Points fixes

Définition : treille complet

Un poset $(D, \sqsubseteq)$ est un treille complet ssi $\sqcap X$ et $\sqcup X$ existent pour tout $X \subseteq D$
Pour un treille complet $(D, \sqsubseteq)$ on appelle *bottom* $\bot = \sqcap D$ et *top* $\top = \sqcup D$
Par exemple pour $(\mathcal{P}(P), \subseteq)$ $\bot = \emptyset$ et $\top = P$

Définition : fonction monotone

Soit $(D, \sqsubseteq)$ un poset, une fonction $f : D \to D$ est monotone ssi $d \sqsubseteq d' \Rightarrow f(d) \sqsubseteq f(d')$ pour tout $d, d' \in D$

Définition : point fixe

Soit $(D, \sqsubseteq)$ et $f : D \to D$, $d \in D$ est un point fixe ssi $d = f(d)$

Théorèmes Knaster-Tarski 1

Soit $(D, \sqsubseteq)$ un treille complet et $f : D \to D$ une fonction monotone, alors f a un plus grand point fixe z_{max} et un plus petit point fixe z_{min} donnés par :

1. $z_{max} = \sqcup\{x \in D | x \sqsubseteq f(x)\}$
2. $z_{min} = \sqcap\{x \in D | f(x) \sqsubseteq x\}$

Théorème Knaster-Tarski 2

Soit $(D, \sqsubseteq)$ un treille complet et $f : D \to D$ une fonction monotone et continue á gauche et soit $f^n(d)$ définie par :

$$f^0(d) = d$$
$$f^{n+1}(d) = f(f^n(d))$$

alors,

1. $z_{min} = f^m(\bot)$ pour $m \in \mathbb{N}$ si la suite croissante $f^n(\bot)$ est constante à partir d'un certain k.
2. $z_{max} = f^M(\top)$ pour $M \in \mathbb{N}$ si la suite décroissante $f^n(\top)$ est constante à partir d'un certain k.

Sorbonne Université
Paradigmes de Programmation Concurrente MU5IN553

Cours 4 - Hennessy-Milner Logic

Carlos Agon - Romain Demangeon

8 octobre 2024

Plan du cours

- Logiques modales
- Logique de Hennessy-Milner
- Logique de Hennessy-Milner avec récursion

Introduction

- Voir un terme de CCS comme une spécification
- On peut faire de *queries* au système pour voir si elles sont vérifiées
- Pour vérifier les propriétés d'un processus donné, de fois il est plus facile de le faire en explorant l'espace des états du processus que en cherchant des équivalences.
- On a besoin d'un langage pour exprimer les propriétés (syntaxe, sémantique et un algorithme de vérification)

Logiques modales

- Connue par les grecques, mais formalisée fin XIX siècle
- Introduction des opérateurs
  L (il est nécessaire) $\Box$
  M (il est possible) $\Diamond$
  liées par les formules : $Lp = \neg M \neg p$ et $Mp = \neg L \neg p$

Sémantique


Saul Kripke

- La sémantique est donnée par un modèle $I = (W, R, s)$ avec W un ensemble non vide de mondes. R une relation sur $W \times W$ et $s : (w, p) \rightarrow \{tt, ff\}$ avec p une proposition et $w \in W$
- On note $I_{w,s} f$ le fait que s satisfait la formule f dans le monde w du modèle I

1. $I_{w,s}\, f \iff s(w, f)$ est vrai
2. $I_{w,s}\, A \vee B \iff I_{w,s} A \vee I_{w,s} B$
3. $I_{w,s}\, \neg A \iff \neg I_{w,s} A$
4. $I_{w,s}\, \Box A \iff I_{w',s} A\ \forall w'$ tq. wRw'
5. $I_{w,s}\, \Diamond A \iff \exists w'$ tq. wRw' et $I_{w',s} A$

Logiques temporelles

Dans le cas des logiques temporelles L et M sont remplacés par P et F
Fp : p sera vrai à tout instant future
Pp : p a toujours été vrai dans le passé
W représente les points du temps
R exprime la notion de précédence temporelle

- $I_{w,s}\, FA \iff I_{w',s} A\ \forall w'$ tq. wRw'
- $I_{w,s}\, PA \iff I_{w',s} A\ \forall w'$ tq. $w'Rw$
- Sp : p sera vrai à un instant future : $\neg F \neg p$
- Ep : p a été vrai à un instant dans le passé : $\neg P \neg p$

Systèmes classiques par rapport à R

- K : pas de restriction $F(p \Rightarrow q) \Rightarrow (Fp \Rightarrow Fq)$
- T : réflexive $Fp \Rightarrow p$
- B : symétrique $p \Rightarrow FPp$
- S4 : transitive $Fp \Rightarrow FFp$

Hennessy-Milner logic (HML)

Définition

L'ensemble $\mathcal{M}$ des formules Hennessy-Milner sur un ensemble d'actions Act est donné par :
$F, G ::= tt \mid ff \mid F \wedge G \mid F \vee G \mid \langle a \rangle F \mid [a]F$
avec $a \in Act$, tt = vrai, ff = faux

- Un processus P satisfait $F \vee G$ ssi P satisfait F ou P satisfait G
- Un processus P satisfait $F \wedge G$ ssi P satisfait F et P satisfait G
- Un processus P satisfait $\langle a \rangle$ F ssi il existe une transition a à partir de P menant à un processus satisfaissant F
- Un processus P satisfait $[a]F$ ssi toute transition a partant de P mène à un processus satisfaissant F
- Tous les processus satisfont tt
- Aucun processus satisfait ff

Hennessy-Milner logic (notation)

En fait c'est une logique modale avec :
$\Diamond$ possiblement
$\langle a \rangle F$: Il est *possible* de s'engager dans une action a et satisfaire F
$\Box$ nécessairement
$[a]F$: Peut importe comme on s'engage dans une action a, F sera *nécessairement* satisfaite

Sémantique d'une formule

La sémantique est paramétrisée par un LTS (Proc, Act, $\{\xrightarrow{a} | a \in \text{Act}\}$)
On dénote $[\![F]\!]$ l'ensemble de processus dans Proc qui satisfont F

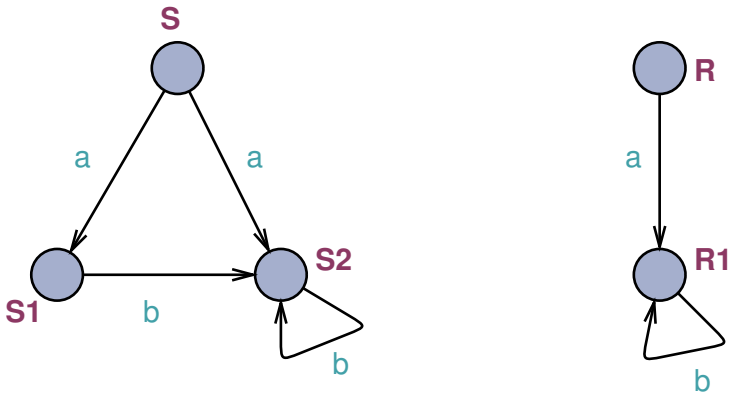
Définition

On défini $[\![F]\!] \subseteq \text{Proc}$ pour $F \in \mathcal{M}$ par :

$$[\![tt]\!] = Proc$$
$$[\![ff]\!] = \emptyset$$
$$[\![F \wedge G]\!] = [\![F]\!] \cap [\![G]\!]$$
$$[\![F \vee G]\!] = [\![F]\!] \cup [\![G]\!]$$
$$[\![\langle a \rangle F]\!] = \langle .a. \rangle [\![F]\!]$$
$$[\![[a]F]\!] = [.a.][\![F]\!]$$

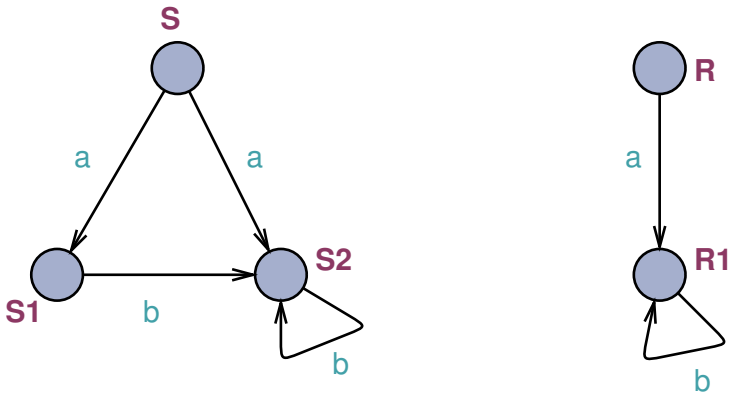
avec
$\langle .a. \rangle$ S $= \{p \in Proc | \exists p' \in S \text{ tq. } p \xrightarrow{a} p'\}$
[.a.] S $= \{p \in Proc \,|\, \forall p' \text{ tq. } p \xrightarrow{a} p' \Rightarrow p' \in S\}$

Exemple 1



$\langle .a. \rangle \{S1, R1\} =$

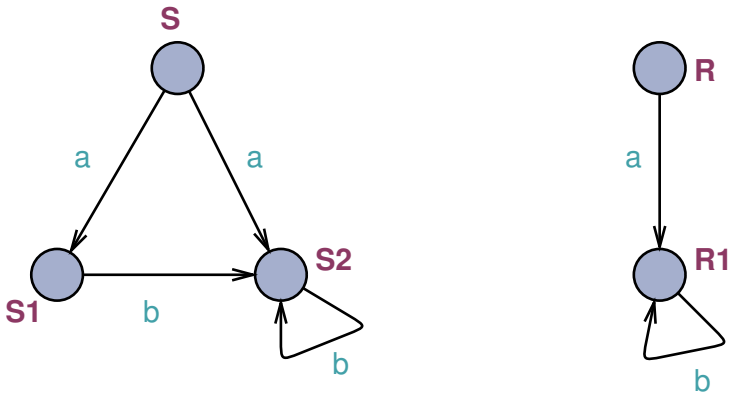
Exemple 1



$\langle .a. \rangle \{S1, R1\} = \{S, R\}$

$[.a.]\{S1, R1\} =$

Exemple 1

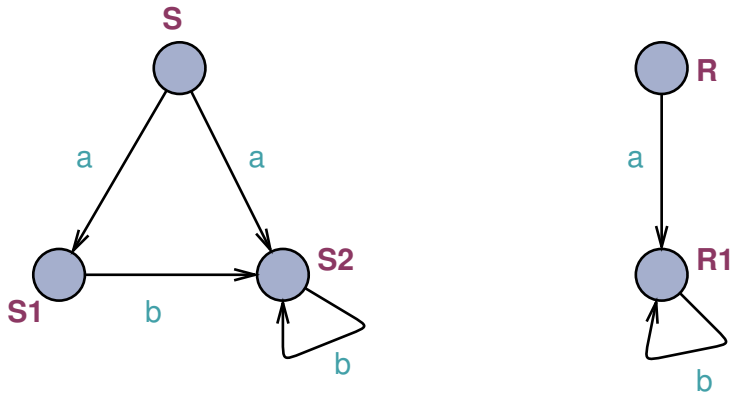


$\langle .a. \rangle \{S1, R1\} = \{S, R\}$

$[.a.]\{S1, R1\} = \{S1, S2, R, R1\}$
Pour $S1$ il est vrai que $\forall p' tq. S1 \xrightarrow{a} p' \Rightarrow p' \in \{S1, R1\}$

$< .b. > \{S1, R1\} =$

Exemple 1

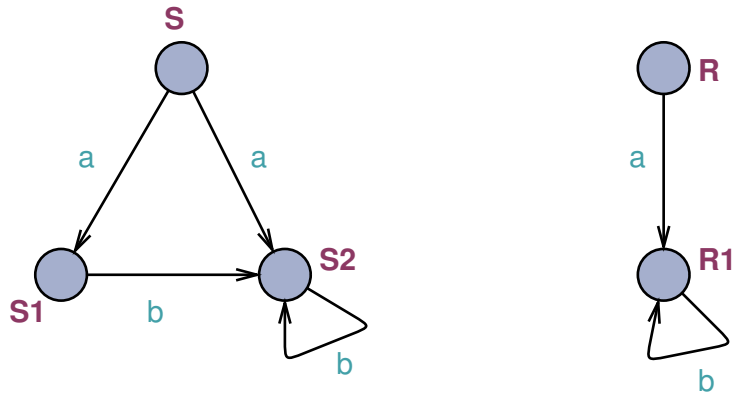


$\langle .a. \rangle \{S1, R1\} = \{S, R\}$

$[.a.]\{S1, R1\} = \{S1, S2, R, R1\}$
Pour $S1$ il est vrai que $\forall p' tq. S1 \xrightarrow{a} p' \Rightarrow p' \in \{S1, R1\}$

$< .b. > \{S1, R1\} = \{R1\}$
$[.b.]\{S1, R1\} =$

Exemple 1

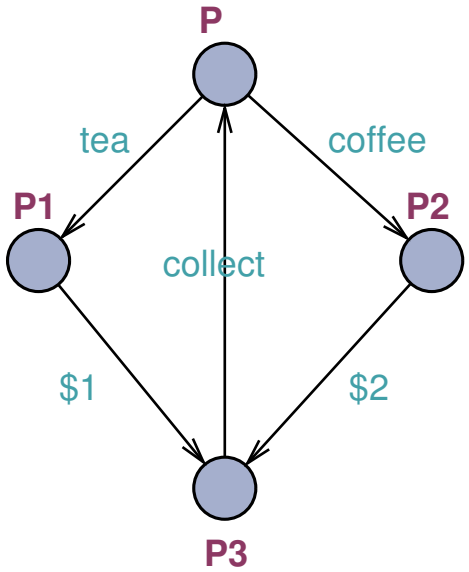


$\langle .a. \rangle \{S1, R1\} = \{S, R\}$

$[.a.]\{S1, R1\} = \{S1, S2, R, R1\}$
Pour $S1$ il est vrai que $\forall p' tq. S1 \xrightarrow{a} p' \Rightarrow p' \in \{S1, R1\}$

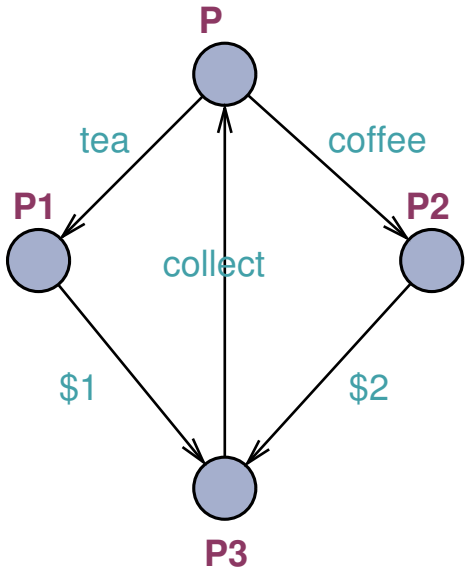
$< .b. > \{S1, R1\} = \{R1\}$
$[.b.]\{S1, R1\} = \{S, R, R1\}$

Exemple 2



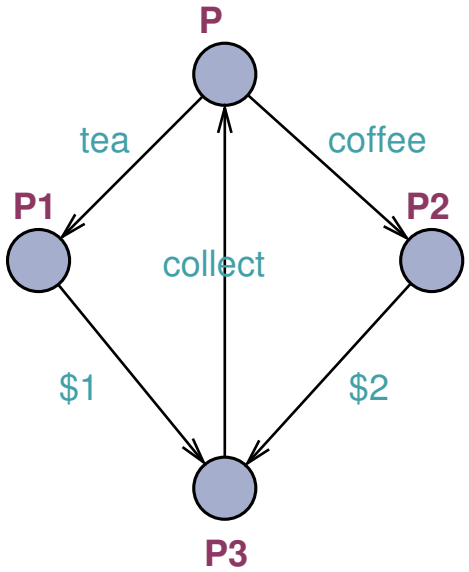
Comment formuler "Le client peut choisir un café" ?

Exemple 2



Comment formuler "Le client peut choisir un café" ?
Il peut faire une transition $\langle \text{coffee} \rangle$
donc la formule est $\langle \text{coffee} \rangle F$, mais c'est qui F ?

Exemple 2



Comment formuler "Le client peut choisir un café" ?
Il peut faire une transition $\langle \mathtt{coffee} \rangle$
donc la formule est $\langle \mathtt{coffee} \rangle F$, mais c'est qui F ?
($F = tt$) Il est *possible* de s'engager dans une action *coffee* et satisfaire tt
$[\![\langle \mathtt{coffee} \rangle tt]\!] = \langle .\mathtt{coffee}. \rangle [\![tt]\!] = \langle .\mathtt{coffee}. \rangle \mathtt{Proc} =$
$\{p \mid \exists p' \in \text{Proc tq. } p \xrightarrow{coffee} p'\} = \{P\}$

Exemple 2

Le client ne peut pas choisir du tea.

Exemple 2

Le client ne peut pas choisir du tea.

$[tea]ff$

Aucun processus satisfait ff donc les seuls processus qui marchent sont ceux qui n'ont pas de transition tea.

$[\![[\texttt{tea}]ff]\!] = [.\texttt{tea}.][\![ff]\!] = [.\texttt{tea}.]\emptyset = \{p \mid \texttt{si}\ p \xrightarrow{tea} p'\ \texttt{alors}\ p \in \emptyset\} = \{p | p \not\xrightarrow{tea}\} = \{P1, P2, P3\}$

Exemple 2

Que veut dire $[\text{coffee}]\langle \texttt{biscuit} \rangle tt$?

Comment écrire :
Le client est prêt pour choisir un tea ou un coffee

Exemple 2

Que veut dire [coffee]$\langle$`biscuit`$\rangle tt$?

Comment écrire :
Le client est prêt pour choisir un tea ou un coffee
$\langle$`coffee`$\rangle tt \vee \langle$`tea`$\rangle tt$

Le client est prêt pour choisir un tea, mais pas un coffee

Exemple 2

Que veut dire $[\texttt{coffee}]\langle\texttt{biscuit}\rangle tt$?

Comment écrire :
Le client est prêt pour choisir un tea ou un coffee
$\langle\texttt{coffee}\rangle tt \vee \langle\texttt{tea}\rangle tt$

Le client est prêt pour choisir un tea, mais pas un coffee
$[\texttt{coffee}]ff \wedge \langle\texttt{tea}\rangle tt$

Le client peut choisir un tea après avoir choisit deux coffee

Exemple 2

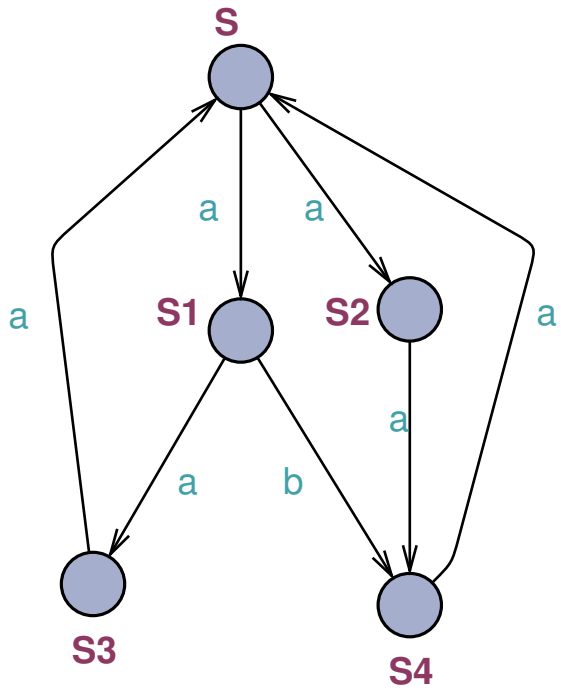
Que veut dire $[\texttt{coffee}]\langle\texttt{biscuit}\rangle tt$?

Comment écrire :
Le client est prêt pour choisir un tea ou un coffee
$\langle\texttt{coffee}\rangle tt \vee \langle\texttt{tea}\rangle tt$

Le client est prêt pour choisir un tea, mais pas un coffee
$[\texttt{coffee}]ff \wedge \langle\texttt{tea}\rangle tt$

Le client peut choisir un tea après avoir choisit deux coffee
$[\texttt{coffee}][\texttt{coffee}]\langle\texttt{tea}\rangle tt$

Exemple 3



On va écrire $p \models F \Leftrightarrow p \in [\![F]\!]$.

Est-ce que $s \models \langle a \rangle (\langle a \rangle tt \wedge \langle b \rangle tt)$?
$s \models [a][a]\langle b \rangle tt$?

La negation

Pour chaque formule $F \in \mathcal{M}$ il y a une formule équivalente à la négation de F, F^C définie par :

- $tt^C = ff$, $ff^C = tt$
- $(F \wedge G)^C = F^C \vee G^C$
- $(F \vee G)^C = F^C \wedge G^C$
- $(\langle a \rangle F)^C = [a]F^C$
- $([a]F)^C = \langle a \rangle F^C$

En particulier :
$(\langle a \rangle tt)^C = [a]ff$
$([a]ff)^C = \langle a \rangle tt$

Exemple 3

Trouver une formule qui soit satisfaite par $a.(b.0 + c.0)$, mais pas pour $a.b.0 + a.c.0$?

Exemple 3

Trouver une formule qui soit satisfaite par a.(b.0 + c.0), mais pas pour a.b.0 + a.c.0 ?
$[a](< b > tt \wedge < c > tt)$

Trouver une formule qui soit satisfaite par a.(b.c.0 + b.c.0), mais pas pour a.b.c.0 + a.b.c.0 ?

Théorème de Hennessy-Milner

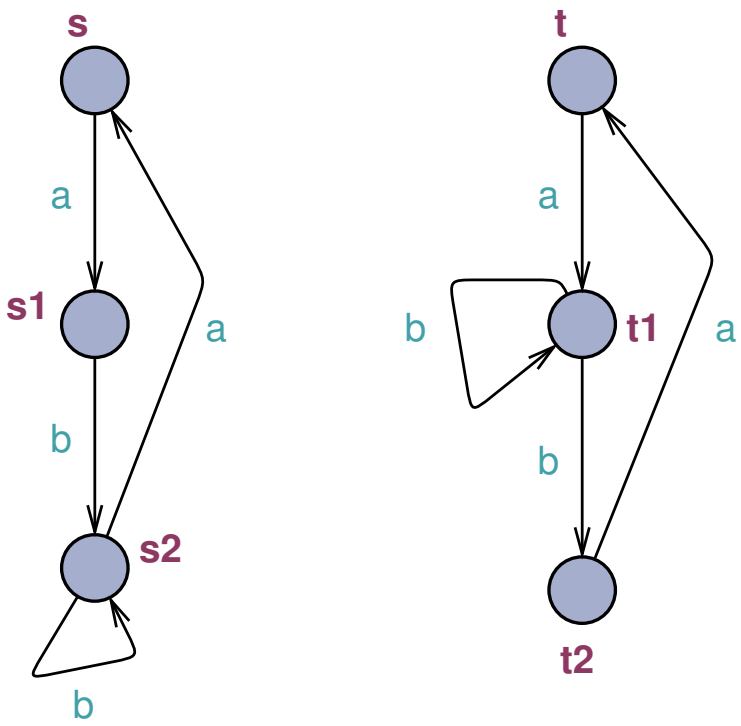
Définition : LTS à image finie

Un état est à image finie si pour toute a l'ensemble $\{P' \mid P \xrightarrow{a} P'\}$ est fini. Un LTS est à image finie si tous ses états le sont.

Théorème : Hennessy-Milner

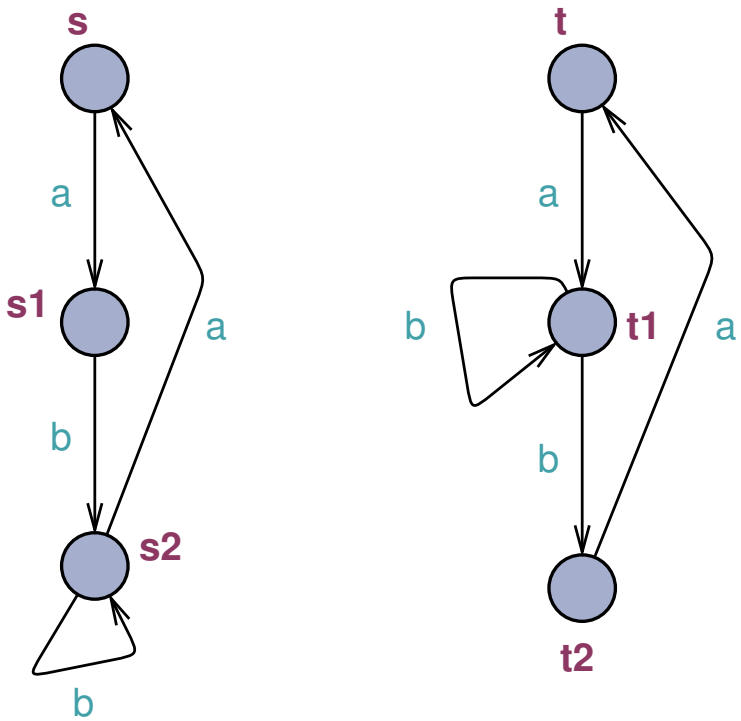
Soit (Proc, Act, $\{\xrightarrow{a} | a \in \text{Act}\}$) an LTS à image finie et P, Q $\in$ Proc. Alors P $\sim$ Q ssi P et Q satisfont exactement les mêmes formules dans la logique de Hennessy-Milner.

Excercice



Prouver que $s \not\sim t$

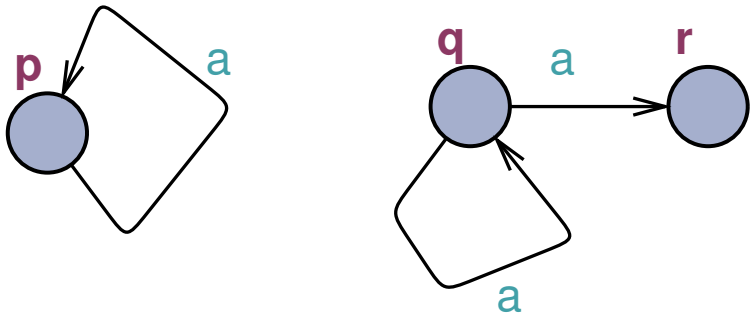
Excercice



Prouver que $s \not\sim t$

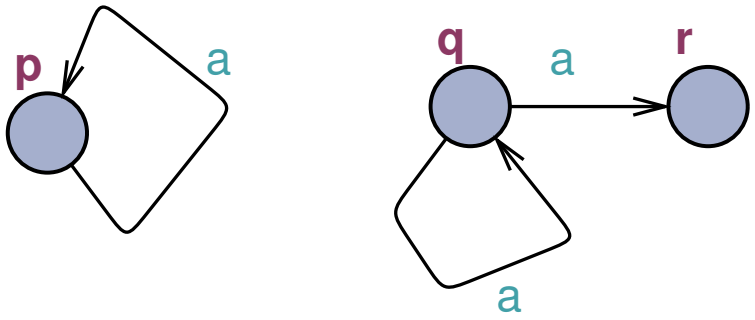
$[a][b](\langle a \rangle tt \wedge \langle b \rangle)$

Formules récursives



Après avoir fait a p peut toujours faire une autre a (ce n'est pas vrai pour q)
$p \models [a]\langle a\rangle tt$, mais $q \not\models [a]\langle a\rangle tt$

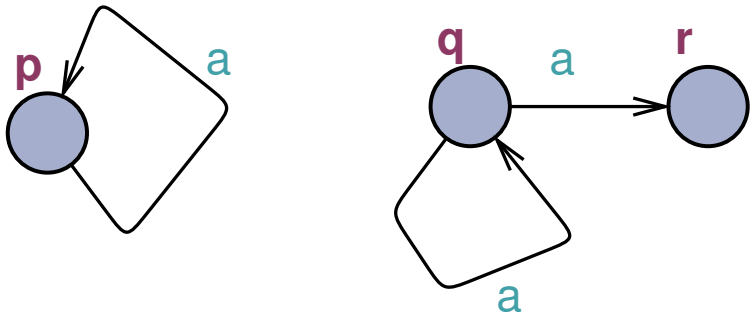
Formules récursives



Après avoir fait a p peut toujours faire une autre a (ce n'est pas vrai pour q)
$p \models [a]\langle a\rangle tt$, mais $q \not\models [a]\langle a\rangle tt$

On aimerait écrire les formules :
$Inv(\langle a\rangle tt) = \langle a\rangle tt \wedge [a]\langle a\rangle tt ... \wedge [a]^n\langle a\rangle tt \wedge ...$
Toujours on peut faire une a transition (Invariante)

Formules récursives

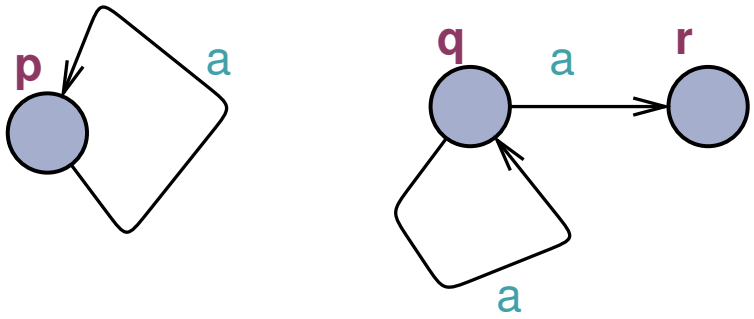


Après avoir fait a p peut toujours faire une autre a (ce n'est pas vrai pour q)
$p \models [a]\langle a\rangle tt$, mais $q \not\models [a]\langle a\rangle tt$

On aimerait écrire les formules :
$Inv(\langle a\rangle tt) = \langle a\rangle tt \wedge [a]\langle a\rangle tt ... \wedge [a]^n\langle a\rangle tt \wedge ...$
Toujours on peut faire une a transition (Invariante)

$Pos([a]ff) = [a]ff \vee \langle a\rangle[a]ff ... \vee \langle a\rangle^n[a]ff \vee ...$
On peut refuser a tout suite ou après un certain nombre de a (Proprieté)

Formules récursives



Après avoir fait a p peut toujours faire une autre a (ce n'est pas vrai pour q)
$p \models [a]\langle a\rangle tt$, mais $q \not\models [a]\langle a\rangle tt$

On aimerait écrire les formules :
$Inv(\langle a\rangle tt) = \langle a\rangle tt \wedge [a]\langle a\rangle tt ... \wedge [a]^n\langle a\rangle tt \wedge ...$
Toujours on peut faire une a transition (Invariante)

$Pos([a]ff) = [a]ff \vee \langle a\rangle [a]ff ... \vee \langle a\rangle^n [a]ff \vee ...$
On peut refuser a tout suite ou après un certain nombre de a (Proprieté)

$X \equiv \langle a\rangle tt \wedge [a]X$
$Y \equiv [a]ff \vee \langle a\rangle Y$

HML avec récursion

Le sense S d'une formule X par rapport à un LTS est donné par l'ensemble de processus dans P qui satisfont X.

- $X \equiv \langle a \rangle tt \wedge [a]X$
  $S = \langle .a. \rangle Proc \cap [.a.]S$
  $S = \emptyset$ est un solution, mais il y a d'autres ...
  $S = \{p\}$ c'est celle qu'on cherche, alors on va dire que :
  $X \equiv^{max} \langle a \rangle tt \wedge [a]X$

HML avec récursion

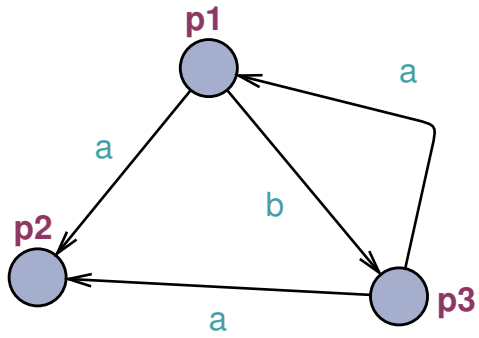
- $Y \equiv [a]ff \vee \langle a \rangle Y$
  $S = [.a.]\emptyset \cup \langle .a. \rangle S$
  il y a $\{p, q, r\}$ comme solution, mais nous on ne veut pas p
  $S = \{q, r\}$ c'est celle qu'on cherche, alors on va dire que :
  $Y \equiv^{min} [a]ff \vee \langle a \rangle Y$

Syntaxe et sémantique de $M_{\{X\}}$

Logique de Hennessy-Milner avec une variable X

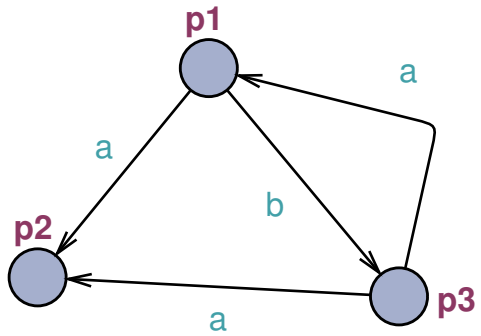
- $F, G ::= X | tt | ff | F \wedge G | F \vee G | <a>F | [a]F$
- Sémantiquement, une formule F(qui peut contenir une variable X) est interprétée comme une fonction
  $\mathcal{O}_F : \mathcal{P}(Proc) \rightarrow \mathcal{P}(Proc)$ que pour un ensemble de processus satisfaisant X retourne un ensemble de processus que satisfont F

Exemple



Soit $F = \langle a \rangle X$
$\mathcal{O}_{\langle a \rangle X}(\{p1\}) = \langle .a. \rangle \{p1\} = \{p3\}$
si X est satisfaite par p_1 alors $\langle a \rangle X$ sera satisfaite par p_3

Exemple



Soit $F = \langle a \rangle X$
$\mathcal{O}_{\langle a \rangle X}(\{p1\}) = \langle .a. \rangle \{p1\} = \{p3\}$
si X est satisfaite par p_1 alors $\langle a \rangle X$ sera satisfaite par p_3

$\mathcal{O}_{\langle a \rangle X}(\{p1, p2\}) = \langle .a. \rangle \{p1, p2\} = \{p1, p3\}$
si X est satisfaite par p_1 et p_2 alors $\langle a \rangle X$ sera satisfaite par p_1 et p_3

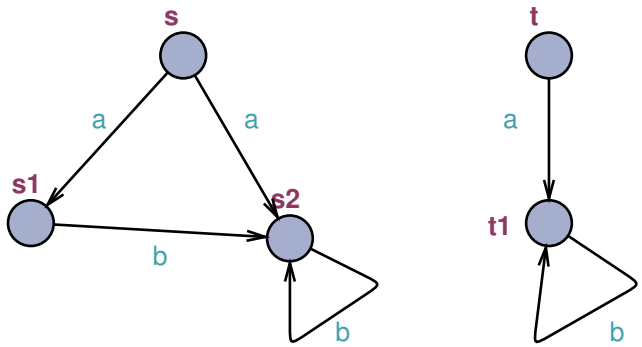
Sémantique de $M_{\{X\}}$

- $\mathcal{O}_X(S) = S$
- $\mathcal{O}_{tt}(S) = proc$
- $\mathcal{O}_{ff}(S) = \emptyset$
- $\mathcal{O}_{F_1 \wedge F_2}(S) = \mathcal{O}_{F_1}(S) \cap \mathcal{O}_{F_2}(S)$
- $\mathcal{O}_{F_1 \vee F_2}(S) = \mathcal{O}_{F_1}(S) \cup \mathcal{O}_{F_2}(S)$
- $\mathcal{O}_{\langle a \rangle F}(S) = \langle .a. \rangle \mathcal{O}_F(S)$
- $\mathcal{O}_{[a]F}(S) = [.a.]\mathcal{O}_F(S)$

Points fixes

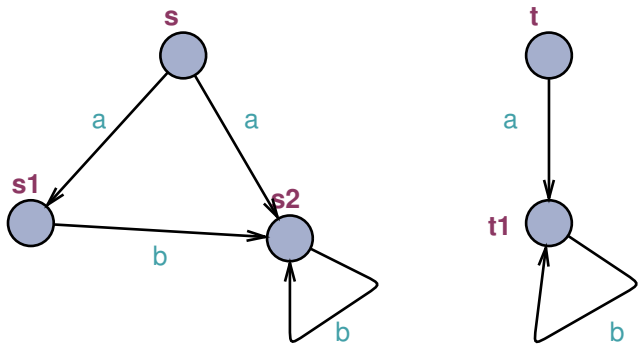
$\mathcal{O}_{F_X}$ est monotone et continue à droîte et on est sur un treille complet ($\top$ = Proc et $\bot = \emptyset$)
FIX $\mathcal{O}_{F_X} = (\mathcal{O}_{F_X})^M Proc$ et
fix $\mathcal{O}_{F_X} = (\mathcal{O}_{F_X})^m \emptyset$

Points fixes



$X \equiv^{max} \langle b \rangle tt \wedge [b]X$

Points fixes

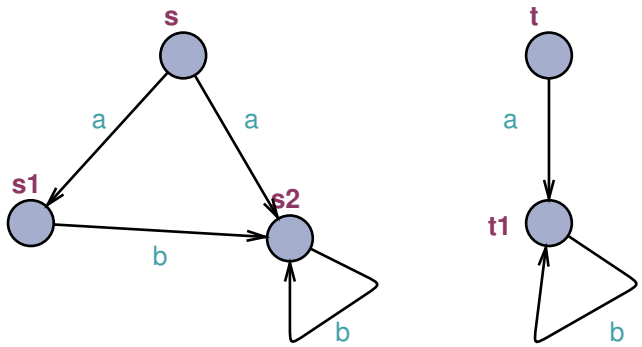


$X \equiv^{max} \langle b \rangle tt \wedge [b]X$

Pour trouver la solution on itère en partant de $\top$

$\mathcal{O}_{\langle b \rangle tt \wedge [b]X}(\{s, s1, s2, t, t1\}) = (\langle .b. \rangle \{s, s1, s2, t, t1\}) \cap ([.b.]\{s, s1, s2, t, t1\})$

Points fixes



$X \equiv^{max} \langle b \rangle tt \wedge [b]X$

Pour trouver la solution on itère en partant de $\top$

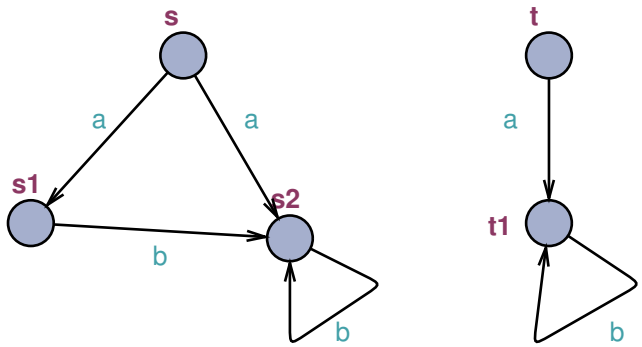
$\mathcal{O}_{\langle b \rangle tt \wedge [b]X}(\{s, s1, s2, t, t1\}) = (\langle .b. \rangle \{s, s1, s2, t, t1\}) \cap ([.b.]\{s, s1, s2, t, t1\})$

$= \{s2, s1, t1\} \cap \{s, s1, s2, t, t1\}$

$= \{s2, s1, t1\}$

$\mathcal{O}_{\langle b \rangle tt \wedge [b]X}(\{s2, s1, t1\}) = (\langle .b. \rangle \{s, s1, s2, t, t1\}) \cap ([.b.]\{s2, s1, t1\})$

Points fixes



$X \equiv^{max} \langle b \rangle tt \wedge [b]X$

Pour trouver la solution on itère en partant de $\top$

$\mathcal{O}_{\langle b \rangle tt \wedge [b]X}(\{s, s1, s2, t, t1\}) = (\langle .b. \rangle \{s, s1, s2, t, t1\}) \cap ([.b.]\{s, s1, s2, t, t1\})$

$= \{s2, s1, t1\} \cap \{s, s1, s2, t, t1\}$

$= \{s2, s1, t1\}$

$\mathcal{O}_{\langle b \rangle tt \wedge [b]X}(\{s2, s1, t1\}) = (\langle .b. \rangle \{s, s1, s2, t, t1\}) \cap ([.b.]\{s2, s1, t1\})$

$= \{s2, s1, t1\} \cap \{s, s1, s2, t, t1\}$

$= \{s2, s1, t1\}$