Composition d'algorithmes autostabilisants

Stéphane Devismes

Université Joseph Fourier, Grenoble I

13 octobre 2020

Plan

Plan

Objectif

La réutilisation d'algorithmes est une préoccupation majeure en génie logiciel.

Objectif

La réutilisation d'algorithmes est une préoccupation majeure en génie logiciel.

En autostabilisation, cette réutilisation se matérialise par la notion de composition.

Objectif

La réutilisation d'algorithmes est une préoccupation majeure en génie logiciel.

En autostabilisation, cette réutilisation se matérialise par la notion de composition.

L'objectif principale des techniques de composition est de **simplifier l'écriture des algorithmes autostabilisants**.

Remarque fondamentale

On définit une composition à l'aide d'un opérateur, noté ici $\circ$.

Remarque fondamentale

On définit une composition à l'aide d'un opérateur, noté ici $\circ$.

La sémantique de l'opérateur ne doit pas augmenter la puissance du modèle.

Remarque fondamentale

On définit une composition à l'aide d'un opérateur, noté ici $\circ$.

La sémantique de l'opérateur ne doit pas augmenter la puissance du modèle.

Il doit exister des **règles de réécriture** pour traduire la composition de plusieurs algorithmes en un seul algorithme n'utilisant pas la composition et ayant exactement le même comportement.

Approches classiques

1. Emuler le comportement d'un démon faible avec un démon fort

Approches classiques

1. Emuler le comportement d'un démon faible avec un démon fort
2. Calculer puis utiliser un sous-graphe couvrant particulier

Emuler le comportement d'un démon plus faible

1. Créer un algorithme autostabilisant qui émule le comportement d'un démon considéré comme plus simple sous l'hypothèse d'un démon plus fort.
   1.1 Par exemple, émuler un démon synchrone, central, ou localement central sous l'hypothèse d'un démon distribué.

Exemple : utiliser une circulation de jeton pour émuler un démon central.

Emuler le comportement d'un démon plus faible

1. Créer un algorithme autostabilisant qui émule le comportement d'un démon considéré comme plus simple sous l'hypothèse d'un démon plus fort.
   1.1 Par exemple, émuler un démon synchrone, central, ou localement central sous l'hypothèse d'un démon distribué.
2. Puis, écrire un algorithme autostabilisant qui résout la tâche voulue en supposant le démon plus simple.

Exemple : utiliser une circulation de jeton pour émuler un démon central.

Emuler le comportement d'un démon plus faible

1. Créer un algorithme autostabilisant qui émule le comportement d'un démon considéré comme plus simple sous l'hypothèse d'un démon plus fort.
   1.1 Par exemple, émuler un démon synchrone, central, ou localement central sous l'hypothèse d'un démon distribué.
2. Puis, écrire un algorithme autostabilisant qui résout la tâche voulue en supposant le démon plus simple.
3. **La composition des deux algorithmes permet alors de résoudre la même tâche sous l'hypothèse du démon le plus général.**

Exemple : utiliser une circulation de jeton pour émuler un démon central.

Calculer puis utiliser un sous-graphe couvrant particulier

1. Résoudre le problème avec algorithme autostabilisant en supposant une topologie particulière.
   1.1 Par exemple, un arbre ou un anneau.

Calculer puis utiliser un sous-graphe couvrant particulier

1. Résoudre le problème avec algorithme autostabilisant en supposant une topologie particulière.
   1.1 Par exemple, un arbre ou un anneau.
2. Puis, écrire un algorithme qui construit cette structure au dessus du réseau réel (généralement quelconque).

Calculer puis utiliser un sous-graphe couvrant particulier

1. Résoudre le problème avec algorithme autostabilisant en supposant une topologie particulière.
   1.1 Par exemple, un arbre ou un anneau.
2. Puis, écrire un algorithme qui construit cette structure au dessus du réseau réel (généralement quelconque).
3. **La composition permet alors d'obtenir un algorithme autostabilisant résolvant le problème posé sur une topologie plus complexe.**

Calculer puis utiliser un sous-graphe couvrant particulier

1. Résoudre le problème avec algorithme autostabilisant en supposant une topologie particulière.
   1.1 Par exemple, un arbre ou un anneau.
2. Puis, écrire un algorithme qui construit cette structure au dessus du réseau réel (généralement quelconque).
3. **La composition permet alors d'obtenir un algorithme autostabilisant résolvant le problème posé sur une topologie plus complexe.**

La composition collatérale hiérarchique permet de réaliser cela !

Plan

Idées

Soit $\mathcal{A}$ et $\mathcal{B}$ deux algorithmes.

Idées

Soit $\mathcal{A}$ et $\mathcal{B}$ deux algorithmes.

Chaque processus p exécute ses programmes (locaux) $\mathcal{A}(p)$ et $\mathcal{B}(p)$ concurremment.

Idées

Soit $\mathcal{A}$ et $\mathcal{B}$ deux algorithmes.

Chaque processus p exécute ses programmes (locaux) $\mathcal{A}(p)$ et $\mathcal{B}(p)$ concurremment.

$\mathcal{B}(p)$ utilise la sortie de $\mathcal{A}(p)$

Idées

Soit $\mathcal{A}$ et $\mathcal{B}$ deux algorithmes.

Chaque processus p exécute ses programmes (locaux) $\mathcal{A}(p)$ et $\mathcal{B}(p)$ concurremment.

$\mathcal{B}(p)$ utilise la sortie de $\mathcal{A}(p)$

Localement à p, les règles de $\mathcal{A}$ sont prioritaires sur les règles de $\mathcal{B}$.

Idées

Soit $\mathcal{A}$ et $\mathcal{B}$ deux algorithmes.

Chaque processus p exécute ses programmes (locaux) $\mathcal{A}(p)$ et $\mathcal{B}(p)$ concurremment.

$\mathcal{B}(p)$ utilise la sortie de $\mathcal{A}(p)$

Localement à p, les règles de $\mathcal{A}$ sont prioritaires sur les règles de $\mathcal{B}$.

Exemple : $\mathcal{A}$ calcule un arbre couvrant et $\mathcal{B}$ utilise la structure de l'arbre pour faire un calcul global.

Définition

Définition 1

Soit $\mathcal{A}$ et $\mathcal{B}$ deux algorithmes tels qu'aucune variable écrite par $\mathcal{B}$ n'apparaît dans $\mathcal{A}$.

Dans la composition collatérale hiérarchique de $\mathcal{A}$ et $\mathcal{B}$, **notée** $\mathcal{B} \circ \mathcal{A}$, le programme local de chaque processus p, $(\mathcal{B} \circ \mathcal{A})(p)$, est défini comme suit :

- $(\mathcal{B} \circ \mathcal{A})(p)$ contient toutes les variables de $\mathcal{A}(p)$ et $\mathcal{B}(p)$;
- $(\mathcal{B} \circ \mathcal{A})(p)$ contient toutes les actions de $\mathcal{A}(p)$;
- pour toute action $G_i \rightarrow S_i$ de $\mathcal{B}(p)$, $(\mathcal{B} \circ \mathcal{A})(p)$ contient l'action $\neg C_p \wedge G_i \rightarrow S_i$ où C_p est la disjonction de toutes les gardes des actions de $\mathcal{A}(p)$.

Plan

Propriété admise sans preuve

Théorème 1

$\mathcal{B} \circ \mathcal{A}$ stabilise à *SP* en **supposant un démon faiblement équitable** si :

- $\mathcal{A}$ est un algorithme (autostabilisant) silencieux en supposant un démon faiblement équitable ;
- $\mathcal{B}$ stabilise à *SP* à partir de toute configuration où aucune action de $\mathcal{A}$ n'est activable en supposant un démon faiblement équitable.[1]

1. Pour rappel, la spécification de $\mathcal{A}$ est satisfaite dans une telle configuration.

Plan

Calcul du maximum (1/2)

Soit $G = (V, E)$ un réseau **connexe et enraciné en** $R \in V$ où chaque processus p dispose en entrée une constante entière E_p.

Calcul du maximum (1/2)

Soit $G = (V, E)$ un réseau **connexe et enraciné en** $R \in V$ où chaque processus p dispose en entrée une constante entière E_p.

Hypothèse supplémentaire : chaque processus p connait son numéro de canal dans l'ensemble $\mathcal{N}_q$ de chacun de ces voisins q.

Calcul du maximum (1/2)

Soit $G = (V, E)$ un réseau **connexe et enraciné en** $R \in V$ où chaque processus p dispose en entrée une constante entière E_p.

Hypothèse supplémentaire : chaque processus p connait son numéro de canal dans l'ensemble $\mathcal{N}_q$ de chacun de ces voisins q.

p pourra tester si, par exemple, le pointeur $père_q$ le désigne comme le père de q dans l'arbre.

Dans la suite, un tel test sera simplement écrit $père_q = p$.

Calcul du maximum (1/2)

On souhaite écrire un algorithme

- autostabilisant et silencieux
- sous l'hypothèse d'un démon distribué faiblement équitable
- où tous les processus p calculent dans la variable de sortie $Sortie_p$ la valeur maximum parmi l'ensemble des entrées

Dans la configuration terminale, tout processus p doit vérifier $Sortie_p = \max_{q \in V}\{E_q\}$.

Décomposition du problème

Décomposition du problème

1. Calcul silencieux d'arbre couvrant T enraciné en R.

Décomposition du problème

1. Calcul silencieux d'arbre couvrant T enraciné en R.

   Dans le suite, $T(p)$ dénotera le sous-arbre de T enraciné en p, $H_{T(p)}$ dénotera la hauteur de $T(p)$; la hauteur de T sera noté simplement H.

Décomposition du problème

1. Calcul silencieux d'arbre couvrant T enraciné en R.

   Dans le suite, $T(p)$ dénotera le sous-arbre de T enraciné en p, $H_{T(p)}$ dénotera la hauteur de $T(p)$; la hauteur de T sera noté simplement H.
2. Un algorithme silencieux qui affecte la variable $MaxT_p$ de chaque processus p à $\max_{q \in T(p)}\{E_q\}$ en supposant que T est bien défini par les sorties *père* du premier algorithme.

Décomposition du problème

1. Calcul silencieux d'arbre couvrant T enraciné en R.

   Dans le suite, $T(p)$ dénotera le sous-arbre de T enraciné en p, $H_{T(p)}$ dénotera la hauteur de $T(p)$; la hauteur de T sera noté simplement H.
2. Un algorithme silencieux qui affecte la variable $MaxT_p$ de chaque processus p à $\max_{q \in T(p)}\{E_q\}$ en supposant que T est bien défini par les sorties *père* du premier algorithme.
3. Un algorithme silencieux qui diffuse la valeur de $MaxT_R$ dans les variables *Sortie* de tous les processus en supposant que la valeur de $MaxT_R$ est constante et T est bien défini par les sorties *père* du premier algorithme.

Partie 1

Nous utilisons l'**algorithme silencieux de calcul d'arbre couvrant en largeur d'abord** vu précédemment.

Partie 1

Nous utilisons l'**algorithme silencieux de calcul d'arbre couvrant en largeur d'abord** vu précédemment.

Dans la suite, cet algorithme est nommé $\mathcal{BFS}$.

Partie 2

Algorithme $\mathcal{MSA}$:

$$MaxT_p \neq \max(\{E_p\} \cup \{MaxT_q, q \in N_p \wedge père_q = p\})$$

$$\rightarrow$$

$$MaxT_p \leftarrow \max(\{E_p\} \cup \{MaxT_q, q \in N_p \wedge père_q = p\})$$

Schéma de la preuve

Pour démontrer que $\mathcal{MSA} \circ \mathcal{BFS}$ stabilise, sous l'hypothèse d'un démon distribué faiblement équitable, nous devons montrer que :

Schéma de la preuve

Pour démontrer que $\mathcal{MSA} \circ \mathcal{BFS}$ stabilise, sous l'hypothèse d'un démon distribué faiblement équitable, nous devons montrer que :

1. A partir de toute configuration terminale de $\mathcal{BFS}$ (où T est bien défini), $\mathcal{MSA}$ **converge en un nombre fini de rondes** vers une configuration terminale.

Schéma de la preuve

Pour démontrer que $\mathcal{MSA} \circ \mathcal{BFS}$ stabilise, sous l'hypothèse d'un démon distribué faiblement équitable, nous devons montrer que :

1. A partir de toute configuration terminale de $\mathcal{BFS}$ (où T est bien défini), $\mathcal{MSA}$ **converge en un nombre fini de rondes** vers une configuration terminale.
2. Dans **toute configuration terminale de** $\mathcal{MSA} \circ \mathcal{BFS}$, $MaxT_p = \max_{q \in T(p)}\{E_q\}$ pour tout processus p.

Preuve (1/3)

Lemme 1

A partir de toute configuration terminale de $\mathcal{BFS}$, $\mathcal{MSA}$ converge en $H + 1$ rondes vers une configuration terminale.

Preuve. Par récurrence sur la hauteur des sous-arbres.

Soit γ_t une configuration terminale de $\mathcal{BFS}$.

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Par définition, p est une feuille de T, qui est bien défini.

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Par définition, p est une feuille de T, qui est bien défini.

Donc $\{MaxT_q, q \in N_p \wedge père_q = p\} = \emptyset$.

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Par définition, p est une feuille de T, qui est bien défini.

Donc $\{MaxT_q, q \in N_p \wedge père_q = p\} = \emptyset$.

Ainsi, si p vérifie $MaxT_p = E_p$, alors p est inactivable pour toujours car E_p est constante.

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Par définition, p est une feuille de T, qui est bien défini.

Donc $\{MaxT_q, q \in N_p \wedge père_q = p\} = \emptyset$.

Ainsi, si p vérifie $MaxT_p = E_p$, alors p est inactivable pour toujours car E_p est constante.

Sinon, p est continûment activable car le prédicat de la règle de p ne dépend que des variables de p. Ainsi, $MaxT_p$ est affecté à E_p lors de la première ronde à partir de γ_t et nous retrouvons le cas précédent. Ainsi, p est inactivable pour toujours au plus 1 ronde après γ_t.

Hypothèse de récurrence

Supposons que tout processus p tel que $H_{T(p)} \leq k$ est inactivable pour toujours au plus $k + 1$ rondes après γ_t.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Au début de la $k + 2$ rondes, la valeur de $\{MaxT_q, q \in N_p \wedge père_q = p\}$ est fixe pour toujours, par hypothèse de récurrence.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Au début de la $k + 2$ rondes, la valeur de $\{MaxT_q, q \in N_p \wedge père_q = p\}$ est fixe pour toujours, par hypothèse de récurrence.

Ainsi, la valeur de prédicat de la règle de p ne dépend que de $MaxT_p$ et de constantes.

Ainsi, si p est activable, il l'est continûment et exécute sa règle lors de la ronde : à la fin de la ronde, p devient inactivable pour toujours.

En posant, $k = H$, nous obtenons l'intitulé du lemme. □

Preuve (2/3)

Lemme 2

Dans toute configuration terminale de $\mathcal{MSA} \circ \mathcal{BFS}$, nous avons $MaxT_p = \max_{q \in T(p)}\{E_q\}$ pour tout processus p.

Preuve. Par récurrence sur la hauteur des sous-arbres.

Soit γ_t une configuration terminale de $\mathcal{MSA} \circ \mathcal{BFS}$.

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Par définition, p est une feuille de T (qui est bien défini).

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Par définition, p est une feuille de T (qui est bien défini).

Donc $\{MaxT_q, q \in N_p \wedge père_q = p\} = \emptyset$.

Cas de base

Soit p un processus tel que $H_{T(p)} = 0$ dans γ_t.

Par définition, p est une feuille de T (qui est bien défini).

Donc $\{MaxT_q, q \in N_p \wedge père_q = p\} = \emptyset$.

Par suite, $MaxT_p = E_p$. Puisque $T(p) = \{p\}$, le lemme est vérifié dans ce cas.

Hypothèse de récurrence

Supposons que tout processus p tel que $H_{T(p)} \leq k$ vérifie $MaxT_p = \max_{q \in T(p)}\{E_q\}$ dans γ_t.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)\}$ dans γ_t.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)\}$ dans γ_t.

Or, pour processus q tel que $q \in N_p \wedge père_q = p$ dans γ_t vérifie $H_{T(q)} \leq k$.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)\}$ dans γ_t.

Or, pour processus q tel que $q \in N_p \wedge père_q = p$ dans γ_t vérifie $H_{T(q)} \leq k$.

Donc, par hypothèse de récurrence, $MaxT_q = \max_{x \in T(q)}\{E_x\}$.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)\}$ dans γ_t.

Or, pour processus q tel que $q \in N_p \wedge père_q = p$ dans γ_t vérifie $H_{T(q)} \leq k$.

Donc, par hypothèse de récurrence, $MaxT_q = \max_{x \in T(q)}\{E_x\}$.

Par suite, $MaxT_p = \max(\{E_p\} \cup \{MaxT_q, q \in N_p \wedge père_q = p\}) = \max(\{E_p\} \cup \{\max_{x \in T(q)}\{E_x\}, q \in N_p \wedge père_q = p\})$.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)\}$ dans γ_t.

Or, pour processus q tel que $q \in N_p \wedge père_q = p$ dans γ_t vérifie $H_{T(q)} \leq k$.

Donc, par hypothèse de récurrence, $MaxT_q = \max_{x \in T(q)}\{E_x\}$.

Par suite, $MaxT_p = \max(\{E_p\} \cup \{MaxT_q, q \in N_p \wedge père_q = p\}) = \max(\{E_p\} \cup \{\max_{x \in T(q)}\{E_x\}, q \in N_p \wedge père_q = p\})$.

Or, par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)$.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)\}$ dans γ_t.

Or, pour processus q tel que $q \in N_p \wedge père_q = p$ dans γ_t vérifie $H_{T(q)} \leq k$.

Donc, par hypothèse de récurrence, $MaxT_q = \max_{x \in T(q)}\{E_x\}$.

Par suite, $MaxT_p = \max(\{E_p\} \cup \{MaxT_q, q \in N_p \wedge père_q = p\}) = \max(\{E_p\} \cup \{\max_{x \in T(q)}\{E_x\}, q \in N_p \wedge père_q = p\})$.

Or, par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)$.

Donc, $MaxT_p = \max_{q \in T(p)}\{E_q\}$ dans γ_t.

Pas de récurrence

Soit p un processus tel que $H_{T(p)} = k + 1$ dans γ_t.

Par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)\}$ dans γ_t.

Or, pour processus q tel que $q \in N_p \wedge père_q = p$ dans γ_t vérifie $H_{T(q)} \leq k$.

Donc, par hypothèse de récurrence, $MaxT_q = \max_{x \in T(q)}\{E_x\}$.

Par suite, $MaxT_p = \max(\{E_p\} \cup \{MaxT_q, q \in N_p \wedge père_q = p\}) = \max(\{E_p\} \cup \{\max_{x \in T(q)}\{E_x\}, q \in N_p \wedge père_q = p\})$.

Or, par définition, $T(p) = \{p\} \cup \bigcup_{q \in N_p \wedge père_q = p} T(q)$.

Donc, $MaxT_p = \max_{q \in T(p)}\{E_q\}$ dans γ_t.

En posant, $k = H$, nous obtenons l'intitulé du lemme.

□

Preuve (3/3)

D'après le corollaire 1, les deux lemmes précédents, et le fait que $\mathcal{BFS}$ calcule en au plus $\mathcal{D}+2$ rondes un arbre couvrant de hauteur $\mathcal{D}$ (le diamètre du réseau), nous avons :

Théorème 2

$\mathcal{MSA} \circ \mathcal{BFS}$ est autostabilisant et silencieux dans un réseau enraciné connexe.

Son temps de stabilisation est au plus $2\mathcal{D}+3$ rondes.

Dans toute configuration terminale de $\mathcal{MSA} \circ \mathcal{BFS}$, nous avons $MaxT_p = \max_{q \in T(p)}\{E_q\}$ pour tout processus p.

Partie 3

Algorithme $\mathcal{MAX}$:

Règle pour la racine R :

$$Sortie_R \neq MaxT_R \rightarrow Sortie_R \leftarrow MaxT_R$$

Règle pour les autres processus p :

$$Sortie_p \neq Sortie_{père_p} \rightarrow Sortie_p \leftarrow Sortie_{père_p}$$

Schéma de la preuve

Pour démontrer que $\mathcal{MAX} \circ \mathcal{MSA} \circ \mathcal{BFS}$ stabilise, sous l'hypothèse d'un démon distribué faiblement équitable, nous devons montrer que :

Schéma de la preuve

Pour démontrer que $\mathcal{MAX} \circ \mathcal{MSA} \circ \mathcal{BFS}$ stabilise, sous l'hypothèse d'un démon distribué faiblement équitable, nous devons montrer que :

1. A partir de toute configuration terminale de $\mathcal{MSA} \circ \mathcal{BFS}$ (où T est bien défini et $MaxT_p = \max_{q \in T(p)}\{E_q\}$ pour tout processus p), $\mathcal{MAX}$ **converge en un nombre fini de rondes** vers une configuration terminale.

Schéma de la preuve

Pour démontrer que $\mathcal{MAX} \circ \mathcal{MSA} \circ \mathcal{BFS}$ stabilise, sous l'hypothèse d'un démon distribué faiblement équitable, nous devons montrer que :

1. A partir de toute configuration terminale de $\mathcal{MSA} \circ \mathcal{BFS}$ (où T est bien défini et $MaxT_p = \max_{q \in T(p)}\{E_q\}$ pour tout processus p), $\mathcal{MAX}$ **converge en un nombre fini de rondes** vers une configuration terminale.
2. Dans **toute configuration terminale de** $\mathcal{MAX} \circ \mathcal{MSA} \circ \mathcal{BFS}$, $Sortie_p = \max_{q \in V}\{E_q\}$ pour tout processus p.

Preuve (1/3)

Lemme 3

A partir de toute configuration terminale de $\mathcal{MSA} \circ \mathcal{BFS}$, $\mathcal{MAX}$ converge en $H + 1$ rondes vers une configuration terminale.

Preuve. Par récurrence sur le niveau dans l'arbre.

Soit γ_t une configuration terminale de $\mathcal{MSA} \circ \mathcal{BFS}$.

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Par définition, $p = R$, c'est-à-dire, p est la racine de T (qui est bien défini).

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Par définition, $p = R$, c'est-à-dire, p est la racine de T (qui est bien défini).

Si p vérifie $Sortie_p = MaxT_p$, alors p est inactivable pour toujours car $MaxT_p$ est constante.

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Par définition, $p = R$, c'est-à-dire, p est la racine de T (qui est bien défini).

Si p vérifie $Sortie_p = MaxT_p$, alors p est inactivable pour toujours car $MaxT_p$ est constante.

Sinon, p est continûment activable car le prédicat de la règle de p ne dépend que des variables de p. Ainsi, $Sortie_p$ est affecté à $MaxT_p$ lors de la première ronde à partir de γ_t et nous retrouvons le cas précédent.

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Par définition, $p = R$, c'est-à-dire, p est la racine de T (qui est bien défini).

Si p vérifie $Sortie_p = MaxT_p$, alors p est inactivable pour toujours car $MaxT_p$ est constante.

Sinon, p est continûment activable car le prédicat de la règle de p ne dépend que des variables de p. Ainsi, $Sortie_p$ est affecté à $MaxT_p$ lors de la première ronde à partir de γ_t et nous retrouvons le cas précédent.

Ainsi, p est inactivable pour toujours au plus 1 ronde après γ_t.

Hypothèse de récurrence

Supposons que tout processus p de niveau inférieur ou égal à k est inactivable pour toujours au plus $k + 1$ rondes après γ_t.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

Au début de la $k+2$ rondes, la valeur de $Sortie_{père_p}$ est fixe pour toujours, par hypothèse de récurrence.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

Au début de la $k+2$ rondes, la valeur de $Sortie_{père_p}$ est fixe pour toujours, par hypothèse de récurrence.

Ainsi, la valeur de prédicat de la règle de p ne dépend que de $Sortie_p$ et d'une constante.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

Au début de la $k+2$ rondes, la valeur de $Sortie_{père_p}$ est fixe pour toujours, par hypothèse de récurrence.

Ainsi, la valeur de prédicat de la règle de p ne dépend que de $Sortie_p$ et d'une constante.

Ainsi, si p est activable, il l'est continûment et exécute sa règle lors de la ronde : à la fin de la ronde, p devient inactivable pour toujours.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

Au début de la $k+2$ rondes, la valeur de $Sortie_{père_p}$ est fixe pour toujours, par hypothèse de récurrence.

Ainsi, la valeur de prédicat de la règle de p ne dépend que de $Sortie_p$ et d'une constante.

Ainsi, si p est activable, il l'est continûment et exécute sa règle lors de la ronde : à la fin de la ronde, p devient inactivable pour toujours.

En posant, $k = H$, nous obtenons l'intitulé du lemme. □

Preuve (2/3)

Lemme 4

Dans toute configuration terminale de $\mathcal{MAX} \circ \mathcal{MSA} \circ \mathcal{BFS}$, nous avons $\mathit{Sortie}_p = \max_{q \in V}\{E_q\}$ pour tout processus p.

Preuve. Par récurrence sur le niveau dans l'arbre.

Soit γ_t une configuration terminale de $\mathcal{MAX} \circ \mathcal{MSA} \circ \mathcal{BFS}$.

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Par définition, $p = R$, c'est-à-dire, p est la racine de T (qui est bien défini).

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Par définition, $p = R$, c'est-à-dire, p est la racine de T (qui est bien défini).

Puisque p est inactivable, nous avons
$Sortie_p = MaxT_p = MaxT_R$.

Cas de base

Soit p un processus de niveau 0 dans γ_t.

Par définition, $p = R$, c'est-à-dire, p est la racine de T (qui est bien défini).

Puisque p est inactivable, nous avons $Sortie_p = MaxT_p = MaxT_R$.

D'après le lemme 2, $Sortie_p = \max_{q \in V}\{E_q\}$. Ainsi, le lemme est vérifié dans ce cas.

Hypothèse de récurrence

Supposons que tout processus p niveau inférieur ou égal à k vérifie $\mathit{Sortie}_p = \max_{q \in V}\{E_q\}$ dans γ_t.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

$p \neq R$

Puisque p est inactivable, nous avons $Sortie_p = Sortie_{père_p}$ dans γ_t.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

$p \neq R$

Puisque p est inactivable, nous avons $Sortie_p = Sortie_{père_p}$ dans γ_t.

Or, par hypothèse de récurrence, $Sortie_{père_p} = \max_{q \in V}\{E_q\}$ dans γ_t.

Pas de récurrence

Soit p un processus de niveau $k+1$ dans γ_t.

$p \neq R$

Puisque p est inactivable, nous avons $Sortie_p = Sortie_{père_p}$ dans γ_t.

Or, par hypothèse de récurrence, $Sortie_{père_p} = \max_{q \in V}\{E_q\}$ dans γ_t.

En posant, $k = H$, nous obtenons l'intitulé du lemme.

□

Preuve (3/3)

D'après le corollaire 1, les deux lemmes précédents, le théorème 2 et le fait que $\mathcal{BFS}$ calcule en au plus $\mathcal{D}+2$ rondes un arbre couvrant de hauteur $\mathcal{D}$ (le diamètre du réseau), nous avons :

Théorème 3

$\mathcal{MAX} \circ \mathcal{MSA} \circ \mathcal{BFS}$ est autostabilisant et silencieux dans un réseau enraciné connexe.

Son temps de stabilisation est au plus $3\mathcal{D}+4$ rondes.

Dans toute configuration terminale de $\mathcal{MAX} \circ \mathcal{MSA} \circ \mathcal{BFS}$, nous avons $\mathit{Sortie}_p = \max_{q \in V}\{E_q\}$ pour tout processus p.