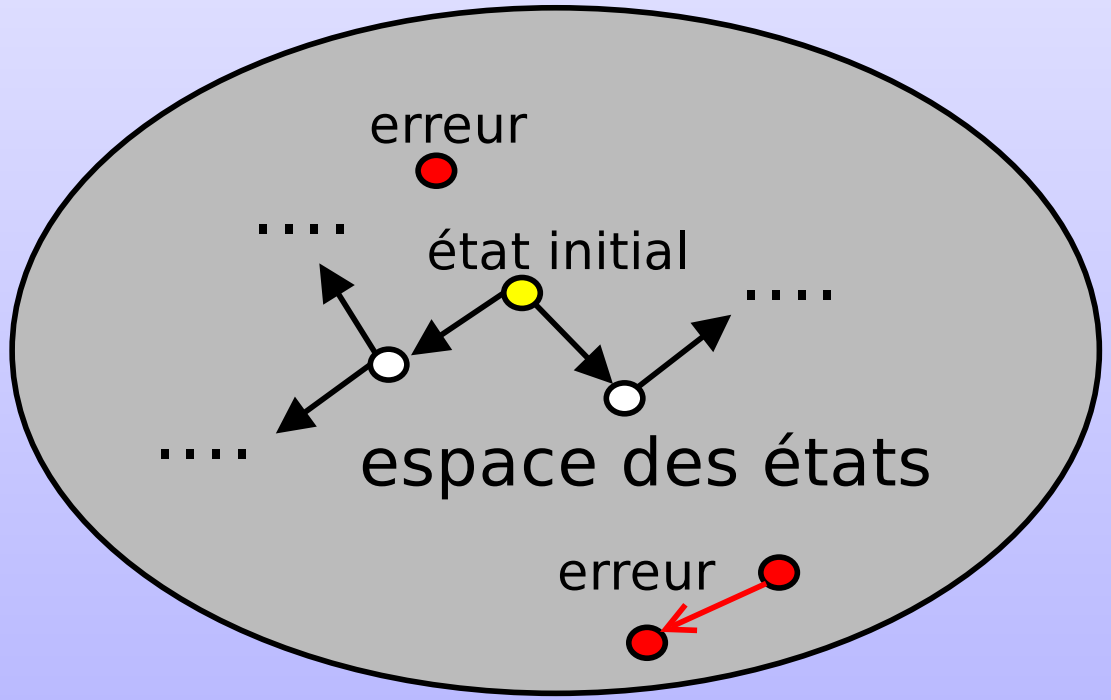
Algorithmes BSP pour la vérification LTL et CTL* de Protocoles de Sécurité

Michael Guedj
sous la direction de
Frédéric Gava, Franck Pommereau et Gaétan Hains
11 octobre 2012

Laboratoire d'**A**lgorithmique, **C**omplexité et **L**ogique (LACL)
Université Paris-Est Créteil

Problématique

- Modélisation
- Vérification
  $\Rightarrow$ **Model-Checking**
- Problème : explosion combinatoire



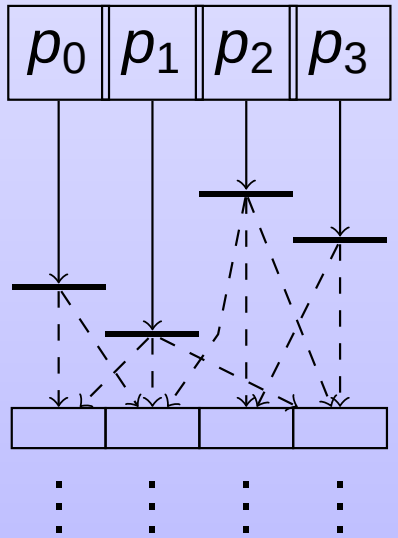
Sujet de thèse

Une approche pour attaquer la combinatoire
$\Rightarrow$ Parallelisation (Calcul Haute Performance)

- Problème en général non parallelisable *a priori*
  - Tarjan, NDFS non parallelisable
- Littérature $\Rightarrow$ amortissement
- Application sur des modèles particuliers
  $\Rightarrow$ Protocoles de sécurité

Modèle BSP

- Machine parallèle
  - Ensemble homogène de CPU + RAM
  - Interconnectés
- Modèle de calcul



calculs locaux

communications

barrière de synchronisation

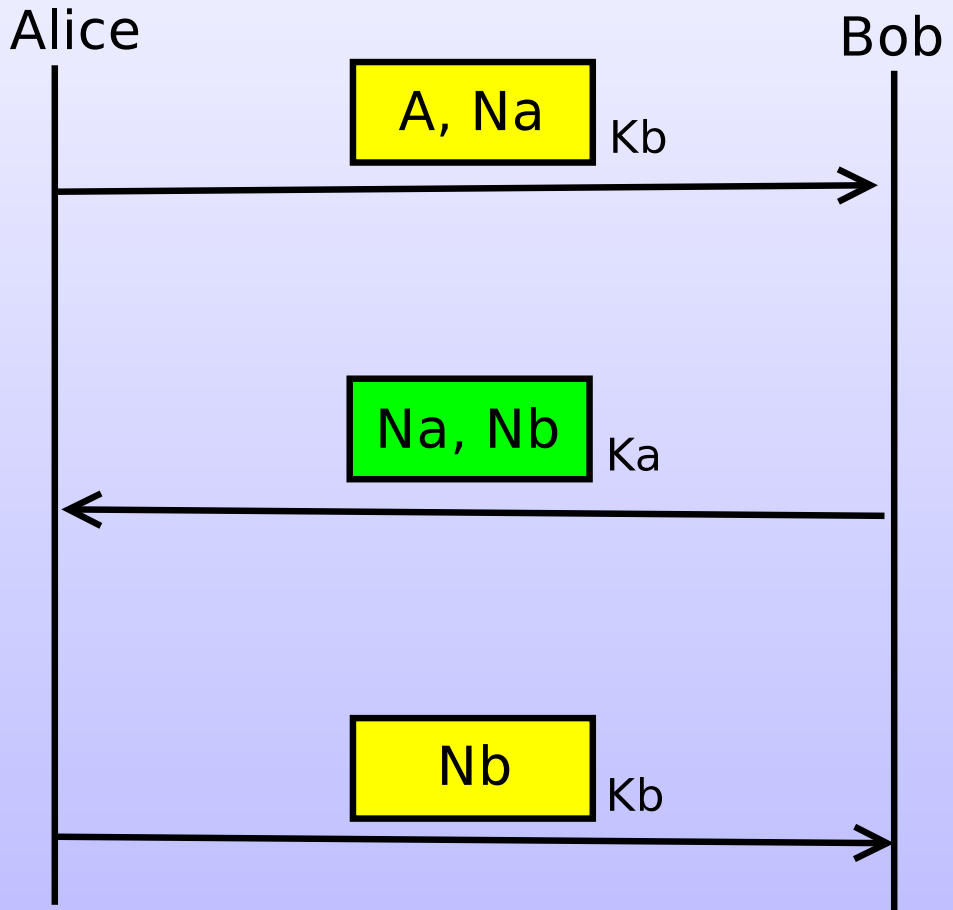
prochaine super-étape

Plan

1. Protocoles de sécurité
2. Calcul de l'espace des états
3. Model checking
   1. LTL
   2. CTL*

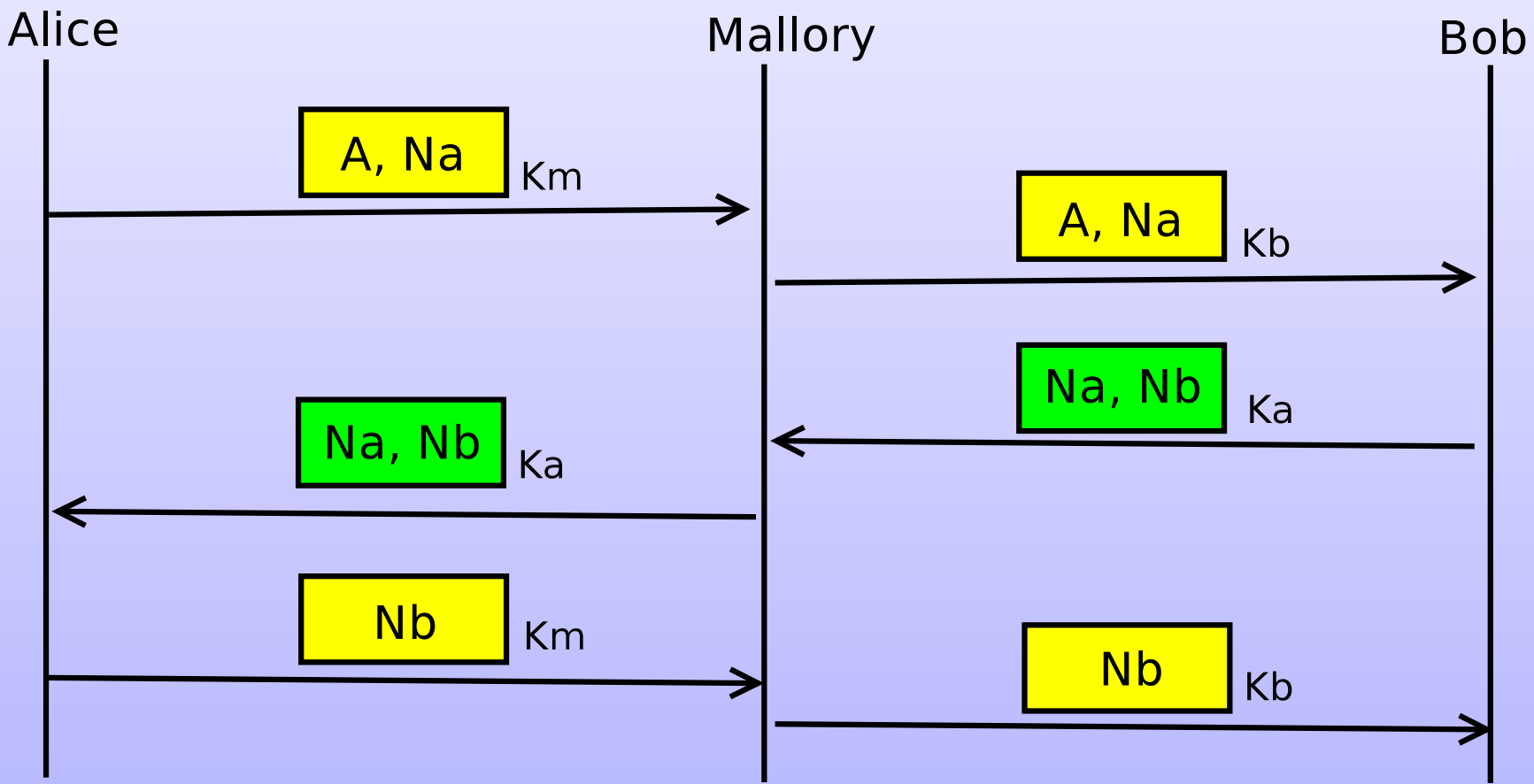
Protocoles de sécurité

Exemple : le protocole de Needham-Schroeder

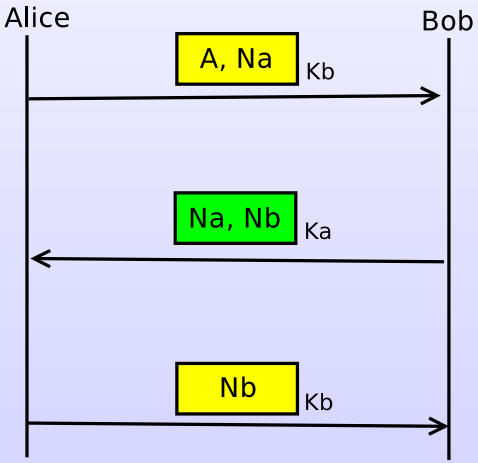


Attaque du protocole de Needham-Schroeder

"*Security protocols are three line programs that people still manage to get wrong*" (Roger Needham)



Modélisation / Vérification



Manuelle

Modèle ABCD

Automatique

?

Propriété de sécurité
(Logique temporelle)

$\models$

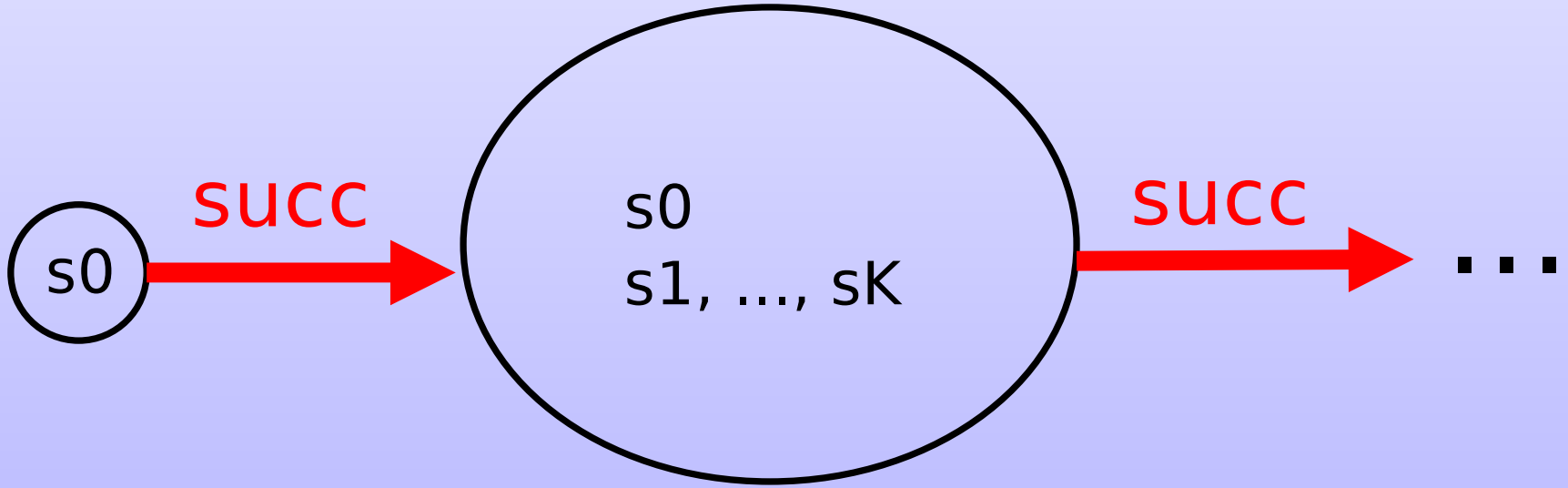
Réseau de Petri

Model checker

Espace des états

Algorithme séquentiel

1. Etat initial s_0
2. Fonction successeur *succ* : $s \rightarrow \{s_1, ..., s_k\}$
3. Calcul d'un point fixe

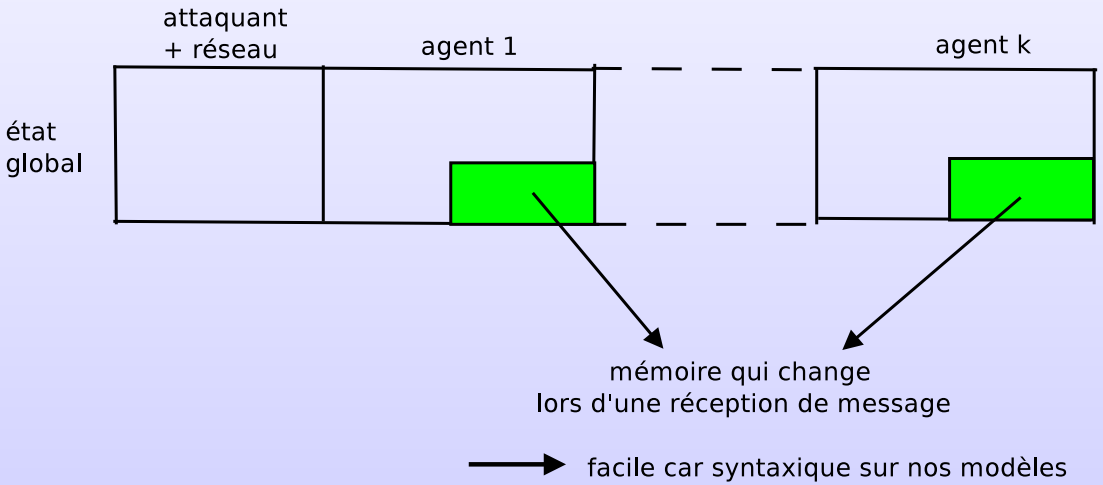


Parallélisation naïve

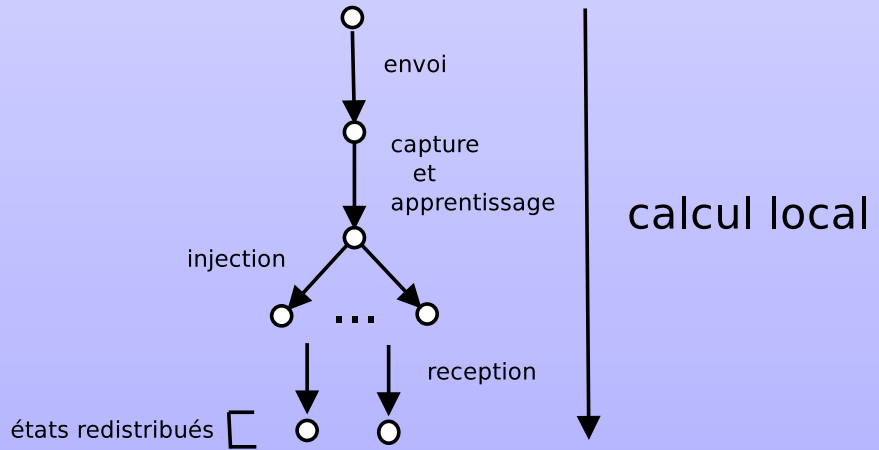
- Fonction de partition **cpu**
  $\Rightarrow$ hachage sur l'état modulo le nombre de processeurs
- Le processeur i calcule $succ(s)$
  $\Leftrightarrow$ **cpu(**s**)** = i $\rightarrow$ état possédé par un processeur
- Succession de calculs sur les états locaux
- Et d'envois des états possédés par les autres processeurs
  $\Rightarrow$ *cross-transition*

Améliorer la localité

- Exploiter la structure des protocoles

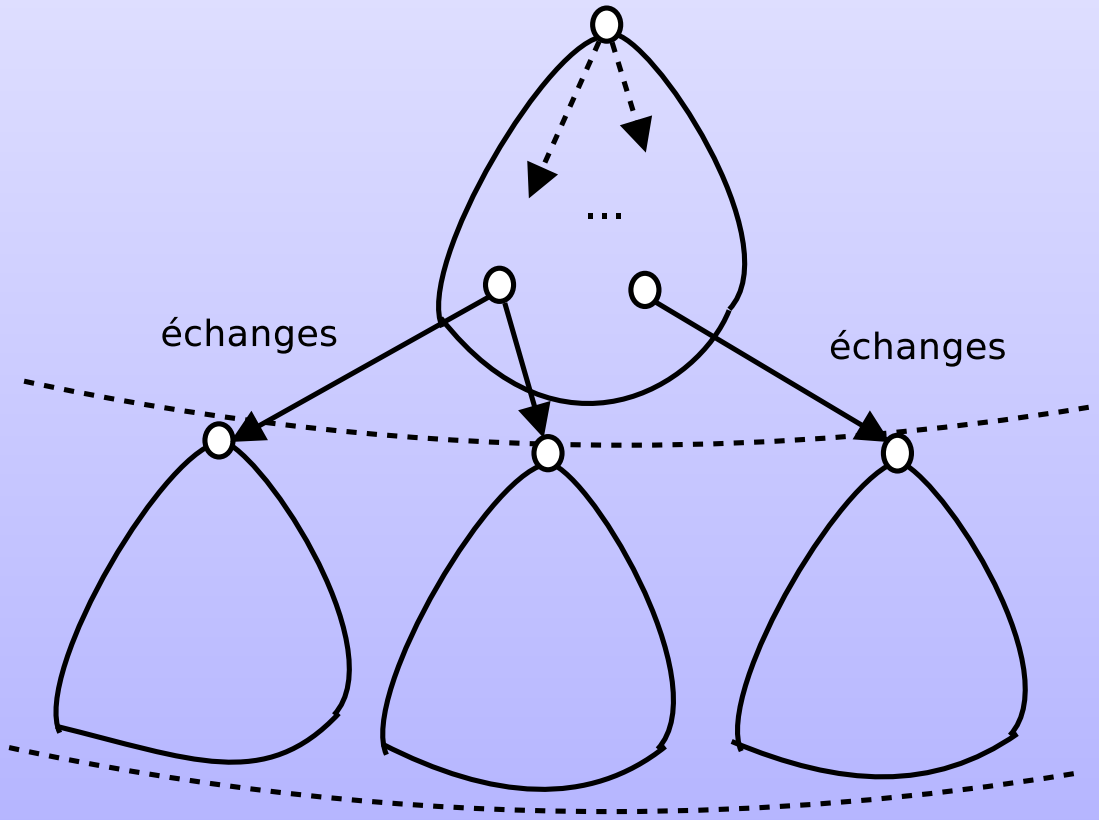


- Fonction **cpu** sur la partie  uniquement



Améliorer la consommation mémoire

- Progression du protocole
  $\Longleftrightarrow$ progression des super-étapes
- Un état calculé à la super-étape k ne sera plus jamais rencontré dans une super-étape $> k$
- Vider la mémoire à chaque super-étape

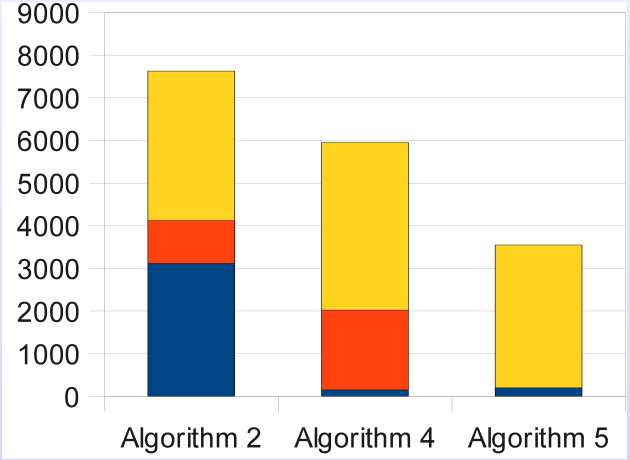


Améliorer l'équilibrage

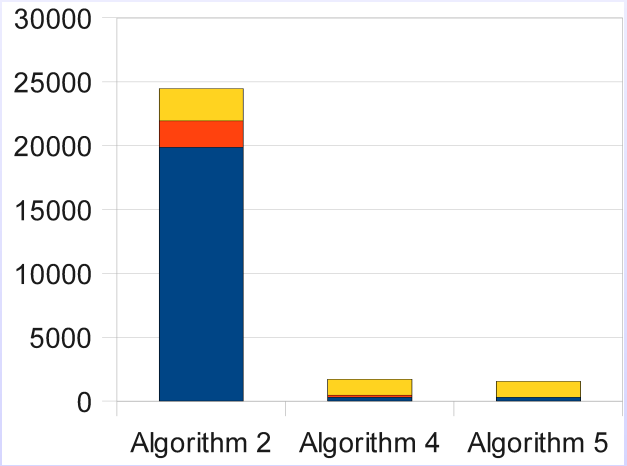
- On ne localise plus les états directement sur les processeurs
  $\Rightarrow$ on calcule un hachage sans modulo
- On calcule un histogramme de répartition des états
  - 1er temps : local
  - 2ème temps : global
- Chaque processeur équilibre ses envois selon cet histogramme

Evaluation pratique des performances

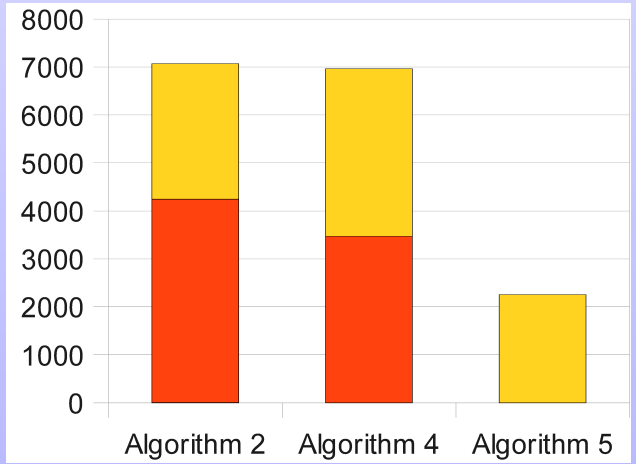
Needham-Schroeder



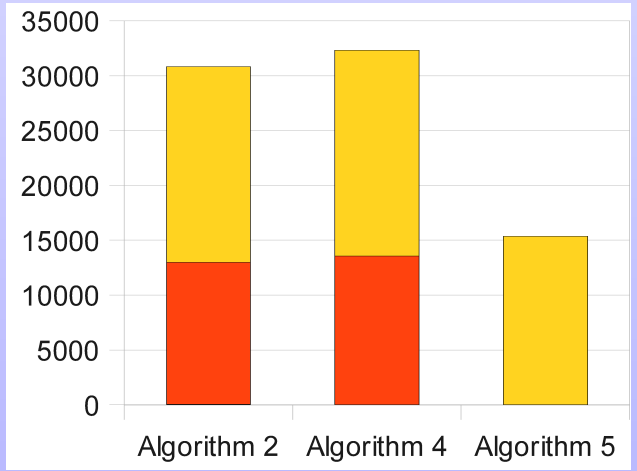
Yahalom



Otway-Rees

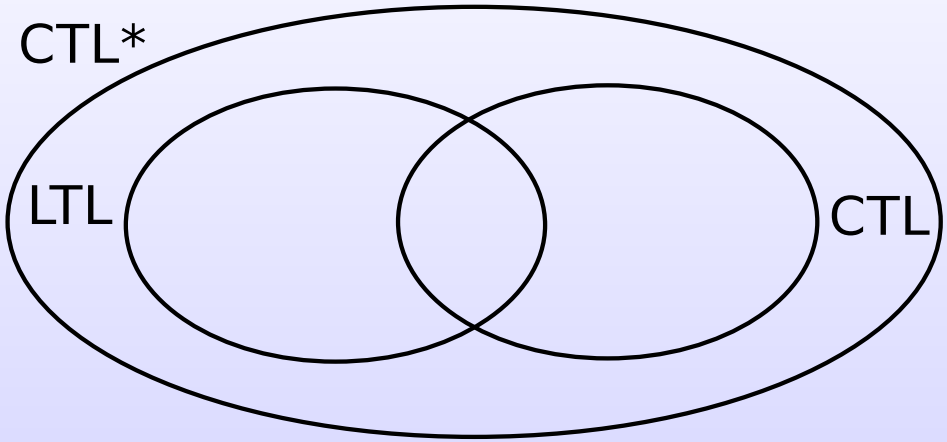


Kao-Chow



Model checking

La logique CTL*

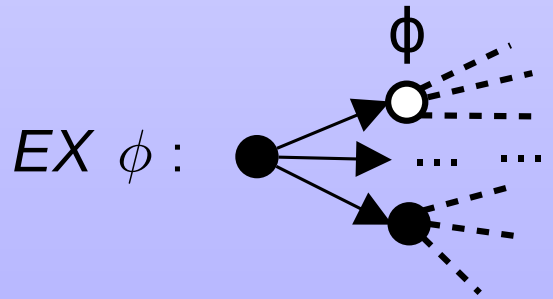


- LTL

$$X\phi : \bullet \rightarrow \bullet^{\phi} \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \cdots$$

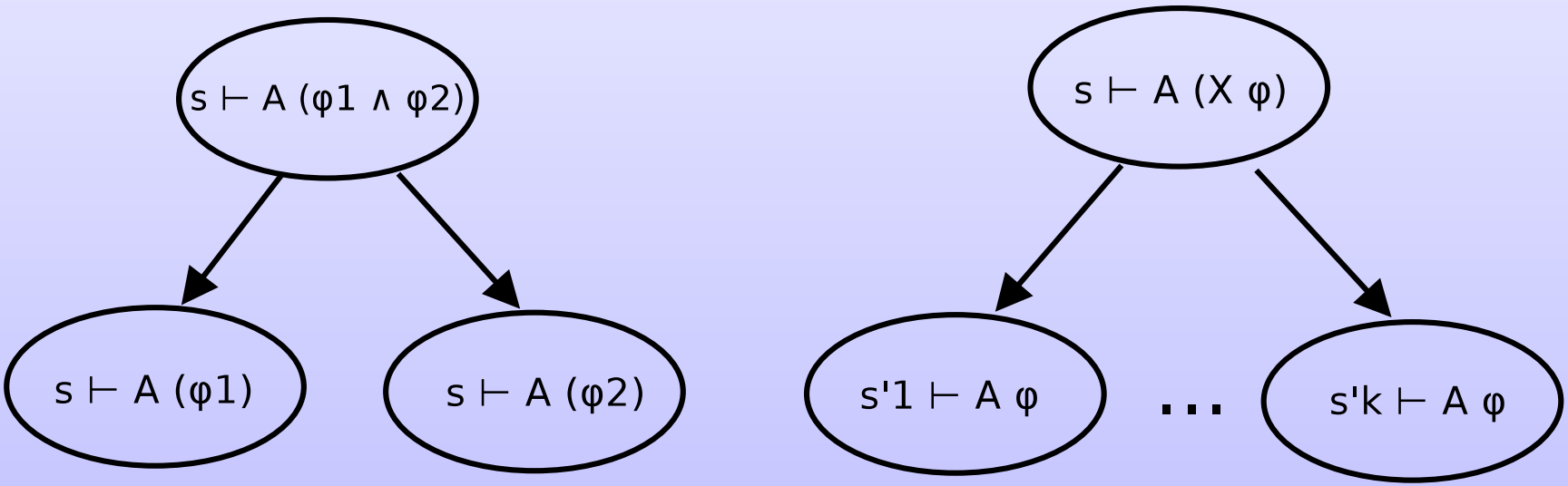
$$\phi_1 U \phi_2 : \bullet^{\phi_1} \rightarrow \bullet^{\phi_1} \rightarrow \bullet^{\phi_1} \rightarrow \bullet^{\phi_2} \rightarrow \bullet \rightarrow \cdots$$

- CTL



Principe du model checking de LTL

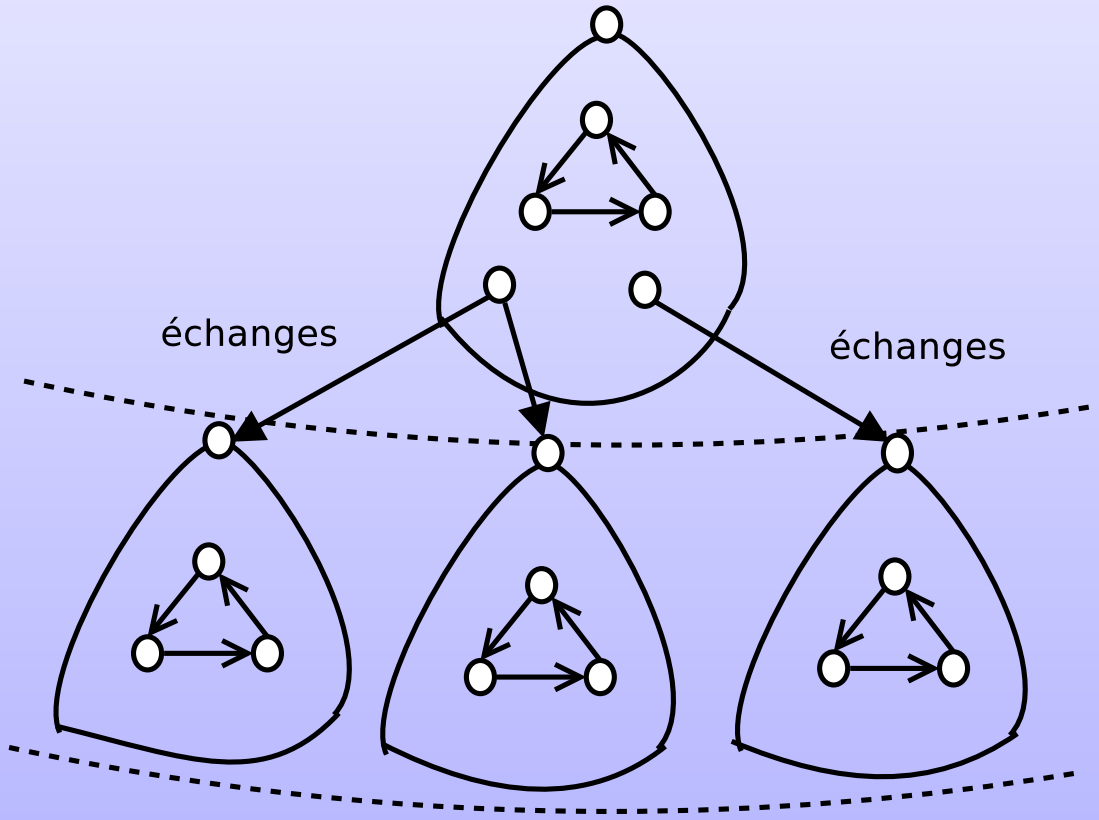
A partir d'un algorithme de [Bhat, Cleaveland, Grumberg 1995]
$\Rightarrow$ construction d'un "proof-graph"



$\Rightarrow$ calcul de CFC (Tarjan)
$\Rightarrow$ règles de validation

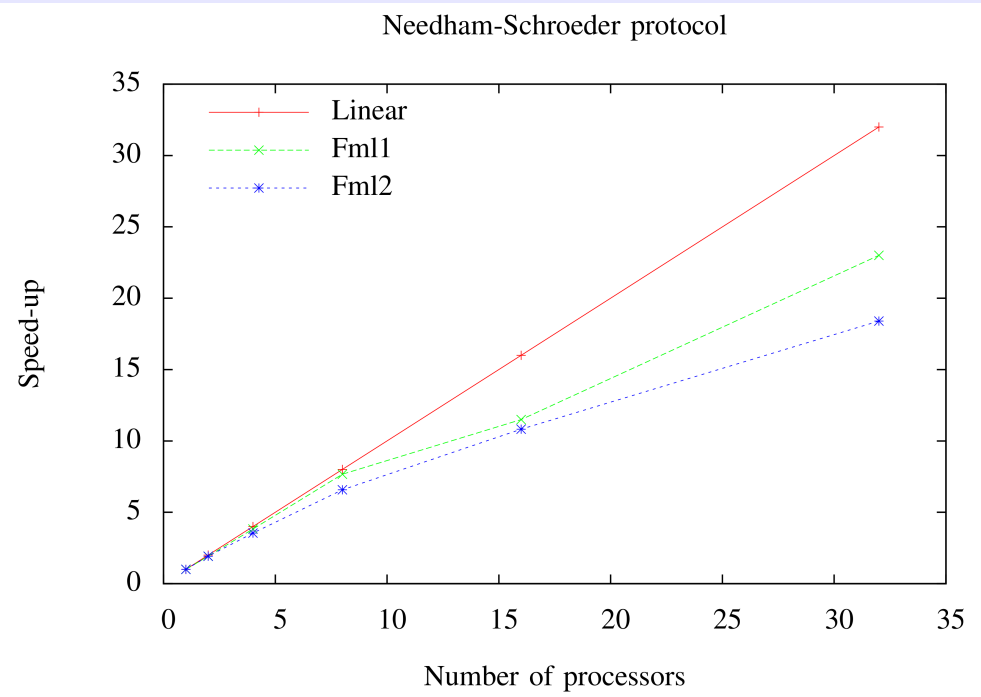
Parallelisation du model checking de LTL

1. Parallelisation de Tarjan
2. Intégration dans l'algorithme précédent
3. Possible car chaque CFC est locale

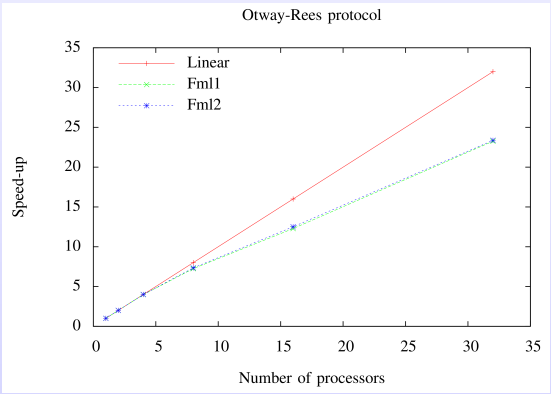


Accélérations sur LTL

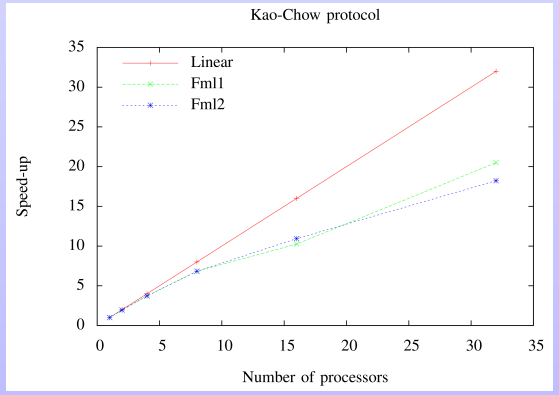
Needham-Schroeder



Otway-Rees



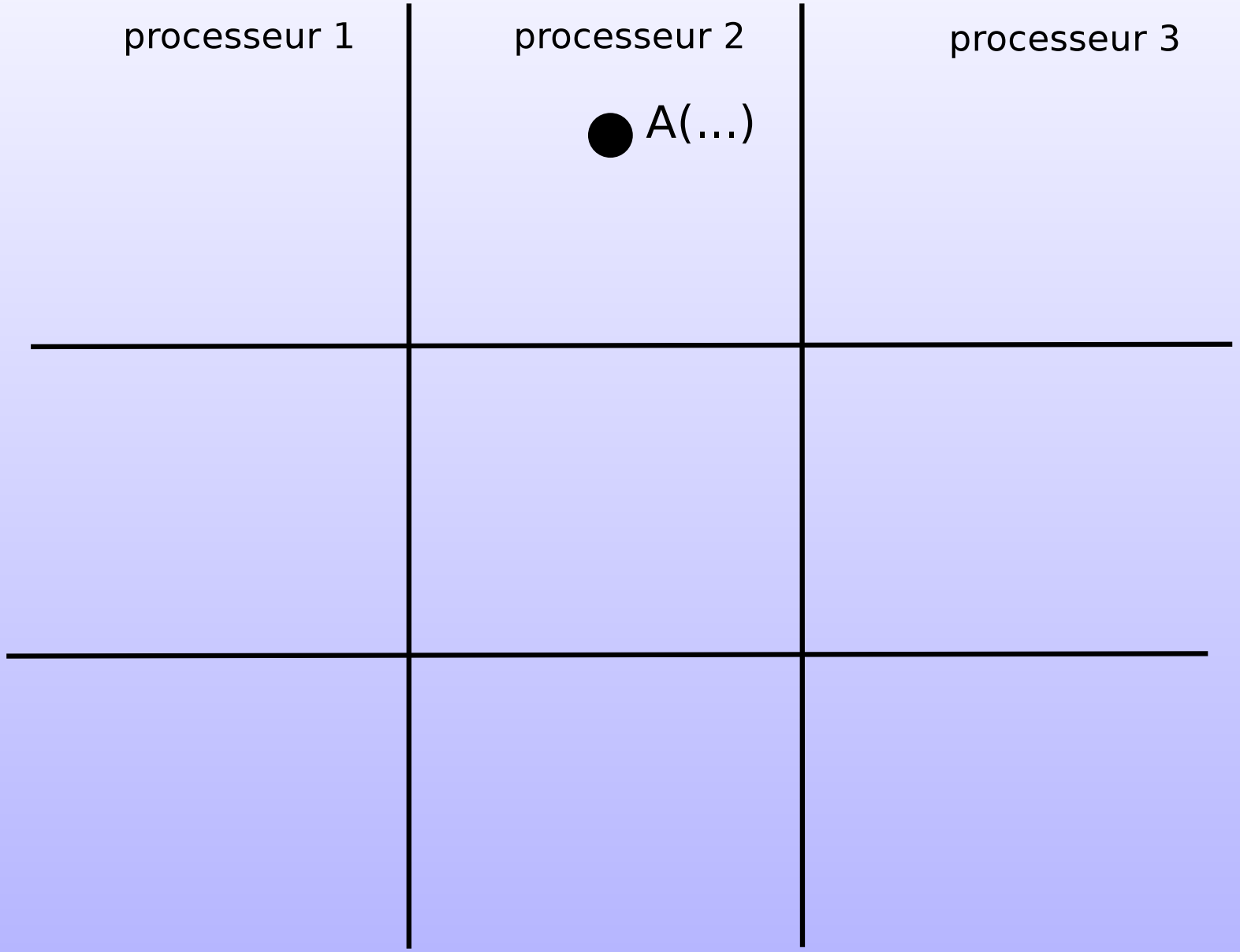
Kao-Chow



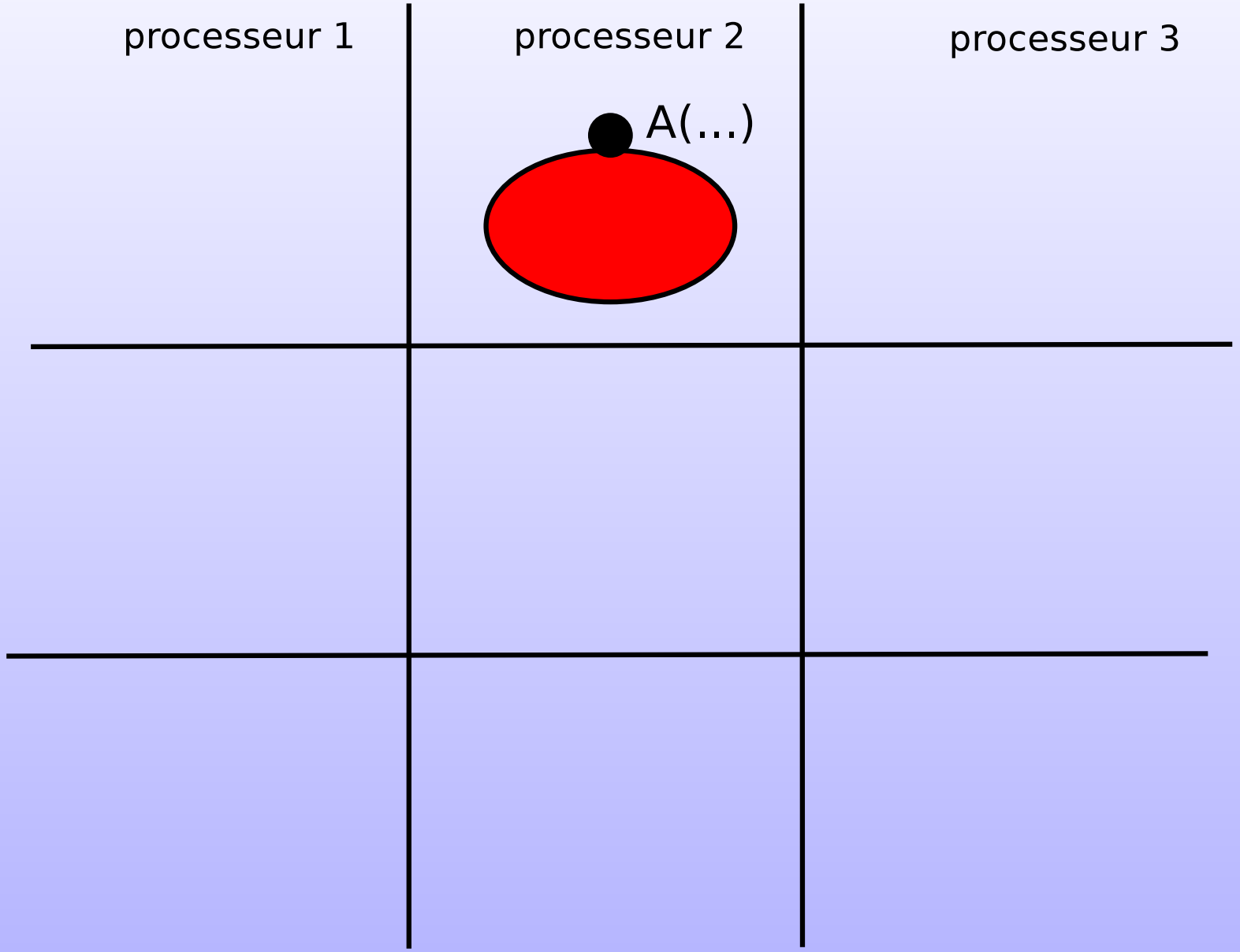
Principe du model checking de CTL*

1. Mêmes auteurs, même article
2. Deux fonctions mutuellement récursives
   1. ModCheckLTL ($\sim$ celle parallélisée précédemment)
   2. ModCheckCTL*
3. Conditions de validation

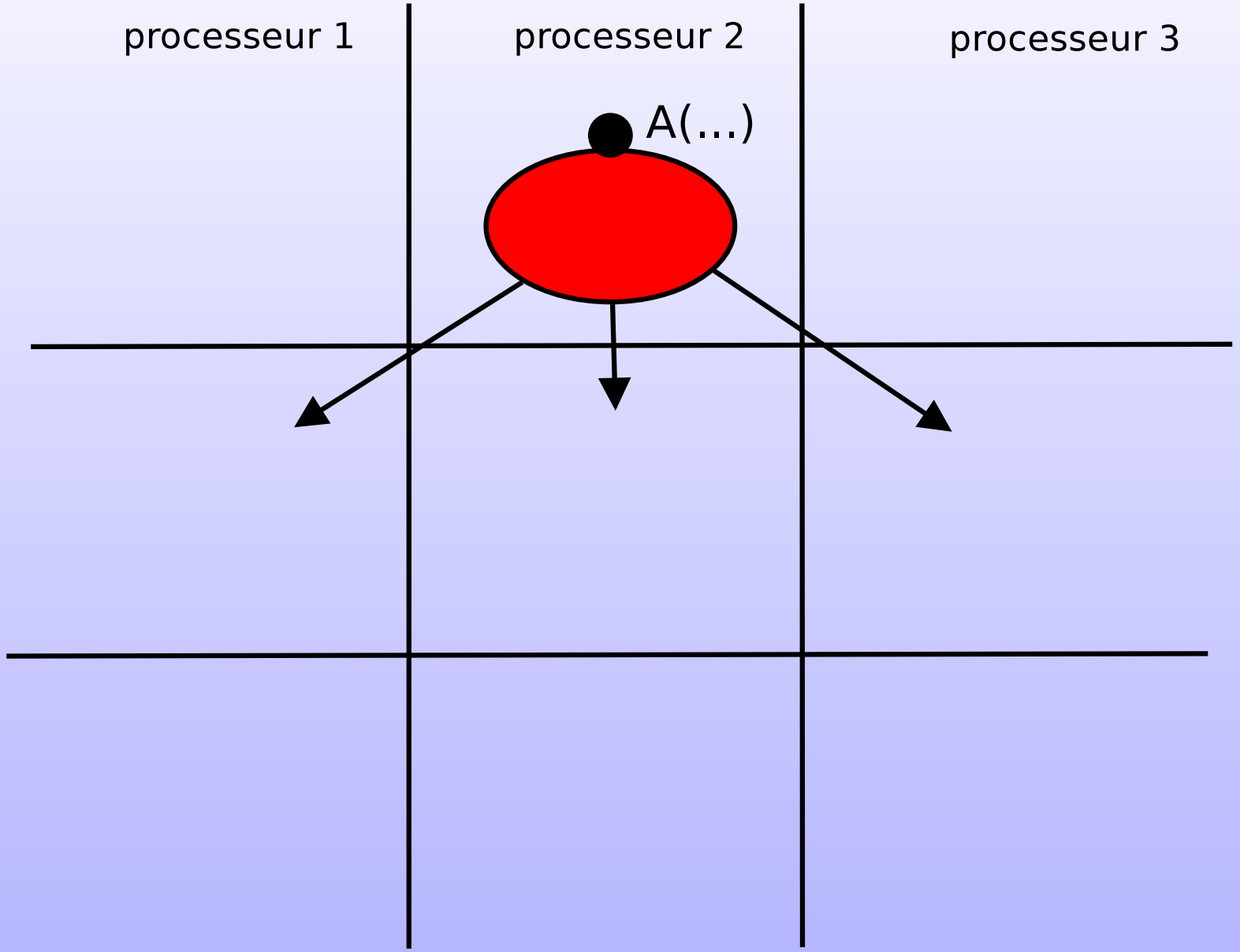
Parallelisation du model checking de CTL*



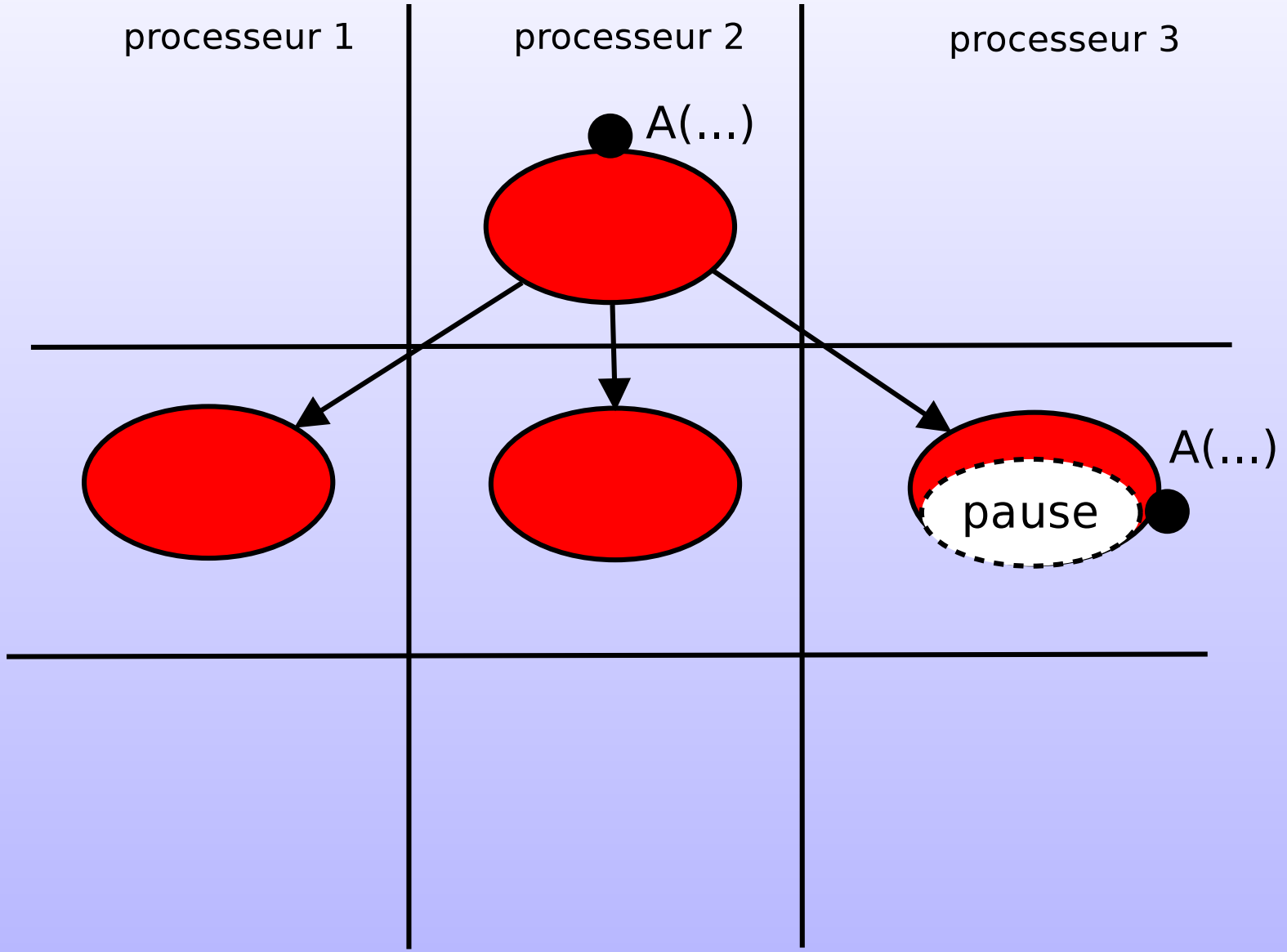
Parallelisation du model checking de CTL*



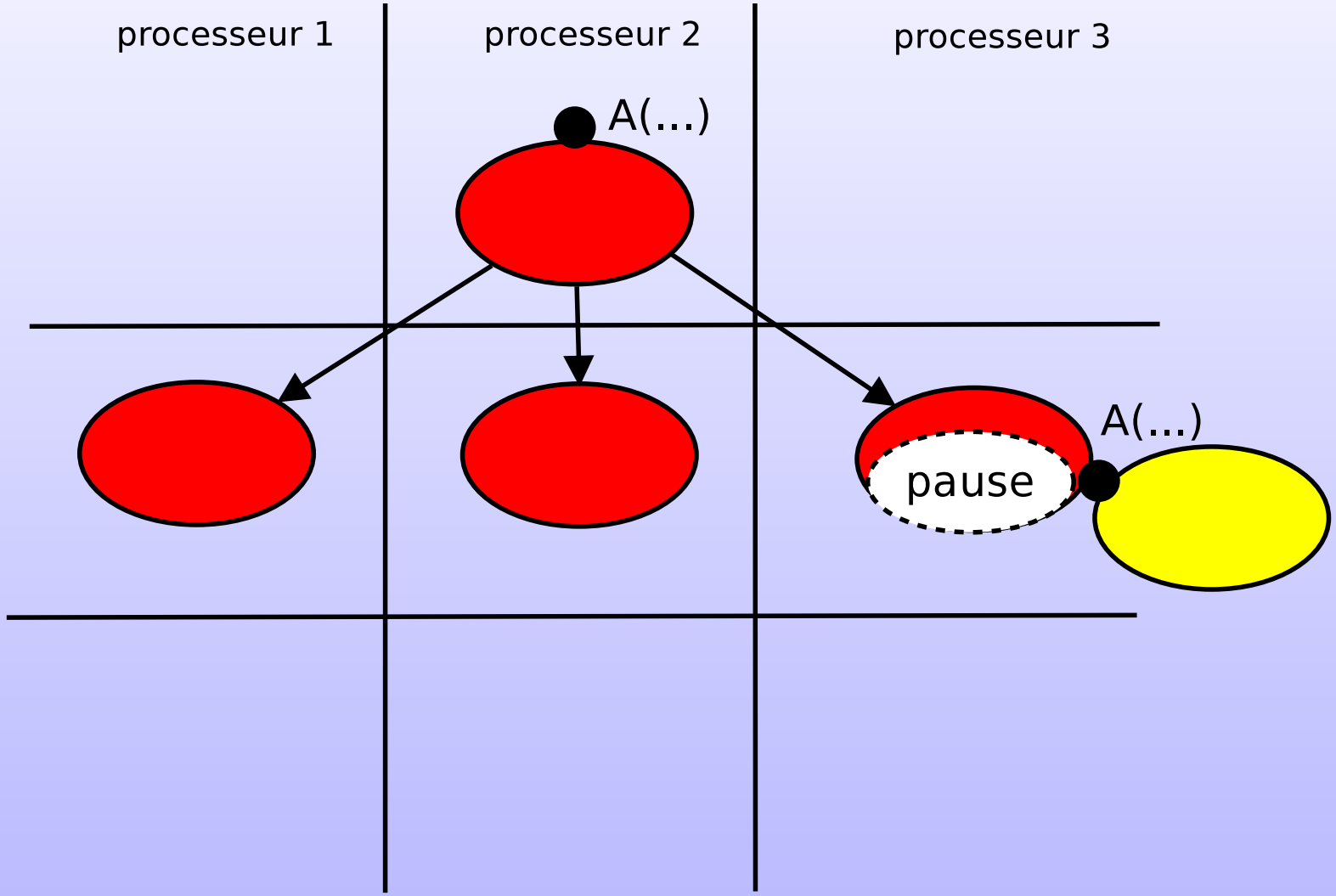
Parallelisation du model checking de CTL*



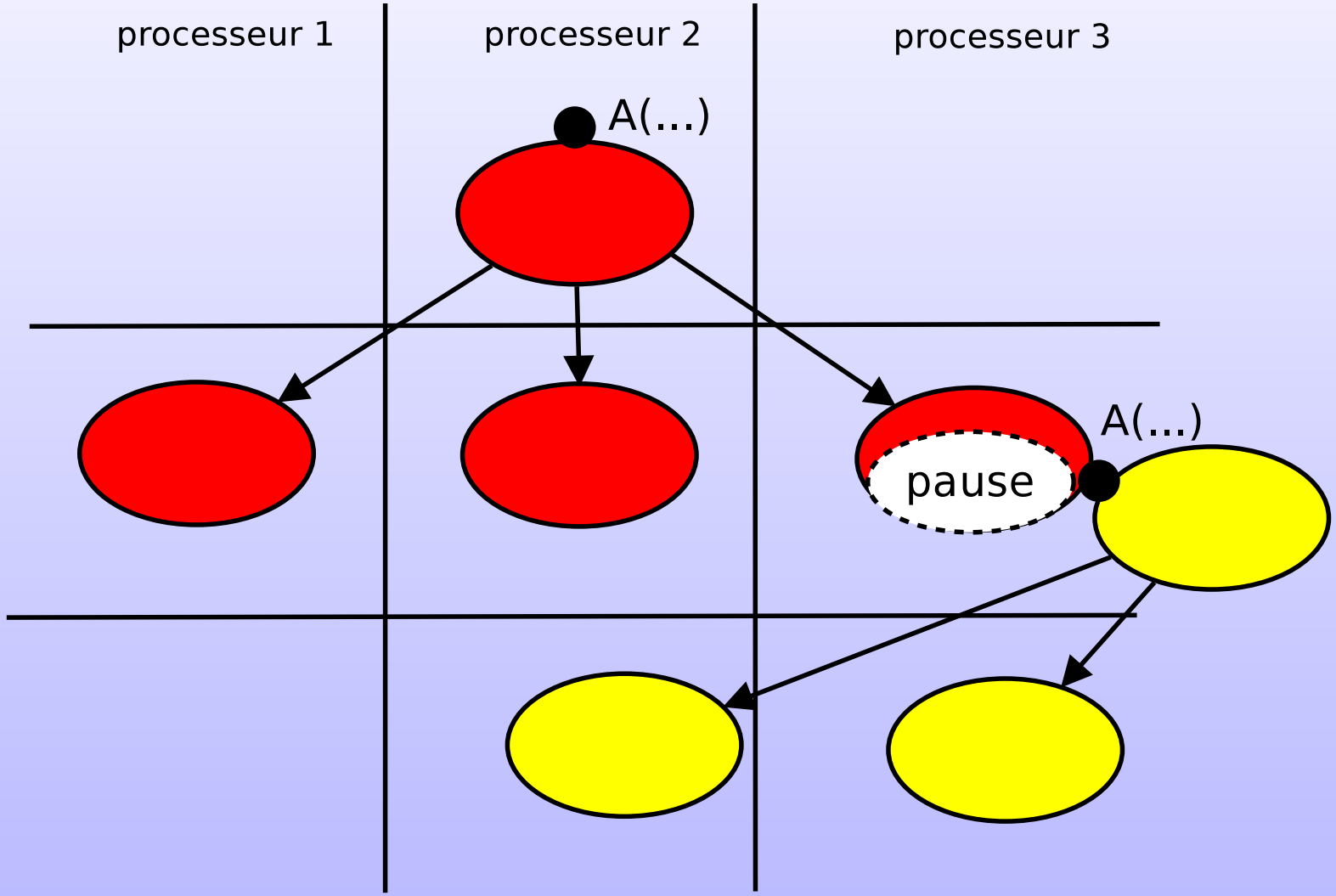
Parallelisation du model checking de CTL*



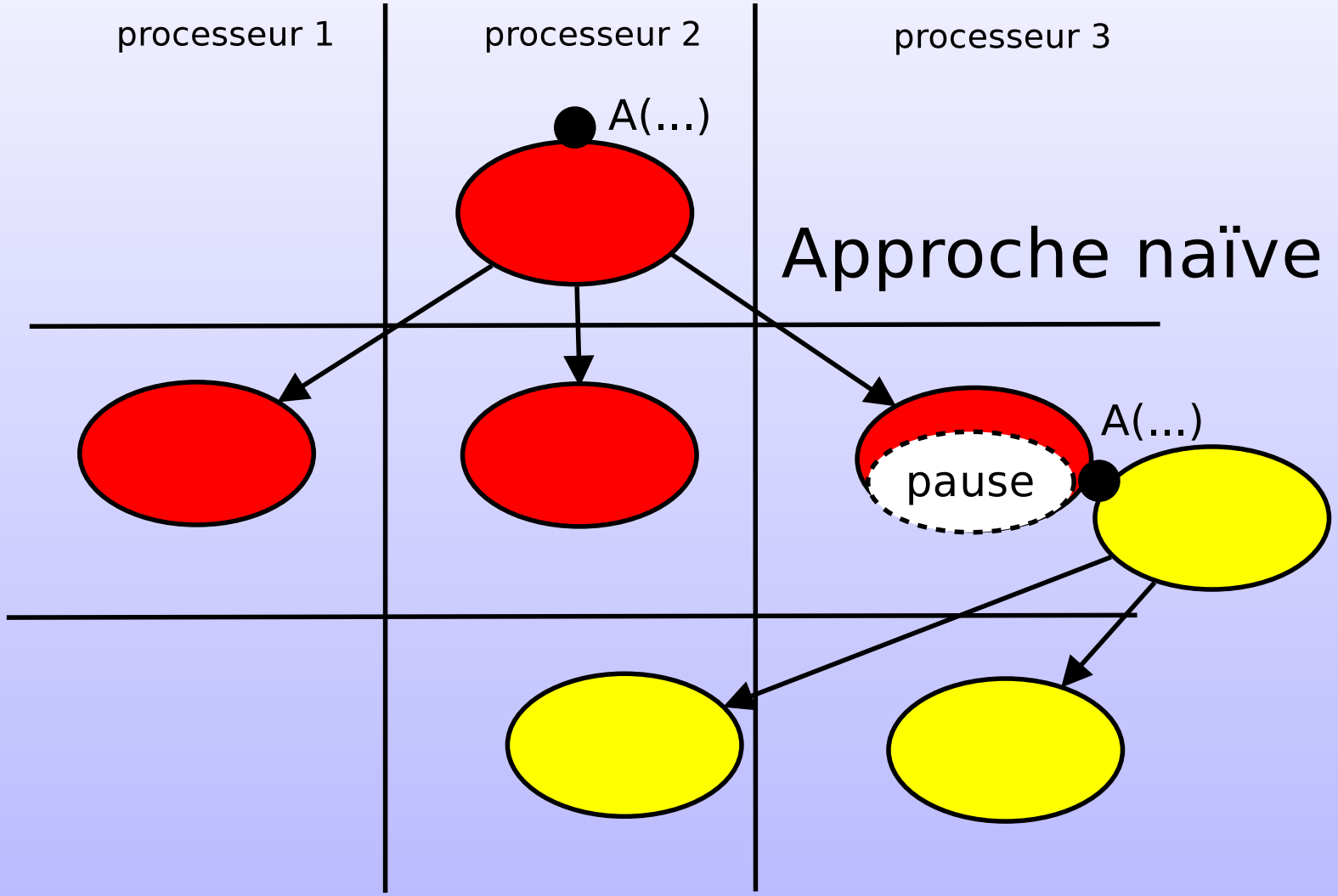
Parallelisation du model checking de CTL*



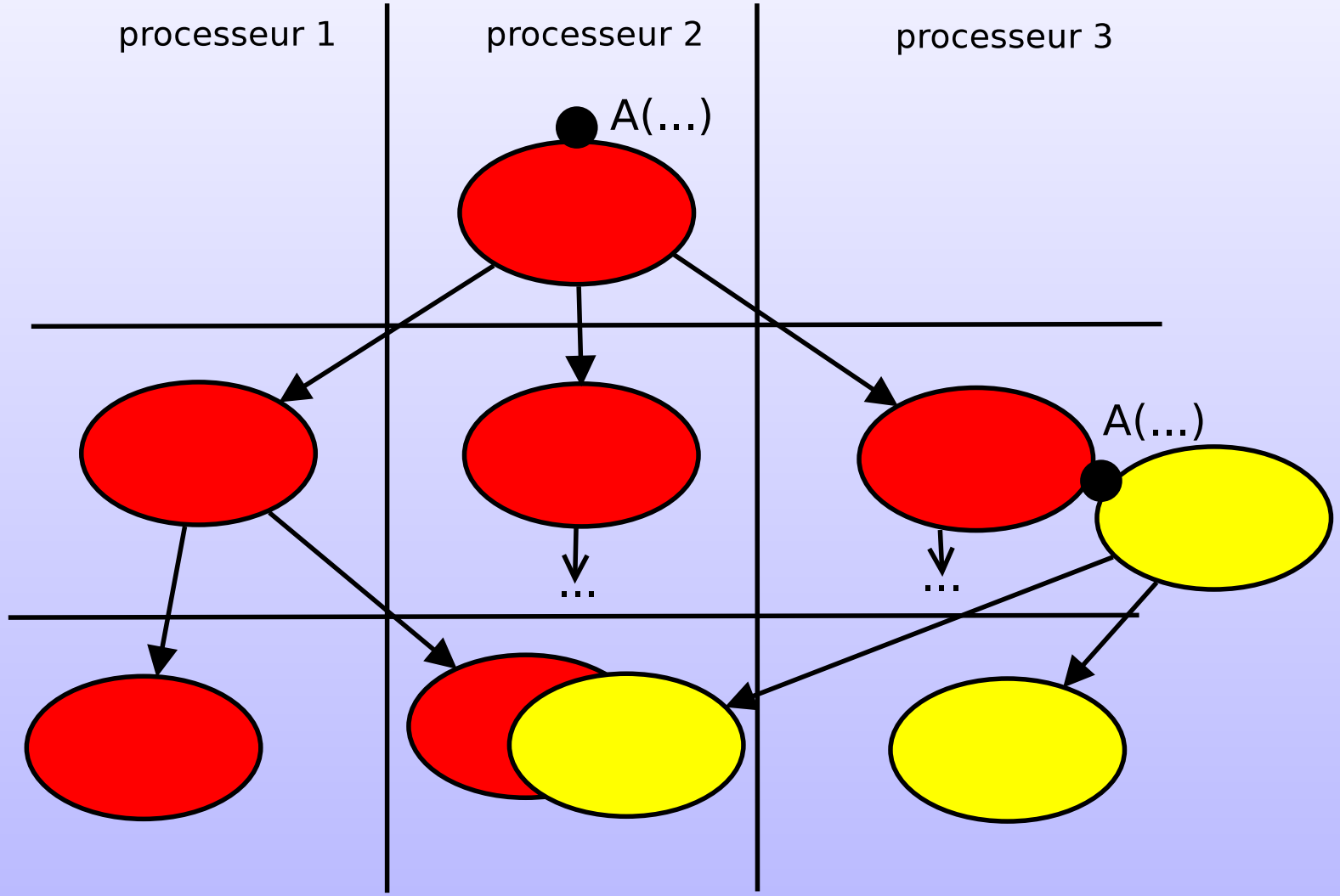
Parallelisation du model checking de CTL*



Parallelisation du model checking de CTL*

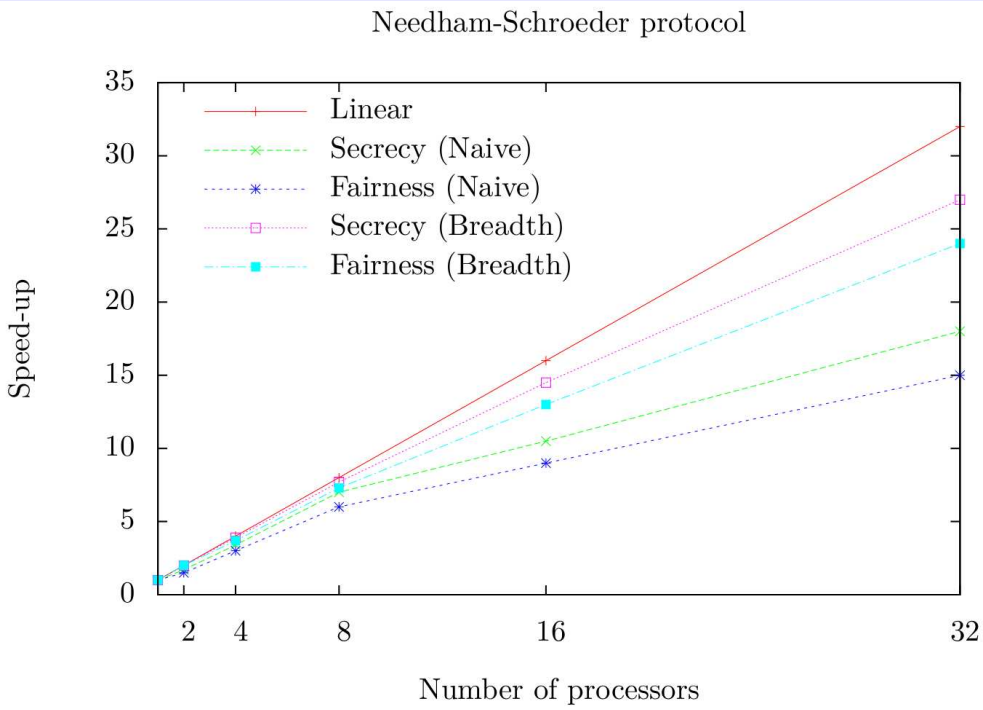


Parallelisation du model checking de CTL*

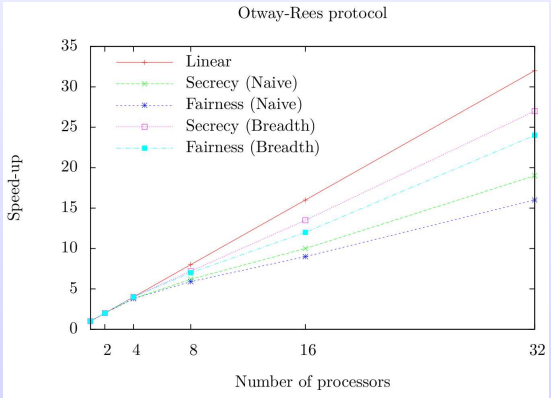


Accélérations sur CTL*

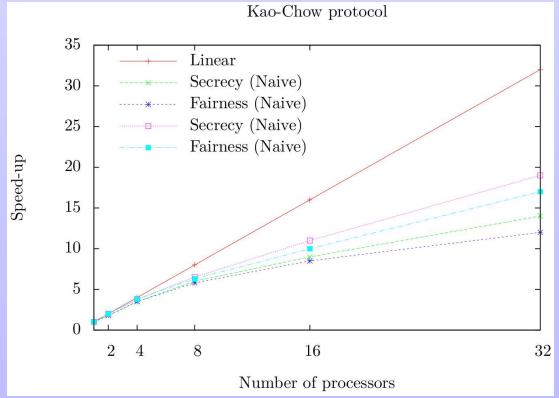
Needham-Schroeder



Otway-Rees



Kao-Chow



Conclusion

Contributions

- Des algorithmes parallèles pour :
  - le calcul de l'espace des états
  - le model-checking de LTL
  - le model-checking de CTL*
- Prototypes
- Evaluation sur des protocoles modélisés

Perspectives

- Etudes de cas plus poussés sur CTL* (LTL)
- Tester le passage à l'échelle sur plus de processeurs
- Preuves formelles
  – travail de thèse de Jean Fortin
  – travail de M2 de Arthur Hidalgo
- Génération facilité des modèles
  – syntaxe "conviviale" compilée en ABCD
- Etendre le domaine d'application
  – protocole plus complexes (boucle, choix, ...)
    $\Rightarrow$ P2P (cf. thèse de Samira Chaou), routage ad hoc
- autres applications que les protocoles ?
- Logiques plus expressives (ou différentes)
  – Past LTL
  – ATL