Calculs de processus
TD1

Exercice 1. Considérer la définition suivante (dans le langage $Imp_{\|}$) :

$$b_1.P_1 + \cdots + b_n.P_n := \mathsf{new}\, x = 1 \,\mathsf{in}\, (Q_1 \mid \cdots \mid Q_n)$$

où

$$\begin{aligned} Q_i := \mathsf{new}\, y = 1 \,\mathsf{in}& \\ & \mathsf{await}\, b_i \,\mathsf{do}\, \mathsf{if}\, x = 1 \,\mathsf{then}\, x := 0 \,\mathsf{else}\, y := 0; \\ & \mathsf{if}\, y = 1 \,\mathsf{then}\, P_i \,\mathsf{else}\, \mathsf{skip} \end{aligned}$$

1. Modifier la définition de façon à ce que, une fois que la branche i est sélectionnée, le processus P_i est exécuté de manière atomique.
2. Avec la définition actuelle, le processus $\mathsf{true.skip} + \mathsf{false.skip}$ tourne à l'infini. Modifier la définition pour éviter cela.

Exercice 2. Supposons de modifer la syntaxe du langage $Imp_{\|}$ en rajoutant un constructeur $\mathsf{spawn}\, P$ qui lance un thread parallèle pour l'exécution de P. Formaliser ce comportement dans une sémantique opérationnelle manipulant des n-uplets de la forme $(P_1, \ldots, P_n; s)$, au lieu de (P, s) comme pour la définition habituelle du langage $Imp_{\|}$. Montrer comment coder le parallèle à l'aide de spawn.

Exercice 3. Soient

$$\begin{aligned} P_1 &:= x := 1; x := x + 1, \\ P_2 &:= x := 2. \end{aligned}$$

Montrer que

1. $[\![P_1]\!]^{IO} = [\![P_2]\!]^{IO}$;
2. $[\![P_1 \mid P_2]\!]^{IO} \neq [\![P_2 \mid P_2]\!]^{IO}$;
3. $[\![\mathsf{atomic}(P_1) \mid Q]\!]^{IO} = [\![P_2 \mid Q]\!]^{IO}$ pour tout Q.

Exercice 4. Soient P_1, P_2 comme dans l'exercice précédent, et soit

$$P_3 := x := 1; x := 2.$$

Montrer que

1. $[\![P_1]\!]^{T} \neq [\![P_2]\!]^{T}$;
2. $[\![P_1]\!]^{T} = [\![P_3]\!]^{T}$;
3. $[\![P_1 \mid P_2]\!]^{T} \neq [\![P_3 \mid P_2]\!]^{T}$.