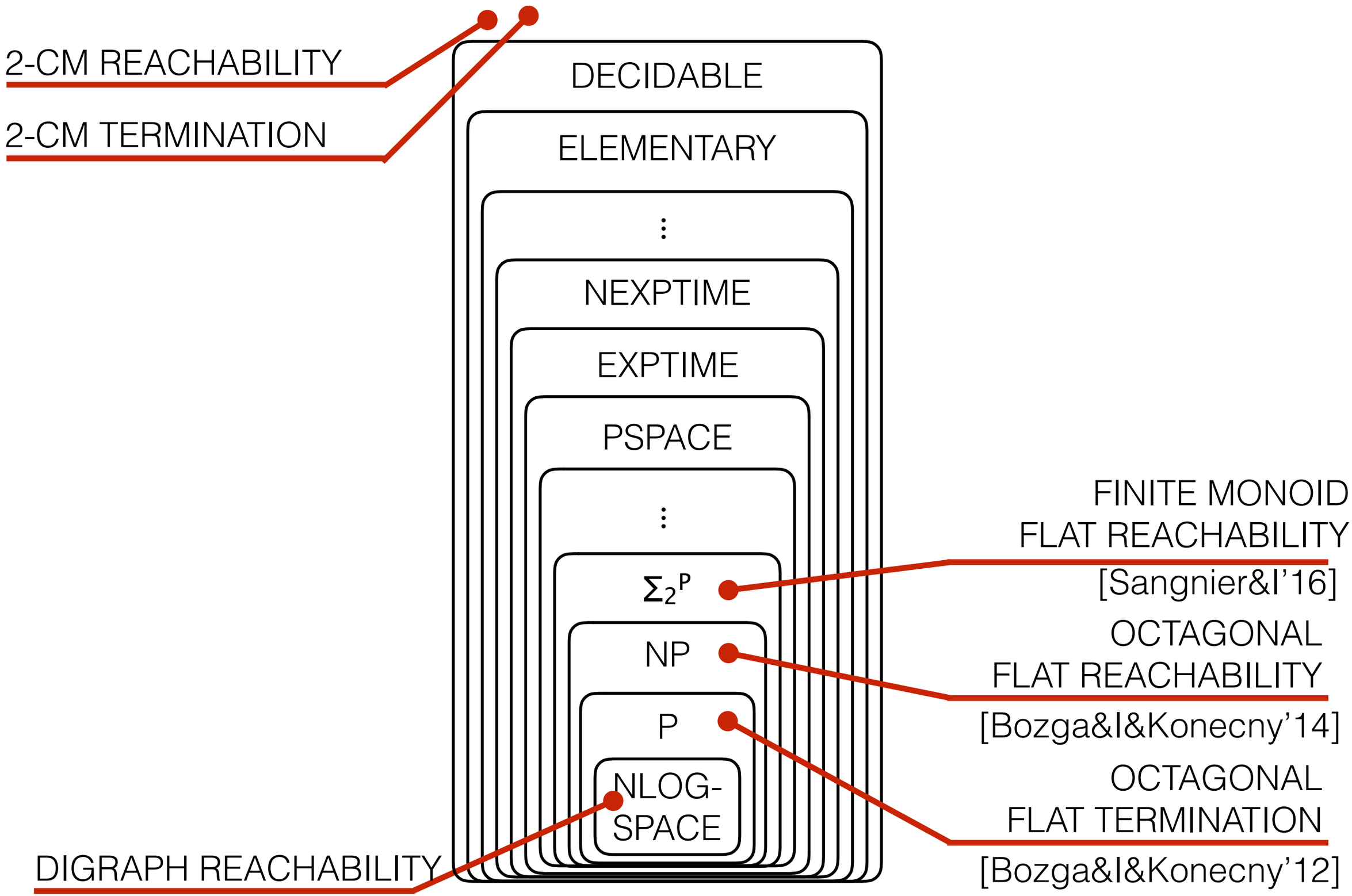
Complexité


2-CM REACHABILITY
2-CM TERMINATION
DECIDABLE
ELEMENTARY
FINITE MONOID
FLAT REACHABILITY
[Boigelot'99,Finkel&Leroux'02]
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ_2^P
NP
P
NLOG-
SPACE
DIGRAPH REACHABILITY

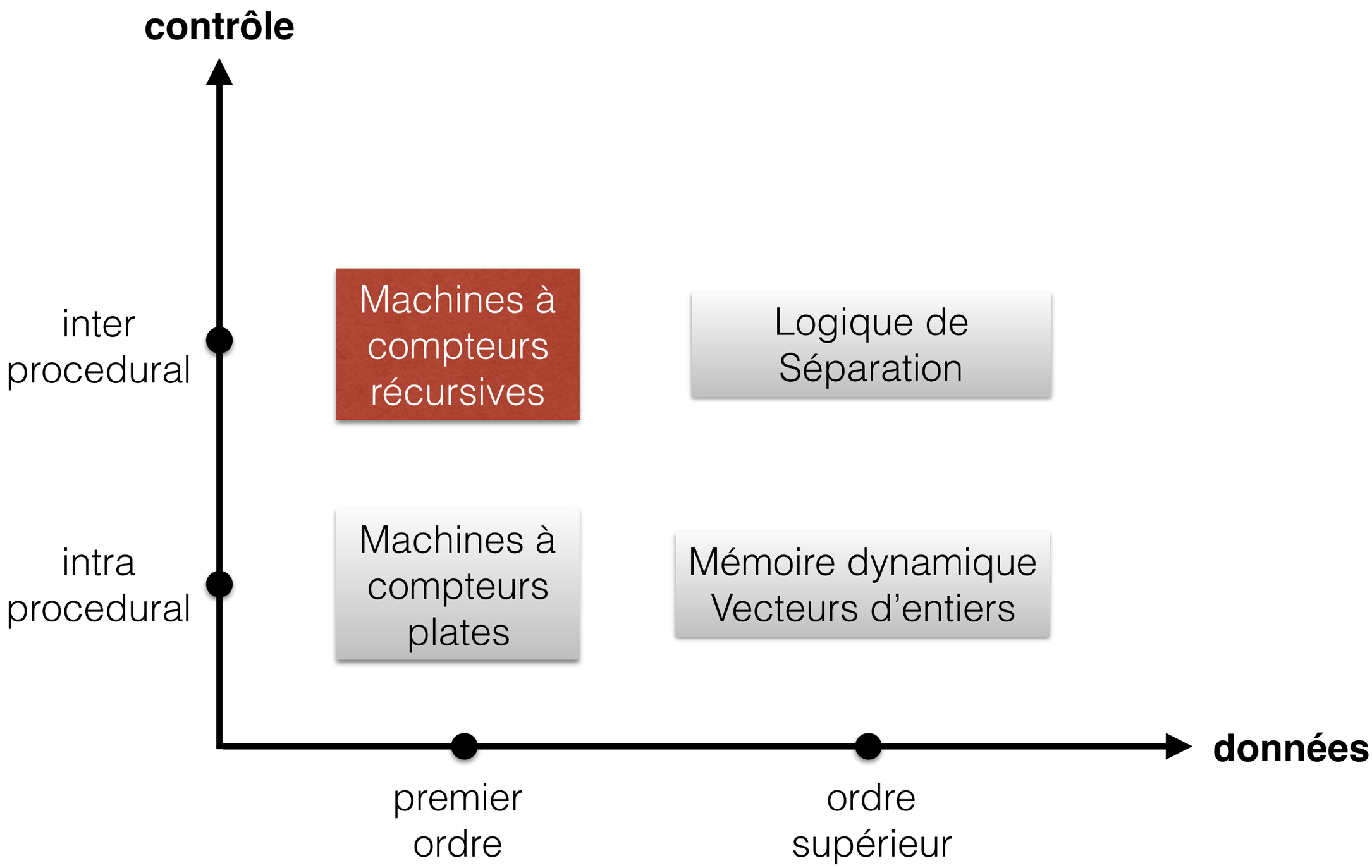
Complexité


2-CM REACHABILITY
2-CM TERMINATION
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ_2^P
NP
P
NLOG-SPACE
FINITE MONOID
FLAT REACHABILITY
[Boigelot'99,Finkel&Leroux'02]
DIFFERENCE BOUNDS
FLAT REACHABILITY
[Comon&Jurski'98]
DIGRAPH REACHABILITY

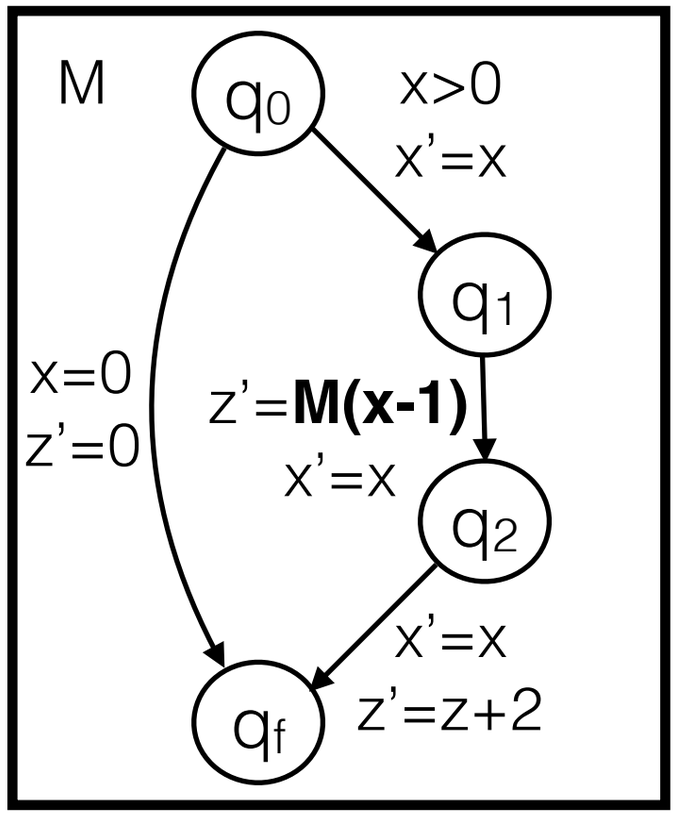
Complexité


2-CM REACHABILITY
2-CM TERMINATION
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ_2^P
NP
P
NLOG-SPACE
FINITE MONOID FLAT REACHABILITY
[Sangnier&I'16]
OCTAGONAL FLAT REACHABILITY
[Bozga&I&Konecny'14]
OCTAGONAL FLAT TERMINATION
[Bozga&I&Konecny'12]
DIGRAPH REACHABILITY

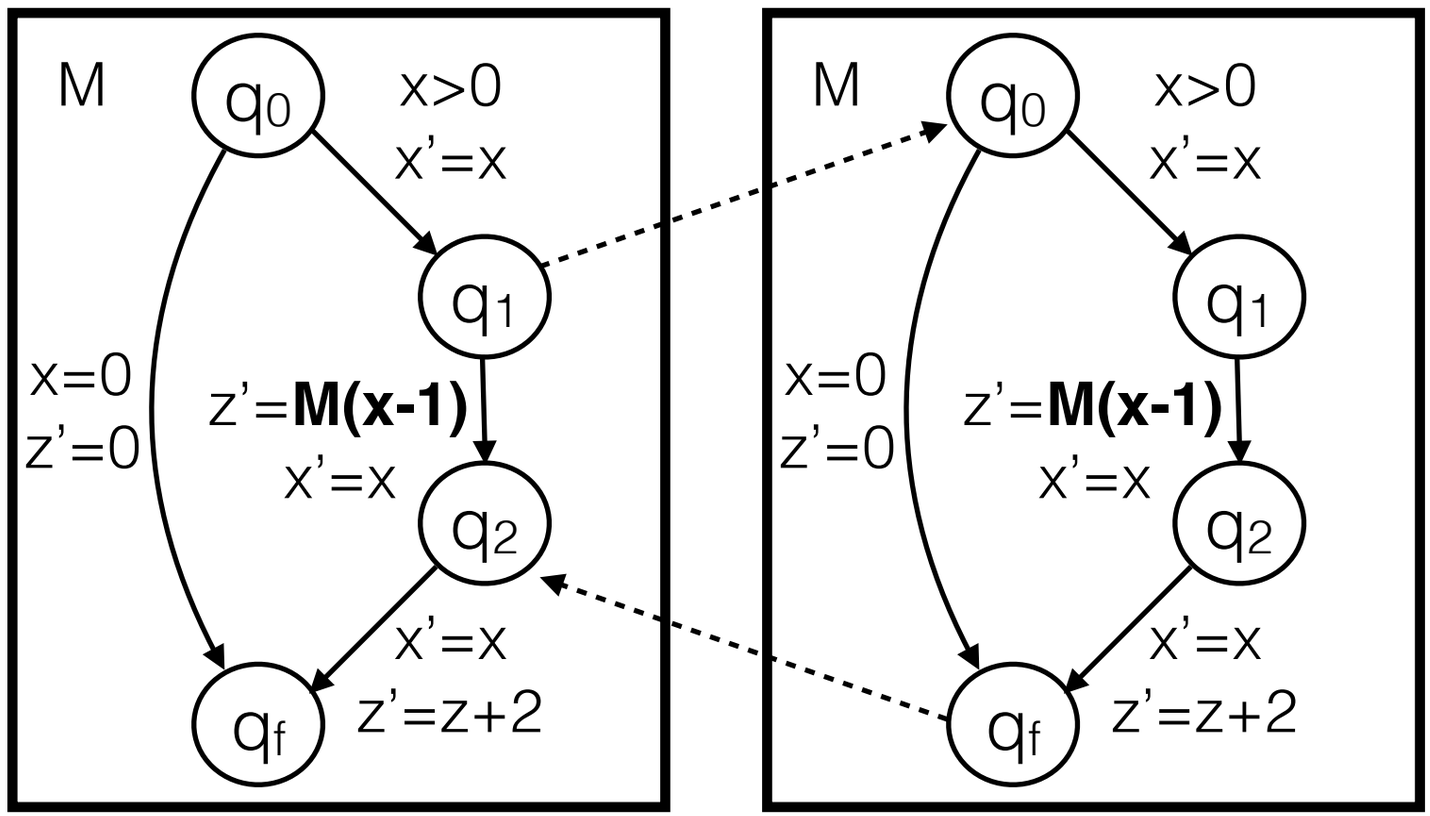
Recapitulatif des résultats (2002-2017)



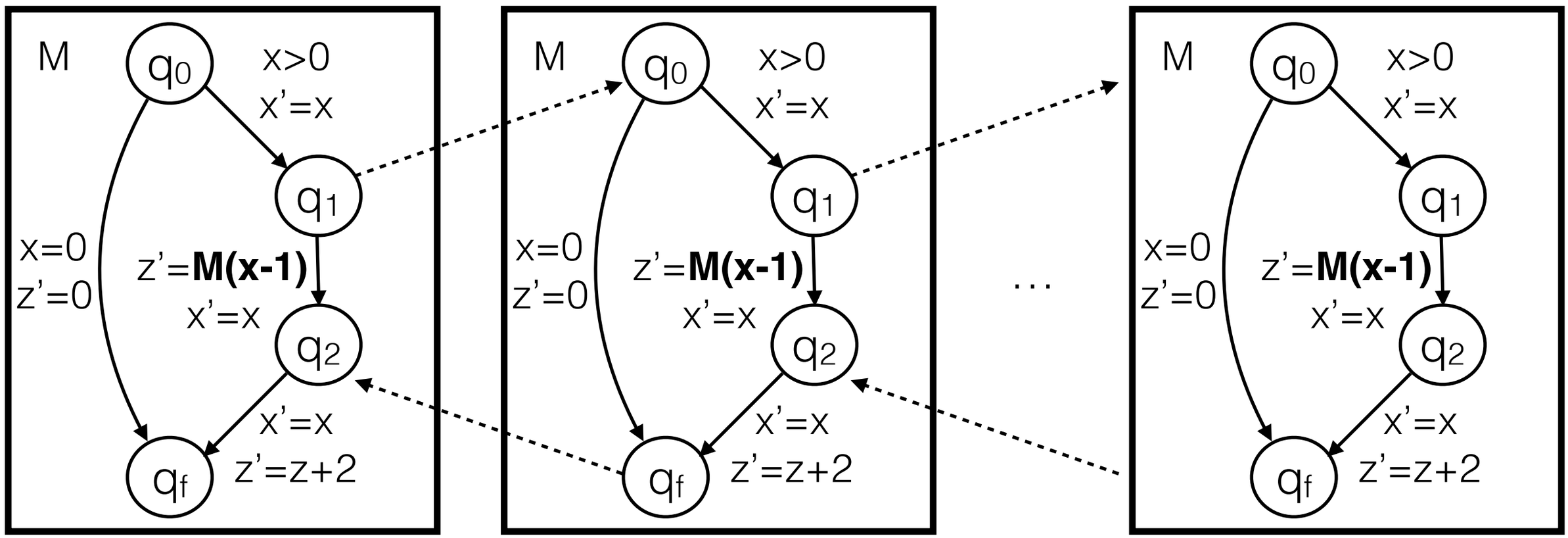
Machines à compteurs récursives



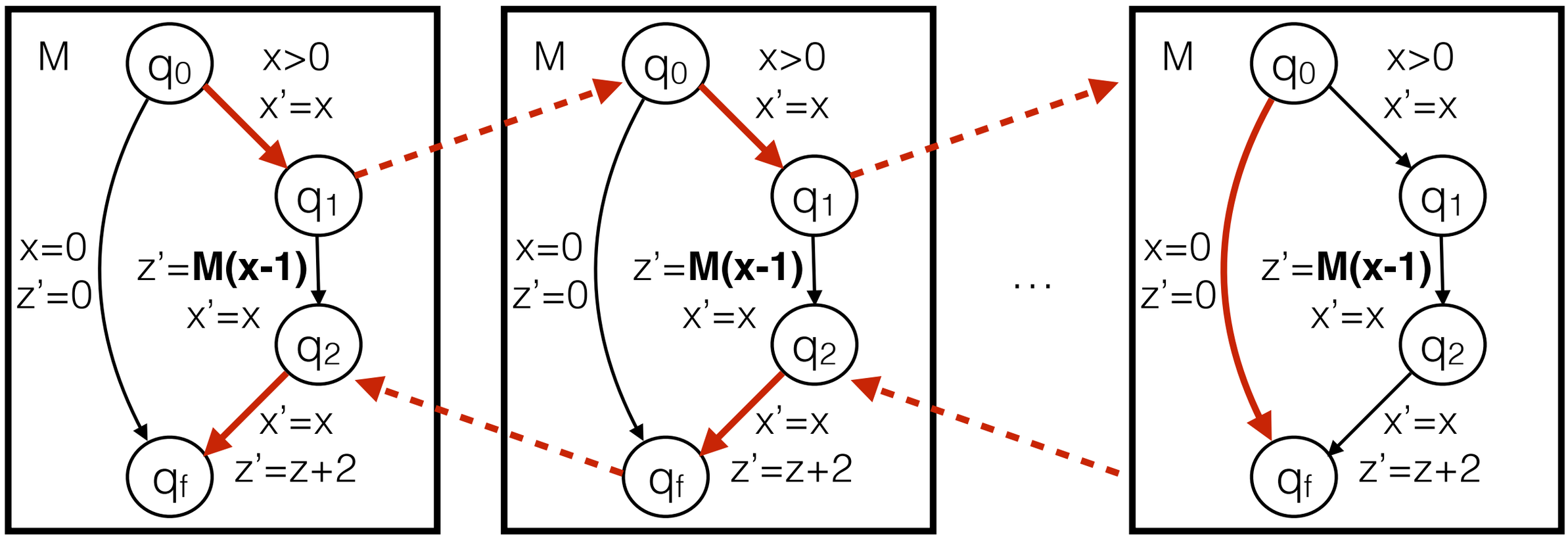
Machines à compteurs récursives



Machines à compteurs récursives

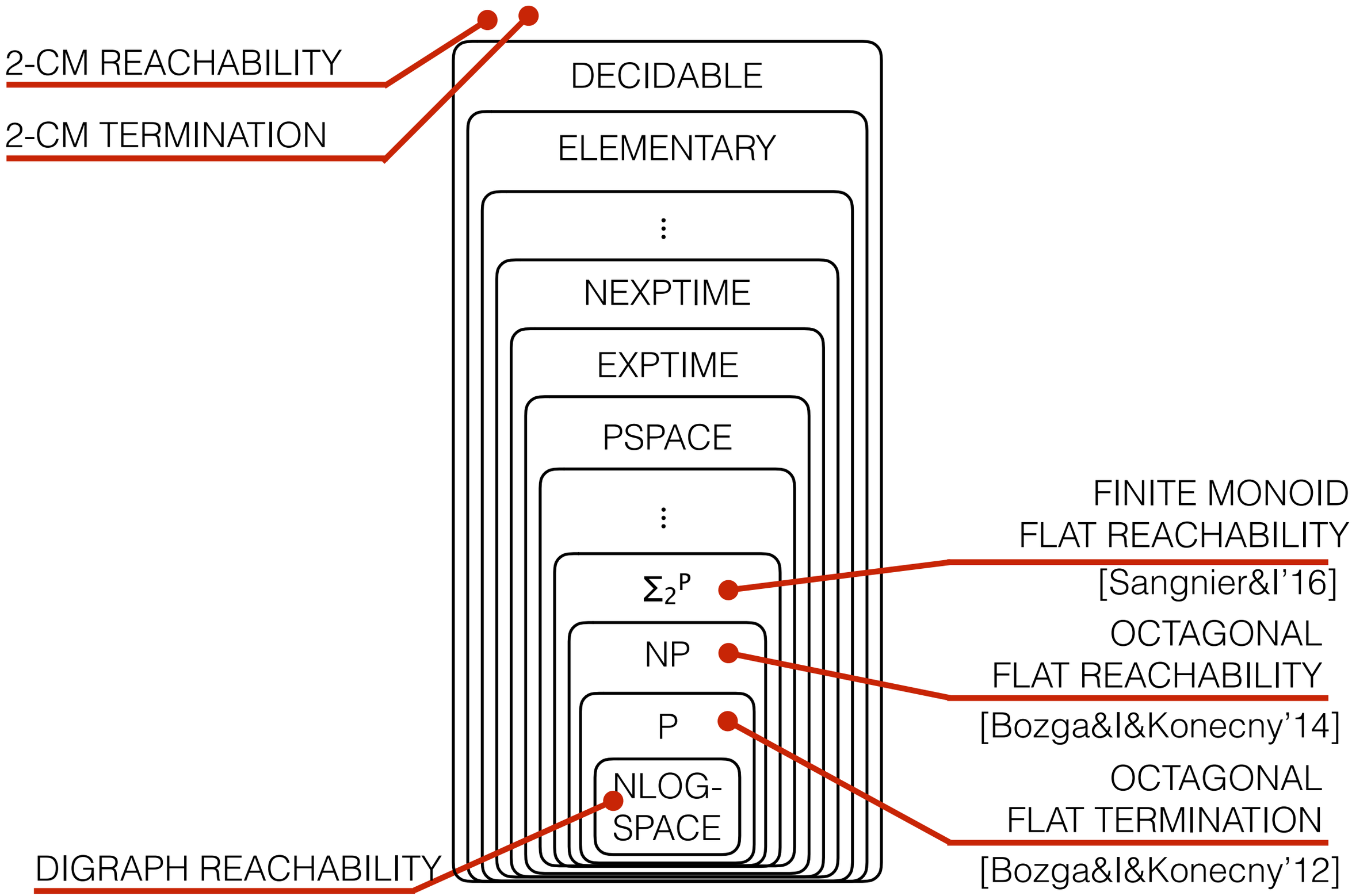


Machines à compteurs récursives

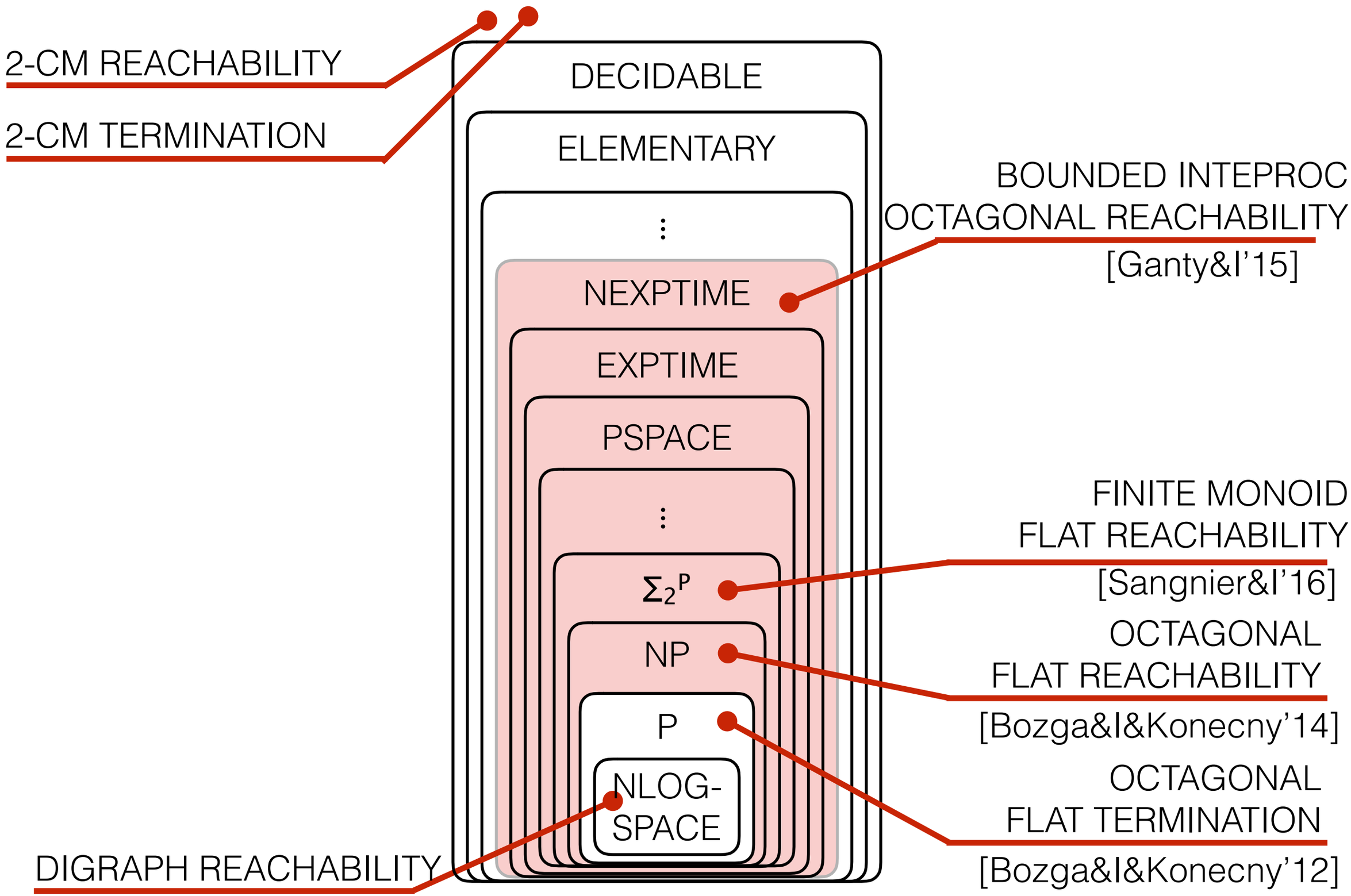


On considère que des executions décrites par un langage hors-contexte borné, c.a.d. inclus dans un langage régulièr de la forme $\mathbf{w_1^* w_2^* \ldots w_n^*}$

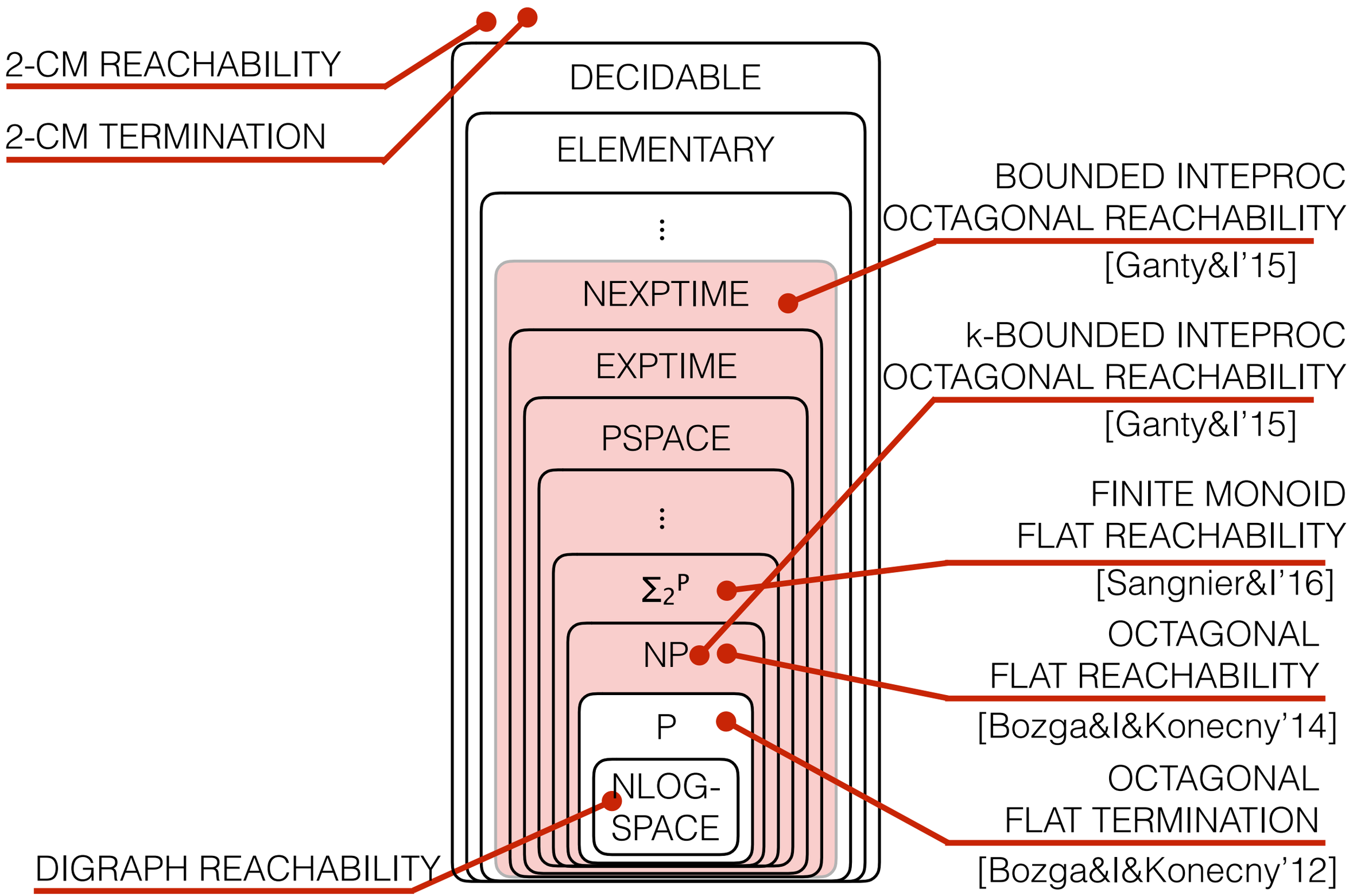
Complexité


2-CM REACHABILITY
2-CM TERMINATION
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ_2^P
NP
P
NLOG-SPACE
FINITE MONOID FLAT REACHABILITY
[Sangnier&I'16]
OCTAGONAL FLAT REACHABILITY
[Bozga&I&Konecny'14]
OCTAGONAL FLAT TERMINATION
[Bozga&I&Konecny'12]
DIGRAPH REACHABILITY

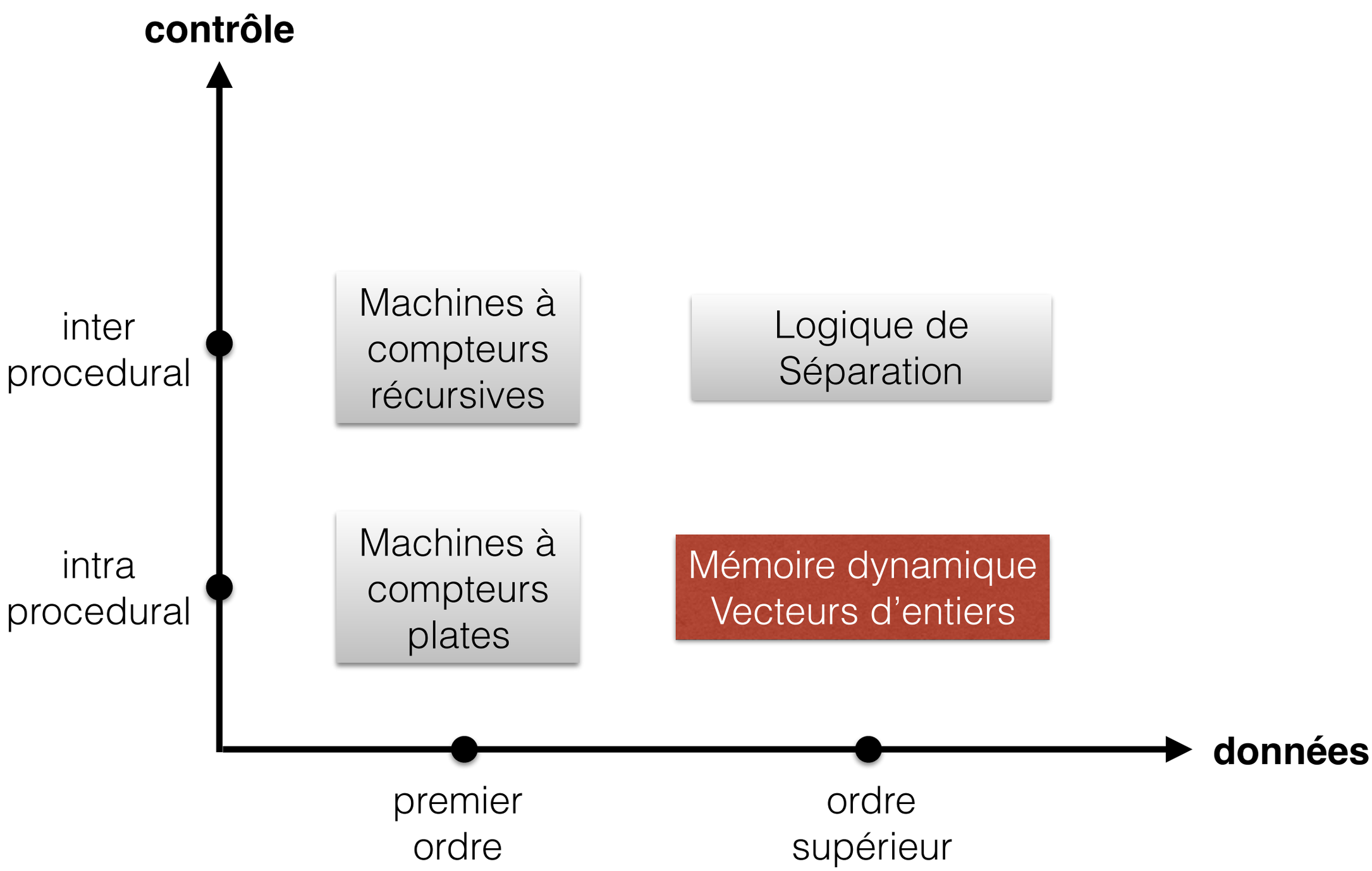
Complexité


2-CM REACHABILITY
2-CM TERMINATION
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ_2^P
NP
P
NLOG-SPACE
BOUNDED INTEPROC
OCTAGONAL REACHABILITY
[Ganty&I'15]
FINITE MONOID
FLAT REACHABILITY
[Sangnier&I'16]
OCTAGONAL
FLAT REACHABILITY
[Bozga&I&Konecny'14]
OCTAGONAL
FLAT TERMINATION
[Bozga&I&Konecny'12]
DIGRAPH REACHABILITY

Complexité

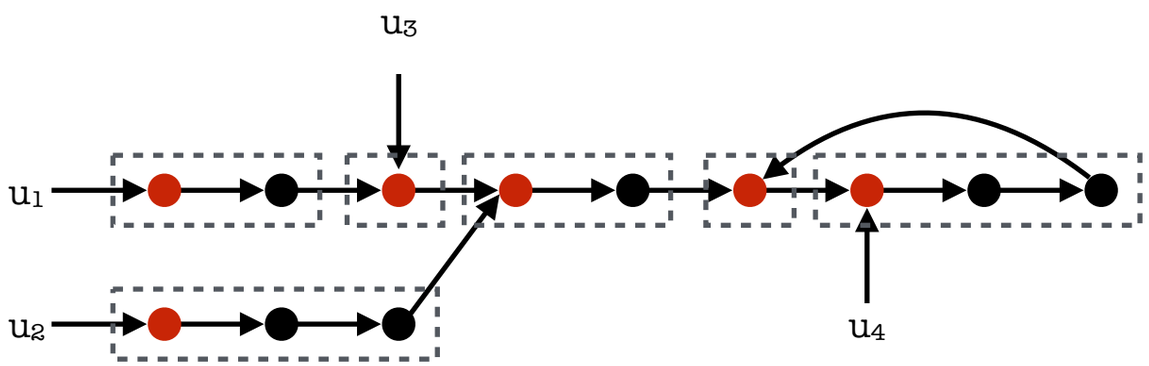

2-CM REACHABILITY
2-CM TERMINATION
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ_2^P
NP
P
NLOG-SPACE
BOUNDED INTEPROC OCTAGONAL REACHABILITY [Ganty&I'15]
k-BOUNDED INTEPROC OCTAGONAL REACHABILITY [Ganty&I'15]
FINITE MONOID FLAT REACHABILITY [Sangnier&I'16]
OCTAGONAL FLAT REACHABILITY [Bozga&I&Konecny'14]
OCTAGONAL FLAT TERMINATION [Bozga&I&Konecny'12]
DIGRAPH REACHABILITY

Recapitulatif des résultats (2002-2017)



Listes simplement chaînées

```
struct ListNode {
   struct ListNode * next;
} * List;
```



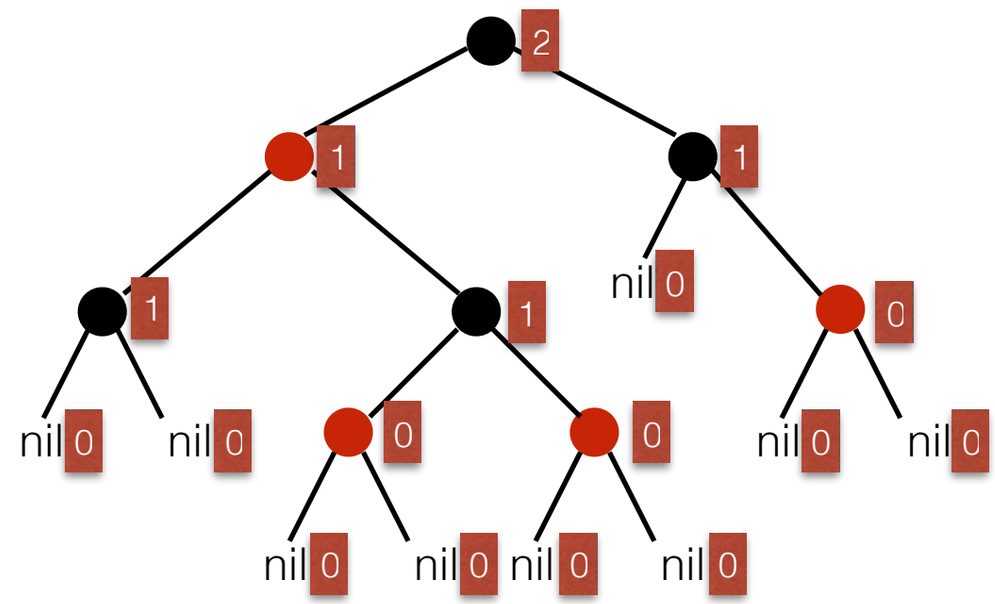
Théorème de bisimulation permettant de reduire les problèmes de verification pour des programmes avec des listes à de problèmes d'atteignabilité/terminaison pour des machines à compteurs [Bouajjani&Bozga&Habermehl&I&Moro&Vojnar'06]

Résultats de (in)décidabilité sur de classes de programmes avec des listes [Bozga&I'07]

- extension de l'arithmétique de Presburger avec divisibilité [Bozga&I'05]

Arbres et contraintes numériques

```
struct TreeNode {
  enum {red, black} color;
   struct TreeNode *up;
   struct TreeNode *left;
   struct TreeNode *right;
} * Tree;
```



Vérification des pre/post-conditions et invariants pour arbres equilibrés (AVL, rouge-noir) [Habermehl&I&Vojnar'06]

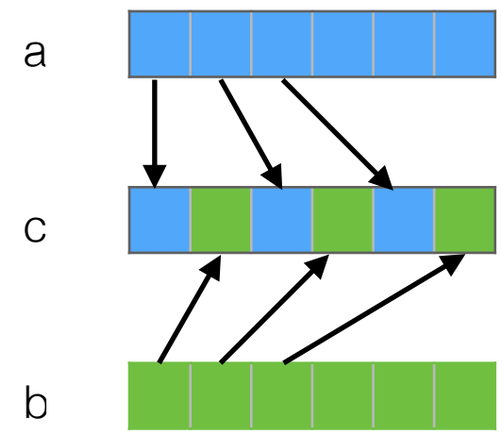
- automates d'arbre avec contraintes de taille (TASC)
- classe fermée par operations booléennes (union, intersection, complement)
- determinisation, minimisation

Preuves de terminaison utilisant des simulations par machines à compteurs [Habermehl&I&Rogalewicz&Vojnar'07, I&Rogalewicz'09]

Vecteurs d’entiers

```
for (k=0, l=0; k < n; k++, l+=2) {

   c[l]=a[k];
   c[l+1]=b[k];
}
```



Vecteurs d'entiers

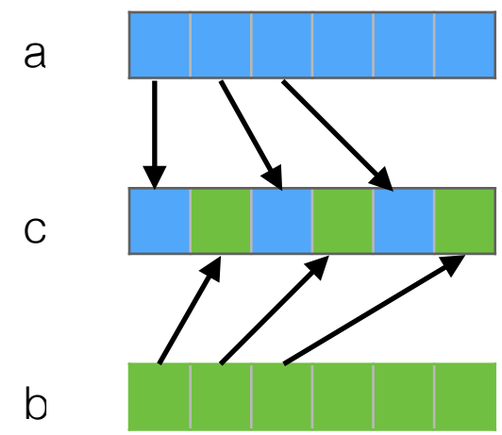
$n>0 \wedge \forall\, i,j\,.\, 0 \leq i,j < n \rightarrow a[i] \leq b[j]$

```
for (k=0, l=0; k < n; k++, l+=2) {
```

$n>0 \wedge k \leq n \wedge l=2k \wedge$
$\forall\, i,j\,.\, 0 \leq i,j < 2k \wedge i \equiv_2 0 \wedge j \equiv_2 1 \rightarrow c[i] \leq c[j] \wedge$
$\forall\, i,j\,.\, 0 \leq i,j < n \rightarrow a[i] \leq b[j]$

```
    c[l]=a[k];
    c[l+1]=b[k];
}
```

$n>0 \wedge \forall\, i,j\,.\, 0 \leq i,j < 2n \wedge i \equiv_2 0 \wedge j \equiv_2 1 \rightarrow c[i] \leq c[j]$



Vecteurs d'entiers

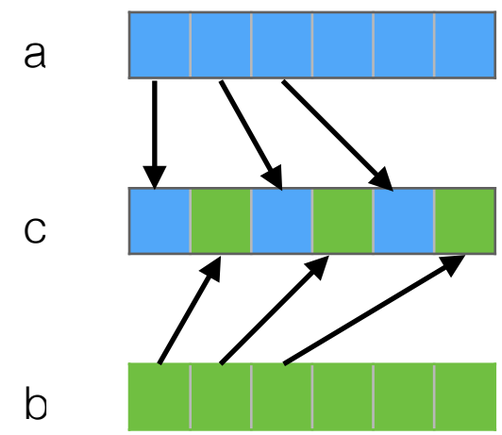
$n>0 \wedge \forall\, i,j\,.\, 0 \le i,j < n \rightarrow a[i] \le b[j]$

```
for (k=0, l=0; k < n; k++, l+=2) {
```

$n>0 \wedge k \le n \wedge l=2k \wedge$
$\forall\, i,j\,.\, 0 \le i,j < 2k \wedge i \equiv_2 0 \wedge j \equiv_2 1 \rightarrow c[i] \le c[j] \wedge$
$\forall\, i,j\,.\, 0 \le i,j < n \rightarrow a[i] \le b[j]$

```
    c[l]=a[k];
    c[l+1]=b[k];
}
```

$n>0 \wedge \forall\, i,j\,.\, 0 \le i,j < 2n \wedge i \equiv_2 0 \wedge j \equiv_2 1 \rightarrow c[i] \le c[j]$

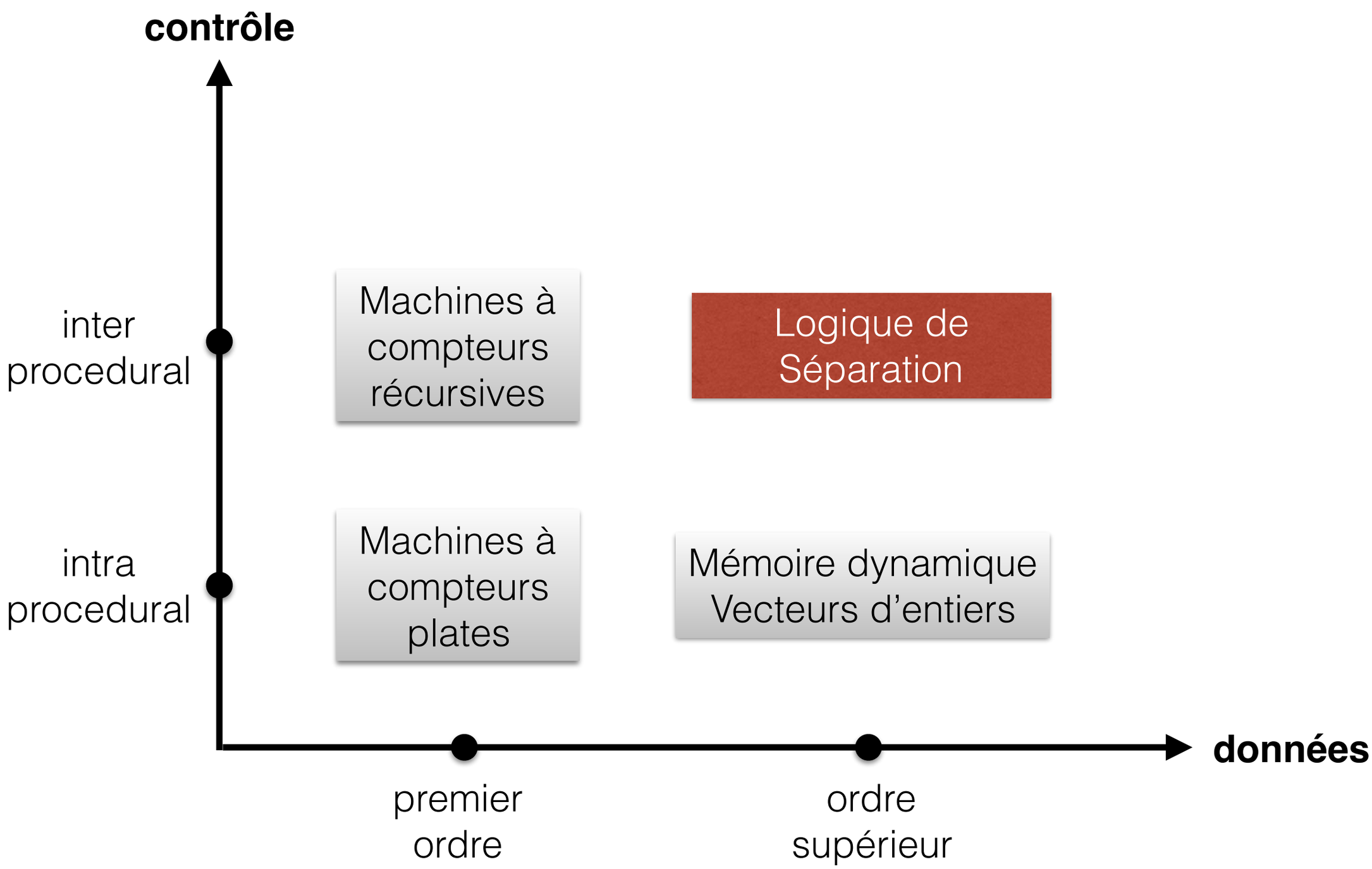


Logiques décidables pour la vérification de conditions de preuve [Habermehl&I&Vojnar'08]

- combinaisons booléennes de formules universelles $\forall\, i,j\,.\, g(i,j) \rightarrow v(a[i+n],b[j+m])$
- traduction vers machines à compteurs plates avec boucles octogonales

Codage de la sémantique des boucles de programme comme machines à compteurs [Bozga,Habermehl&I&Vojnar'09]

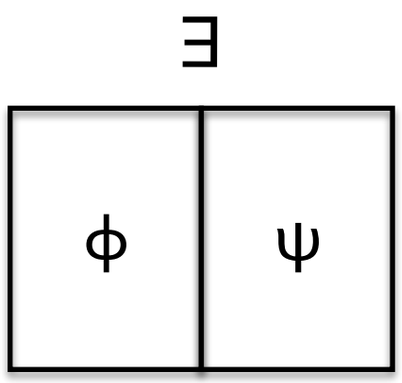
Recapitulatif des résultats (2002-2017)



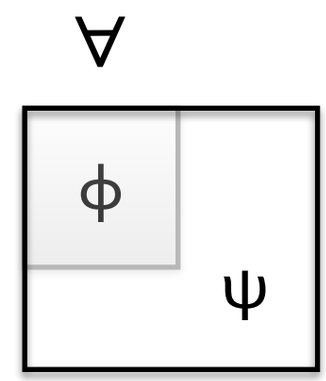
Logique de séparation

Logique de premier ordre sous-structurelle [Ishtiaq,O'Hearn'99] [Reynolds'99]

- opérateurs adjoints booléens ($\wedge$,$\rightarrow$) et spatiaux ($*$,$-*$)



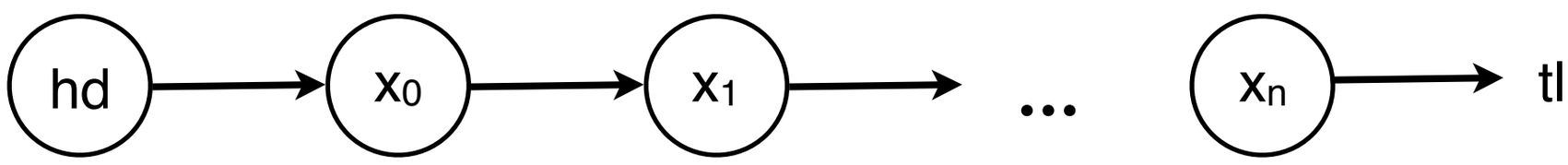
$\phi * \psi$



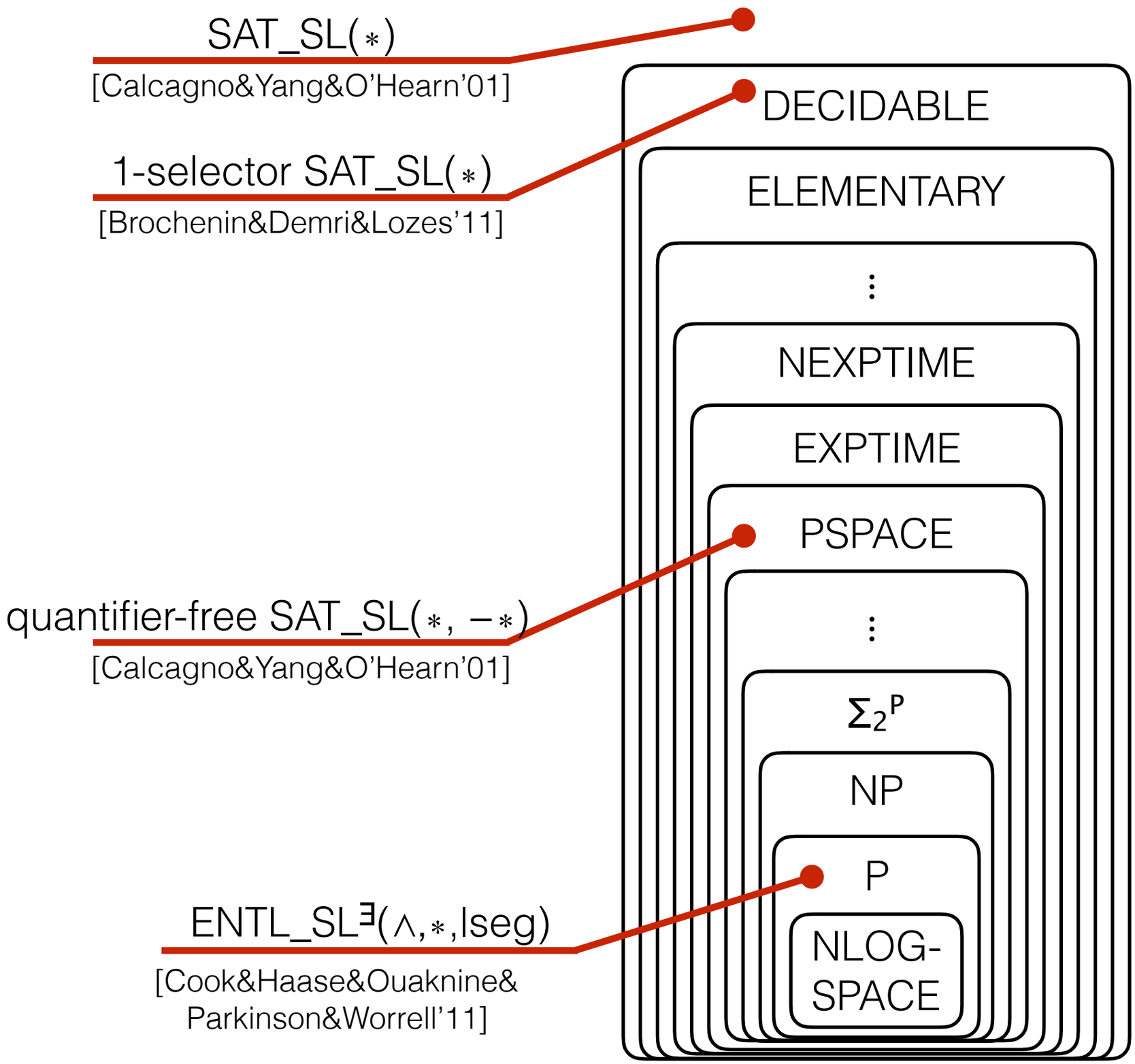
$\phi -* \psi$

Définitions inductives pour décrire des structures de données récursives

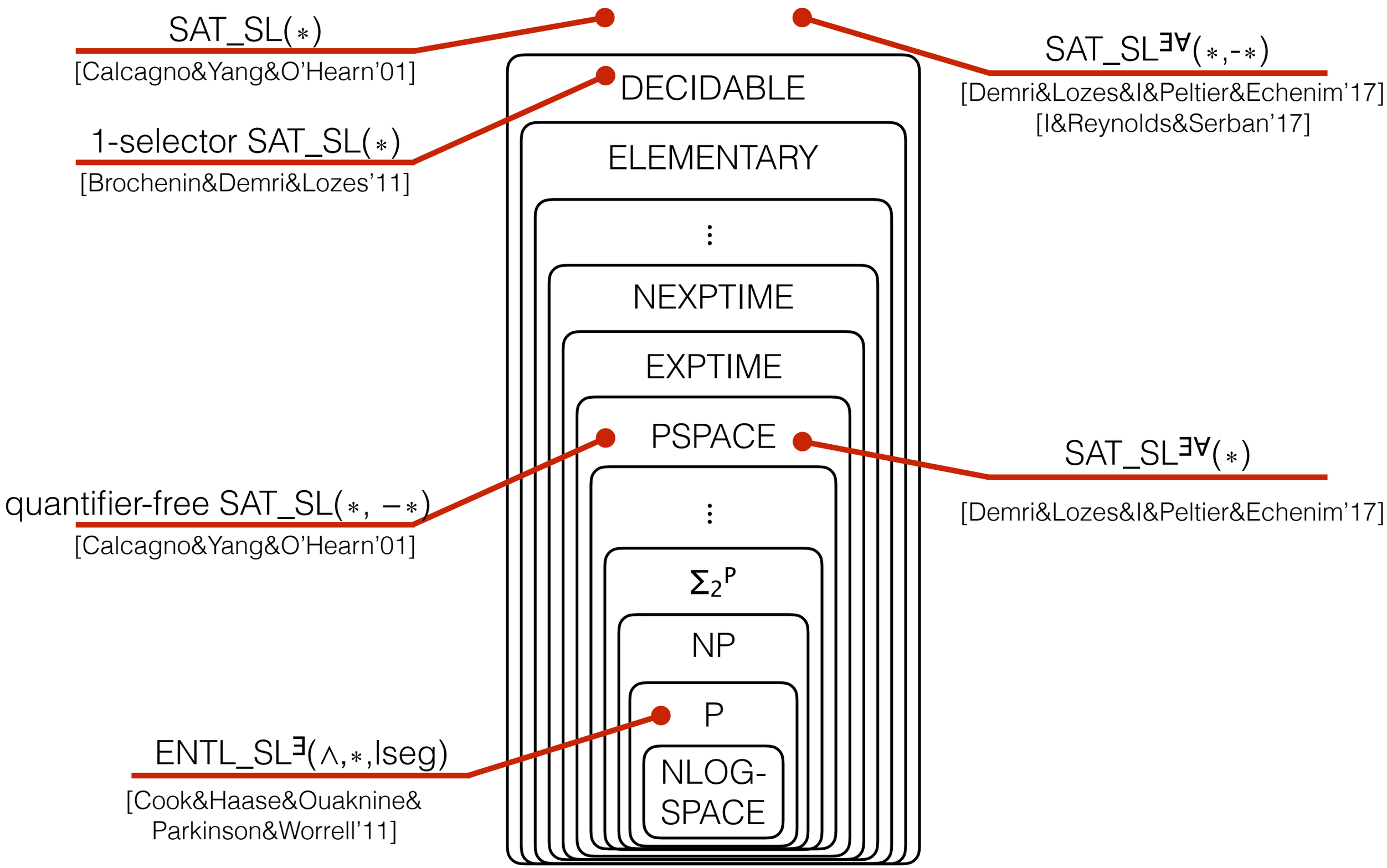
$$\mathsf{lseg}(hd,tl) \equiv emp \wedge hd=tl \vee \exists x \,.\, hd \rightarrow x * \mathsf{lseg}(x,tl)$$



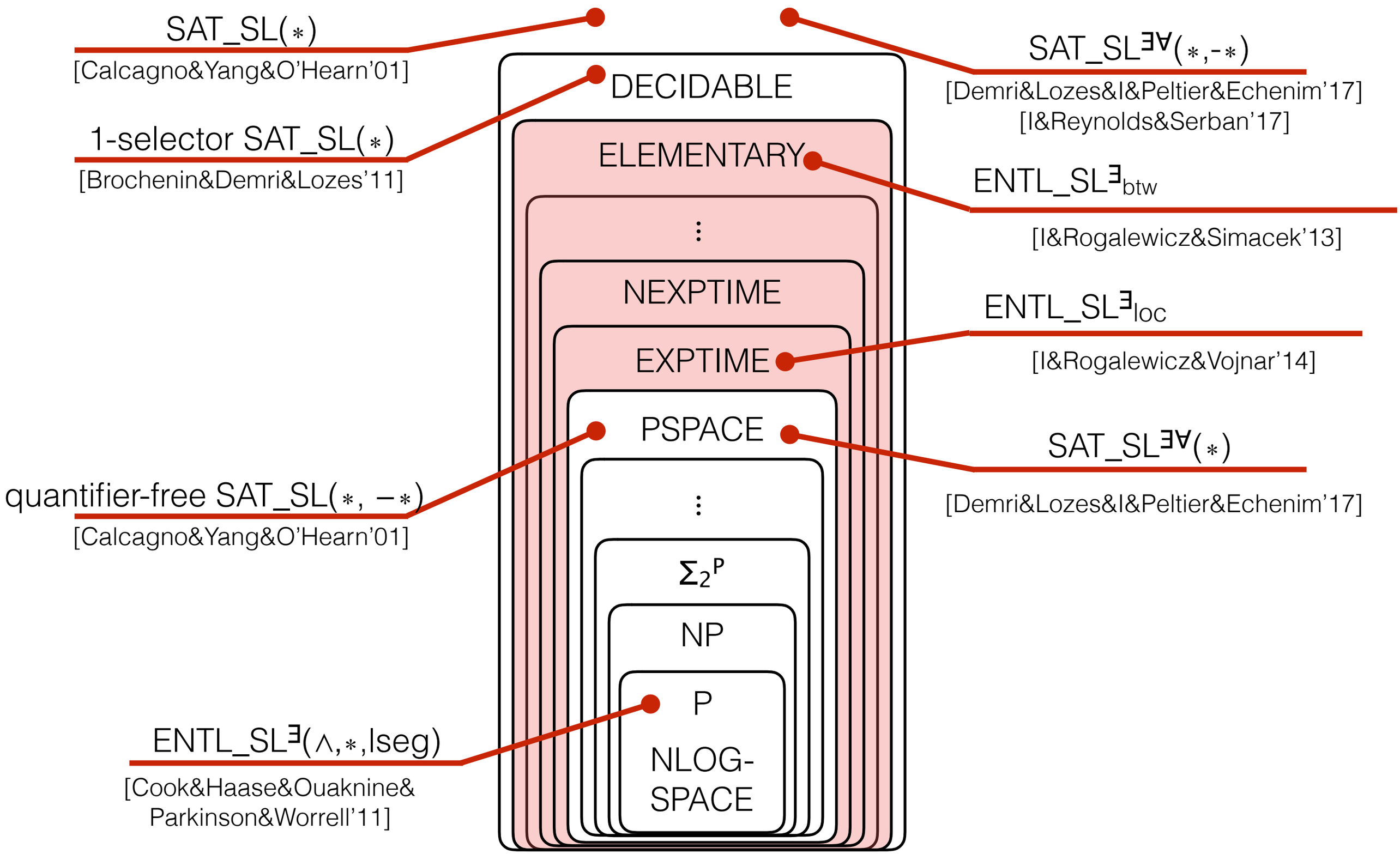
Complexité


SAT_SL(∗)
[Calcagno&Yang&O'Hearn'01]
1-selector SAT_SL(∗)
[Brochenin&Demri&Lozes'11]
quantifier-free SAT_SL(∗, −∗)
[Calcagno&Yang&O'Hearn'01]
ENTL_SL∃(∧,∗,lseg)
[Cook&Haase&Ouaknine&
Parkinson&Worrell'11]
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ2P
NP
P
NLOG-
SPACE

Complexité

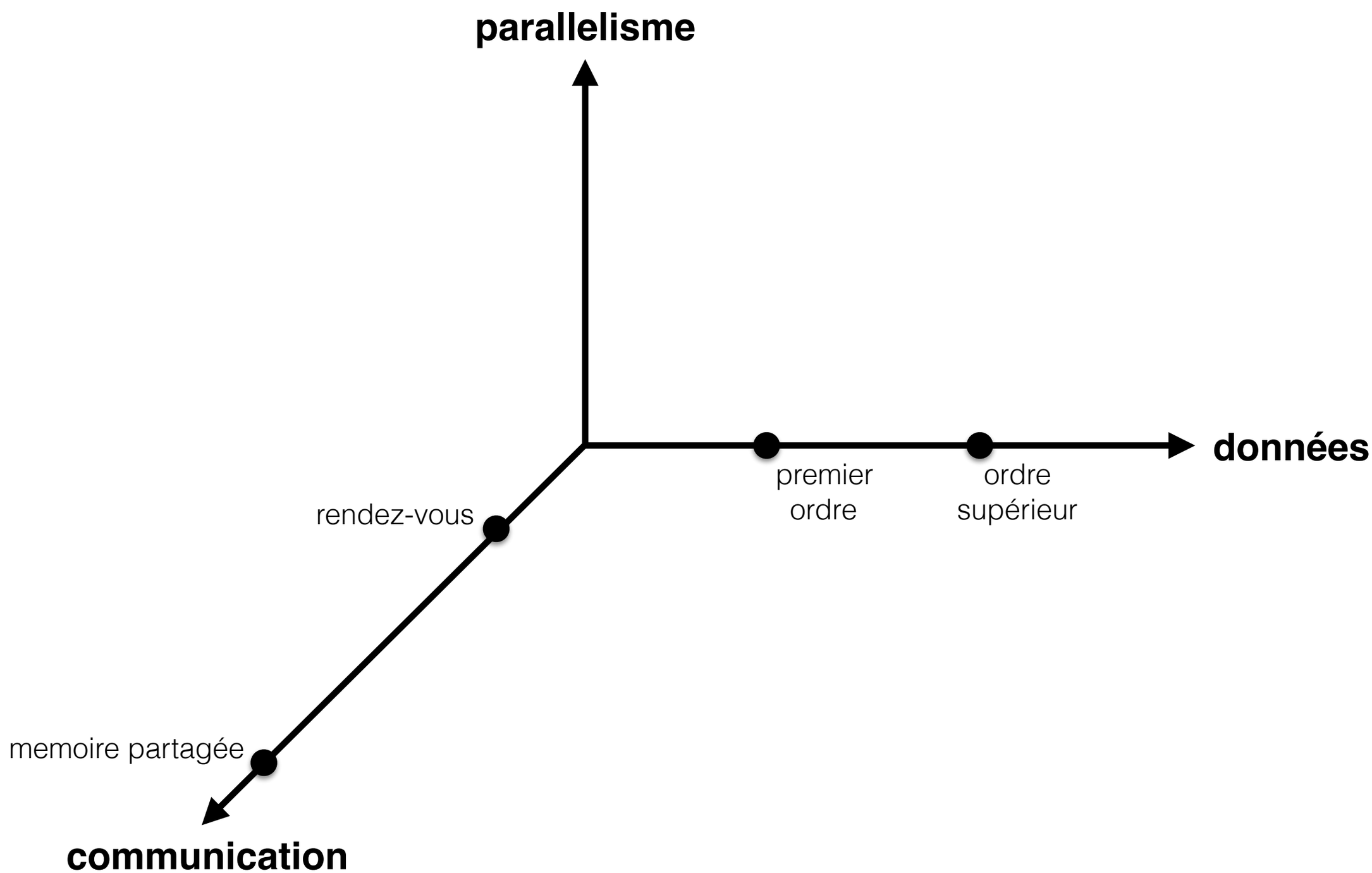

SAT_SL(∗)
[Calcagno&Yang&O'Hearn'01]
1-selector SAT_SL(∗)
[Brochenin&Demri&Lozes'11]
SAT_SL$^{\exists\forall}$(∗,-∗)
[Demri&Lozes&I&Peltier&Echenim'17]
[I&Reynolds&Serban'17]
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ_2^P
NP
P
NLOG-SPACE
SAT_SL$^{\exists\forall}$(∗)
[Demri&Lozes&I&Peltier&Echenim'17]
quantifier-free SAT_SL(∗, −∗)
[Calcagno&Yang&O'Hearn'01]
ENTL_SL$^{\exists}$(∧,∗,lseg)
[Cook&Haase&Ouaknine&
Parkinson&Worrell'11]

Complexité

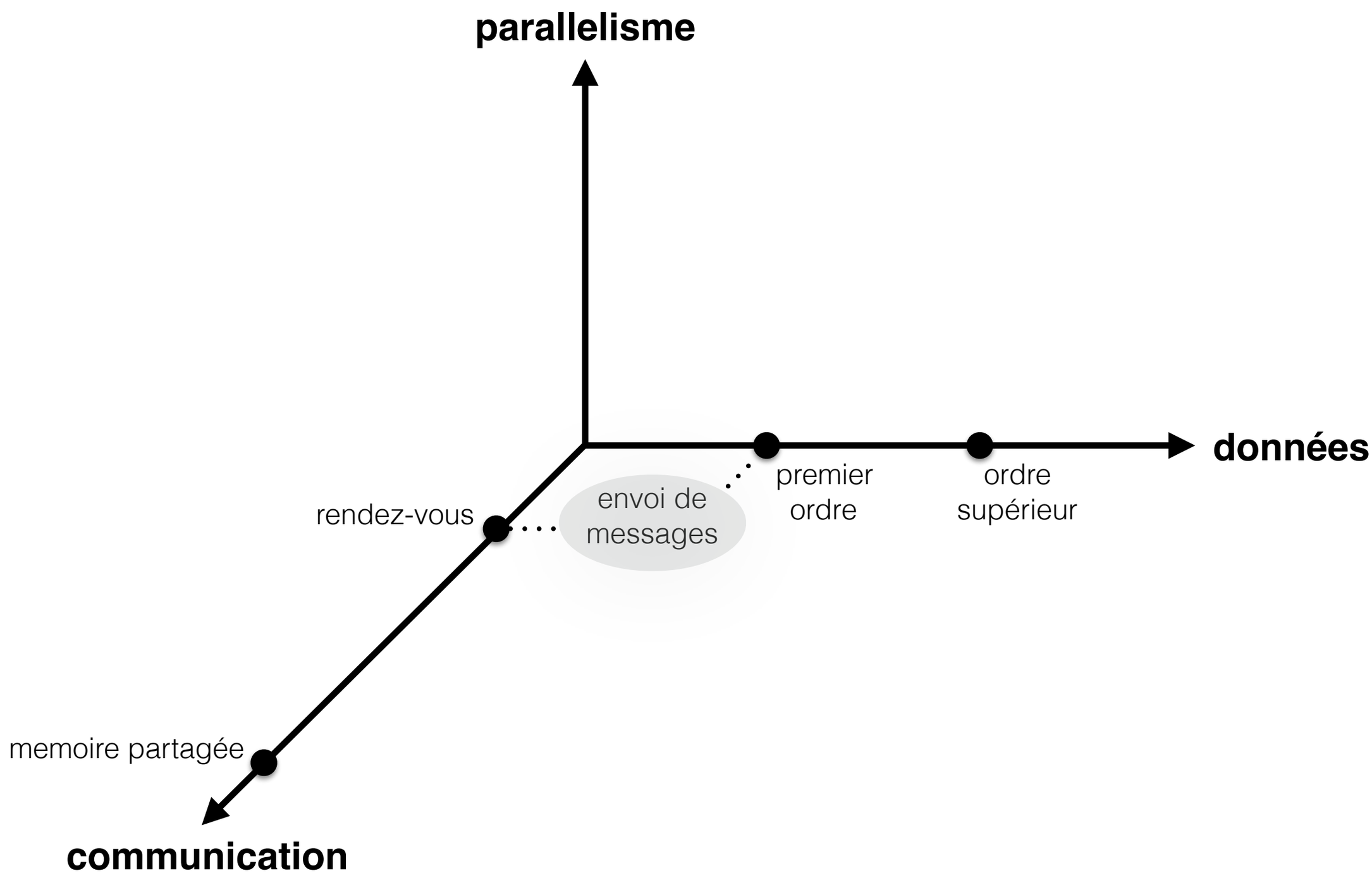

SAT_SL(∗)
[Calcagno&Yang&O'Hearn'01]
1-selector SAT_SL(∗)
[Brochenin&Demri&Lozes'11]
quantifier-free SAT_SL(∗, −∗)
[Calcagno&Yang&O'Hearn'01]
ENTL_SL∃(∧,∗,lseg)
[Cook&Haase&Ouaknine&
Parkinson&Worrell'11]
DECIDABLE
ELEMENTARY
⋮
NEXPTIME
EXPTIME
PSPACE
⋮
Σ2P
NP
P
NLOG-
SPACE
SAT_SL∃∀(∗,-∗)
[Demri&Lozes&I&Peltier&Echenim'17]
[I&Reynolds&Serban'17]
ENTL_SL∃btw
[I&Rogalewicz&Simacek'13]
ENTL_SL∃loc
[I&Rogalewicz&Vojnar'14]
SAT_SL∃∀(∗)
[Demri&Lozes&I&Peltier&Echenim'17]

Travaux en cours et futurs

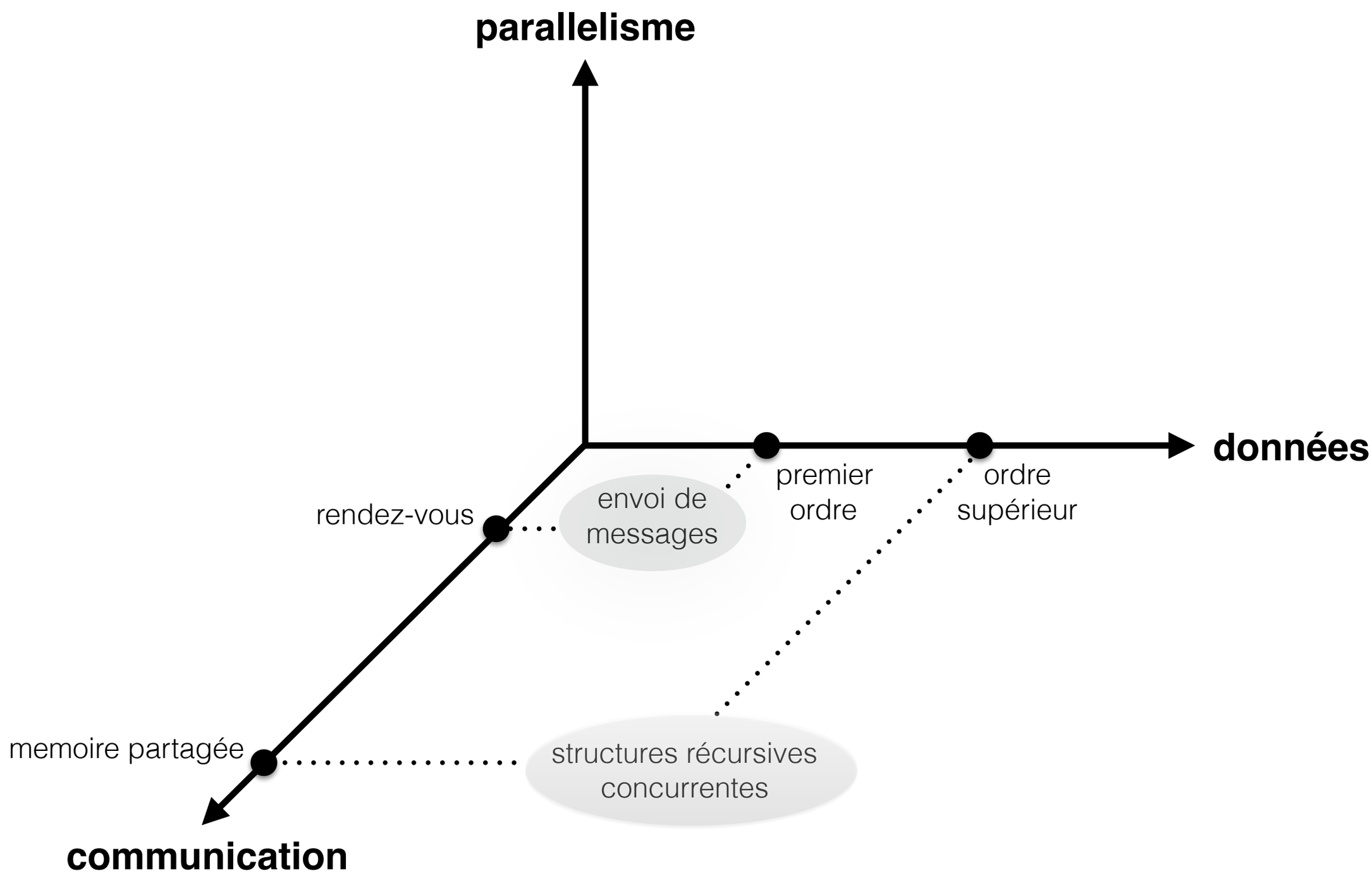
Parallelisme et communication



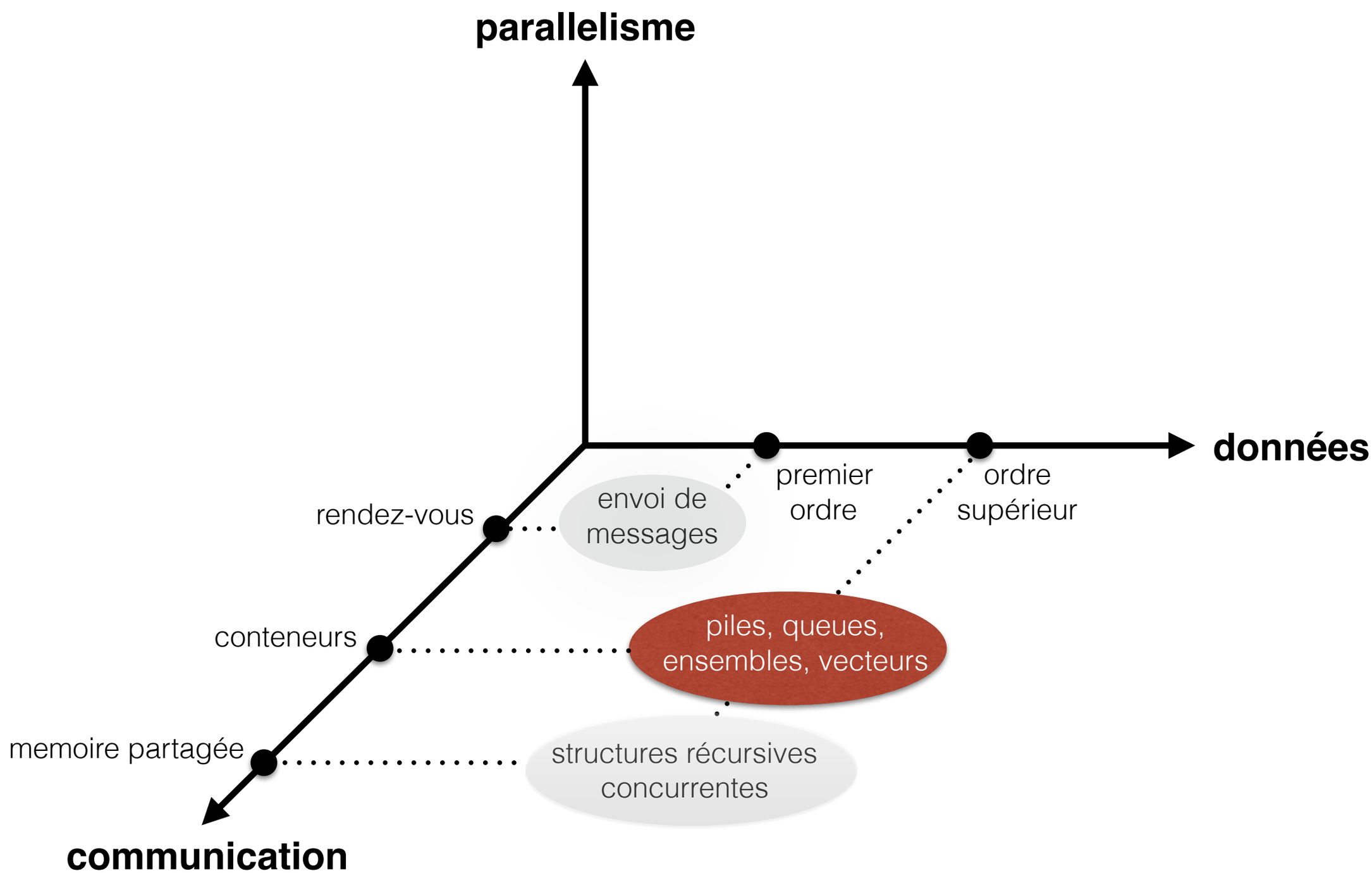
Parallelisme et communication



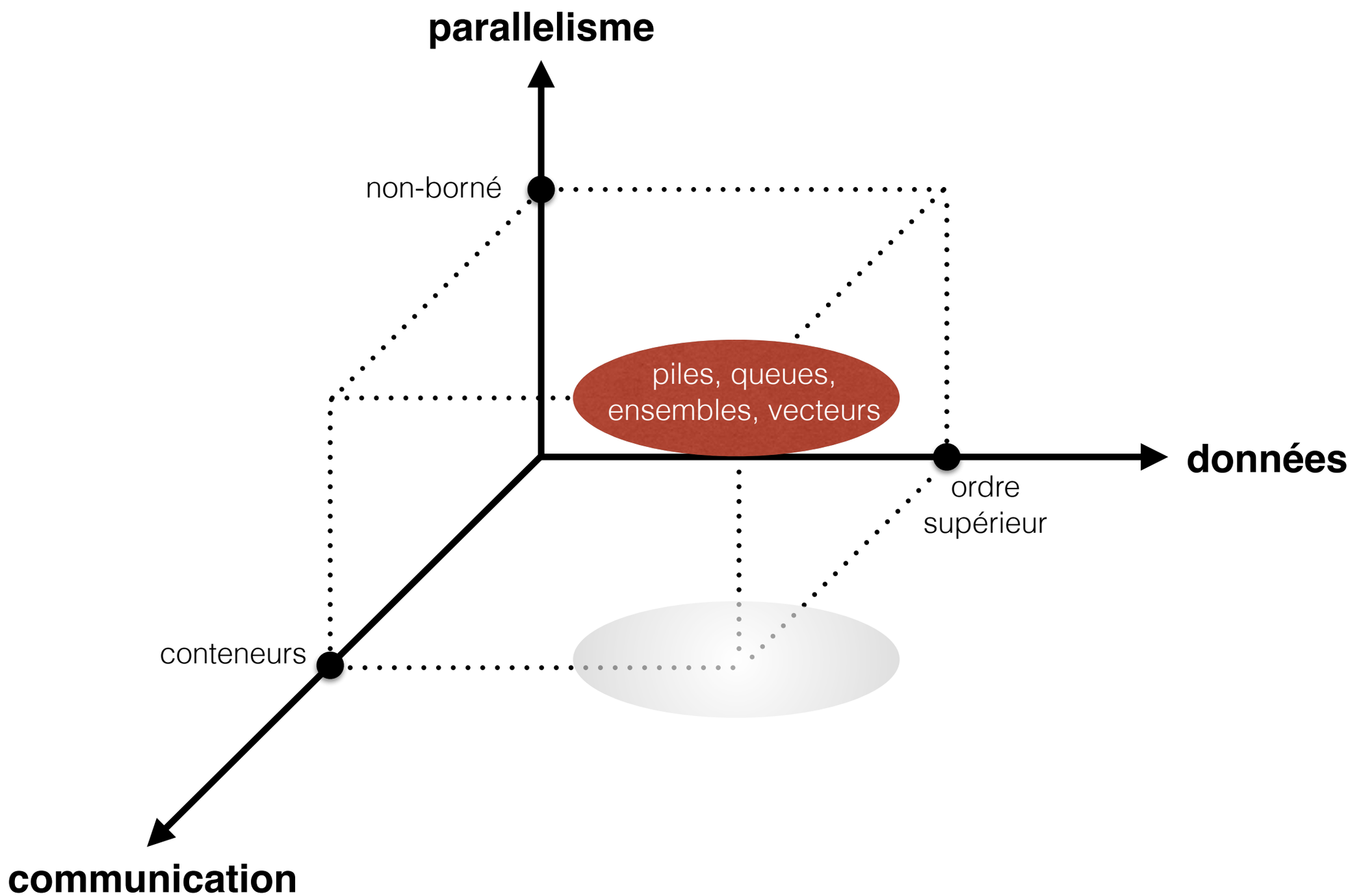
Parallelisme et communication



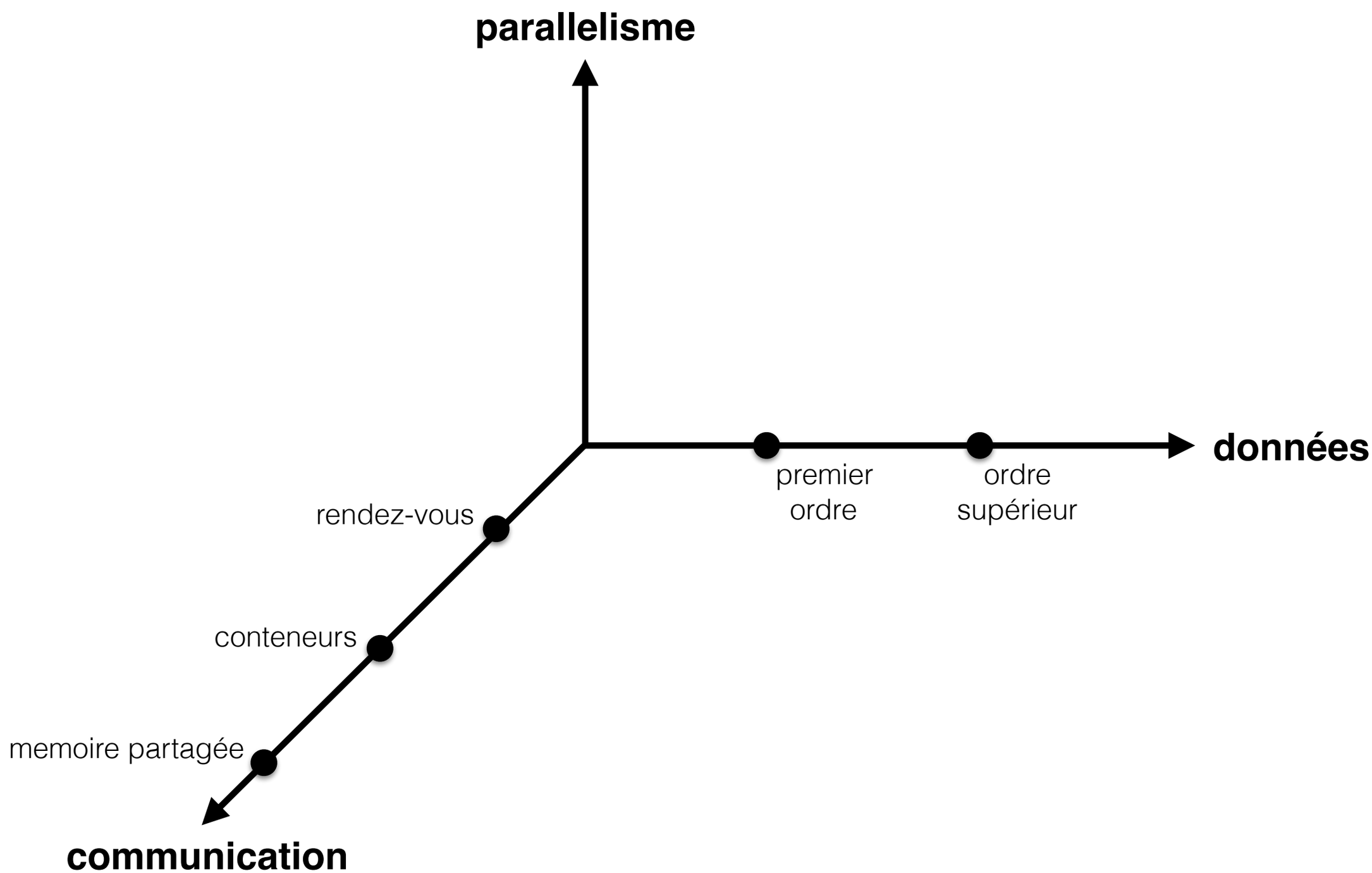
Parallelisme et communication



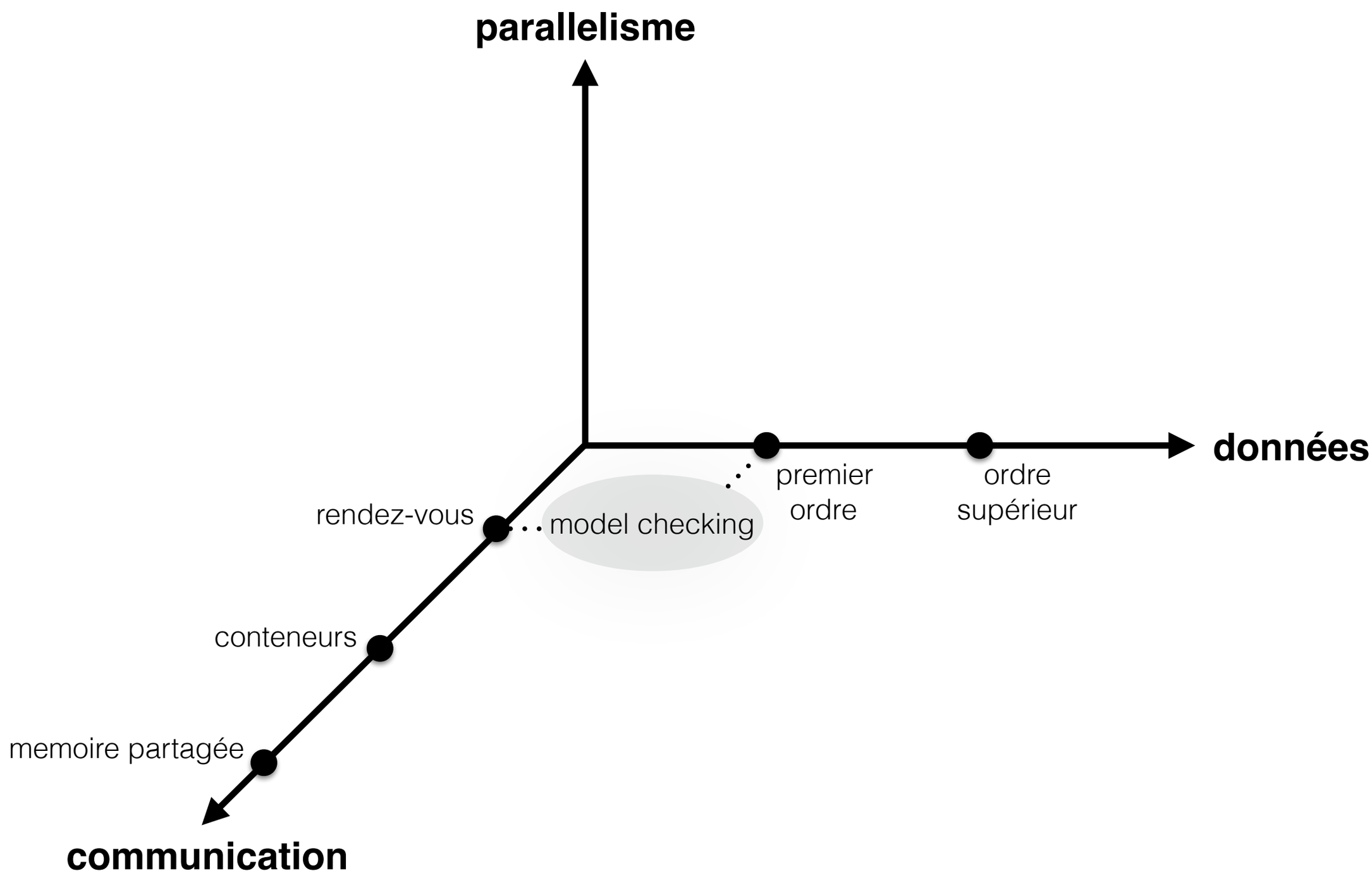
Parallelisme et communication



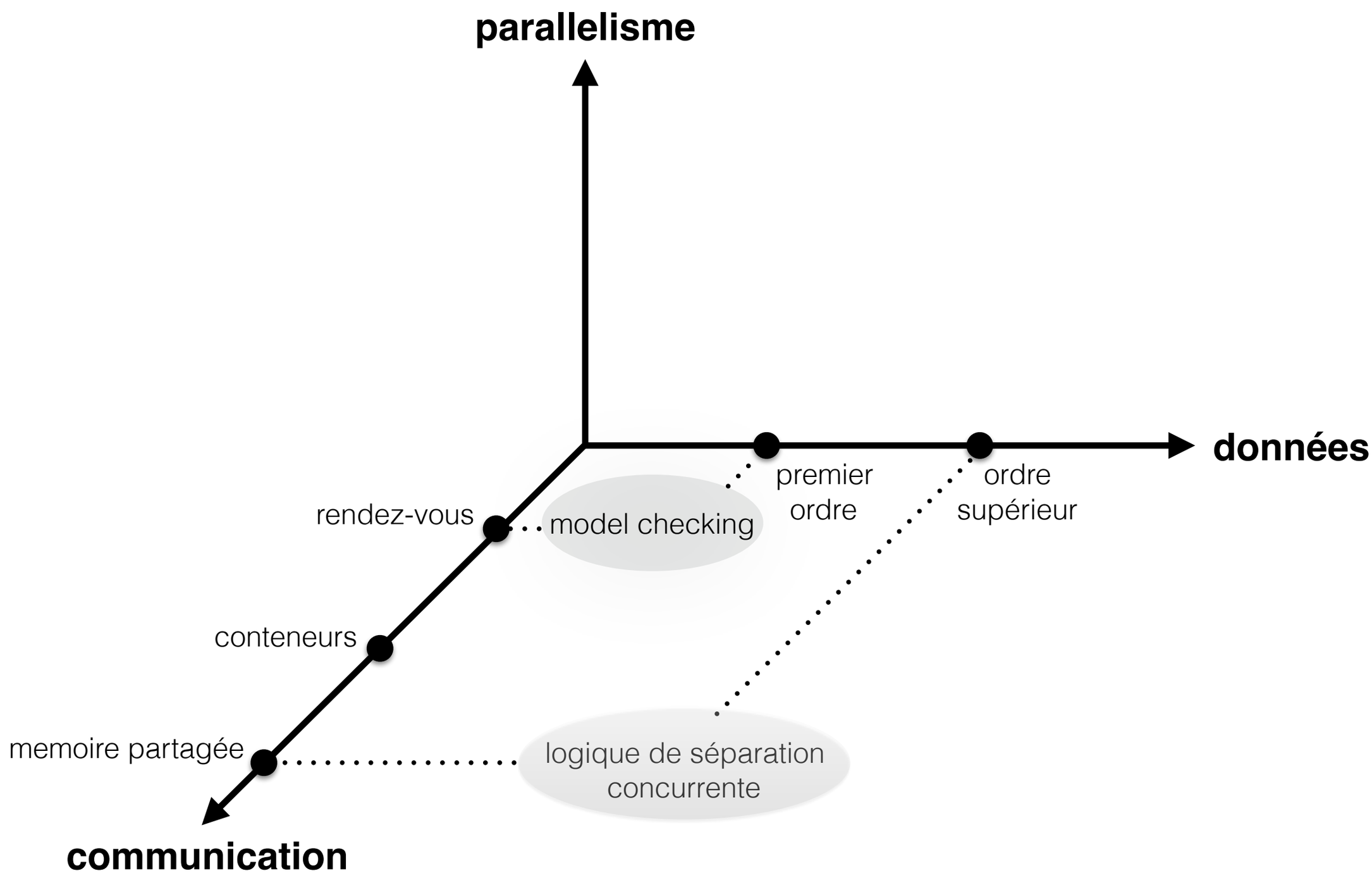
Vérification de programmes concurrents



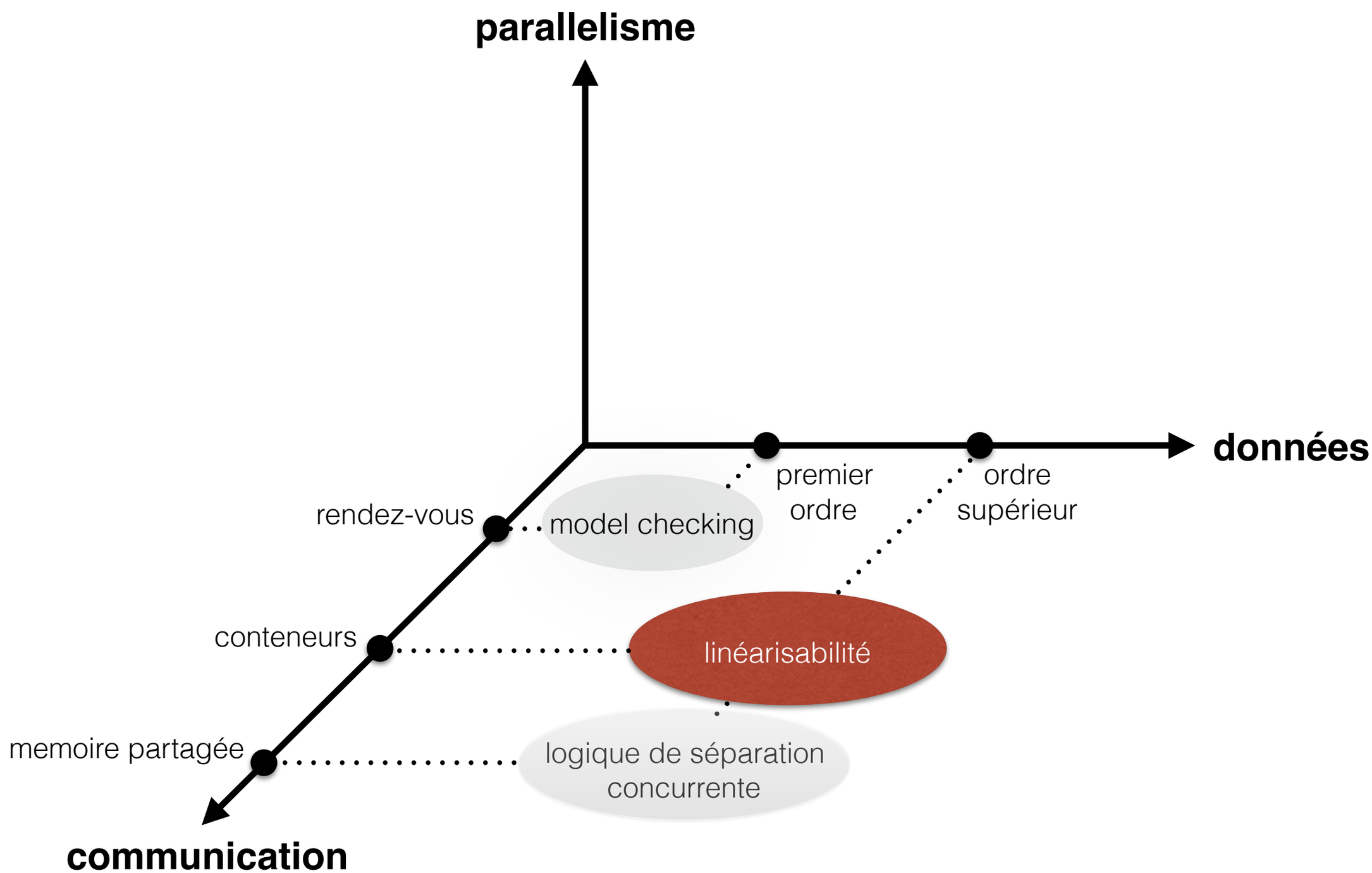
Vérification de programmes concurrents



Vérification de programmes concurrents



Vérification de programmes concurrents



Linéarisabilité

Implementation $op_1;op_2;op_3;op_4;op_5;op_6;op_7;op_8;op_9;op_{10}$ ...

Spécification

Linéarisabilité

Implementation $op_1;op_2;op_3;op_4;op_5;op_6;op_7;op_8;op_9;op_{10}$ ...

Spécification push(1);push(2);pop(2);pop(1); … ✅

Linéarisabilité

Implementation

$op_1;op_2;op_3;op_4;op_5;op_6;op_7;op_8;op_9;op_{10}$ …

Spécification

push(1);push(2);pop(2);pop(1); …

Linéarisabilité

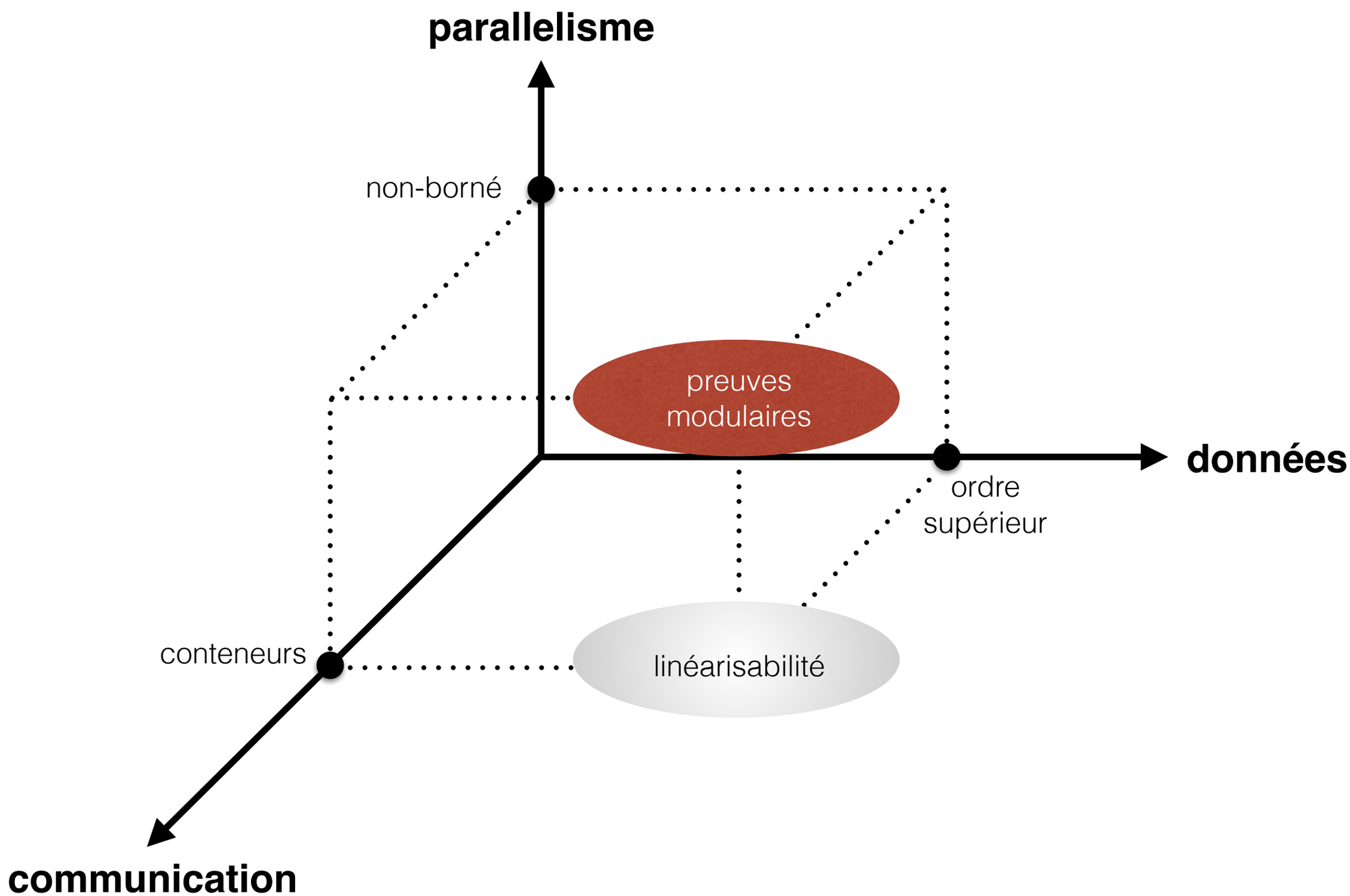
Implementation $op_1;op_2;op_3;op_4;op_5;op_6;op_7;op_8;op_9;op_{10}$ ...

Spécification push(1);push(2);pop(2);pop(1); …

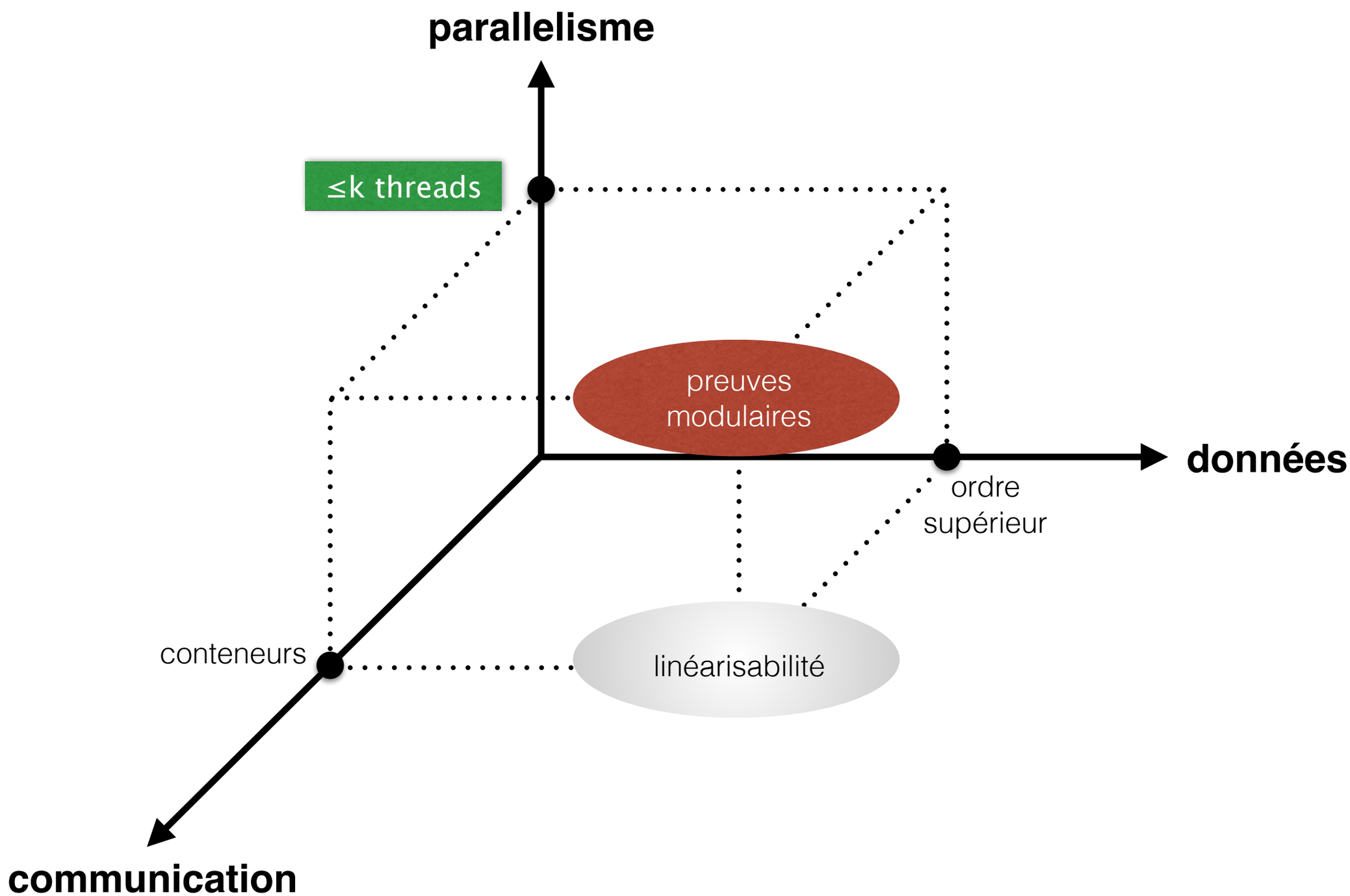
Défis relevés:

- identification de points de linéarisation
- synthèse de relations de raffinement entre les traces d'implementation et spécification
- implementations utilisant des structures recursives chaînées
- spécifications axiomatiques fournies par l'utilisateur
- conteneurs paramètres par de theories de données

Preuves modulaires de programmes concurrents



Preuves modulaires de programmes concurrents



Preuves modulaires de programmes concurrents

Hypothèse: toutes les implementations de conteneurs sont linéarisables

Défis relevés:

- preuves de bornes de type cut-off pour des spécifications de données arbitraires
- bornes de complexité combinatoire

Directions de travail:

- logique de séparation avec definitions inductives paramètrées par la théorie de données
- recherche de preuves par descente infinie combinée avec SMT
- automates alternants étendus avec données paramètrées pour l'inférence d'invariants

Preuves modulaires de programmes concurrents

Hypothèse: toutes les implementations de conteneurs sont linéarisables

Défis relevés:

- preuves de bornes de type cut-off pour des spécifications de données arbitraires
- bornes de complexité combinatoire

Directions de travail:

- logique de séparation avec definitions inductives paramètrées par la théorie de données
- recherche de preuves par descente infinie combinée avec SMT
- automates alternants étendus avec données paramètrées pour l'inférence d'invariants

Merci!