UFR 919 Informatique – Master Informatique

Spécialité STL – UE 5I553 – PPC

Paradigmes de programmation concurrente
TD 2 — (2 h)
CCS

Romain Demangeon

Excercie 1 : Machine à café en CCS

Soit la description suivante d'une machine à café :
- la machine accepte les pièces quand elle est disponible,
- on peut appuyer sur un des deux boutons de la machine : *thé* ou *café*,
- quand on appuie sur un bouton, si une piece est présente, et si aucun gobelet n'est présent, la machine sert la boisson requise,
- une personne peut boire la boisson et jeter le gobelet, cela rend la machine disponible.

Question 1

Proposer un processus CCS qui modélise la machine à café.
Explorer son LTS.

Question 2

Proposer un processus CCS qui modélise un utilisateur riche des 3 pieces et qui boit un thé et un café.
Explorer son LTS.

Question 3

Proposer un processus mettant en relation les deux processus précédents.
Explorer ses réductions.

Exercice 2 : Modéles de programmes en CCS

Pour chaque question, proposer un processus CCS modélisant le système de threads.
Explorer les LTS.

Question 1

Un unique thread exécutant le programme suivant :

1. Tâche locale τ
2. Création de deux threads fils exécutant chacun deux tâches locales τ
3. Attente de synchronisation des deux threads et destruction.
4. Retour à l'étape 1.

Question 2

Un thread t_1 exécutant le programme suivant :

1. Tâche locale τ
2. Envoi d'un message asynchrone m à t_2
3. Tâche locale τ
4. Réception d'une réponse asynchrone r de t_2.
5. Tâche locale τ
6. Retour à l'étape 1.

et un thread t_2 adéquat (qui permet à t_1 de ne pas être bloqué).

Question 3

Trois threads qui exécutent chacun successivement trois tâches locales τ et qui se synchronisent les trois ensemble avant de recommencer.

Question 4

Trois threads exécutant chacun le programme suivant :

1. Tâche locale indépendante τ
2. Utilisation d'une ressource partagée r
3. Tâche locale indépendante τ
4. Retour à l'étape 1.

Question 5

Deux threads exécutant chacun le programme suivant :

1. Tâche locale indépendante τ
2. Prise d'une ressource partagée r1
3. Prise d'une ressource partagée r2
4. Tâche locale τ
5. Relâche de la ressource r2
6. Relâche de la ressource r1
7. Retour à l'étape 1.

Question 6

Des threads réalisant le dîner des philosophes.

Question 7

Quatre threads accédant régulièrement à une base de données n'acceptant que deux connexions simultanées.

Question 8

Un thread implémentant le programme p suivant :

1. Tâche locale indépendante τ
2. Création d'un nouveau thread exécutant p
3. Tâche locale indépendante τ
4. Choix non-déterministe entre i) terminaison ou ii) retour à l'étape 1

Question 9

Trois threads qui bouclent en effectuant une tâche τ et un ordonnanceur qui choisit de manière non déterministe entre les trois threads.

Exercice 3 : Sémantique de traces

On écrit $\stackrel{a}{\Rightarrow}$ pour $(\stackrel{\tau}{\rightarrow})^* \stackrel{a}{\rightarrow} (\stackrel{\tau}{\rightarrow})^*$ quand $a \neq \tau$.
Une *trace observable* d'un processus P est une suite d'actions $a_1.a_2...a_n$ telle qu'il existe $P_1, ..., P_n$ t.q. $P \stackrel{a_1}{\Longrightarrow} P_1 \stackrel{a_2}{\Longrightarrow} P_2 \stackrel{a_3}{\Longrightarrow} ... \stackrel{a_n}{\Longrightarrow} P_n$.

La *sémantique de traces* de P est l'ensemble des traces observables de P.

Question 1

Quelle est la sémantique de traces de :

$P_1 = \overline{a}.\overline{a}.b.\overline{a} \mid a.\overline{c} \mid a.\overline{b}$
$P_2 = (\nu b, c)(\overline{a}.\overline{a}.b.\overline{a} \mid a.\overline{c} \mid a.\overline{b})$
$P_3 = \overline{a} \mid \overline{b}$
$P_4 = \overline{a}.\overline{b} + \overline{b}.\overline{a}$

Question 2

Décrire la sémantique de traces des processus de la question 2.

Question 3

Considérer les machines à café suivantes :

$M_1 = p.\tau.(b_1.\overline{the}.M_1 + b_2.\overline{cafe}.M_1)$
$M_2 = p.(\tau.b_1.\overline{the}.M_2 + \tau.b_2.\overline{cafe}.M_2)$

Comparer leurs sémantiques de traces.

Exercice 4 : Réplication et Terminaison

CCS avec réception répliquée est donné par la syntaxe suivante :
$P ::= O \mid P|P \mid \sum_i \alpha_i.P_i \mid (\nu a)P \mid !a.P$
avec la règle : $(\overline{a}.Q \mid !a.P) \rightarrow (Q \mid P \mid !a.P)$

Question 1

Faire l'exercice 2 en CCS avec réception répliquée.

Question 2

Donner des exemples de processus qui bouclent en CCS avec réception répliquée.

Question 3

Donner un exemple de processus dont la taille des réduits est non bornée en CCS avec réception répliquée.

Question 4

Donner un exemple de processus dont le nombre de noms liés des réduits (espace mémoire) est non borné en CCS avec réception répliquée.

Question 5

Se convaincre qu'un processus de CCS avec réception répliquée qui ne contient pas de sous-processus $!a.P$ est terminant.

On appelle *outputs à top level* les éemissions $\overline{a}$ d'un processus qui ne se trouvent pas derrière un input répliqué (i.e. dans le P d'un $!a.P$).

Etudier l'évolution de l'ensemble des *outputs à top level* lors d'une réduction. En déduire un critère de terminaison.