Sorbonne Université
Paradigmes de Programmation Concurrente MU5IN553

Cours 3 - Bisimulation et points fixes

Carlos Agon - Romain Demangeon

28 septembre 2024

Plan du cours

- Equivalence de trace
- Bisimulation forte
- Bisimulation faible
- Bisimulation comme un point fixe

Rappel : réactions et transitions

- $P = a.b|\overline{a}$
- $Q = (\nu c) \quad (a|\overline{a}.c|\overline{c})$
- $R = (\nu c) \quad (!c.a.\overline{c}|!c.b.\overline{c}|\overline{c})$

Equivalence entre processus

- $P = a.b|\overline{a}$ et $Q = \overline{a}|a.b$ (Equivalence structurelle)
- $P = a|b$ et $Q = a.b + b.a$ (Même LTS)
- $P = a.b.a.b.P$ et $Q = a.b.Q$ (Pas le même LTS, **même traces ?**)

Equivalence de traces

Trace forte (Strong trace)

Soit LTS(P) = $<\mathcal{Q},\mathcal{T}>$ alors
str(P) $=_{def} \{< \alpha_1, ..., \alpha_n > | \exists q_1, .., q_n \in \mathcal{Q} : P \xrightarrow{\alpha_1} q_1 \xrightarrow{\alpha_2} ... \xrightarrow{\alpha_n} q_n\}$

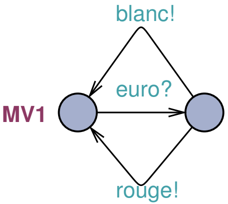
Trace faible (Weak trace)

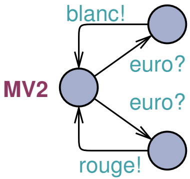
Soit LTS(P) = $<\mathcal{Q},\mathcal{T}>$ alors
tr(P)
$=_{def} \{< \alpha_1, ..., \alpha_n > | \exists q_1, .., q_n \in \mathcal{Q} : P \xrightarrow{\tau}^* \xrightarrow{\alpha_1} \xrightarrow{\tau}^* q_1 ... \xrightarrow{\tau}^* \xrightarrow{\alpha_n} \xrightarrow{\tau}^* q_n\}$

Equivalence de trace

Forte : $P =_{STR} Q$ ssi str(P) = str(Q)
Faible : $P =_{TR} Q$ ssi tr(P) = tr(Q)

Equivalence de traces

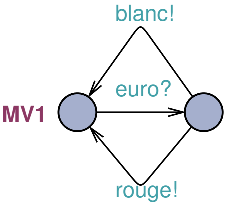


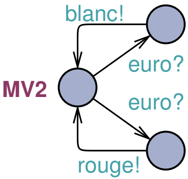


str(MV1) = tr(MV1) = {
<euro, blanc, euro, rouge, euro, rouge,...>,
<euro, blanc, euro, blanc, euro, blanc,...>...
}

str(MV2) = tr(MV2) ={
<euro, blanc, euro, rouge, euro, rouge,...>,
<euro, blanc, euro, blanc, euro, blanc,...>...
}

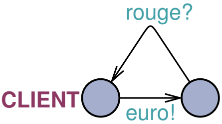
Equivalence de traces





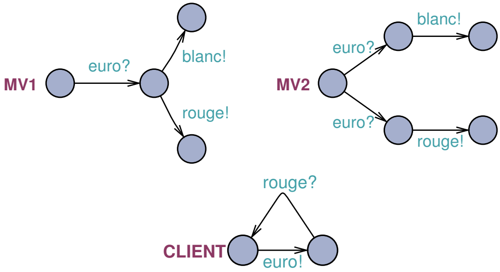
str(MV1) = tr(MV1) = {
<euro, blanc, euro, rouge, euro, rouge,...>,
<euro, blanc, euro, blanc, euro, blanc,...>...
}

str(MV2) = tr(MV2) ={
<euro, blanc, euro, rouge, euro, rouge,...>,
<euro, blanc, euro, blanc, euro, blanc,...>...
}



Equivalence de traces

(MV1 | MV2 | CLIENT)



Nous cherchons une équivalence qui distingue les *deadlock* potentiels d'un processus quand il interagit avec un autre.

Equivalence entre processus

- $P = a.b|\overline{a}$ et $Q = \overline{a}|a.b$ (Equivalence structurelle)
- $\subseteq$
- $P = a|b$ et $Q = a.b + b.a$ (Même LTS)
- $\subseteq$
- $P = a.b.a.b.P$ et $Q = a.b.Q$ (Pas le même LTS, mais même comportement)
- $\subseteq$
- $P = a.b + a.c$ et $Q = a.(b + c)$ (même traces **MAIS...**, ça marche pas en tout contexte)

Congruence de processus

Définition context

Un contexte de processus $\mathcal{C}$ est une expression avec un trou [ ].
$\mathcal{C} ::= [\,] \;\mid\; \alpha.\mathcal{C} + \mathcal{M} \;\mid\; (\nu a)\mathcal{C} \;\mid\; \mathcal{C}|P \;\mid\; P|\mathcal{C}$
Les contextes élémentaires sont :
$\alpha.[\,] + \mathcal{M}$, (νa) [ ], [ ] | P et P | [ ]
$\mathcal{C}[Q]$ dénote le fait de remplir le contexte $\mathcal{C}$ avec le processus Q.

Définition Congruence de processus

Soit $\cong$ une relation d'équivalence sur $\mathcal{P}$ alors $\cong$ est une congruence de processus si elle est preservée par tous les contextes élémentaires,
c-à-d si P $\cong$ Q alors :

$$\alpha.P + M \cong \alpha.Q + M$$

$$(\nu a)P \cong (\nu a)Q$$

$$P \mid R \cong Q \mid R$$

$$R \mid Q \cong R \mid Q$$

$\cong$ arbitraire est une congruence de processus ssi $\forall\mathcal{C}\ P \cong Q \Rightarrow \mathcal{C}[P] \cong \mathcal{C}[Q]$

Equivalence entre processus

- $P = a.b|\overline{a}$ et $Q = \overline{a}|a.b$ (Equivalence structurelle)
- $\subseteq$
- $P = a|b$ et $Q = a.b + b.a$ (Même LTS)
- $\subseteq$
- $P = a.b.a.b.P$ et $Q = a.b.Q$ (Pas le même LTS, mais même comportement)
- $\subseteq$
- **Bisimulation**
- $\subseteq$
- $P = a.b + a.c$ et $Q = a.(b + c)$ (même traces **MAIS...**, ça marche pas en tout contexte)

Simulation forte

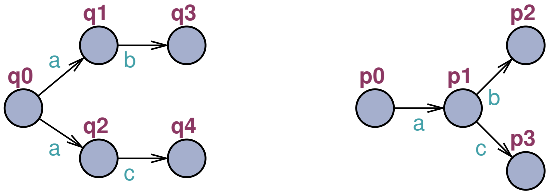
Pour que 2 processus soient équivalents, il doivent avoir les mêmes traces, et les états qu'ils atteignent doivent aussi être équivalents.

Définition simulation forte

Soit $\mathcal{S}$ une relation sur l'ensemble des états d'un LTS $= (\mathcal{Q}, \mathcal{T})$, $\mathcal{S}$ est une simulation forte ssi à chaque fois que $p\mathcal{S}q$:
si $p \xrightarrow{\alpha} p'$ alors il existe $q' \in \mathcal{Q}$ tq. $q \xrightarrow{\alpha} q'$ et. $p'\mathcal{S}q'$

- on dit que q simule fortement p s'il existe une simulation $\mathcal{S}$ contenant (p,q) (i.e. p$\mathcal{S}$q)
- q et p peuvent appartenir à deux LTS differents, cela ne change pas la définition.

Simulation forte



p_0 simule fortement q_0 car $(q_0, p_0) \in R$ avec
$\mathrm{R} = \{(q_0, p_0), (q_1, p_1), (q_2, p_1), (q_3, p_2), (q_4, p_3)\}$

Cette relation n'est pas trop réflexive...

Bisimulation forte (strong bisimulation)

Une relation $\mathcal{R}$ est une bisimulation si $\mathcal{R}$ et $\mathcal{R}^{-1}$ sont des simulations.

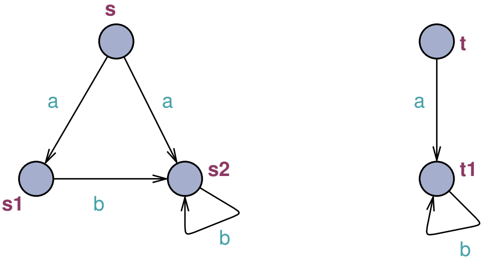
Définition : bisimulation forte

Soit $\mathcal{R}$ une relation sur l'ensemble des états d'un LTS, $\mathcal{R}$ est une bisimulation forte ssi à chaque fois que $s_1 \mathcal{R} s_2$:

- si $s_1 \xrightarrow{\alpha} s_1'$ alors il existe une transition $s_2 \xrightarrow{\alpha} s_2'$ tq. $s_1' \mathcal{R} s_2'$
- si $s_2 \xrightarrow{\alpha} s_2'$ alors il existe une transition $s_1 \xrightarrow{\alpha} s_1'$ tq. $s_1' \mathcal{R} s_2'$

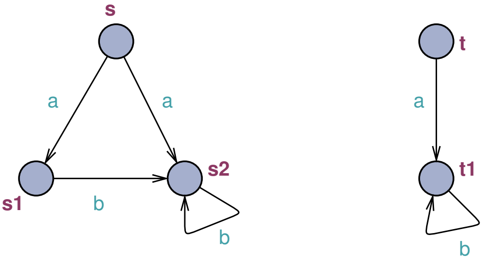
Deux états p et q sont bisimilaires ($p \sim q$) ssi il existe une bisimulation forte $\mathcal{R}$ tq. $p \mathcal{R} q$.

Exemple 1



Prouver que $s \sim t$

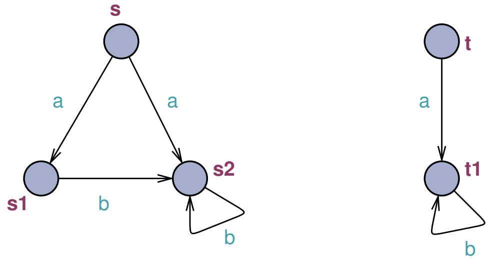
Exemple 1



Prouver que $s \sim$ t
$\mathcal{R} = \{(s,t),(s_1,t_1),(s_2,t_1)\}$

Egalement $s1 \sim$ s2

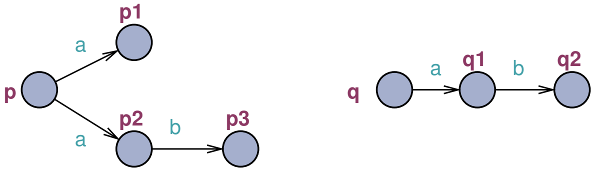
Exemple 1



Prouver que $s \sim$ t
$\mathcal{R} = \{(s, t), (s_1, t_1), (s_2, t_1)\}$

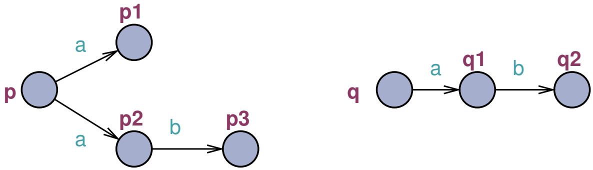
Egalement $s1 \sim$ s2
$\mathcal{R} = \{(s_1, s_2), (s_2, s_2)\}$

Exemple 2



p simule fortement q et q simule fortement p, alors $p \sim q$?

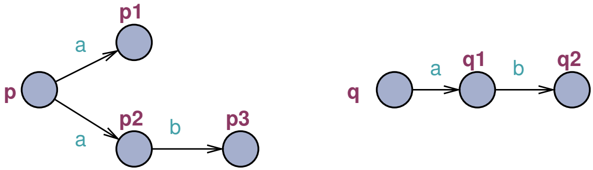
Exemple 2



p simule fortement q et q simule fortement p, alors $p \sim q$?

$S1 = \{(q, p), (q1, p2), (q2, p3)\}$
$S2 = \{(p, q), (p1, q1), (p2, q1), (p3, q2)\}$

Exemple 2



p simule fortement q et q simule fortement p, alors $p \sim q$?

$S1 = \{(q, p), (q1, p2), (q2, p3)\}$
$S2 = \{(p, q), (p1, q1), (p2, q1), (p3, q2)\}$

mais $S1 \neq S2^{-1}$, alors $p \not\sim q$

Exemple 3

$$\mathcal{Q} = \{s_i | i \geqslant 1\} \cup \{t\}$$
$$\xrightarrow{a} = \{(s_i, s_{i+1}) | i \geqslant 1\} \cup \{(t, t)\}$$

Prouver que $s_1 \sim$ t

Exemple 3

$$\mathcal{Q} = \{s_i | i \geqslant 1\} \cup \{t\}$$
$$\xrightarrow{a} = \{(s_i, s_{i+1}) | i \geqslant 1\} \cup \{(t, t)\}$$

Prouver que $s_1 \sim$ t

$$\mathcal{R} = \{(s_i, t) | i \geqslant 1\}$$

$\sim$ est une relation d'équivalence

Soit LTS $= (\mathcal{Q}, \mathcal{T})$

- Réflexive : $p \sim p$
  $\mathcal{I} : \{(p, p) | p \in \mathcal{Q}\}$
- Symétrique $p \sim q$ alors $q \sim p$
  Par définition de bisimulation
- Transitive : $p \sim q$ et $q \sim r$ alors $p \sim r$
  Soit S_1 et S_2 deux bisimulations avec $(p, q) \in S_1$ et $(q, r) \in S_2$
  $S = \{(s_1, s_3) | \exists s_2((s_1, s_2) \in S_1 \wedge (s_2, s_3) \in S_2)\}$
  $(p, r) \in S$ et S est une bisimulation

Jeu de caractérisation de bisimilarité

Pour montrer que $s \sim t$, on trouve un $\mathcal{R}$. Mais pour prouver que $s \not\sim t$ comment faire ? Deux joueurs : l'attaquant qui essaie de montrer $s \not\sim t$ et le défenseur qui essaie de montrer le contraire.

Strong bisimulation game

Soit un LTS (Proc, Act, $\{\xrightarrow{\alpha}\ \alpha \in \mathrm{Act}\}$)

1. Le jeu commence par une paire d'états $(s, t) \in \mathcal{Q} \times \mathcal{Q}$, appelée la configuration courante
2. L'attaquant choisit soit s soit t et une action α
   - S'il choisit gauche alors il fait la transition $s \xrightarrow{\alpha} s'$ pour un $s' \in \mathcal{Q}$
   - S'il choisit droite alors il fait la transition $t \xrightarrow{\alpha} t'$ pour un $t' \in \mathcal{Q}$
3. Le défenseur doit répondre à l'attaquant
   - Si l'attaquant a choisi gauche alors il doit jouer à droite avec $t \xrightarrow{\alpha} t'$ pour un $t' \in \mathcal{Q}$
   - Si l'attaquant a choisi droite alors il il doit jouer à gauche avec $s \xrightarrow{\alpha} s'$ pour un $t' \in$,
4. (s', t') devient la configuration courante et le jeu continue.

Jeu de caractérisation de bisimilarité

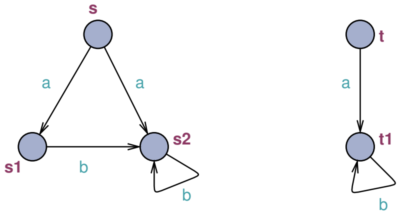
Le jeu est perdu quand un joueur est coincé, il ne peut pas choisir un état en suivant les règles.

1. L'attaquant perd si $s \not\rightarrow$ et $t \not\rightarrow$
2. Le défenseur perd s'il ne trouve pas une transition pour répondre
3. Si jamais ils sont bloqués (le jeu est infini) c'est le défenseur qui gagne

$s1 \not\sim s2$ si l'attaquant a une stratégie gagnante (i.e. indépendante de la manière de jouer du défenseur)

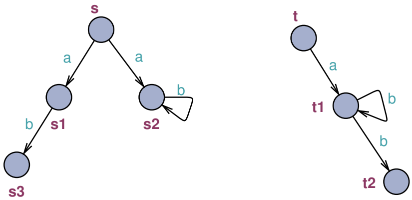
Exemple $s \sim t$

Le défenseur a une stratégie gagnante



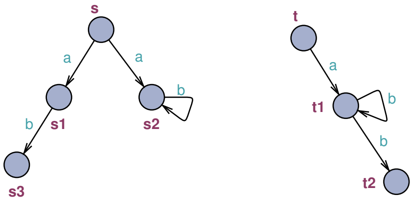
Exemple $s \not\sim t$

L'attaquant a une stratégie gagnante



Exemple $s \not\sim t$

L'attaquant a une stratégie gagnante



$s \xrightarrow{a} s1$, $t1 \xrightarrow{b} t1$ et $t1 \xrightarrow{b} t1$

Bisimulation faible

Soient P et Q $\in \mathcal{P}$, P $\stackrel{\alpha}{\Rightarrow}$ Q ssi

1. si $\alpha \neq \tau$ et il existe P' et Q' tq. $P(\xrightarrow{\tau})^* P' \xrightarrow{\alpha} Q'(\xrightarrow{\tau})^* Q$
2. si $\alpha = \tau$ et $P(\xrightarrow{\tau})^* Q$

Définition bisimulation faible

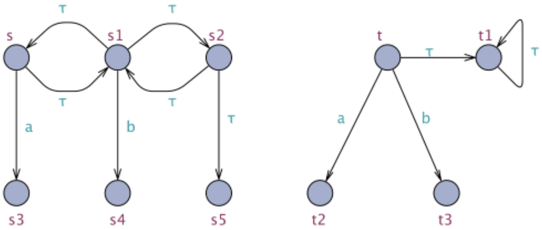
Soit $\mathcal{R}$ une relation sur l'ensemble des états d'un LST, $\mathcal{R}$ est une bisimulation faible ssi à chaque fois que $s_1 \mathcal{R} s_2$:
si $s_1 \xrightarrow{\alpha} s_1'$ alors il existe une transition $s_2 \stackrel{\alpha}{\Rightarrow} s_2'$ tq. $s_1' \mathcal{R} s_2'$
si $s_2 \xrightarrow{\alpha} s_2'$ alors il existe une transition $s_1 \stackrel{\alpha}{\Rightarrow} s_1'$ tq. $s_1' \mathcal{R} s_2'$

Deux états s et s' sont faiblement bisimilaires ou observationellement équivalents (s $\approx$ s') ssi il y a une bisimulation faible qui les met en relation.

$$s \xrightarrow{\tau} s_1 \xrightarrow{a} s_2 \qquad t \xrightarrow{a} t_1$$

s $\not\sim$ t mais s $\approx t$ avec $\mathcal{R} = \{(s,t),(s_1,t),(s_2,t_1)\}$
Par contre, si $s \sim t$ alors $s \approx t$

Exemple



$\approx$ autres propriétés

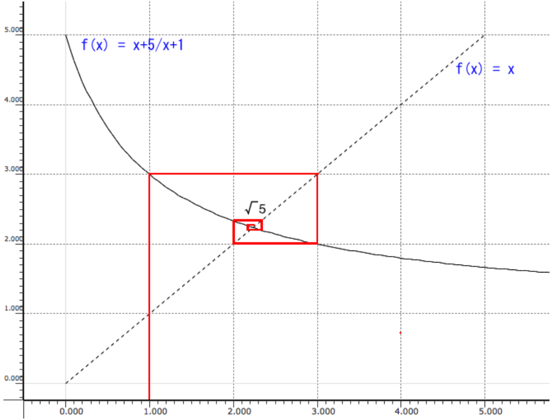
Si $P \sim Q$ alors $P \approx Q$

$\approx$ est une relation d'équivalence

$\approx$ est elle même une bisimulation faible

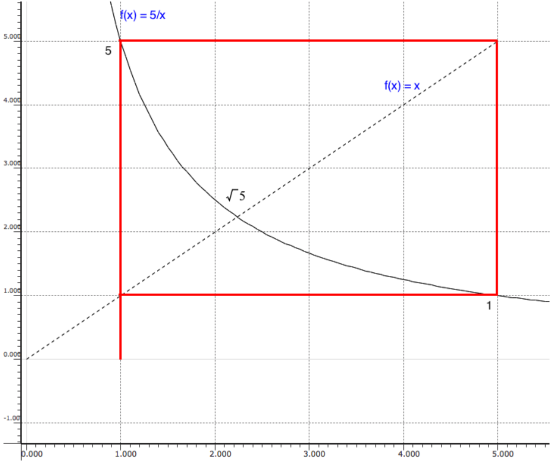
Bisimulation comme un point fixe

Reformuler la notion de bisimulation forte grâce à un théorème sur le point fixe
On gagne un algorithme pour calculer la bisimulation forte pour un LTS avec transitions, états et actions finies.



$u_0 = 1$ et $u_{n+1} = f(u_n)$, la suite 1,3,2,7/3,11/5,.... converge vers $\sqrt{5}$

Points fixes



$u_0 = 1$ et $u_{n+1} = f(u_n)$, la suite 5,1,5,1,.... ne converge pas

Posets et treillis complets

Définition : ensemble partiellement ordonné

Un poset est une paire $(D, \sqsubseteq)$ avec D un ensemble et $\sqsubseteq$ une relation (i.e. $\subseteq D \times D$) tq.

1. réflexive
2. antisymétrique
3. transitive

Définition : ensemble totalement ordonné

$(D, \sqsubseteq)$ est totalement ordonné si en plus $\forall d, e \in D \quad (d \sqsubseteq e \vee e \sqsubseteq d)$

$(\mathcal{P}(P), \subseteq)$ est un poset ? Est-il totalement ordonné ?

Least Upper Bounds (lup) et Greatest Lower Bounds (glb)

Définition : suprême (lup)

Soit $(D, \sqsubseteq)$ un poset et $X \subseteq D$, d est un upper bound (supérieur) de X ssi $x \sqsubseteq d$ pour tout $x \in X$.
d est le least upper bound (suprême) de X ($\sqcup X$) ssi

1. d est un upper bound
2. $d \sqsubseteq d'$ pour tout d' upper bound X

Définition : infime (glb)

Soit $(D, \sqsubseteq)$ un poset et $X \subseteq D$, d est un lower bound (inférieur) de X ssi $d \sqsubseteq x$ pour tout $x \in X$.
d est le greatest lower bound (infime) de X ($\sqcap X$) ssi

1. d est un lower bound
2. $d' \sqsubseteq d$ pour tout d' lower bound X

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 $(\mathcal{P}(P), \subseteq)$ $\quad \bot = \emptyset$ 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 $d = f(d)$

Exemple

Soit $f : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ définie pour tout $X \subseteq \mathbb{N}$ par :
$f(X) = X \cup \{1, 2\}$ alors f est monotone et son point fixe est ...

Exemple

Soit $f : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ définie pour tout $X \subseteq \mathbb{N}$ par :
$f(X) = X \cup \{1,2\}$ alors f est monotone et son point fixe est ...
$f(\{1,2\}) = \{1,2\} \cup \{1,2\} = \{1,2\}$
Il y a un autre point fixe ?

Exemple

Soit $f : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ définie pour tout $X \subseteq \mathbb{N}$ par :
$f(X) = X \cup \{1,2\}$ alors f est monotone et son point fixe est ...
$f(\{1,2\}) = \{1,2\} \cup \{1,2\} = \{1,2\}$
Il y a un autre point fixe ?
$f(\mathbb{N}) = \mathbb{N} \cup \{1,2\} = \mathbb{N}$

Théorème 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. $z_{min} = \sqcap\{x \in D | f(x) \sqsubseteq x\}$

Pour $(\mathcal{P}(\mathbb{N}), \subseteq)$ et une $f : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ définie par $f(X) = X \cup \{1, 2\}$ on a :

$z_{max} = \sqcup\{X \subseteq \mathbb{N} | X \subseteq X \cup \{1, 2\}\} = \mathbb{N}$
$z_{min} = \sqcap\{X \subseteq \mathbb{N} | X \cup \{1, 2\} \subseteq X\} = \{1, 2\}$

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. $z_{min} = f^m(\perp)$ pour $m \in \mathbb{N}$ si la suite croissante $f^n(\perp)$ 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.

Soit g : $\mathcal{P}(\{0, 1, 2\}) \to \mathcal{P}(\{0, 1, 2\})$ définie par $g(X) = (X \cap \{1\}) \cup \{2\}$
Calculer z_{min} et z_{max}...

Bisimulation comme un point fixe

Définition : bisimilarité forte

On va définir $\sim$ comme un point fixe d'une fonction monotone.
Notez que $(\mathcal{P}(Q \times Q), \subseteq)$ est un treille complet avec *lup* et *glb*
On définit l'ensemble $\mathcal{F}(\mathcal{R})$ ainsi :
$(p, q) \in \mathcal{F}(\mathcal{R})$ $\forall p, q \in Q$ ssi

1. $p \xrightarrow{\alpha} p'$ implique qu'il existe $q \xrightarrow{\alpha} q'$ tq. $(p', q') \in \mathcal{R}$
2. $q \xrightarrow{\alpha} q'$ implique qu'il existe $p \xrightarrow{\alpha} p'$ tq. $(p', q') \in \mathcal{R}$

$$\sim = \bigcup \{\mathcal{R} \in \mathcal{P}(Q \times Q) \mid \mathcal{R} \subseteq \mathcal{F}(\mathcal{R})\}$$

Algorithme pour calculer $\sim$ comme un point fixe

L'algorithme n'est pas le plus rapide...

On sait que $\mathcal{R} \subseteq \mathcal{S}$ alors $\mathcal{F}(\mathcal{R}) \subseteq \mathcal{F}(\mathcal{S})$

Alors par le théorème de Knaster-Tarski 2, on a :
$\sim = \mathcal{F}^M(Q \times Q)$

Exemple

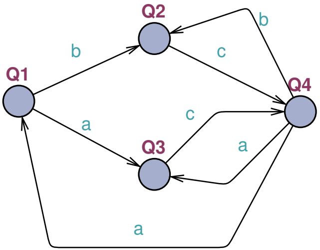
Trouver $\sim$ pour :

$$Q_1 = b.Q_2 + a.Q_3$$
$$Q_2 = c.Q_4$$
$$Q_3 = c.Q_4$$
$$Q_4 = b.Q_2 + a.Q_3 + a.Q_1$$

Exemple

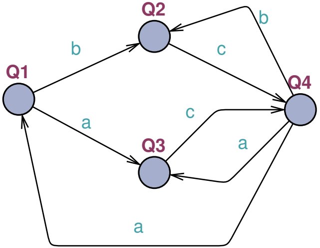
Trouver $\sim$ pour :

$$Q_1 = b.Q_2 + a.Q_3$$
$$Q_2 = c.Q_4$$
$$Q_3 = c.Q_4$$
$$Q_4 = b.Q_2 + a.Q_3 + a.Q_1$$



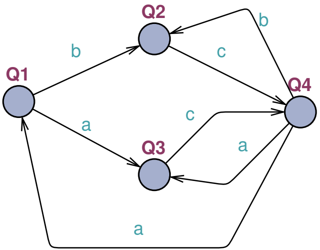
Exemple

$\top = \{(Q_1, Q_2), (Q_2, Q_1), (Q_1, Q_3), (Q_3, Q_1), (Q_1, Q_4), (Q_4, Q_1),$
$(Q_2, Q_3), (Q_3, Q_2), (Q_2, Q_4), (Q_4, Q_2), (Q_3, Q_4), (Q_4, Q_3), (Q_i, Q_i)\}$



Exemple

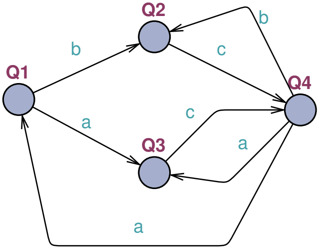
$\top = \{(Q_1, Q_2), (Q_2, Q_1), (Q_1, Q_3), (Q_3, Q_1), (Q_1, Q_4), (Q_4, Q_1),$
$(Q_2, Q_3), (Q_3, Q_2), (Q_2, Q_4), (Q_4, Q_2), (Q_3, Q_4), (Q_4, Q_3), (Q_i, Q_i)\}$



$\mathcal{F}^0(\top) = \{(Q_1, Q_4), (Q_4, Q_1), (Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$

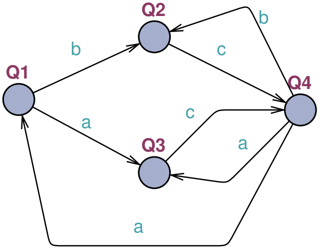
Exemple

$$\mathcal{F}^0(\top) = \{(Q_1, Q_4), (Q_4, Q_1), (Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$$



Exemple

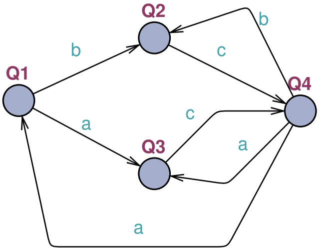
$$\mathcal{F}^0(\top) = \{(Q_1, Q_4), (Q_4, Q_1), (Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$$



$$\mathcal{F}^1(\mathcal{F}^0(\top)) = \{(Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$$

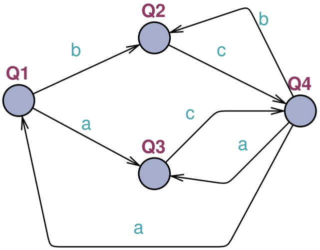
Exemple

$$\mathcal{F}^1(\mathcal{F}^0(\top)) = \{(Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$$



Exemple

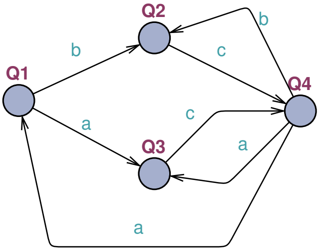
$$\mathcal{F}^1(\mathcal{F}^0(\top)) = \{(Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$$



$$\mathcal{F}^2(\mathcal{F}^1(\mathcal{F}^0(\top))) = \{(Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$$

Exemple

$$\mathcal{F}^1(\mathcal{F}^0(\top)) = \{(Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$$



$$\mathcal{F}^2(\mathcal{F}^1(\mathcal{F}^0(\top))) = \{(Q_2, Q_3), (Q_3, Q_2), (Q_i, Q_i)\}$$

alors on peut dire que $Q_2 \sim Q_3$

$\sim$ est une congruence

Si $P \sim Q$ alors on peut remplacer P par Q dans n'importe quel contexte.

Si $P \sim Q$ alors :

1. $\alpha.P + M \sim \alpha.Q + M$
2. $(\nu a)P \sim (\nu a)Q$
3. $P|Q \sim Q|P$
4. $Q|P \sim P|Q$

Références

- Anne Dicky
  Cours "Points fixes de fonctions monotones"
  Master d'informatique Universite de Bordeaux 1
- Luca Aceto et *al.*
  "Reactive Systems" Modeling, specification and verification
- Robin Milner
  "Communicating and mobile systems : the π-calculus"