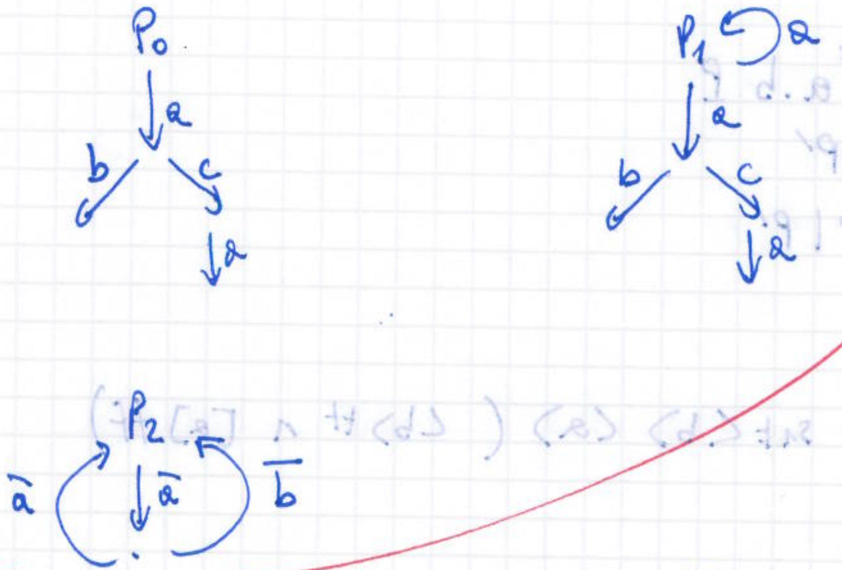
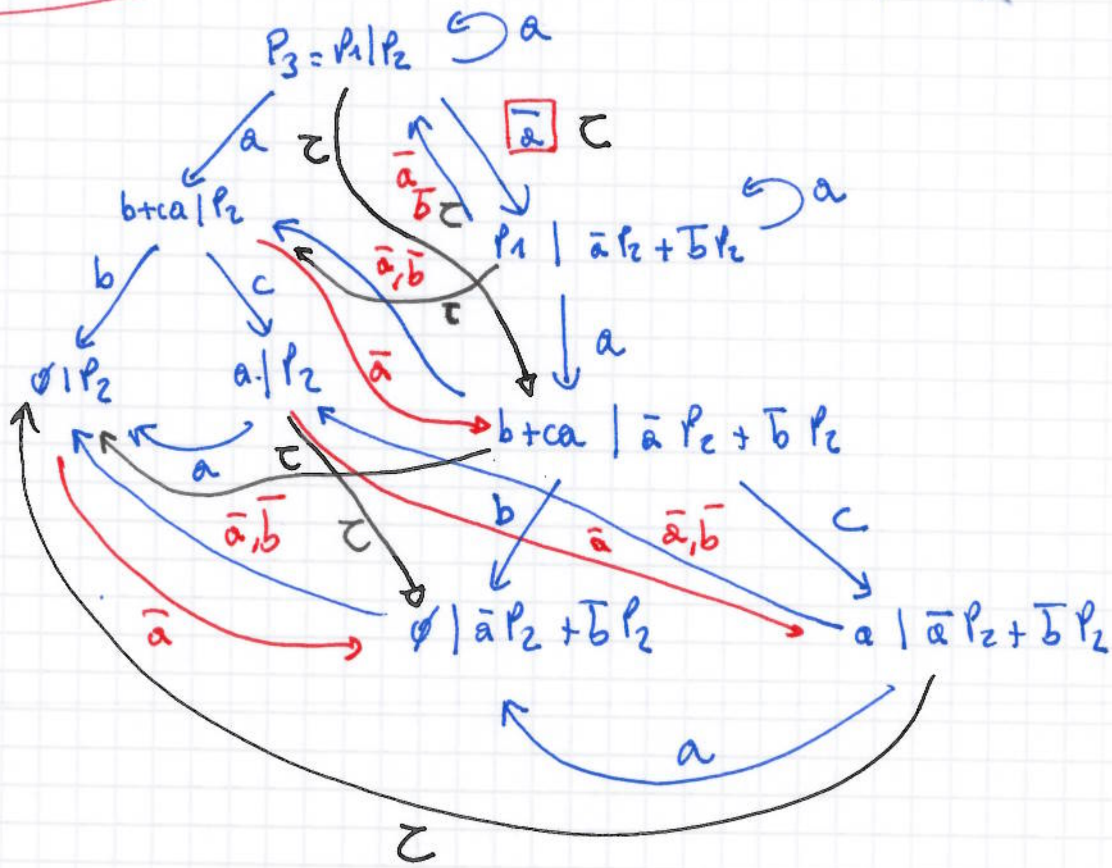
1

exo 1



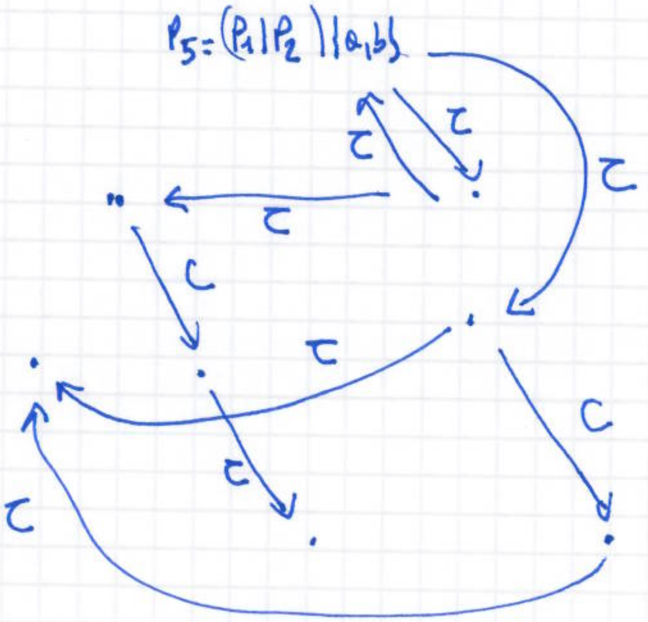
P0
a
b
c
a
P1
a
a
b
c
a
P2
ā
ā
b̄

0.5



P3 = P1 | P2
a
ā
τ
b+ca | P2
P1 | āP2 + b̄P2
ā, b̄
τ
b
c
a
∅ | P2
a | P2
ā
b+ca | āP2 + b̄P2
τ
ā, b̄
b
c
∅ | āP2 + b̄P2
a | āP2 + b̄P2

1



P5 = (P1 | P2) \ {a,b}
τ
c

0.5

2]

Exercice 1 (suite)

2] (i) $P \stackrel{\text{def}}{=} a.b.P$

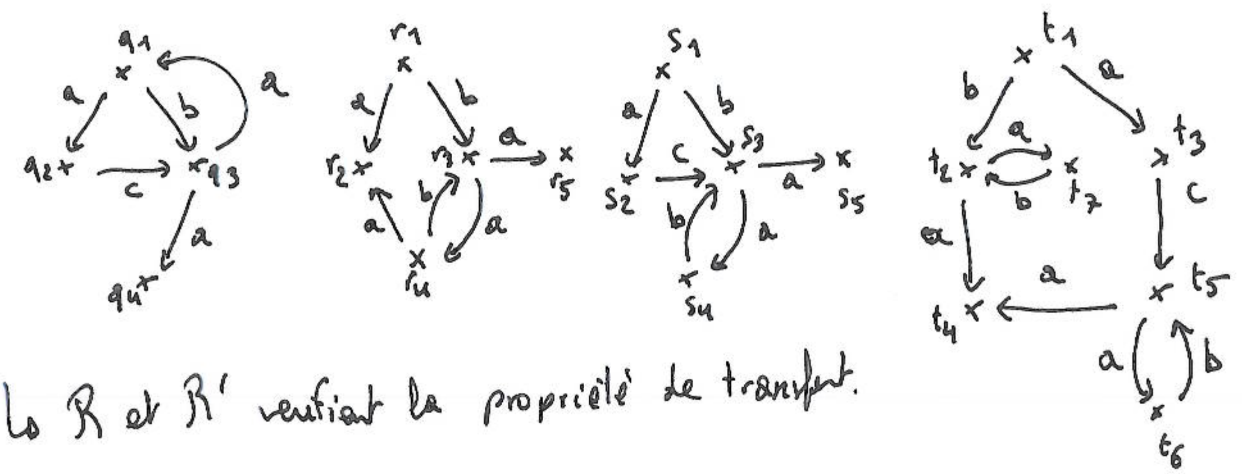
(ii) $\begin{cases} P' \stackrel{\text{d}}{=} a.P' \\ \underbrace{P''}_{\text{résultat}} \stackrel{\text{d}}{=} P \mid P' \end{cases}$

3] $q_1 \sim r_1$ et $s_1 \sim t_1$

justification:

1) $R = \{(q_1, r_1), (q_2, r_2), (q_3, r_3), (q_4, r_5), (q_1, r_4)\}$

2) $R' = \{(s_1, t_1), (s_3, t_2), (s_5, t_4), (s_4, t_7), (s_2, t_3), (s_3, t_5), (s_4, t_6)\}$



↳ R et R' vérifient la propriété de transfert.

mais: $q_1 \nsim s_1$

car $s_1 \models \langle b \rangle \langle a \rangle (\langle b \rangle \mathbb{T} \wedge [a] \bot)$

donc: $q_1 \nsim t_1$, $r_1 \nsim s_1$, $r_1 \nsim t_1$.

3/ Exercice 4:

1.] $s \sqsubseteq t$:

tout ce que s peut faire, t peut aussi le faire.

Plus précisément: pour toute transition $s \xrightarrow{a} s'$, il existe une transition $t \xrightarrow{a} t'$ qui permet à t de produire la même action que s et qui conduit à un état où t' peut encore simuler s'.

2.] . $q_0 \sqsubseteq s_0$: et $r_0 \sqsubseteq s_0$

Justification:

$$R = \{ (q_0, s_0), (q_1, s_0), (q_2, s_0) \}$$

et $R' = \{ (r_0, s_0), (r_1, s_0) \}$

Ces deux relations sont des simulations: s_0 peut produire soit un a, soit un b et recommence... Donc tout système étiqueté par des a et des b peut être simulé par s_0.

Mais:

. $s_0 \not\sqsubseteq q_0$ et $s_0 \not\sqsubseteq r_0$: il suffit de voir que depuis s_0, on peut produire une exécution avec uniquement des a... ce n'est pas le cas de q_0 et r_0.

. $q_0 \not\sqsubseteq r_0$: q_0 peut faire 2 a de suite... faux pour r_0.

. $r_0 \not\sqsubseteq q_0$: r_0 peut faire b.

3.] ⓐ On veut montrer que $s_1 \sqsubseteq s_2 \Rightarrow (\forall \varphi \in HML_{\wedge,\diamond},\ s_1 \models \varphi \Rightarrow s_2 \models \varphi)$.

On fait une preuve par induction structurelle sur les formules φ (comme en cours lorsqu'on a montré:

$$s_1 \sim s_2 \Rightarrow (\forall \varphi \in HML : s_1 \models \varphi \Leftrightarrow s_2 \models \varphi) \quad)$$

4S

(suite exercice 4)

▷ On se limite ici au cas $\underline{\langle a\rangle\varphi'}$:

. $s_1 \models \langle a\rangle\varphi' \overset{\text{sémantique}}{\Rightarrow} \exists\, s_1 \xrightarrow{a} s_1'$ t.q. $s_1' \models \varphi'$

Comme $s_1 \sqsubseteq s_2$, il existe $s_2 \xrightarrow{a} s_2'$ avec $s_1' \sqsubseteq s_2'$.

Si $s_1' \sqsubseteq s_2'$, $s_1' \models \varphi'$ implique (par induction) $s_2' \models \varphi'$

Conclusion: $\exists\, s_2 \xrightarrow{a} s_2'$ t.q. $s_2' \models \varphi'$

et donc: $s_2 \models \langle a\rangle\varphi'$.

ⓑ . On peut faire la m̃ chose qu'au ⓐ ...

Ou alors:

Remarque: pour toute formule $\varphi \in HML_u$, il existe $\overline{\varphi} \in HML_e$ telle que

$$q \models \varphi \iff q \not\models \overline{\varphi}$$

def de $\overline{\varphi}$:

$$\begin{cases} \overline{\top} = \bot & \overline{\bot} = \top \\ \overline{\varphi \wedge \psi} = \overline{\varphi} \vee \overline{\psi} & \overline{\varphi \vee \psi} = \overline{\varphi} \wedge \overline{\psi} \\ & \overline{[a]\varphi} = \langle a\rangle\overline{\varphi} \end{cases}$$

ex: $\varphi = [a]([b]\bot \wedge [c]\bot)$: "après tout "a", on ne peut faire ni "b", ni "c"".

$\overline{\varphi} = \langle a\rangle(\langle b\rangle\top \vee \langle c\rangle\top)$: "il existe un a après lequel on peut faire b ou c"

On a bien $q \models \varphi \iff q \not\models \overline{\varphi}$.

Supposons $s_1 \sqsubseteq s_2$

Si $\varphi \in HML_u$ et $s_2 \models \varphi$ alors $s_2 \not\models \overline{\varphi}$ et donc d'après ⓐ $s_1 \not\models \overline{\varphi}$

et donc $s_1 \models \varphi$ ...

5) Suite exercice 4)

4\.

on prend le m̂ jeu qu'en cours ... sauf que l'attaquant (joueur) ne peut pas choisir le système où il prendra une transition : il doit toujours jouer du "coté gauche".

5\.

Soit $\mathcal{R} = \{ (s, r) \mid \forall \varphi \in HMLe, \ s \models \varphi \Rightarrow r \models \varphi \}$

On va montrer que $\mathcal{R}$ est une <u>simulation</u>.

Supposons le contraire : donc il existe $(s, r) \in \mathcal{R}$ et

$\exists\, s \xrightarrow{a} s'$ telle que $\forall\, r \xrightarrow{a} r'$ on a : $(s', r') \notin \mathcal{R}$

[<u>NB</u>: aucune transition $\underline{a}$ depuis r ne peut simuler $s \xrightarrow{a} s'$.]

Si $(s', r') \notin \mathcal{R} \Rightarrow \exists\, \varphi_{r'} \in HMLe$ t.q. $s' \models \varphi_{r'}$ et $r' \not\models \varphi_{r'}$

On définit la formule Φ suivante:

$$\Phi = \langle a \rangle \underbrace{\bigwedge_{r' \in Succ(r, a)} \varphi_{r'}}$$

= conjonction des $\varphi_{r'}$ pour tous les successeurs r' de r par une transition $\underline{a}$.

→ ensemble fini par hypothèse ...

Alors on a : $s \models \Phi$ car la transition $s \xrightarrow{a} s'$ ci-dessus permet d'atteindre l'état s' où tous les $\varphi_{r'}$ sont <u>vrais</u>.

Mais $r \not\models \Phi$: pour tout $r \xrightarrow{a} r'$, on sait que $r' \not\models \varphi_{r'}$ et donc la conjonction sera fausse.

—

6)

Exercice 2 :

2.a) Dans l'algo K, toutes les variables partagées sont single writer / single reader.

2.b) ▪ $\square \neg (P_0 : p_6 \wedge P_1 : p_6)$

▪ $\begin{cases} \square\ (P_0 : p_2 \Rightarrow \diamond\ P_0 : p_6) \\ \square\ (P_1 : p_2 \Rightarrow \diamond\ P_1 : p_6) \end{cases}$

3.a) le codage de la priorité est différent mais le principe reste le même (...).

3.b) Abs. d'interblocage :

Si il y a interblocage, on a :

$\underbrace{B_1 = B_0 = T}_{\text{car } P_0 \text{ et } P_1 \text{ sont ds le await}}$ et $\underbrace{local0 == turn1 \text{ et } local1 \,!= turn0}_{\text{car blocage...}}$ (1)

Mais on sait aussi : turn0 = local0 car seul P_0 peut modifier ces deux variables.

De même : on a : turn1 = local1

On en déduit que la condition (1) implique :

turn1 == turn0

et turn1 != turn0 : contradiction !

3.c] Supposons que P_0 soit bloqué en p5.

Alors on a : $\underbrace{B_1 = T}$ et local0 == turn1

↳ donc P_1 est entre $p_3 \ldots p_7$.

Comme il n'y a pas d'interblocage, P_1 finira par arriver en p_7 et reviendra en SNC : soit il y reste et P_0 passe le await, soit P_1 refait son préprotocole et donnera la priorité à P_0 ... Donc P_0 ne reste pas bloqué.