INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Vérification de Protocoles

François Vernadat
INSA-DGEI

vernadat@laas.fr

Université de Toulouse

Outline

1. Introduction
   - Une définition du Model-Checking
   - Classe de propriétés visée
   - Model-Checking : les principes
   - Panorama du cours & Organisation/Evaluation

Introduction

What is model checking ? – Amir Pnueli 2000

"Model checking is the method by which a desired behavioral property of a reactive system is verified over a given system (the model) through exhaustive enumeration (explicit or implicit) of all the states reachable by the system and the behaviors that traverse through them."

Programme "Classique"

- Termine
- Retourne un résultat
- Données +/- complexes mais contrôle séquentiel ("simple")

Ex : Algo de tri, compilateur, , ...

Correction = correction partielle + terminaison

Système Réactif (> 70)

- Interactions entre Σ et son envt
- Ne retourne pas de résultat
- Terminaison en général indésirée (deadlock)
- Données +/- simples mais contrôle concurrent ("compliqué")

Ex : Protocoles, Contrôle/Commande, Σ-Embarqués, ...

Correction = sûreté, vivacité, équité, ...

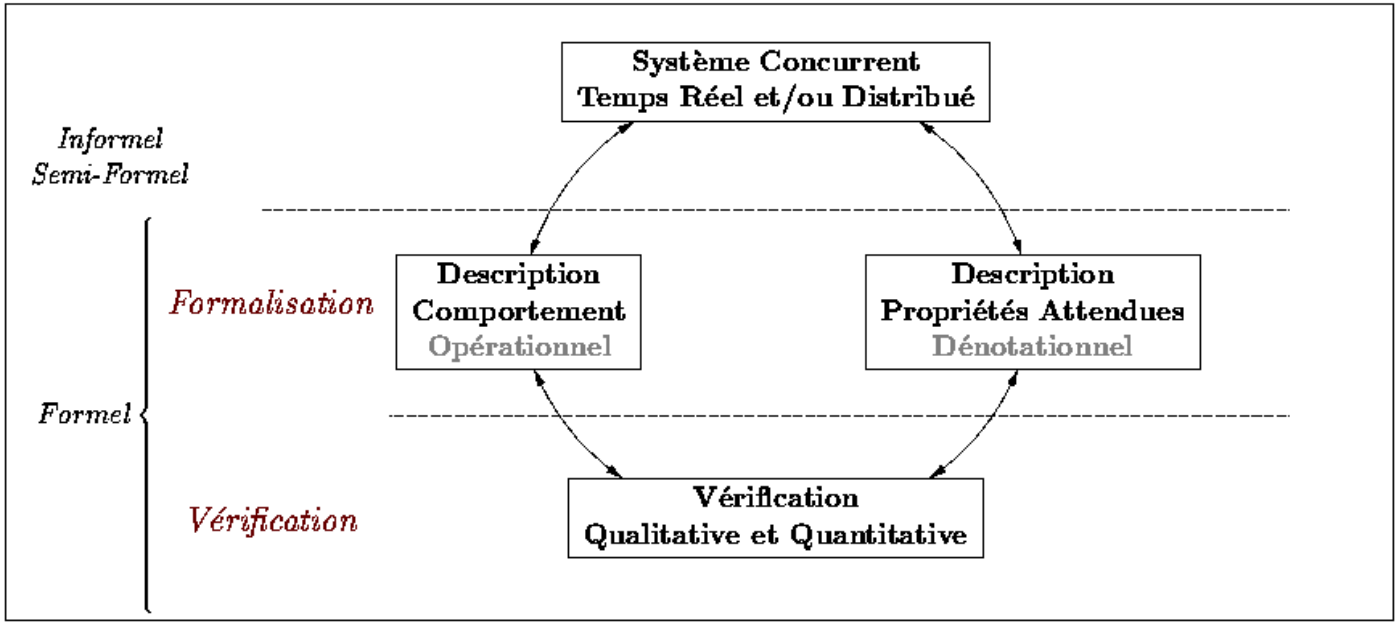
Classe de propriétés des systèmes réactifs

- **Accessibilité**
  - Une certaine situation peut être atteinte
  - Chaque action peut avoir lieu
- **Invariant**
  - Propriété vraie en chaque état
    La barrière est fermée lorsqu'un train est sur le passage à niveau
- **Sûreté**
  - Quelque chose de mauvais n'arrive jamais
    Jamais plus de 2 processus en section critique
    Jamais de débordement de buffers
- **Vivacité**
  - Quelque chose de bon finit par arriver
    Un philosophe affamé finira par manger
- **Equité(s)**
  - Si une action a lieu une infinité de fois alors une autre action aura lieu une (infinité) de fois
  - Quelque chose (de bon) se répète infiniment souvent
    S'il n'y a pas une infinité de pertes, tout message émis - perdu et retransmis - finira par arriver (Alternating Bit Protocol)

Model-Checking

Ingrédients

$\mathcal{M}$: Modèle (formel) du comportement (**issu de** Rdp, Automates, Alg.Proc,..)
ϕ : Modèle (formel) des propriétés (formalisme dédié)
MC : Algorithme de vérification permettant de décider si $\mathcal{M}$ satisfait ϕ?
nb : Formel est une condition requise pour envisager l'automaticité



Assertionnel

Propriété ϕ exprimée dans une logique ad hoc
$\mathcal{M} \models \phi$ (cours IDM)

Comportemental

Propriété ϕ exprimée par un "graphe" G_ϕ
$\mathcal{M} \equiv G_\phi$ (cours MRP)

Exemple de langage d'expression des propriétés

Rappel : Query UPPAAL

Rappels L_{Uppaal}

Syntaxe

$$Prop ::= \ 'A[]' \ Expression \mid 'E <>' \ Expression \mid 'E[]' \ Expression$$
$$\mid 'A <>' \ Expression \mid Expression \ ' - - >' \ Expression$$

Pas d'imbrication de formules !

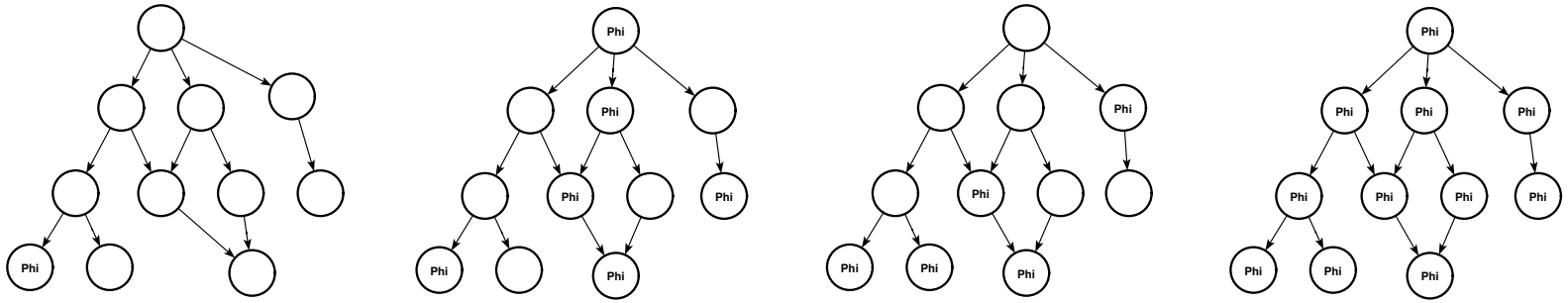
Sémantique : double quantification chemins/états

A/E	Chemins	$(\forall/\exists)$
$[]/<>$	Etats	$(\forall/\exists)$

$\phi - - > \psi \equiv [](\phi \Rightarrow \Diamond\psi)$

Evaluation à la racine

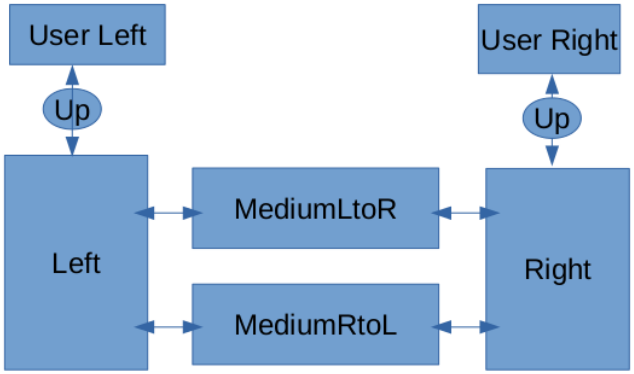
$E\Diamond\phi$ $E[]\phi$ $A\Diamond\phi$ $A[]\phi$



Expression des propriétés par un graphe

Rappel : TP Système concurrent Ouverture/Fermeture de connexion

Architecture



Ouverture

initiation : L'usager demande l'ouverture de la communication (side(sup) ? appel), il peut recevoir un accord (side(sup) ! ok), ou un refus (side(sup) ! nok). En cas d'accord, la connexion est ouverte, en cas de refus la tentative a échoué.

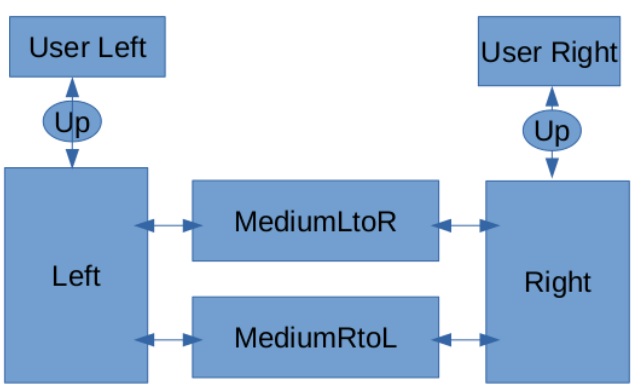
réponse : L'usager peut recevoir une demande de communication (side(sup) ! dring), il peut l'accepter (side(sup) ? accept) ou la refuser (side(sup) ? reject). Après avoir accepté, il considère la connexion ouverte.

Fermeture :

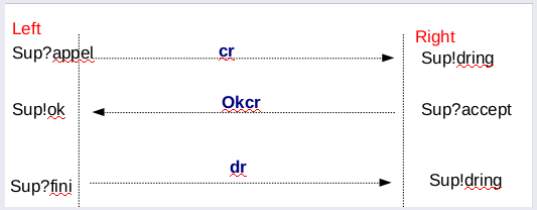
Un usager connecté, ou ayant initié une connexion peut l'interrompre (side(sup) ? fini). Dans ce cas, l'autre usager, s'il est connecté, doit recevoir une indication de déconnexion : (side(sup) ! bip)

Rappel TP Ouverture/Fermeture de connexion

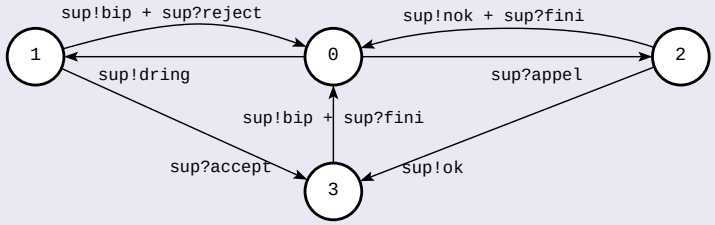
Architecture



Quelques Scénarios



Expression du service local **attendu** par un automate



Service : Langage (Ensemble infini) des séquences de messages échangés par deux (ou plus) entités de protocole.

local : seuls les messages transitant entre l'entité et son usager (ceux transitant sur up) sont considérés.

Service Attendu Vs Service Offert

Tp Systèmes concurrents de 4I

Vu (ou entrevu) comment on pouvait formaliser la notion de service par un automate pour **spécifier** les propriétés **attendues** du protocole

$\mapsto$ Langage / Automate (accepteur de langages)

Cours de vérification

- Voir comment vérifier que le service **offert** correspond bien au service **attendu**
- Envisager d'autres approches que l'équivalence langage
- Esquisser le lien avec la vérification assertionnelle
- Equivalence Vs congruence (cf opérations de composition des Algèbres de processus)

Panorama du cours

1. Approche Linéaire (Automates)
   1. Automates (Rappels)
   2. Spécification via des Automates
   3. Spécification vers la vérification (Compléments Automates)
   4. Du modèle à l'implantation : runtime vérification
2. Approche Arborescente (LTS)
   1. Spécification via des LTS
   2. Spécification vers la vérification
3. Compléments
   1. Notion de simulation/bisimulation
   2. Logique d'Hennesy-Milner
   3. Relation entre les approches assertionnelles (logiques) et comportementales (graphiques)
   4. Equivalence vs Congruence
4. Conclusion : Comparaisons, Bibliographie, Outils, Table des matières

Organisation & Evaluation

Volumes 10H CM + 2*TD + 2*TP

Evaluation 1H Exam - Documents autorisés

Outline

2. Approche linéaire (Automates)
   - Spécification via des automates
   - De la spécification à la vérification
     - Automate vs LTS
     - Abstration des ϵ-transitions
     - Rédution & Déterminisation
     - Minimisation
   - Vérification par comparaison ou par introspection/projection
   - Du modèle à l'implantation : Vérification en ligne
   - Premier bilan de l'approche linéaire/Automates

Automates : Rappels et compléments

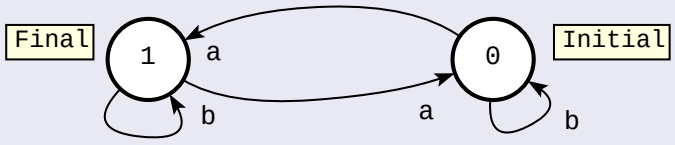
Définitions de base

- **Alphabet** α **:** Ensemble fini de symboles appelés *lettres*
- **Mot de** α est une suite ordonnée de lettres de α
  ex : $\alpha = \{a, b\} \mapsto$ a,b,a.b,b.a,a.a, ... sont des mots de α
- $\star$ **Etoile de Kleene**
  $\alpha^\star$ l'ensemble de tous les mots construits à partir de α
  nb : $\alpha^\star$ est infini (dés que $\alpha \neq \emptyset$)

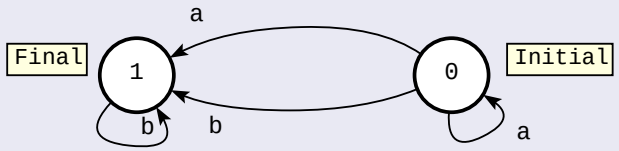
Automate

$\mathcal{A} =< S, I, F, \Sigma, \rightarrow >$ où
S est un ensemble (fini) d'états,
$I \subset S$: états initiaux
$F \subset S$: états finals
$\rightarrow \subset S \times \sigma \times S$
Σ est un alphabet

Automate **Déterministe** ssi
$|I| = 1$ et la relation $\rightarrow$ est une fonction
i.e $\forall s \in S, \forall \sigma \in \Sigma : |\rightarrow_\sigma (s)| \leq 1$



Automate déterministe reconnaisant les mots sur $\{a, b\}$ contenant un nb impair de a



Automate **non** déterministe car $\rightarrow_a (s) = \{0, 1\}$

Automates et Langages

Acceptation

Un mot $u \in \Sigma^*$ est accepté par $\mathcal{A}$ ssi
il existe un chemin $< q_o, e_0, q_1, e_1, \ldots q_{n-1}, e_{n-1} q_n >$ vérifiant : (1), (2) et (3)
(1) $q_o \in I, q_n \in F$, (2) $u = e_0.e_1....e_{n-1}$, (3) $\forall i \in [0, n-1] : (q_i, e_i, q_{i+1}) \in \rightarrow$

Langage reconnu par un automate

$L(\mathcal{A}) =_{Def} \{w \in \Sigma^* : w$ est accepté par $\mathcal{A}\}$

Rationnel & Reconnaisable

- On appelle **Rat** la plus petite classe des langages stable par union, produit, étoile et qui contient les lettres de l'alphabet. Un langage de cette classe est appelé rationnel (régulier)
  (cf expression rationnelle b(a+b)*ab)
- On appelle **Rec** la classe des langages reconnus par un automate. Un langage de cette classe est appelé reconnaissable.
- $Rat = Rec$ (Kleene 1951)
- Exemple de langages non reconnaissables : $a^n b^n$, expressions bien parenthésées, ...

Spécification de Services/Protocoles

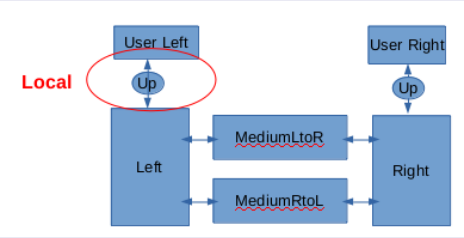
Principe

Définir le "service" attendu par l'ensemble des séquences de messages échangés. i.e., Définir le langage associé via son automate.

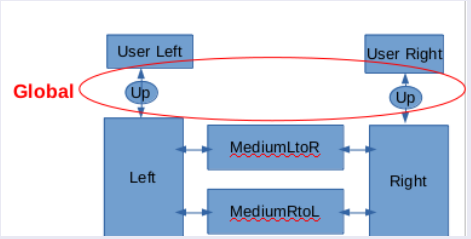
Différents points de vue complémentaires possibles

- Point de vue orienté "service" :

"Local"

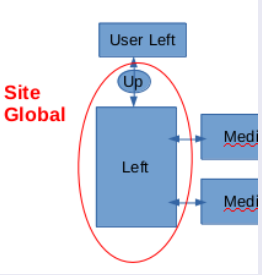


"Global"



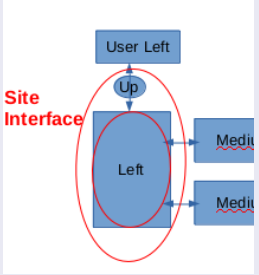
- Point de vue orienté "site" :

"Global"



(Messages + Evt internes)

"Interface"



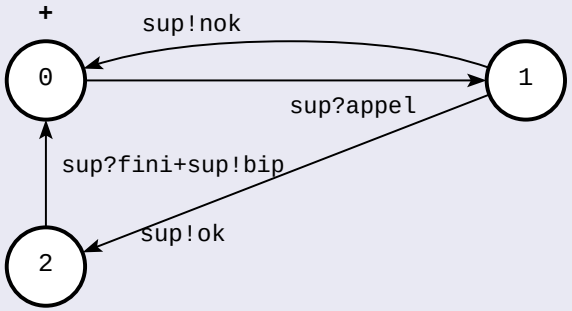
(Messages uniquement)

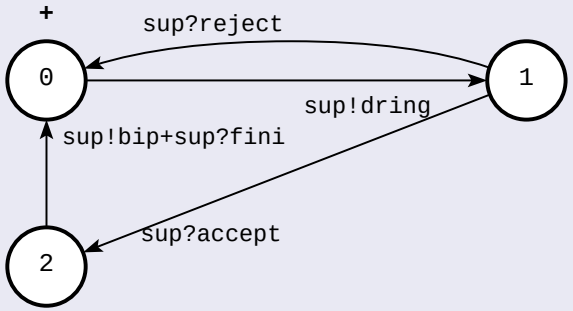
Ouverture/fermeture de connexion : Ex de spécifications

Service local : Spécifications de complexité croissante

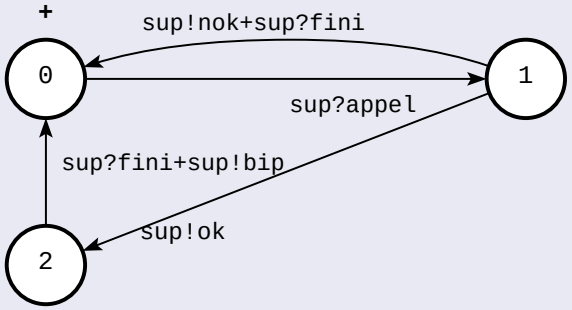
- **Version Asymétrique** : Initiateur/récepteur

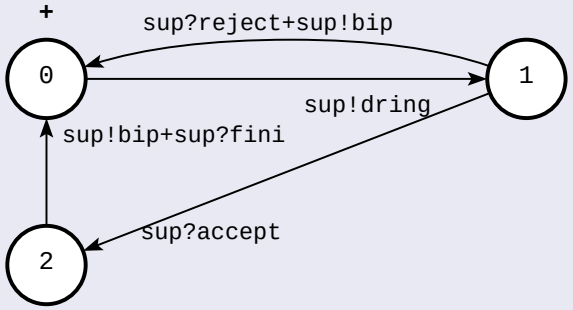
  a) On ne clôt que des connexions ouvertes.



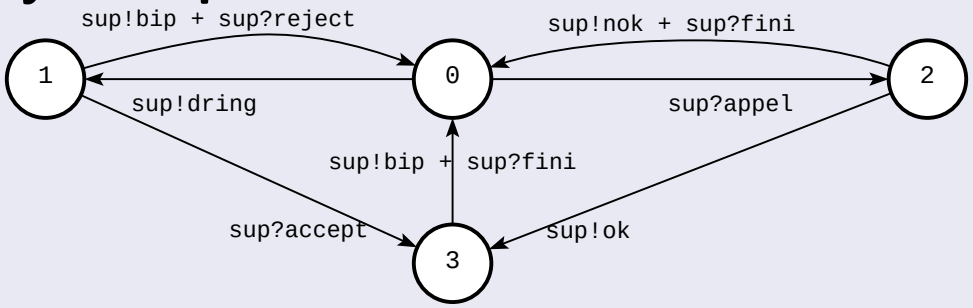


  b) L'initiateur peut clore une connexion en cours d'établissement





- **Version Asymétrique**



De la spécification à la vérification

Principe général

1. Disposer d'une(*) spécification du système sous forme d'automates (Une ou plusieurs suivant les "points de vue" à considérer !)
2. Disposer d'un modèle du système
3. Engendrer le graphe d'états associé représentant le comportement **fourni**
4. Comparer l'automate spécifiant le comportement **attendu** et l'automate représentant le comportement fourni

La spécification est satisfaite par le modèle (modèle $\models$ spécification) ssi le **service fourni** correspond au **service attendu** : égalité des langages

Quelques petits pbs à régler :

(3) Passer du graphe d'états à un automate

(4.1) Le comportement *réel* du modèle est plus *fin/détaillé* que la description du service $\mapsto$ nécessité d'**abstraire**
(Automates : Réduction des λ transitions)

(4.2) Décider de l'égalité de deux langages
Equivalence entre les automates respectifs
(Automates : Déterminiser/Minimiser ...)

(1) Du modèle à l'automate

Modélisation : Réseau de Petri (RdP), Automates Communicants (AC), Algèbre de Processus (AP), ...

Modèle	Graphe d'états
RdP	Graphe de marquages
AC	"Automate" étiqueté
AP	LTS/STE

Vs Automate ?
$\mathcal{A} =< S, I, F, \Sigma, \rightarrow >$

En plus : Etiquetage des sommets
(marquages pour les RdP, propositions atomiques pour AC)
$\mapsto$ les oublier !

En moins : Ensemble d'états Initiaux : un seul état initial s_0
$\mapsto$ Prendre $I = \{s_0\}$
Ensemble d'états Finals : **?**
$\mapsto$ Prendre $F = S$ – tous les états seront finals !
(dans ce cas, le langage reconnu est fermé par préfixe)

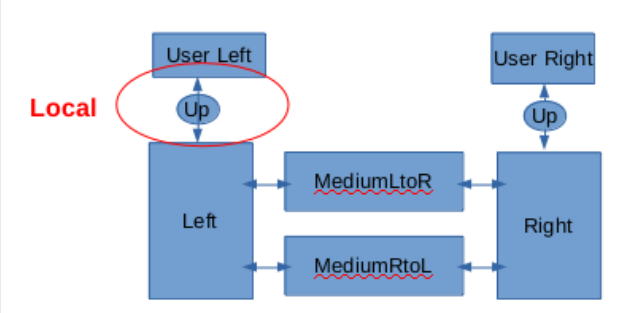
Globalement : $\mathcal{A} =< S, \{s_0\}, S, \rightarrow, \Sigma >$

En simplifiant les éléments redondants $\mapsto$ LTS : $< S, s_0, \Sigma, \rightarrow >$

Abstraire les événements non observés

Retour sur l'exemple Ouverture/Fermeture :

Modèle global (MG)
(LeftUser $\otimes$ LeftEntity)
$\otimes$ (LtoRmed $\otimes$ RtoLmed)
$\otimes$ (RightUser $\otimes$ RightEntity)



Evénements du modèle global (Σ_{MG})
Evénements du service (local) attendu (Σ_{Serv})

Pour comparer les langages $\mathcal{L}(Serv)$ et $\mathcal{L}(MG)$ il va falloir effacer dans MG toutes les lettres de $\Sigma_{MG} \setminus \Sigma_{Serv}$
Au niveau langage, on utilise une gomme !

Gomme : $\Sigma = \Sigma_{Obs} \cup \Sigma_{Inv}$

Gomme : $g_{\Sigma_{Obs}} : \Sigma^\star \mapsto \Sigma^\star_{Obs}$ définie par

$$g_{\Sigma_{Obs}}(1_\Sigma) = 1_{\Sigma_{Obs}}$$

$$g_{\Sigma_{Obs}}(a.w) = \begin{cases} a.g_{\Sigma_{Obs}}(w) & \text{Si } a \in \Sigma_{Obs} \\ g_{\Sigma_{Obs}}(w) & \text{Sinon} \end{cases}$$

g est un morphisme : $g(x.y) = g(x).g(y)$

Vérification

$MG \models Serv$ ssi
$\mathcal{L}(Serv) = g_{Serv}(\mathcal{L}(MG))$

Ok mais pas très opérationnel !

Réduction des ϵ transitions (ϵ :Automates, τ :LTS)

Objectif

Partant d'un automate $\mathcal{A} =< S, I, F, \Sigma_{Obs} \cup \Sigma_\tau, \rightarrow >$ construire un automate déterministe $\mathcal{A}' =< S', I', F', \Sigma_{Obs}, \rightarrow' >$ tel que $\mathcal{L}(\mathcal{A}') = g_{\Sigma_{Obs}}(\mathcal{L}(\mathcal{A}))$

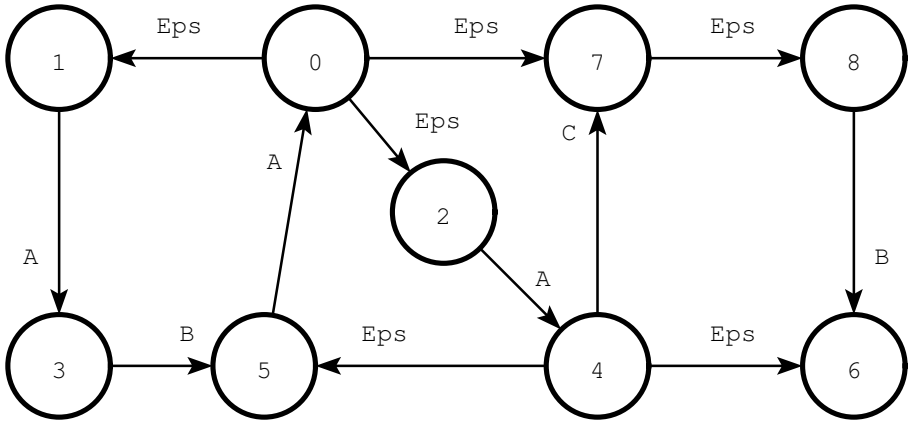
Opérations sur les relations binaires (rappels)

R, R_1, R_2 des relations binaires sur $S \times S$

- Composition de relation o
  Soit R_1, R_2 deux relations binaires on note $R_1 \ o \ R_2$ leur composition :
  $R_1 \ o \ R_2 =_{Def} \{(x, z) : \exists (x, y) \in R_1, \exists (y, z) \in R_2\}$
- Fermeture réflexive et transitive d'une relation
  Soit R une relation, on note R^* la fermeture réflexive et transitive de R
  En posant $R^0 = Id$ et $R^{K+1} = R \ o \ R^K$, on a $R^* =_{Def} \bigcup_{i \geq 0} R^i$

Notations :

- $\xrightarrow{\tau} =_{Def} \bigcup_{t \in \Sigma_\tau} \xrightarrow{t}$
- $\overset{\tau}{\Rightarrow} =_{Def} (\xrightarrow{\tau})^{(*)}$ et pour $\sigma \in \Sigma_{Obs}$: $\overset{\sigma}{\Rightarrow} =_{Def} \overset{\tau}{\Rightarrow} \ o \ \xrightarrow{\sigma}$
- pour $s \in S, t \in \Sigma$: $\overset{t}{\Rightarrow} (s) =_{Def} \{q \in S : s \overset{t}{\Rightarrow} q\}$
- pour $E \subset S, t \in \Sigma$: $\overset{t}{\Rightarrow} (E) =_{Def} \bigcup_{e \in E} \overset{t}{\Rightarrow} (E)$



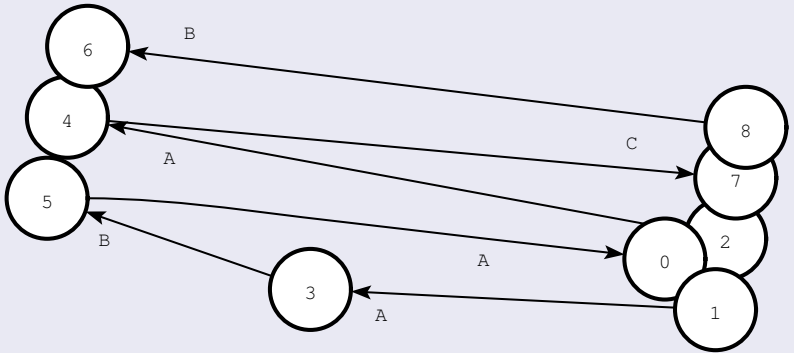
Exemple avec $\Sigma_\tau = \{Esp\}$

$\overset{\tau}{\Rightarrow}(0) = \{0,1,2,7,8\}, \overset{\tau}{\Rightarrow}(1) = \{1\}, \overset{\tau}{\Rightarrow}(4) = \{4,6\}, \overset{\tau}{\Rightarrow}(6) = \{6\}, \ldots$
$\overset{\tau}{\Rightarrow}(\{0,4,7\}) = \{0,1,2,4,5,6,7,8\}, \ldots,$
$\overset{A}{\Rightarrow}(0) = \{3,4\}, \overset{B}{\Rightarrow}(7) = \{6\}, \ldots$
$\overset{B}{\Rightarrow}(\{0,3\}) = \{5,6\}, \ldots$

Pas intermédiaire : automate non déterministe sans ϵ -transition

L'automate réduit à ses états dans 2^S Ex $0' = \{0,1,2,7,8\}$

on a remplacé $\rightarrow$ par $(\overset{t}{\Rightarrow})_{t \in \Sigma_{Obs}}$



Réduction des ϵ-transitions et déterminisation

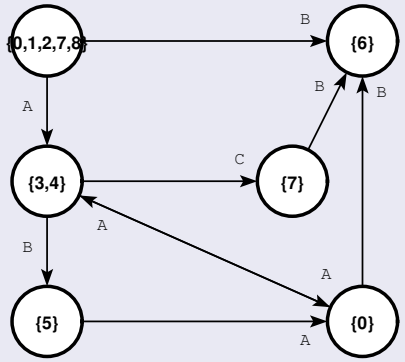
Données

Automate initial (déterministe ou non) $\mathcal{A} =< S, I, F, \Sigma_{Obs} \cup \Sigma_\tau, \rightarrow >$
$\mapsto$ Automate réduit déterministe sans ϵ-transition $\mathcal{A}' =< S', I', F', \Sigma_{Obs}, \rightarrow' >$
tels que $\mathcal{A}$ et $\mathcal{A}'$ sont équivalents (i.e., $\mathcal{L}(\mathcal{A}') = g_{\Sigma_{Obs}}(\mathcal{L}(\mathcal{A})))$

- $S' \subset 2^S$, $I' =_{Def} \overset{\tau}{\Rightarrow} (I)$, $I' \in S'$
- S' et $\rightarrow'$ sont les plus petits ensembles vérifiant
  $\forall s' \in S', \forall t \in \Sigma_{Obs} : \ \overset{\sigma}{\Rightarrow} (s') \in S'$ et $(s', \sigma, \overset{\sigma}{\Rightarrow} (s')) \in \rightarrow'$
- $F' =_{Def} \{E \in S' : \ E \cap F \neq \emptyset\}$

Application avec $I = \{0, 1\}$ et $F = \{8\}$

$$
\begin{aligned}
S' = \quad & \{\{0,1,2,7,8\}, \\
& \{3,4\}, \{6\}, \\
& \{5\}, \{7\}, \\
& \{0\}\} \\
F' = \quad & I'
\end{aligned}
$$

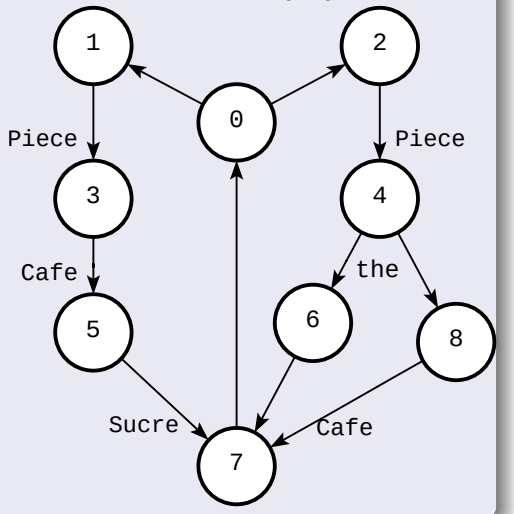


L'automate résultant n'est pas forcément minimal ;-(

Exemple de réduction/déterminisation (principales étapes)

Automate Initial

$$F = I = \{0\}$$



Réduction des ϵ-transitions (non exhaustif)

$\overset{\tau}{\Rightarrow}(0) = \{0, 1, 2\}$

...

$\overset{\tau}{\Rightarrow}(4) = \{4, 8\}$

$\overset{\tau}{\Rightarrow}(6) = \{6, 7, 0, 1, 2\}$

$\overset{\tau}{\Rightarrow}(7) = \{7, 0, 1, 2\}$

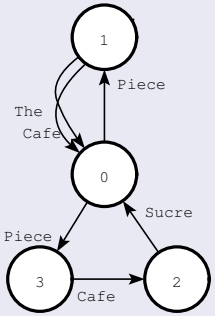
$\overset{Piece}{\Rightarrow}(i) = \{3, 4\}$ $(i \in \{0, 1, 2, 6, 7\})$

$4 : \overset{Piece}{\Rightarrow}(4) = \emptyset, \overset{The}{\Rightarrow}(4) = \{6\}, \overset{Cafe}{\Rightarrow}(4) = \{7\}$

....

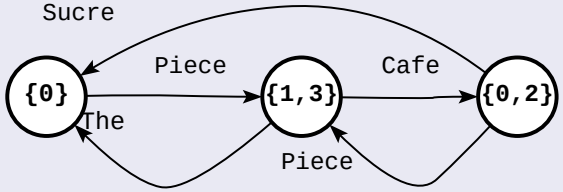
Automate sans ϵ-transition

$$F = I = \{0\}$$



Automate déterministe

$$F = \{\{0\}, \{0, 2\}\}$$



Exo : Montrer qu'il est minimal !

Minimisation d'un Automate par partition

Automate initial déterministe $\mathcal{A} =< S, s_0, F, \Sigma, \rightarrow >$

Equivalence de Nerode : (Anil Nerode)

Soit $\asymp \subset S \times S$ la relation d'équivalence telle que $s \asymp q$ ssi $\mathcal{L}(\mathcal{A})(s) = \mathcal{L}(\mathcal{A})(q)$
i.e. *Deux sommets sont équivalents ssi les langages reconnus par $\mathcal{A}$ en considérant s et q comme état initiaux respectifs sont égaux*

Automate quotient par l'équivalence de Nerode

Soit $\mathcal{A}_{\asymp} =< S_{\asymp}, s_{0\asymp}, F_{\asymp}, \Sigma, \rightarrow_{\asymp} >$, l'automate défini par :

- $S_{\asymp} =_{Def} S/\asymp$, le quotient de S par $\asymp$
- $s_{0\asymp}$ la classe d'équivalence de s_0 pour $\asymp$
- $F_{\asymp} =_{Def} \{f_{\asymp} : \ f \in F\}$, l'ensemble des classes d'équ. des états finals
- $\rightarrow_{\asymp}$ le plus petit ensemble vérifiant
  Si $s \xrightarrow{t} q$ alors $(s_{\asymp}, t, q_{\asymp}) \in \rightarrow_{\asymp}$

Propriété de minimisation

L'automate $\mathcal{A}_{\asymp}$ est équivalent à $\mathcal{A}$ et est minimal wrt son nombre d'états.

Reste plus qu'à savoir calculer $\asymp$!

Minimisation d'un Automate (suite)

Automate initial déterministe $\mathcal{A} =< S, s_0, F, \Sigma, \rightarrow >$

Calcul opérationnel de l'équivalence de Nerode (Moore)

$\asymp$ peut être définie comme la limite d'une suite décroissante de relations d'équ.
$\asymp =_{Def} \bigcap_{n \geq 0} \asymp_n$

En prenant $\asymp_0 = \{S \setminus F, F\}$ et en définissant $\asymp_{K+1}$ à partir de $\asymp_K$ ainsi

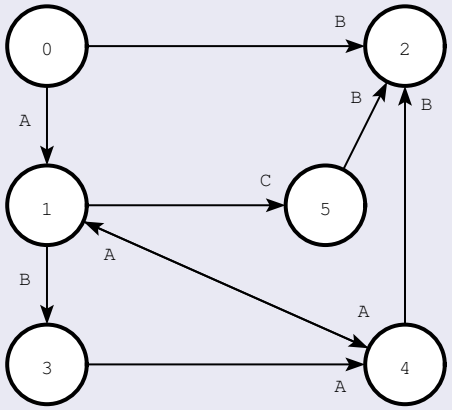
$$s \asymp_{K+1} q \text{ ssi } s \asymp_K q \text{ et } \forall t \in \Sigma \left| \begin{array}{l} \text{Si } s \xrightarrow{t} s' \text{ Alors } \exists q' : q \xrightarrow{t} q' \text{ avec } s' \asymp_K q' \\ \text{Si } q \xrightarrow{t} q' \text{ Alors } \exists s' : s \xrightarrow{t} s' \text{ avec } s' \asymp_K q' \end{array} \right.$$

nb : La suite $\asymp_n$ étant décroissante converge nécessairement.
Il existe un point fixe : i.e. $\exists k \in \mathbb{N} : \asymp_k = \asymp_{k+1}$
Au pire on tombe sur $\asymp_n = \{\{s\} : s \in S\}$

"Accélerateur"

Soit $Out : \ S \mapsto 2^{\Sigma}$ définie par $Out(s) = \{t \in \Sigma : \ s \xrightarrow{t}\}$
et $\asymp_{out} \subset S \times S$ définie par $s \asymp_{out} q$ ssi $Out(s) = Out(q)$
on a $\asymp_1 \subset \asymp_{out}$ on peut donc démarrer avec $\asymp_{out} \cap \asymp_0$

Exemple de minimisation d'un automate



Out

0	$\{A, B\}$
1	$\{A, B, C\}$
2	$\emptyset$
3	$\{A\}$
4	$\{A, B\}$
5	$\{B\}$

Out^{-1}

$\{A, B, C\}$	$\{1\}$
$\{A, B\}$	$\{0, 4\}$
$\{A\}$	$\{3\}$
$\{B\}$	$\{5\}$
$\emptyset$	$\{2\}$

$\asymp_{out} = \{\{0, 4\}, \{1\}, \{2\}, \{3\}, \{5\}\}$

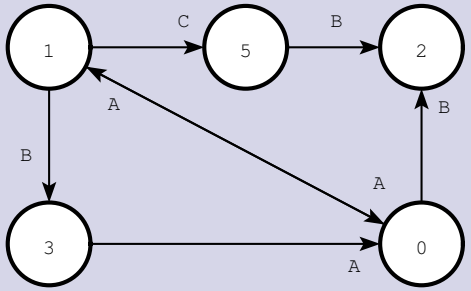
En prenant $F = \{0\}$

Alors $\asymp_0 = \{\{0\}, \{4\}, \{1\}, \{2\}, \{3\}, \{5\}\}$ et $\asymp = \asymp_0$
L'automate ci-dessus est minimal

En prenant $F = \{0, 4\}$ alors $\asymp_1 = \asymp_{out}$

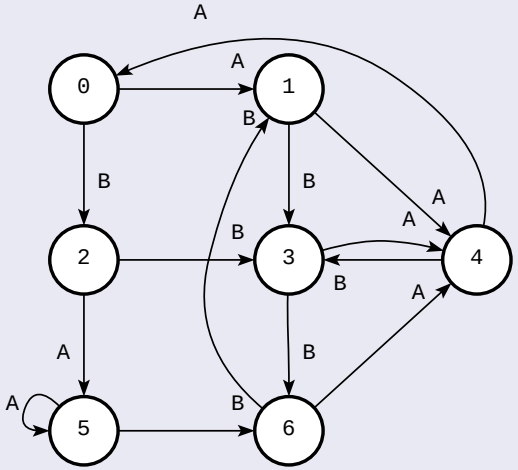
Il faut calculer $\asymp_2$:
On a $\{\{1\}, \{2\}, \{3\}, \{5\}\} \subset \asymp_2$.
Quid de 0 et 4 ?
Rien ne permet de séparer 0 et 4
(ils ont exactement les mêmes "classes" de successeurs)
on a donc $0 \asymp_2 4$
Finalement $\asymp_2 = \asymp_1$ donc $\asymp = \asymp_1$

Automate minimal



Exemple de minimisation (principales étapes)

Automate Initial
($F = \{4\}$)



Calcul de $\asymp_0$

$\asymp_0 =$
$\{\{4\}, \{0,1,2,3,5,6\}\}$
$Out^{-1}(\{A,B\}) = S$
(Pas d'accélération !)

Calcul de $\asymp_1$
$\{0,1,2,3,5,6\}$?

	A	B
0	$0_{\asymp_0}$	$0_{\asymp_0}$
1	$4_{\asymp_0}$	$0_{\asymp_0}$
2	$0_{\asymp_0}$	$0_{\asymp_0}$
3	$4_{\asymp_0}$	$0_{\asymp_0}$
5	$0_{\asymp_0}$	$0_{\asymp_0}$
6	$4_{\asymp_0}$	$0_{\asymp_0}$

$\mapsto \{0,2,5\}, \{1,3,6\}$

Calcul de $\asymp_2$
$\{0,2,5\}$?

	A	B
0	$1_{\asymp_1}$	$0_{\asymp_1}$
2	$0_{\asymp_1}$	$1_{\asymp_1}$
5	$0_{\asymp_1}$	$1_{\asymp_1}$

$\mapsto \{0\}, \{2,5\}$

$\{1,3,6\}$?

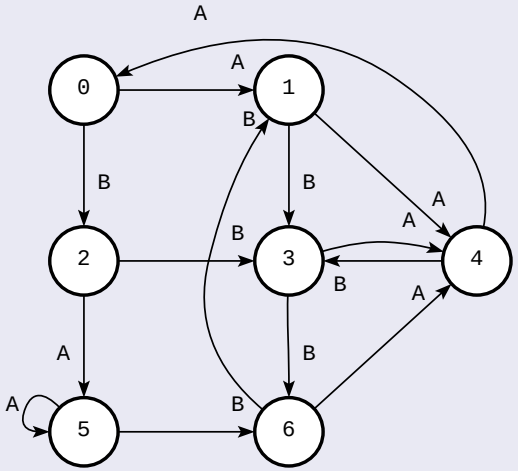
	A	B
1	$4_{\asymp_1}$	$1_{\asymp_1}$
3	$4_{\asymp_1}$	$1_{\asymp_1}$
5	$4_{\asymp_1}$	$1_{\asymp_1}$

$\mapsto \{1,3,6\}$

Bilan provisoire
$\asymp_2 = \{\{0\}, \{2,5\}, \{4\}, \{1,3,6\}\}$
reste à calculer $\asymp_k$ $(k \geq 3)$
i.e Tenter de partitionner
$\{2,5\}$ et $\{1,3,6\}$ etc ...

Exemple de minimisation (suite)

Automate Initial



$\asymp_2 =$
$\{\{0\},\{2,5\},\{4\},\{1,3,6\}\}$

Calcul de $\asymp_3$

$\{2,5\}$?

	A	B
2	$2_{\asymp_2}$	$1_{\asymp_2}$
5	$2_{\asymp_2}$	$1_{\asymp_2}$

Pas de chgt

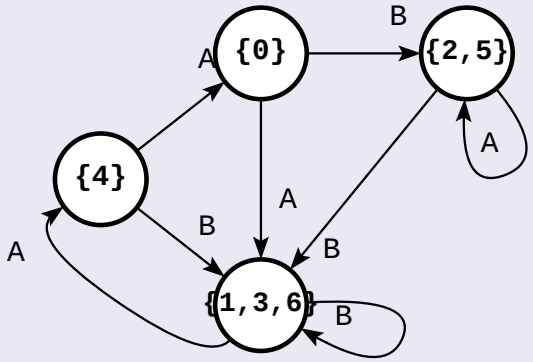
$\{1,3,6\}$?

	A	B
1	$4_{\asymp_2}$	$1_{\asymp_2}$
3	$4_{\asymp_2}$	$1_{\asymp_2}$
6	$4_{\asymp_2}$	$1_{\asymp_2}$

Pas de chgt

Point fixe atteint ($\asymp_2 = \asymp_3$)
Donc $\asymp = \{\{0\},\{2,5\},\{4\},\{1,3,6\}\}$

Automate minimal résultant
$I = \{0\}$,$F = \{4\}$



Méthodes de vérification associées

Vérification par comparaison

1. Disposer d'une(*) spécification du système sous forme d'automates (Une ou plusieurs suivant les "points de vue" à considérer !)
2. Disposer d'un modèle du système
3. Engendrer le graphe d'états associé représentant le comportement **fourni**
4. Comparer l'automate spécifiant le comportement **attendu** et l'automate représentant le comportement fourni (réduire les ϵ-transition, déterminiser, minimiser et s'assurer de l'isomporhisme des automates)
   La spécification est satisfaite par le modèle (modèle $\models$ spécification) ssi le **service fourni** correspond au **service attendu** : égalité des langages

Vérification par "introspection"

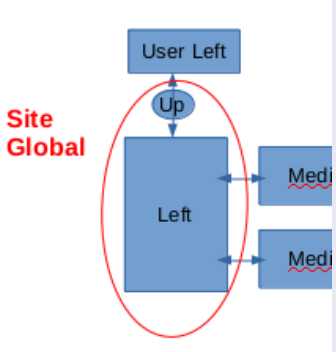
1. Disposer d'un modèle du système
2. Engendrer le graphe d'états associé représentant le comportement **fourni**
3. Sélectionner un "point de vue" (choisir Σ_{Obs})
4. Synthétiser le **"service offert"** : "Projeter" le comportement global wrt Σ_{obs} (réduire les ϵ-transition, déterminiser, minimiser)
5. "Analyser" le service offert

Vérification par projection/Introspection

Principe

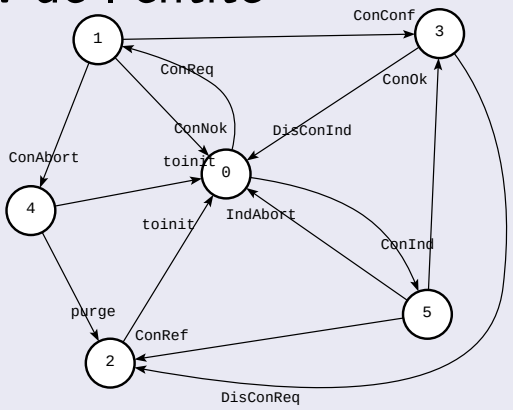
Pré-requis : Disposer d'un modèle
Définir les **"points d' observation"** (choisir Σ_{Obs})
Synthétiser le sevice **offert/fourni** wrt Σ_{Obs}
Analyser le résultat
Difficultés : Ne pas oublier que l'on travaille sur un modèle abstrait
Avoir conscience des propriétés préservées par la relation d'équivalence utilisée **(à suivre)**

Exemple de projection : Focalisation sur une entité du protocole



(Messages + Evt internes)

Abstraction du comportement **effectif** de l'entité



Du modèle à l'implantation : Vérification en ligne

Technique des observateurs, moniteurs, contrôleurs, safety-bags, . . .

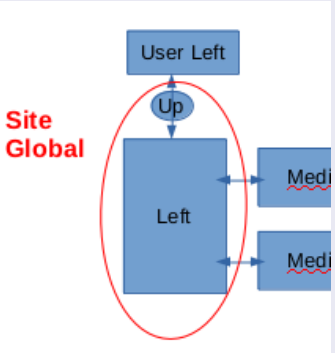
Principe : Réutiliser (automatiquement) les résultats de vérification pour "monitorer" /surveiller **en ligne** l'exécution de l'implantation

Implantation : Encapsuler l'entité à surveiller par une surcouche contenant l'automate de contrôle représentant les exécutions de référence.

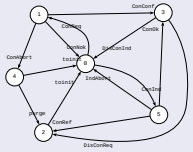
Dès que l'exécution réelle "dévie" de l'exécution de référence, une alarme est déclenchée et d'éventuels mécanismes de tolérance aux fautes peuvent être mis en oeuvre.

Exemple de safety-bag

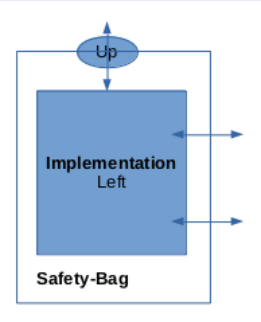
Modèle vérifié de l'Entité



Génération par **projection** de l'automate de **contrôle**



Génération du safety-bag et encapsulation de l'implantation



Premier bilan de l'approche linéaire/Automates

Points forts

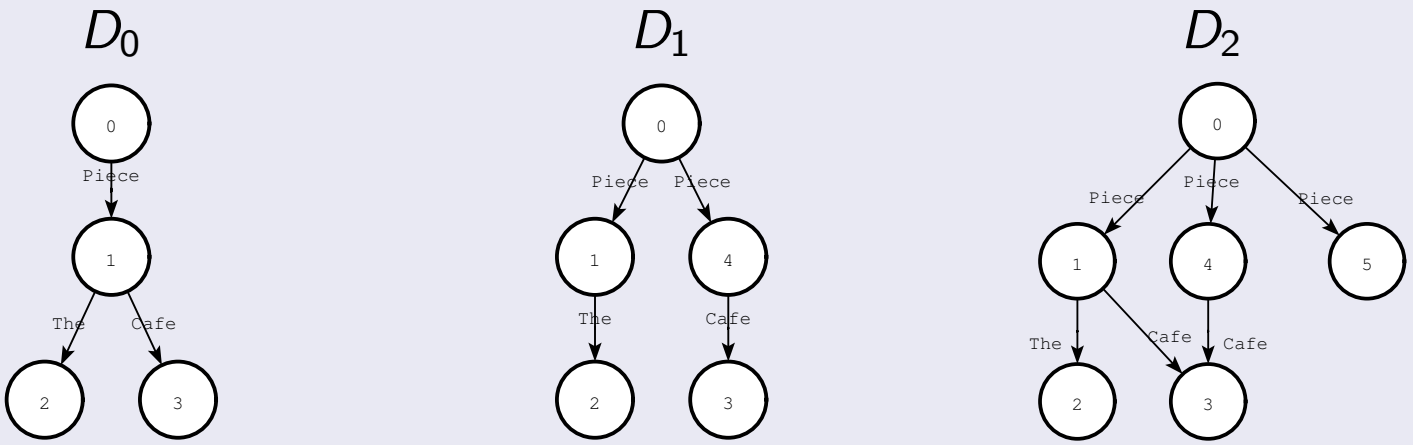
1. "Automatique"
2. Possibilités offertes & Spectre couvert (modèle $\mapsto$ implantation)
   1. Spécification formelle du "service attendu" via des automates
   2. Vérification par comparaison entre le service "attendu" et le service "offert"
   3. Vérification par introspection/projection : synthèse du service "offert"'
      (très utile pour la mise au point)
   4. Réutilisation des projections pour générer des safety-bags : runtime-vérification

Point faible

Identifier la clase de propriétés préservées par ces approches
Faible Expressivité pour la vérification

Tea time : Quelles propriétés sont préservées ?

Diverses spécifications de machines à café



Equivalence de ces spécifications ?

wrt **Utilisateur o/n ?** & **Exploitant o/n ?** & **Langage o/n ?**

Propriétés préservées : séquencement d'actions

On ne peut obtenir une boisson qu'après l'avoir payée, ...

Propriétés non préservées :

blocage : D_0 et D_1 délivrent toujours une boisson payée mais pas D_2

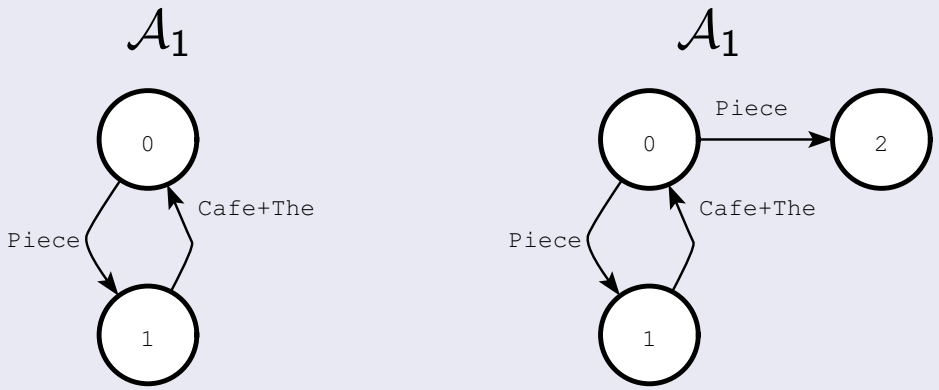
non-déterminisme : D_1 garantit une boisson mais c'est le distributeur qui chosit !

Bilan

On ne pourra pas distinguer entre ces 3 comportements de distributeur !

Langage & Préservation des Blocages

On ne peut donc plus faire la distinction entre bloquant et non bloquant



$$\mathcal{L}(\mathcal{A}_0) = \mathcal{L}(\mathcal{A}_1)$$

Non préservation des blocages

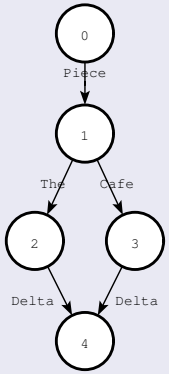
On est passé d'un automate - avec une notion d'états finals (F) - à un LTS sans états finals ($F = S$)

Equivalences de Traces Maximales

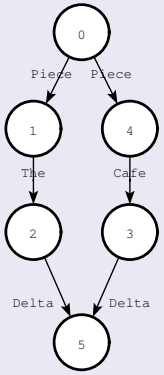
Principe : Pour pallier la perte de la notions d'états finals - et la non préservation des blocages - on enrichit l'automate initial – $\mathcal{A} \mapsto TM(\mathcal{A})$ – en **matéralisant** tous les états sans successeur par une nouvelle transition δ ; on applique ensuite l'approche traditionnelle.
Pour un automate $\mathcal{A}$: $\mathcal{L}_{TM}(\mathcal{A}) =_{Def} \mathcal{L}(TM(\mathcal{A}))$

Equivalence de Traces Maximales (suite)

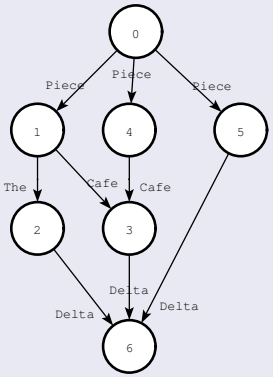
$TM(D_0)$



$TM(D_1)$



$TM(D_2)$



$$\mathcal{L}_{TM}(\mathcal{D}_0) = \mathcal{L}_{TM}(\mathcal{D}_1) \text{ mais } \mathcal{L}_{TM}(\mathcal{D}_0) \neq \mathcal{L}_{TM}(\mathcal{D}_2)$$

Bilan Général

Blocages bien pris en compte par l'extension "Traces Maximales" !
Mais pas de progrès vis à vis du non-déterminisme !

Mécanisme d'abstraction perfectible :

On ne peut pas savoir à l'avance ce qui est important - ce dont on peut faire abstraction - et ce qui ne l'est pas - ce dont on ne doit pas faire abstraction

Approche langage trop radicale : gomme qui efface tout !

Besoin d'une abstraction plus raisonnée qui "synthétiserait" ce dont on peut faire abstraction ....

$\mapsto$ Section suivante : LTS, arborescente, bisimulation ...

Outline

3. Equivalence Observationnelle
   - Equivalence Observationnelle forte ($\sim$)
     - Exemple
     - Extension de $\sim$ aux LTS
     - Projection observationnelle

Equivalence Observationnelle ($\sim$)

Généralités

Introduite dans le contexte de l'algèbre de Processus CCS (R. Milner) $\sim$ corrige un certain nombre de faiblesses de l'équivalence langage **appliquée** dans un contexte de vérification.

Trois grandes différences :

1. La donnée d'entrée sera un LTS et non plus un automate
   Considération pragmatique puisque dans un contexte vérification c'est la seule donnée effectivement disponible.
2. Privilégie une approche arborescente (non-déterminisme) contrairement a l'approche langage (par nature séquentielle/linéaire).
   De facon informelle, on va appliquer un algorithme de minimisation sans chercher à déterminiser auparavant.
3. Le mécansime d'abstraction est moins radical que celui mis en oeuvre dans le contexte langage : au lieu de **gommer** tout ce que l'on ne veut pas observer - l'approche observationnelle "banalise" tous les événements que l'on ne veut pas observer, les "incorpore" aux événements observables .... mais ne les fait pas disparaître (à suivre)
   A posteriori, l'approche observationnelle permet de "synthétiser" les événements dont on peut faire effectivement abstraction.

Equivalence observationnelle forte ($\sim$)
LTS : $< S, s_0, \Sigma, \rightarrow >$ $\quad \sim \subset S \times S$

Terminologie : Forte : pas d'abstraction / Faible : avec abstraction

Définition récursive de $\sim$

$s \sim q$ ssi $\forall t \in \Sigma$

$\forall s'$ Si $s \xrightarrow{t} s'$ Alors $\exists q' : q \xrightarrow{t} q'$ avec $s' \sim q'$

$\forall q'$ Si $q \xrightarrow{t} q'$ Alors $\exists s' : s \xrightarrow{t} s'$ avec $q' \sim s'$

Définition opérationnelle équivalente

$\sim = \cap_{n \geq o} \sim_n$ où $\sim_0 = S \times S$

$s \sim_{k+1} q$ ssi $s \sim_k q$ et $\forall t \in \Sigma$

$\forall s'$ Si $s \xrightarrow{t} s'$ Alors $\exists q' : q \xrightarrow{t} q'$ avec $s' \sim_k q'$

$\forall q'$ Si $q \xrightarrow{t} q'$ Alors $\exists s' : s \xrightarrow{t} s'$ avec $q' \sim_k s'$

nb : $\sim$ est définie car $(\sim_i) \downarrow$ $\qquad$ (i.e $i < j \Rightarrow \sim_j \subset \sim_i$)

Premier Exemple

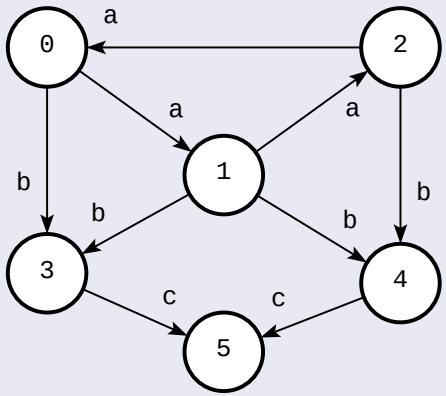
Accélerateur

Soit $Out : \ S \mapsto 2^{\Sigma}$ définie par $Out(s) = \{t \in \Sigma : \ s \xrightarrow{t}\}$

et $\asymp_{out} \subset S \times S$ définie par $s \asymp_{out} q$ ssi $Out(s) = Out(q)$

on a $\asymp_1 = \asymp_{out}$

Exemple



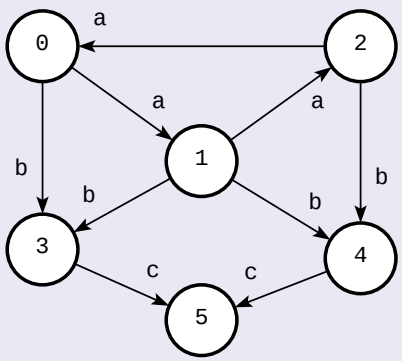
Out^{-1}

$\{a, b, c\}$	$\emptyset$
$\{a, b\}$	$\{0, 1, 2\}$
"$\{x, y\}$"	$\emptyset$
$\{a\}$	$\emptyset$
$\{b\}$	$\emptyset$
$\{c\}$	$\{3, 4\}$
$\emptyset$	$\{5\}$

$\asymp_{out} = \sim_1 = \{\{0, 1, 2\}, \{3, 4\}, \{5\}\}$

Premier Exemple : $\sim_1 = \{\{0,1,2\},\{3,4\},\{5\}\}$

Calcul de $\sim_2$



Quid de $\{0,1,2\}$

0	a	$\{0_{\sim_1}\}$
0	b	$\{3_{\sim_1}\}$
1	a	$\{0_{\sim_1}\}$
1	b	$\{3_{\sim_1}\}$
2	a	$\{0_{\sim_1}\}$
2	b	$\{3_{\sim_1}\}$

Pas de changement

Quid de $\{3,4\}$

3	c	$\{5_{\sim_1}\}$
4	c	$\{5_{\sim_1}\}$

Pas de changement

$\sim_2 = \{\{0,1,2\},\{3,4\},\{5\}\}$

Pas de "bloc" séparé

Conclusion

Point fixe atteint : $\sim_1 \ = \ \sim_2$

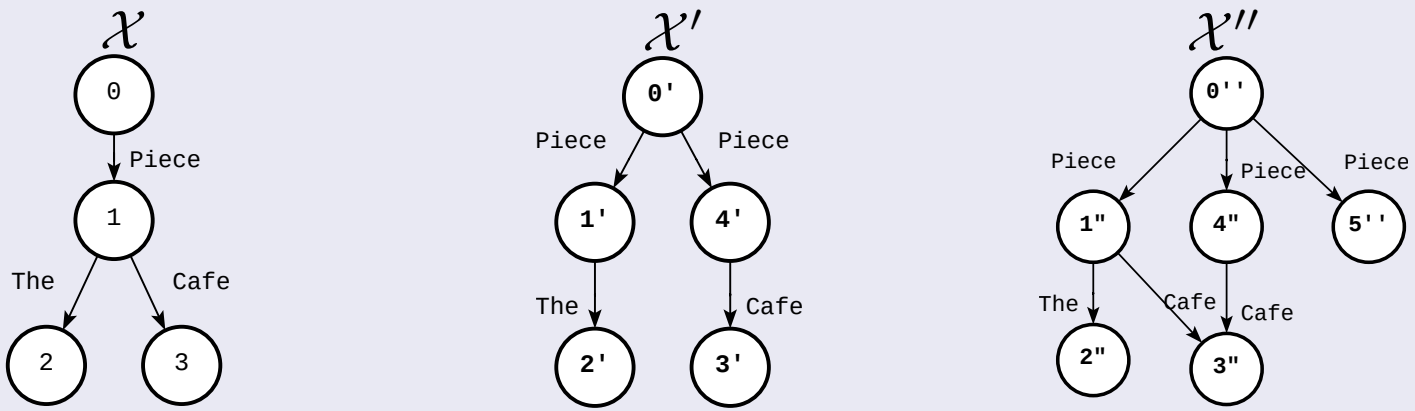
donc $\sim \ = \ \sim_1 = \{\{0,1,2\},\{3,4\},\{5\}\}$

Extension de $\sim$ aux LTS

Définition

Deux LTS, LTS_1 et LTS_2, sont observationnellement équivalents si leurs états initiaux respectifs le sont : $LTS_1 \sim LTS_2$ ssi $s_{01} \sim s_{02}$

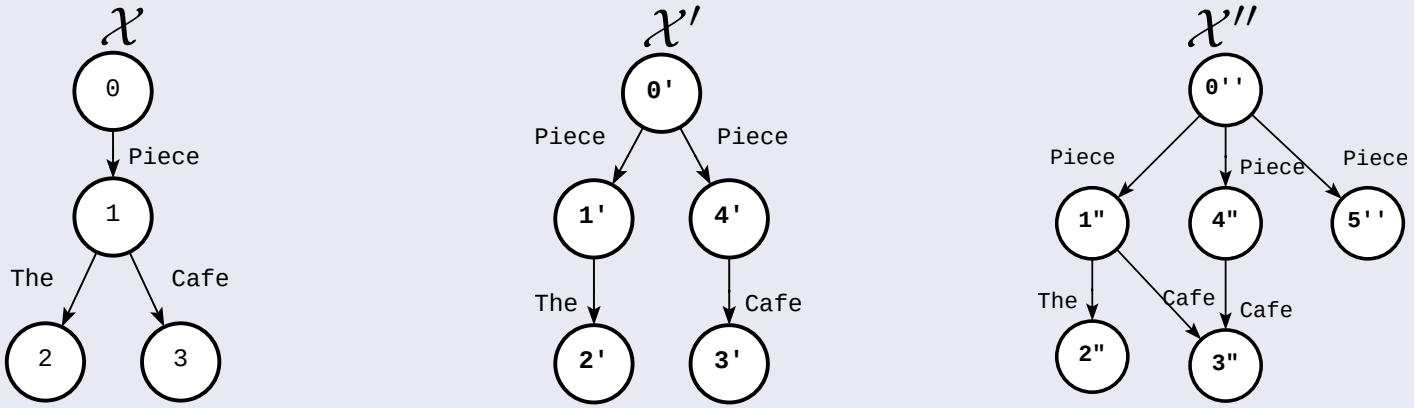
Exemple d'application : 3 machines à café reconnaissant le même langage



Question : l'équivalence observationnelle parvient-elle à les distinguer ?

Approche

On va considérer les 3 LTS "de front" en considérant la "réunion" de ces 3 STE comme un seul STE.
(i.e., on calcule directement $\sim$ sur $(S \cup S' \cup S'') \times (S \cup S' \cup S''))$



Calcul de $\sim_1$ (i.e., calcul de Out^{-1})

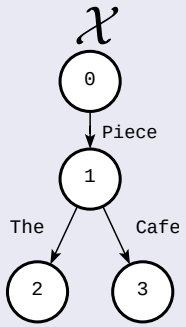
Out^{-1}

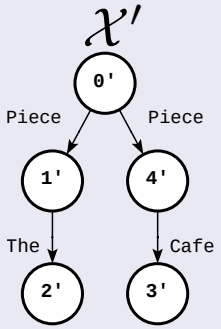
$\{Piece, The, Cafe\}$	$\emptyset$
$\{The, Cafe\}$	$\{1, 1''\}$
"$\{x, y\}$"	$\emptyset$
$\{Piece\}$	$\{0, 0', 0''\}$
$\{Cafe\}$	$\{4', 4''\}$
$\{The\}$	$\{1'\}$
$\emptyset$	$\{2, 3, 2', 3', 2'', 3'', 5''\}$

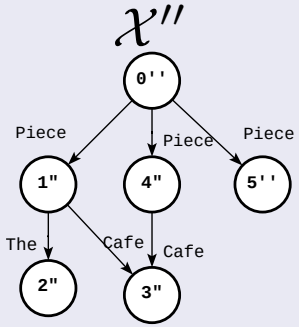
$\asymp_{out} = \sim_1 = \{\{0, 0', 0''\}, \{1, 1''\}, \{1'\}, \{4', 4''\}, \{2, 3, 2', 3', 2'', 3'', 5''\}\}$

Propriété de $Out^{-1}(\emptyset)$: $\forall s, q \in Out^{-1}(\emptyset)\ :\ s \sim q$

Application : $\{2, 3, 2', 3', 2'', 3'', 5''\} \in \sim$







Calcul des $\sim_i$ $(i \geq 2)$

Quid de $\{1, 1''\}$

1	*The*	$\{2_{\sim_1}\}$
1	*Cafe*	$\{2_{\sim_1}\}$
$1''$	*The*	$\{2_{\sim_1}\}$
$1''$	*Cafe*	$\{2_{\sim_1}\}$

Pas de changement

Quid de $\{0, 0', 0''\}$

0	*Piece*	$\{1_{\sim_1}\}$
$0'$	*Piece*	$\{1'_{\sim_1}, 4'_{\sim_1}\}$
$0''$	*Piece*	$\{1_{\sim_1}, 4'_{\sim_1}, 2_{\sim_1}\}$

$\{0, 0', 0''\} \mapsto \{0\}, \{0'\}, \{0''\}$ (*)

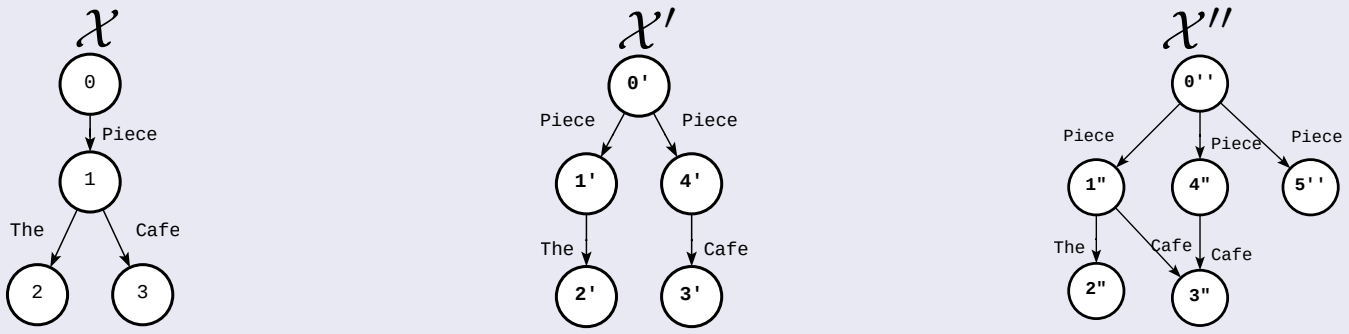
(*) $\mathcal{X}, \mathcal{X}'$ et $\mathcal{X}''$ sont donc deux à deux non équivalent observationnellement.

Quid de $\{4', 4''\}$? A-t-on une chance de pouvoir séparer 4' et 4" ?

On sait que $3' \sim 3''$ on a donc $4' \sim 4''$ (il suffit de vérifier qu'ils satisfont bien les conditions de la définition récursive de $\sim$)

Donc $\sim_2 = \{\{0\}, \{0'\}, \{0''\}, \{1, 1''\}, \{1'\}, \{4', 4''\}, \{2, 3, 2', 3', 2'', 3'', 5''\}\}$

nb : on aurait ainsi pu directement déduire que $1 \sim 1'$ et finalement déduire que $\sim = \{\{0\}, \{0'\}, \{0''\}, \{1, 1''\}, \{1'\}, \{4', 4''\}, \{2, 3, 2', 3', 2'', 3'', 5''\}\}$



Retour sur les LTS d'origine

On vient de calculer $(S \cup S' \cup S'')/\sim_{S \cup S' \cup S''}$
Peut-on en déduire $\sim$ sur chacun d'eux ?

Propriété : $C \in \sim_S$ ssi $(\exists C' \in \sim_{S \cup S' \cup S''}$ avec $C' \cap S = C$ et $C \neq \emptyset)$

Application :

$\sim_{S \cup S' \cup S''} =$
$\{\{0\}, \{0'\}, \{0''\}, \{1, 1''\}, \{1'\}, \{4', 4''\}, \{2, 3, 2', 3', 2'', 3'', 5''\}\}$

$S/\sim = \{\{0\}, \{1\}, \{2, 3\}\}$
$S'/\sim = \{\{0'\}, \{1'\}, \{4'\}, \{2', 3'\}\}$
$S''/\sim = \{\{0''\}, \{1''\}, \{4''\}, \{2'', 3'', 5''\}\}$

Projection observationelle – Même principe que la projection langage

Projection observationnelle (forte) (LTS Quotient)

Soit un LTS $\mathcal{X} = < S, s_0, \Sigma, \rightarrow >$ et $\sim \subset S \times S$

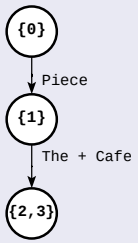
$\mathcal{X}/\sim =_{Def} < S_\sim, s_{0\sim}, \Sigma, \rightarrow_\sim >$ où

- $S_\sim =_{Def} S/\sim$, le quotient de S par $\sim$
- $s_{0\sim}$ la classe d'équivalence de s_0 pour $\sim$
- $\rightarrow_\sim$ le plus petit ensemble vérifiant : Si $s \xrightarrow{t} q$ alors $(s_\sim, t, q_\sim) \in \rightarrow_\sim$

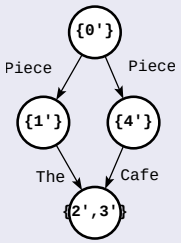
$\mathcal{X}/\sim$ est le plus petit - en nombre d'états - LTS équivalent pour $\sim$ à $\mathcal{X}$

Application à $\mathcal{X}$, $\mathcal{X}'$ et $\mathcal{X}''$

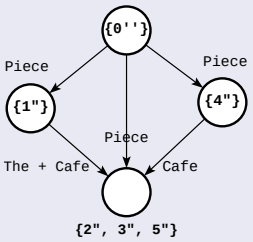
$\mathcal{X}/\sim$



$\mathcal{X}'/\sim$



$\mathcal{X}''/\sim$



Remarque

On savait que $\mathcal{X}$, $\mathcal{X}'$ et $\mathcal{X}''$ étaient non deux à deux équivalents observationnelement, on obtient donc trois projections (quotients) distincts

Outline

4. Equivalence Observationnelle (faible)
   - Introduction
   - LTS abstrait
   - Equivalence Observationnelle faible ($\sim_{\Sigma_{Obs}}$)
     - Extension de $\sim_{\Sigma_{Obs}}$ aux LTS
     - Exemple
   - Projection Observationnelle
   - Simplification de LTS compatible avec $\sim$

Equivalence Observationnelle faible ($\sim$)

Généralités

Abstraction nécessaire pour pouvoir comparer des LTS de nature diverse : par exemple, spécification (partielle) / comportement (global)
Difficulté de savoir à l'avance ce dont on peut (raisonnablement) faire abstraction.

Différence majeure entre les deux relations d'équivalence :

Langage "efface/gomme" tout ce dont on a décidé de faire abstraction : que ce soit pertinent ou non !

Observationnelle "conserve en partie" ce dont on a décidé de faire abstraction : elle "banalise" tous les événements à abstraire (inobservables) en les renommant (τ pour CCS, i pour Lotos), elle les incorpore aux événements concrets (non abstraits/observables).

A posteriori, l'approche observationnelle permet de "synthétiser" 2 classes d'événements abstraits :
- ceux qui ne provoquent pas de changement de classe de comportement et dont on peut faire effectivement abstraction
- ceux qui entraînent des changements de classe de comportement et dont on ne doit/peut pas faire abstraction.

LTS abstrait

Principe

Partant du LTS initial $\mathcal{X} =< S, s_0, \Sigma, \rightarrow >$ et $\Sigma_{obs} \subset \Sigma$
Obtenir un LTS *abstrait* $\mathcal{X}_{\Sigma_{obs}} =< S, s_0, \Sigma_{Obs} \cup \{\tau\}, \Rightarrow >$ permettant de faire abstraction des événements abstraits (i.e $\in \Sigma \setminus \Sigma_{obs}$) et sur lequel on déroulera les calculs habituels.

Expérimentation abstraite – Passage de $(\rightarrow, \Sigma)$ à $(\Rightarrow, \Sigma_{obs} \cup \{\tau\})$

Définition de $\Rightarrow$
$\Rightarrow \subset S \times (\Sigma_{Obs} \cup \{\tau\}) \times S)$ est la plus petite relation vérifiant :

1. $\forall s \in S : s \overset{\tau}{\Rightarrow} s$ (τ-réflexivité)
2. Si $s_1 \overset{e_1}{\rightarrow} s_2 \overset{e_2}{\rightarrow} s_3 \ldots \overset{e_{n-1}}{\rightarrow} s_n$ avec $\forall i \in [1, n-1] : e_i \in \Sigma \setminus \Sigma_{Obs}$ Alors $s_1 \overset{\tau}{\Rightarrow} s_n$
3. Si $s \overset{\tau}{\Rightarrow} s_i$ et $q_i \overset{\tau}{\Rightarrow} q$ et pour $a \in \Sigma_{Obs} : s_i \overset{a}{\rightarrow} q_i$ Alors $s \overset{a}{\Rightarrow} q$

Définition équivalente de $\Rightarrow$

$\overset{\tau}{\Rightarrow} =_{Def}$ Fermeture réflexive et transitive de $\overset{\tau}{\rightarrow}$ avec $\overset{\tau}{\rightarrow} =_{Def} U_{i \in \Sigma \setminus \Sigma_{obs}} \overset{i}{\rightarrow}$
$\overset{\sigma}{\Rightarrow} =_{Def} \overset{\tau}{\Rightarrow} \mathbf{o} \overset{\sigma}{\rightarrow} \mathbf{o} \overset{\tau}{\Rightarrow} \qquad (\forall \sigma \in \Sigma_{obs})$
où **o** désigne la composition des relations

Motivation/Construction de $\Rightarrow$ sur un exemple

$\Rightarrow$ (Rappel)

$\overset{\tau}{\Rightarrow} =_{Def}$ Fermeture réflexive et transitive de $\overset{\tau}{\rightarrow}$ avec $\overset{\tau}{\rightarrow} =_{Def} \bigcup_{i \in \Sigma \setminus \Sigma_{obs}} \overset{i}{\rightarrow}$

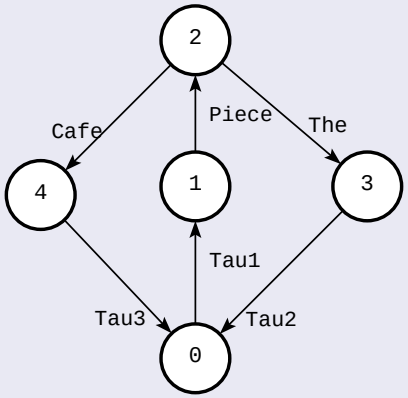
$\overset{\sigma}{\Rightarrow} =_{Def} \overset{\tau}{\Rightarrow} \circ \overset{\sigma}{\rightarrow} \circ \overset{\tau}{\Rightarrow} \qquad (\forall \sigma \in \Sigma_{obs})$

(Quelques) Propriétés de $\Rightarrow$

(P1) : Si $s \overset{i}{\rightarrow} q$ alors $s \overset{\tau}{\Rightarrow} q \qquad (\forall i \notin \Sigma_{Obs})$

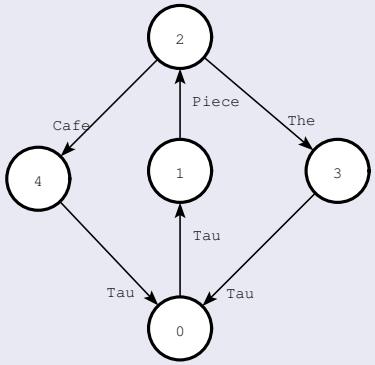
(P2) : Si $s \overset{O}{\rightarrow} q$ alors $s \overset{O}{\Rightarrow} q \qquad (\forall O \in \Sigma_{Obs})$

LTS initial : $\mathcal{X}$
$\Sigma_{Obs} = \{Piece, The, Cafe\}$



Début du calcul de $\Rightarrow$ ...

Prise en compte de (P1) et (P2)



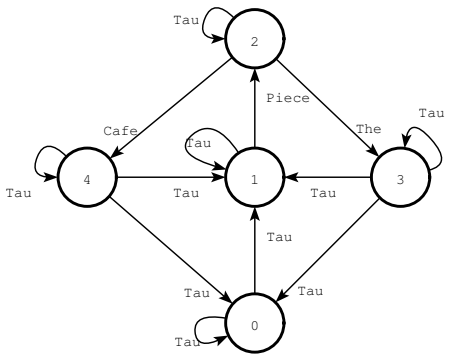
Banalisation des événements inobservables

Motivation/Construction de $\Rightarrow$ sur un exemple (suite)

$\overset{\tau}{\Rightarrow} =_{Def}$ Fermeture réflexive et transitive de $\overset{\tau}{\rightarrow}$ avec $\overset{\tau}{\rightarrow} =_{Def} U_{i \in \Sigma \setminus \Sigma_{obs}} \overset{i}{\rightarrow}$
$\overset{\sigma}{\Rightarrow} =_{Def} \overset{\tau}{\Rightarrow} \circ \overset{\sigma}{\rightarrow} \circ \overset{\tau}{\Rightarrow} \qquad (\forall \sigma \in \Sigma_{obs})$

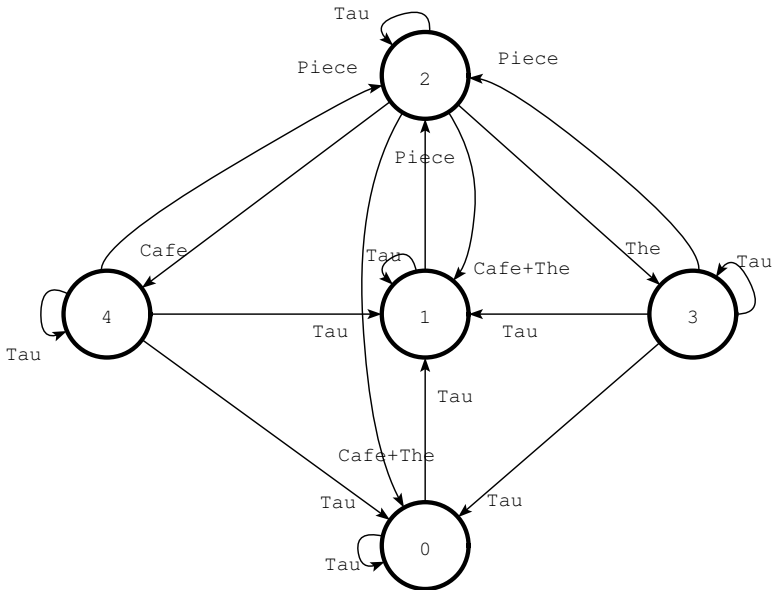
Calcul de $\overset{\tau}{\Rightarrow}$

Réflexivité ;
$0 \overset{\tau}{\Rightarrow} 0, 1 \overset{\tau}{\Rightarrow} 1 \ldots$
Transitivité :
$4 \overset{\tau}{\rightarrow} 0$ et $0 \overset{\tau}{\rightarrow} 1$ donne
$4 \overset{\tau}{\Rightarrow} 1$
...



Calcul de $\overset{O}{\Rightarrow} \quad (0 \in \Sigma_{Obs})$

$0 \overset{\tau}{\Rightarrow} 1$ et $1 \overset{piece}{\Rightarrow} 2$ donne $0 \overset{piece}{\Rightarrow} 2$
...
$+ \; 4 \overset{piece}{\Rightarrow} 2, 3 \overset{piece}{\Rightarrow} 2,$
$+ \; 2 \overset{the}{\Rightarrow} 0, 2 \overset{the}{\Rightarrow} 1$
$+ \; 2 \overset{cafe}{\Rightarrow} 0, 2 \overset{cafe}{\Rightarrow} 1$



Equivalence Observationnelle faible ($\sim_{\Sigma_{Obs}}$)

Définition récursive de $\sim_{\Sigma_{Obs}}$

Eq. observationnelle faible = Eq. observationnelle forte sur le LTS abstrait

$\mathcal{X}$ un LTS : $< S, s_0, \Sigma, \rightarrow >$, $\Sigma_{Obs} \subset \Sigma$ et
$\mathcal{X}_{Obs} =< S, s_0, \Sigma_{Obs} \cup \{\tau\}, \Rightarrow >$, le LTS abstrait associé
On note $\sim_{\Sigma_{Obs}}$, l'équivalence observationnelle faible

$s \sim_{\Sigma_{Obs}} q$ ssi $\forall t \in \Sigma_{Obs} \cup \{\tau\}$
- $\forall s'$ Si $s \overset{t}{\Rightarrow} s'$ Alors $\exists q' : q \overset{t}{\Rightarrow} q'$ avec $s' \sim_{\Sigma_{Obs}} q'$
- $\forall q'$ Si $q \overset{t}{\Rightarrow} q'$ Alors $\exists s' : s \overset{t}{\Rightarrow} s'$ avec $q' \sim_{\Sigma_{Obs}} s'$

Définition opérationnelle équivalente

$\sim_{\Sigma_{Obs}} = \cap_{n \geq o} \sim_{(\Sigma_{Obs}, n)}$ où $\sim_{(\Sigma_{Obs}, 0)} = S \times S$

$s \sim_{(\Sigma_{Obs}, k+1)} q$ ssi $s \sim_{(\Sigma_{Obs}, k)} q$ et $\forall t \in \Sigma_{Obs} \cup \{\tau\}$
- $\forall s'$ Si $s \overset{t}{\Rightarrow} s'$ Alors $\exists q' : q \overset{t}{\Rightarrow} q'$ avec $s' \sim_{(\Sigma_{Obs}, k)} q'$
- $\forall q'$ Si $q \overset{t}{\Rightarrow} q'$ Alors $\exists s' : s \overset{t}{\Rightarrow} s'$ avec $q' \sim_{(\Sigma_{Obs}, k)} s'$

nb : $\sim$ est définie car $(\sim_{(\Sigma_{Obs}, i)}) \downarrow$

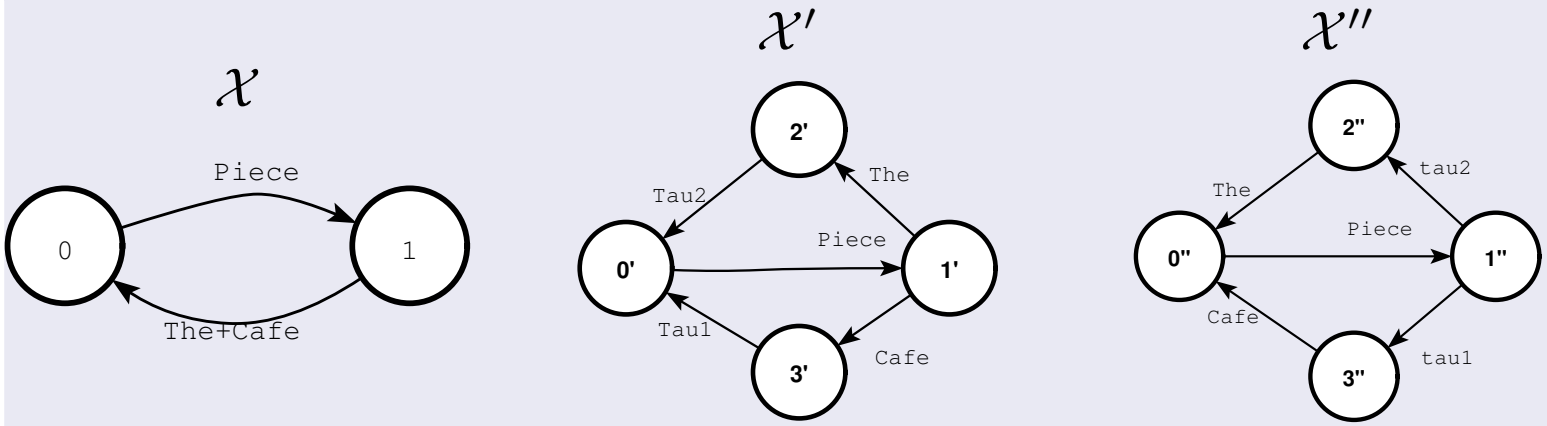
Propriété : $\Sigma_1 \subset \Sigma_2 \Rightarrow \sim_{\Sigma_2} \subset \sim_{\Sigma_1}$

Extension de $\sim_{\Sigma_{Obs}}$ aux LTS

Définition

Deux LTS, $\mathcal{X}_1$ et $\mathcal{X}_2$, sont faiblement observationnellement équivalents ssi leurs etats initiaux respectifs le sont : $\mathcal{X}_1 \sim_{\Sigma_{Obs}} \mathcal{X}_2$ ssi $s_{01} \sim_{\Sigma_{Obs}} s_{02}$

Exemple d'application : 3 machines à café reconnaissant le même langage



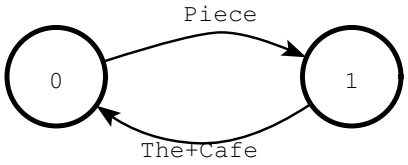
Question : l'équivalence observationnelle faible parvient-elle à les distinguer ?

Approche

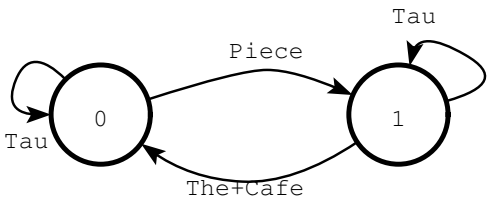
On considère les 3 LTS "de front" en considérant la "réunion" de ces 3 STE comme un seul STE.
(i.e., on calcule directement $\sim_{\Sigma_{Obs}}$ sur $(S \cup S' \cup S'') \times (S \cup S' \cup S''))$

1) Calcul des LTS abstraits ($\Sigma_{Obs} = \{Piece, The, Cafe\}$)

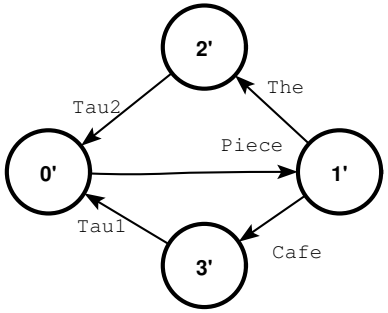
$\mathcal{X}$



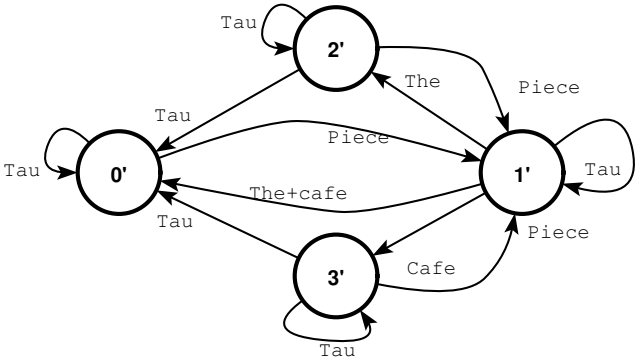
$\mathcal{X}_{\Sigma_{Obs}}$



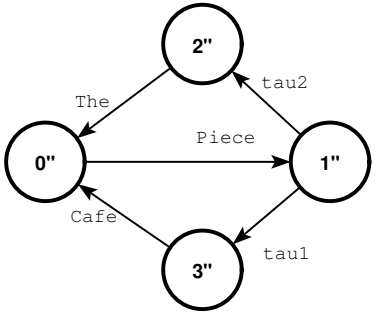
$\mathcal{X}'$



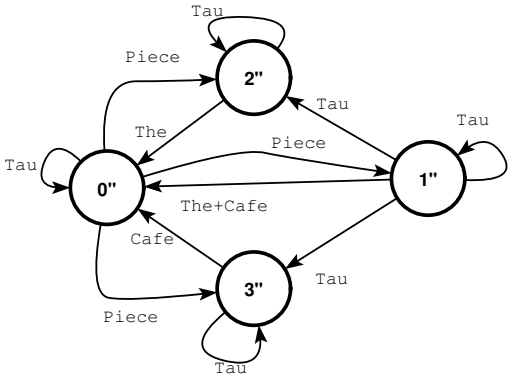
$\mathcal{X}'_{\Sigma_{Obs}}$



$\mathcal{X}''$



$\mathcal{X}''_{\Sigma_{Obs}}$



2) Calcul Σ^{-1} et de $\sim_{Out}$

$\Sigma^{-1} : 2^{\Sigma_{Obs}}$	$\mapsto$	2^S
$\{piece, cafe, the, \tau\}$	$\mapsto$	$\emptyset$
$\{cafe, the, \tau\}$	$\mapsto$	$\{1,1',1''\}$
$''\{x, y, z\}''$	$\mapsto$	$\emptyset$
$\{piece, \tau\}$	$\mapsto$	$\{0,0',2',3',0''\}$
$\{cafe, \tau\}$	$\mapsto$	$\{3''\}$
$\{the, \tau\}$	$\mapsto$	$\{2''\}$
$''\{x, y\}''$	$\mapsto$	$\emptyset$
$\{\tau\}$	$\mapsto$	$\emptyset$
$''\{x\}''$	$\mapsto$	$\emptyset$
$\emptyset$	$\mapsto$	$\emptyset$

$\Rightarrow \sim_{Out} = \{\{0, 0', 2', 3', 0''\}, \{1, 1', 1''\}, \{2''\}, \{3''\}\}$

Remarque

Comme $\overset{\tau}{\Rightarrow}$ est réflexive : $\Sigma^{-1}(\emptyset) = \emptyset$

τ n'est donc pas discriminant : i.e., $\forall E \in 2^{\Sigma_{Obs}} : \tau \notin E \Rightarrow \Sigma^{-1}(E) = \emptyset$

Propriété $\sim_{Out} = \sim_{(\Sigma_{Obs},1)}$

Pas de chgt wrt Eq. obs forte

nb : Par la suite, on note $\sim_k$ au lieu de $\sim_{(\Sigma_{Obs},k)}$

3) Calcul des $\sim_k$ $(k \geq 2)$

Propriétés Equivalence Observationnelle Faible (suite)

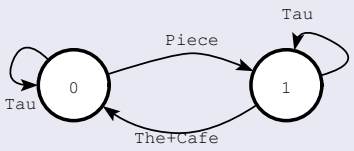
Eq Obs Forte : $s_1, s_2 \in \Sigma^{-1}(\emptyset) \Rightarrow s_1 \sim s_2$ (Rappel)

Inutilisable pour l'equ. faible ..... par contre on a :

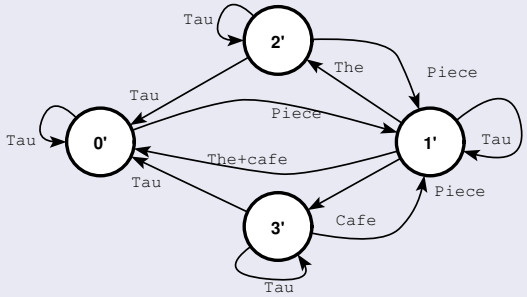
Eq Obs Faible : $s_1, s_2 \in \Sigma^{-1}(\{\tau\}) \Rightarrow s_1 \sim s_2$

Calcul de $\sim_2$

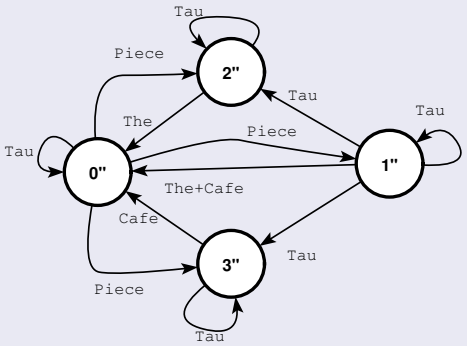
$\mathcal{X}_{\Sigma_{Obs}}$



$\mathcal{X}'_{\Sigma_{Obs}}$



$\mathcal{X}''_{\Sigma_{Obs}}$



$\sim_1 = \{\{0, 0', 2', 3', 0''\}, \{1, 1', 1''\}, \{2''\}, \{3''\}\}$

Quid de $\{0, 0', 2', 3', 0''\}$ à l'ordre 2

0	*Piece*	$\{1_{\sim_1}\}$
0	τ	$\{0_{\sim_1}\}$
$0'$	*Piece*	$\{1_{\sim_1}\}$
$0'$	τ	$\{0_{\sim_1}\}$

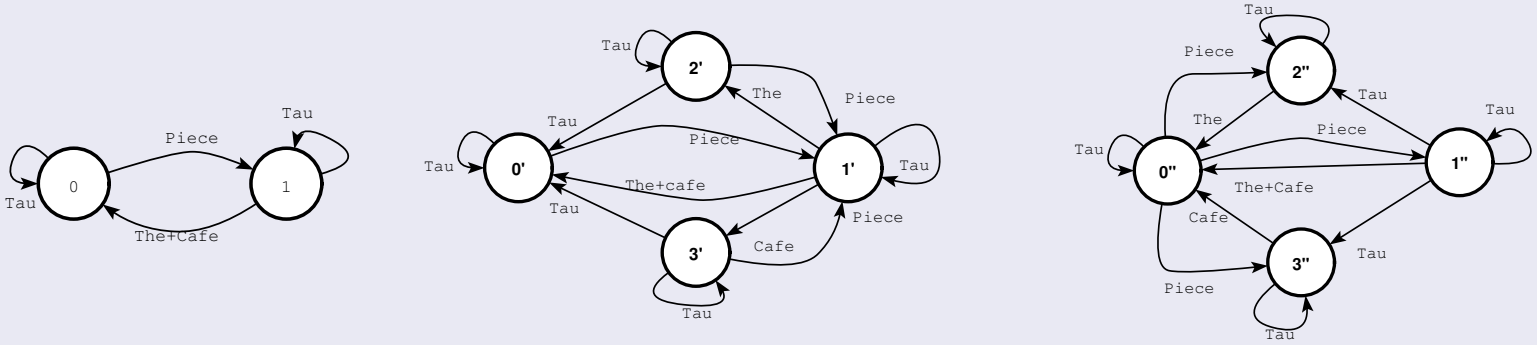
$2'$	*Piece*	$\{1_{\sim_1}\}$
$2'$	τ	$\{0_{\sim_1}\}$
$3'$	*Piece*	$\{1_{\sim_1}\}$
$3'$	τ	$\{0_{\sim_1}\}$

$0''$	*Piece*	$\{0_{\sim_1}, 2'_{\sim_1}, 3'_{\sim_1}\}$
$0''$	τ	$\{0_{\sim_1}\}$

$\mapsto \{0, 0', 2', 3\}, \{0''\}$

$\mathcal{X}'' \not\sim_{\Sigma_{Obs}} \mathcal{X}'$ et $\mathcal{X}'' \not\sim_{\Sigma_{Obs}} \mathcal{X}$

3) Calcul de $\sim$ (fin)



$\sim_1 = \{\{0, 0', 2', 3', 0''\}, \{1, 1', 1''\}, \{2''\}, \{3''\}\}$

$\sim_2 \supset \{\{0, 0', 2', 3'\}, \{0''\}, \{2''\}, \{3''\}\}$

$\{1, 1', 1''\}$ à l'ordre 2 ?

1	*The*	$\{0_{\sim_1}\}$
1	*Cafe*	$\{0_{\sim_1}\}$
1	τ	$\{1_{\sim_1}\}$
$1'$	*The*	$\{0_{\sim_1}\}$
$1'$	*Cafe*	$\{0_{\sim_1}\}$
$1'$	τ	$\{1_{\sim_1}\}$
$1''$	*The*	$\{0_{\sim_1}\}$
$1''$	*Cafe*	$\{0_{\sim_1}\}$
$1''$	τ	$\{1_{\sim_1}, 2''_{\sim_1}, 3''_{\sim_1}\}$

Donc $\{1, 1', 1''\} \mapsto \{1, 1'\}, \{1''\}$

nb : Avec intuition : $1'' \not\sim 1'(1)\}$ 2" et 3" sont "singuliers" : ils n'admettent aucun état équivalent dans $\mathcal{X}$ ou $\mathcal{X}'$.

Les prédécesseurs de 2" et 3" (1" par ex) sont aussi singuliers.

Exercice : terminer les calculs

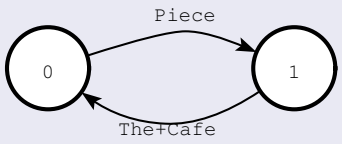
Montrer que $\sim = \sim_2$

En déduire que $\mathcal{X} \not\sim_{\Sigma_{Obs}} \mathcal{X}'$

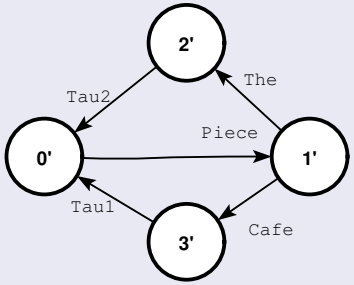
De quoi a-t-on fait abstraction ?

$\sim = \{\{0, 0', 2', 3'\}, \{0''\}, \{1, 1'\}, \{1''\}, \{2''\}, \{3''\}\}$

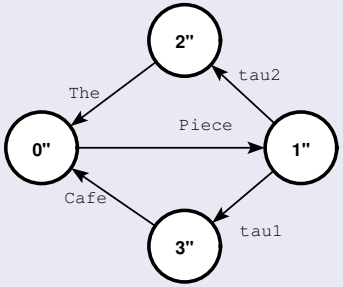
$\mathcal{X}$



$\mathcal{X}'$



$\mathcal{X}''$



$\sim = \{\{0\}, \{1\}\}$ $\quad \sim = \{\{0', 2', 3'\}, \{1'\}\}$ $\quad \sim = \{\{0''\}, \{1''\}, \{2''\}, \{3''\}\}$

Transition inobservable : interne/externe

$\mathcal{X}$ un LTS : $< S, s_0, \Sigma, \rightarrow >$, $\Sigma_{Obs} \subset \Sigma$ et $\sim_{\Sigma_{Obs}}$, l'équivalence associée

- $t_i \in \Sigma \setminus \Sigma_{Obs}$ est **interne** ssi $s \xrightarrow{t_i} q \Rightarrow s \sim_{\Sigma_{Obs}} q \quad (\forall s, q \in S)$
- Une transition non interne est dite **externe**.

Analyse des transitions "non observées"

Pour $\mathcal{X}'$: $tau1$, et $tau2$ sont internes et ne changent pas le comportement observé ; on peut en faire abstraction.

Pour $\mathcal{X}''$: $tau1$, et $tau2$ sont externes. On ne peut pas en faire abstraction, suivant leur occurrence, on obtiendra du thé ou du café

Projection Observationnelle faible

Projection observationnelle (faible) (LTS Quotient)

$\mathcal{X}$ un LTS : $< S, s_0, \Sigma, \rightarrow >$, $\Sigma_{Obs} \subset \Sigma$ et $\sim_{\Sigma_{Obs}}$, l'équivalence associée
$\mathcal{X}/\sim_{\Sigma_{Obs}} =_{Def} < S_{\sim_{\Sigma_{Obs}}}, s_{0\sim_{\Sigma_{Obs}}}, \Sigma, \rightarrow_{\sim_{\Sigma_{Obs}}} >$ où

- $S_{\sim_{\Sigma_{Obs}}} =_{Def} S/\sim_{\Sigma_{Obs}}$, le quotient de S par $\sim_{\Sigma_{Obs}}$
- $s_{0\sim_{\Sigma_{Obs}}}$, la classe d'équivalence de s_0 pour $\sim_{\Sigma_{Obs}}$
- $\rightarrow_{\sim_{\Sigma_{Obs}}}$, le plus petit ensemble vérifiant :

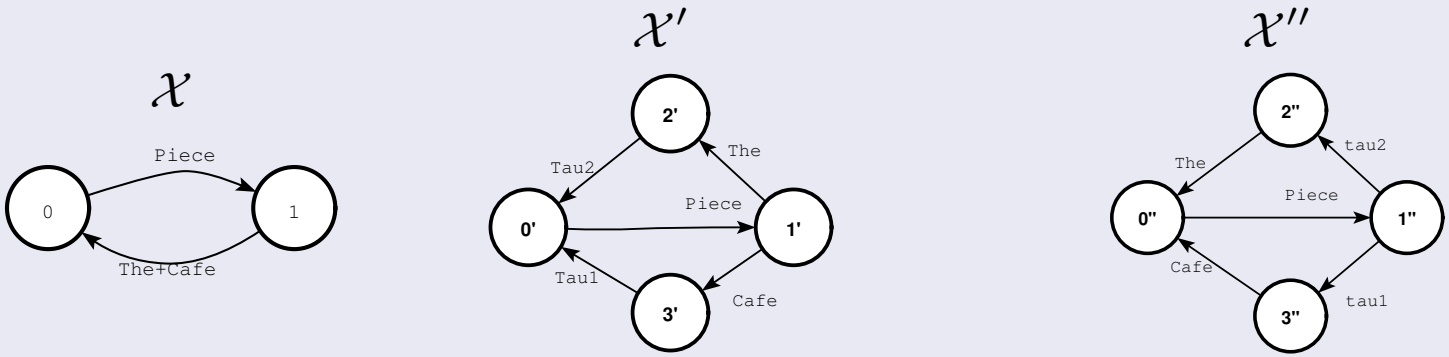
  1. Pour $t \in \Sigma_{Obs}$: Si $s \xrightarrow{t} q$ alors $(s_{\sim_{\Sigma_{Obs}}}, t, q_{\sim_{\Sigma_{Obs}}}) \in \rightarrow_{\sim_{\Sigma_{Obs}}}$
  2. Pour $t \in \Sigma \setminus \Sigma_{Obs}$: Si $s \xrightarrow{t} q$ et $s \not\sim_{\Sigma_{Obs}} q$ alors $(s_{\sim_{\Sigma_{Obs}}}, t, q_{\sim_{\Sigma_{Obs}}}) \in \rightarrow_{\sim_{\Sigma_{Obs}}}$

$\mathcal{X}/\sim_{\Sigma_{Obs}}$ est le plus petit LTS équivalent pour $\sim_{\Sigma_{Obs}}$ à $\mathcal{X}$
plus petit : en nombre d'états

Variante : Considérer $\Rightarrow$ à la place de $\rightarrow$

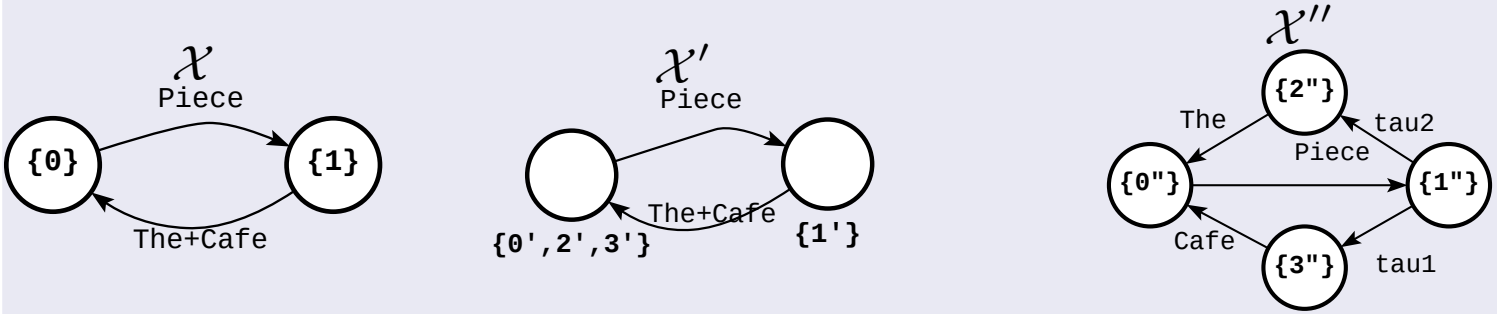
Exemple (fin)

Retour sur les 3 LTS d'origine



$\sim = \{\{0\},\{1\}\}$ $\quad \sim = \{\{0',2',3'\},\{1'\}\}$ $\quad \sim = \{\{0''\},\{1''\},\{2''\},\{3''\}\}$

Projections associées



Remarque

On savait que $\mathcal{X} \sim_{\Sigma_{Obs}} \mathcal{X}'$ et $\mathcal{X} \not\sim_{\Sigma_{Obs}} \mathcal{X}''$. On obtient donc deux projections (quotients) distinctes : une pour $\mathcal{X}$ et $\mathcal{X}'$, l'autre pour $\mathcal{X}''$

Simplification de LTS compatible avec $\sim$

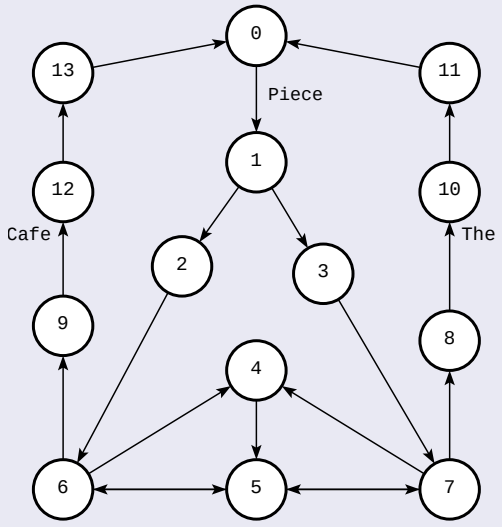
Réduction des τ-chaînes

Si $s \xrightarrow{i} q$ avec $i \notin \alpha_{Obs}$ et $s \xrightarrow{j} q' \Rightarrow (j \notin \alpha_{Obs}$ et $q' \sim q)$ Alors $s \sim q$
Application sur $\mathcal{X}'$: $0' \sim 3'$

Réduction des τ-cycles

Si $s \overset{\tau}{\Rightarrow} q$ et $q \overset{\tau}{\Rightarrow} s$ alors $s \sim q$
Pose le problème de la divergence (à suivre)

Application

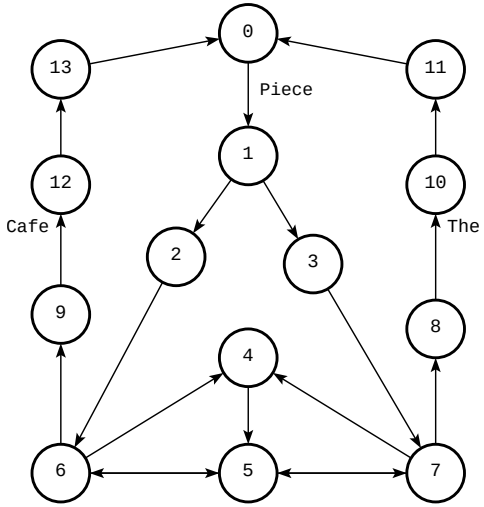


Exemple de réduction

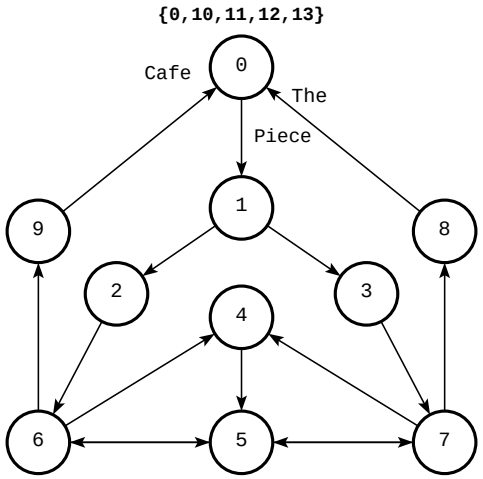
τ-chaîne :
$12 \to 13, 4 \to 5, \ldots$
$\Rightarrow 12 \sim 13,\ 4 \sim 5$
$6 \to 9$ **n'est pas** une τ-chaînes
Qu'en est-il de $1 \to 2$?

τ-cycle :
$4 \to 5 \to 6 \to 7 \to 4$
$\Rightarrow 4 \sim 5 \sim 6 \sim 7$

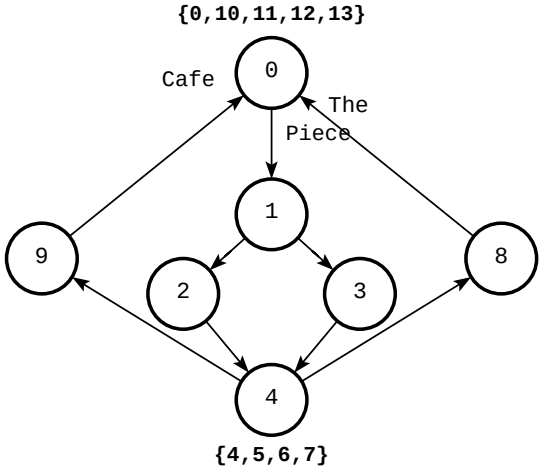
Exemple de réduction/simplification τ-chaînes, τ-cycles



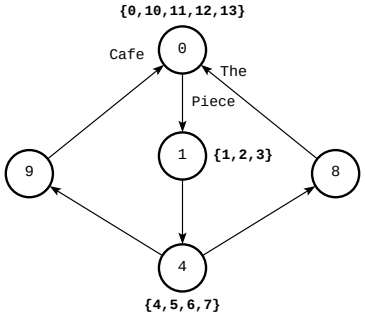
LTS Initial



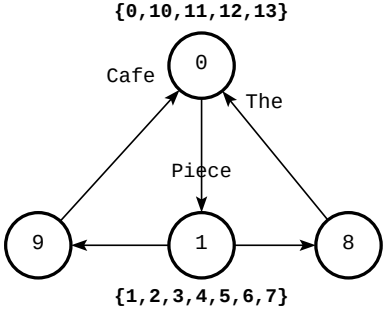
Réduction τ-chaînes :
(12,13,0) et (10,11,0)



Réduction τ-cycle :
(4,5,6,7)



Réduction
τ-chaînes :
(1,2) et (1,3)

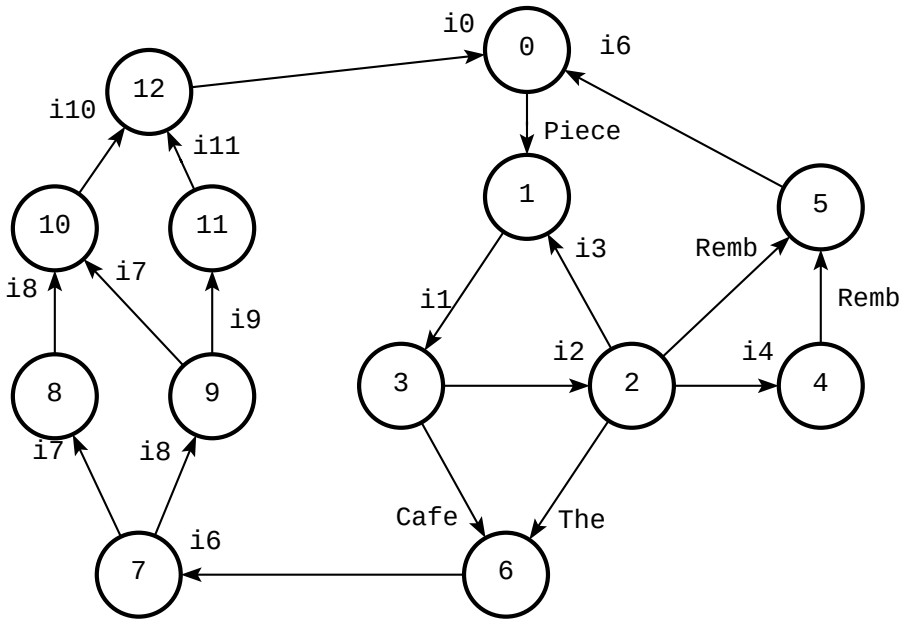


Réduction τ-chaîne :
(1,4)

Le LTS est minimal !

Exercice : Calculez la projection observationnelle du distributeur suivant

$\Sigma_{Obs} = \{Piece, The, Cafe, remb\}$



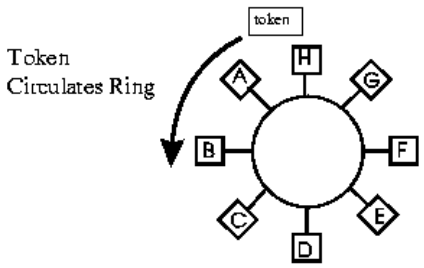
Consignes

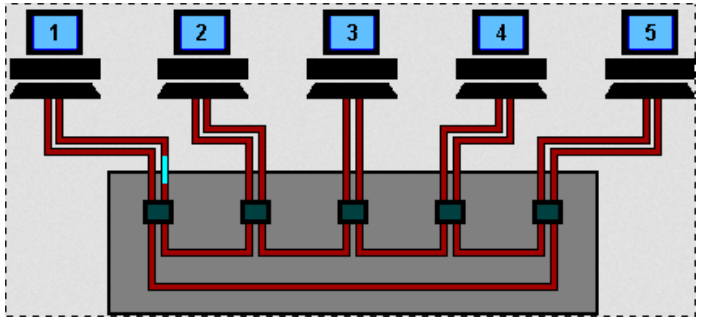
Ne pas oublier de simplifier si c'est possible !
Garder la trace des simplifications réalisées
en TP : CADP simplifie mais ne garde pas la trace des réductions opérées ;-(

Outline

5. Exemple de Vérification d'un protocole (Token-Ring)
   - Introduction
   - Spécification comportementale (via un LTS) des propriétés à vérifier
   - Première Modélisation
     - Génération du LTS
     - Outil CADP
     - Vérification
   - De la vérification à l'aide au diagnostic
     - Principe
     - Identification des problèmes et des corrections
     - Seconde modélisation
     - Vérification après correction

Vérification d'un modèle de Token-Ring





Token-Ring (rappel)

Mécanisme réparti simple permettant d'offrir de facon **équitable** un accés en exclusion mutuelle à une ressource "critique".

Propriétés à garantir

Safety/Sûreté : La ressource est utilisée par au plus un usager.
Liveness/Vivacité : Un usager en attente de la ressource l'obtiendra au bout d'un temps fini (absence de famine).

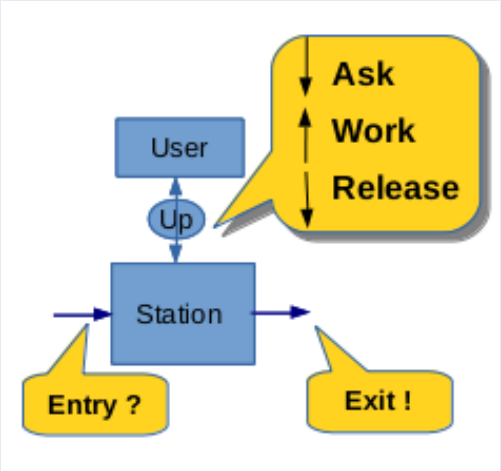
Pourquoi un anneau à jeton

Jeton : matérialise le privilège nécessaire pour l'accès à la ressource. Son unicité "garantit" la propriété d'exclusion mutuelle.
Anneau : la topologie "garantit" la circulation équitable du privilège entre les différents sites.

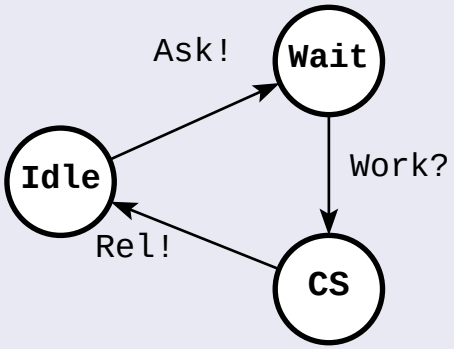
Modélisation du Token-Ring (cf TP Systèmes concurrents 4IR)

Architecture

Chaque site est composé par un usager et une station. La station communique avec ses deux voisines pour assurer la circulation du jeton et dialogue avec **son** usager.



Usager



Expression des propriétés à garantir (Uppaal)

Safety/Sûreté : Exclusion mutuelle

$(\forall i)(\forall j)(i \neq j) : A[]\ not\ (Work_i \wedge Work_j)$

Liveness/Vivacité : absence de famine

$(\forall i) : Wait_i -- > CS_i$

Un usager en attentre inévitablement travaillera

Comment les exprimer/vérifier via des LTS ? !

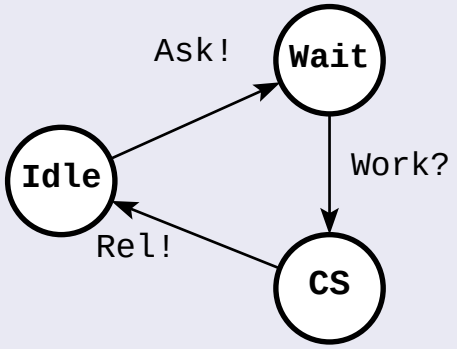
Expression de la propriété de vivacité : absence de famine

Propriété de vivacité (assertionnelle)

Liveness/Vivacité : absence de famine
$(\forall i) : Wait_i -- > CS_i$ (Uppaal)
$(\forall i) :$ "$Ask_i -- > Work_i''$ (à la Uppaal)

Comment les exprimer/vérifier via des LTS ? !
A disposition : choix des Evénements, de l'équivalence, (présence de τ-cycles).

Usager



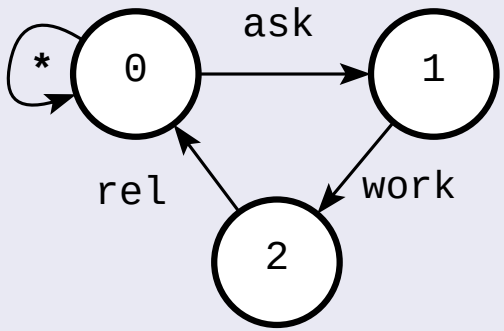
Propriété de vivacité via un LTS ?

Evénements observés : Pour un site i donné $\{Ask_i, Work_i, Rel_i\}$
Choix de l'équivalence ?
- langage ? non (pas de préservation des blocages)
- observationnelle ? oui mais en faisant "attention" à la divergence

Spécification via un LTS de la propriété de vivacité

Spécification via un LTS de la propriété de vivacité

Comportement attendu



La boucle avec $*$ matérialise la possibilité de divergence
- On peut boucler de facon interne dans 0
- On ne peut rester indéfiniment ni dans 1 ni dans 2

"Explicitation" des propriétés exprimées par le LTS

Absence de blocage

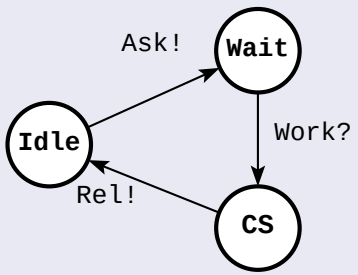
Séquencement : $Ask! - > Work? - > Rel! \ldots$

Divergence :

requise en 0 : un usager doit pouvoir rester indéfiniment dans l'état idle
prohibée en 1 : un usager ne doit pas pouvoir attendre indéfiniment
prohibée en 2 : un usager ne doit pas pouvoir travailler indéfiniment

Expression de la propriété de sûreté : exclusion mutuelle

Usager



Propriété d'exclusion mutuelle

Safety/Sûreté : Exclusion mutuelle
Jamais plus d'un usager en train de travailler
$(\forall i)(\forall j)(i \neq j) : A[]\ not\ (Work_i \wedge Work_j)$

Statique/Evénementiel

L'exclusion mutuelle est une propriété naturellement "statique"
(i.e., pour l'exprimer on utilise plus volontiers les états que les événements)
En comparaison, la propriété d'absence de famine, vue précédemment, s'exprimait aussi facilement à l'aide d'états qu'à l'aide d'événements.

Evidence : L'expression par un LTS d'une propriété événementielle est plus directe que celle d'une propriété statique

Quiz

Exprimer - de facon assertionnelle - la propriété d'exclusion mutuelle en ne faisant intervenir que des événements.

Expression de l'exclusion mutuelle via un LTS

Expression événementielle de l'exclusion mutuelle

Enoncé statique : *Jamais plus d'un usager en train de travailler*

Enoncé événementiel ?

Quels événements considérer ? Entrée(s) et Sortie(s) de chaque site

Essais de formulation :

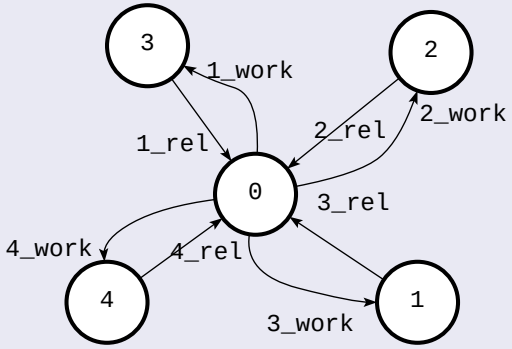
a) *Toute entrée est suivie par une sortie ?*
pas de rapport !

b) *Jamais deux entrées de suite en section critique ?*
trop fort !

c) *Entre deux entrées consécutives en section critique, il y a forcément la sortie du premier entré.*

Spécification comportementale : LTS + Equivalence

LTS



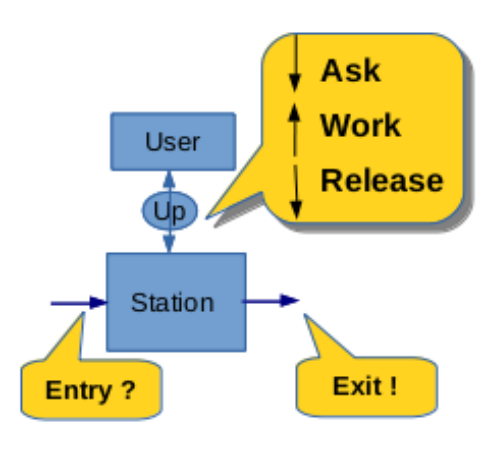
Quelle équivalence considérer ?

Propriété de séquencement :
Equivalence langage est suffisante

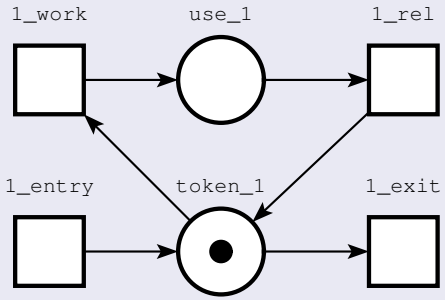
Première Modélisation (cf TP Systèmes concurrents 4IR)

Architecture & Composants

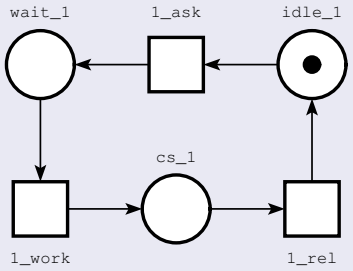
Architecture



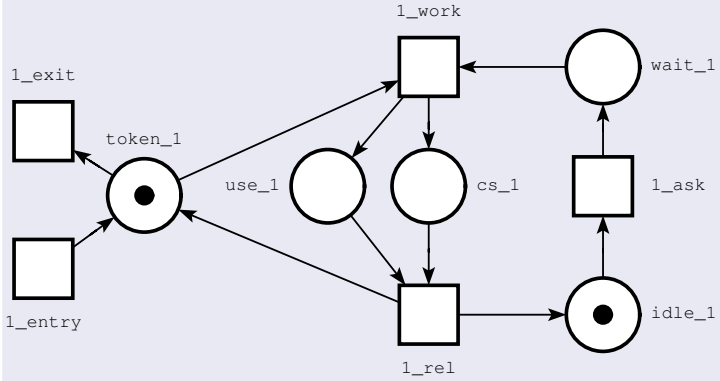
Station



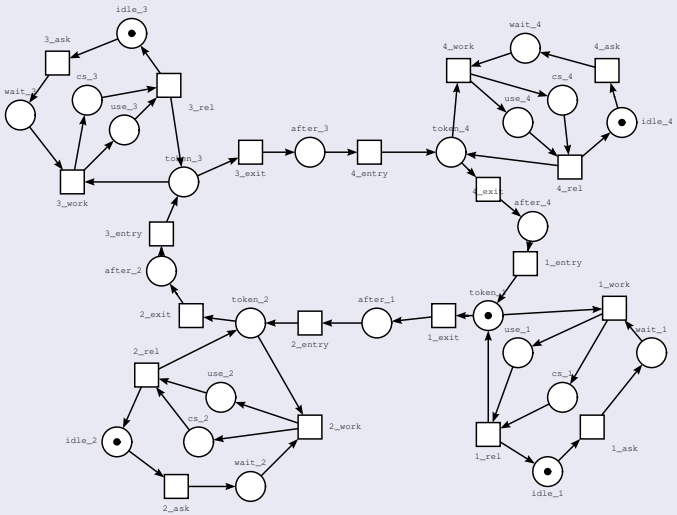
Client



Site = (Client × Station)

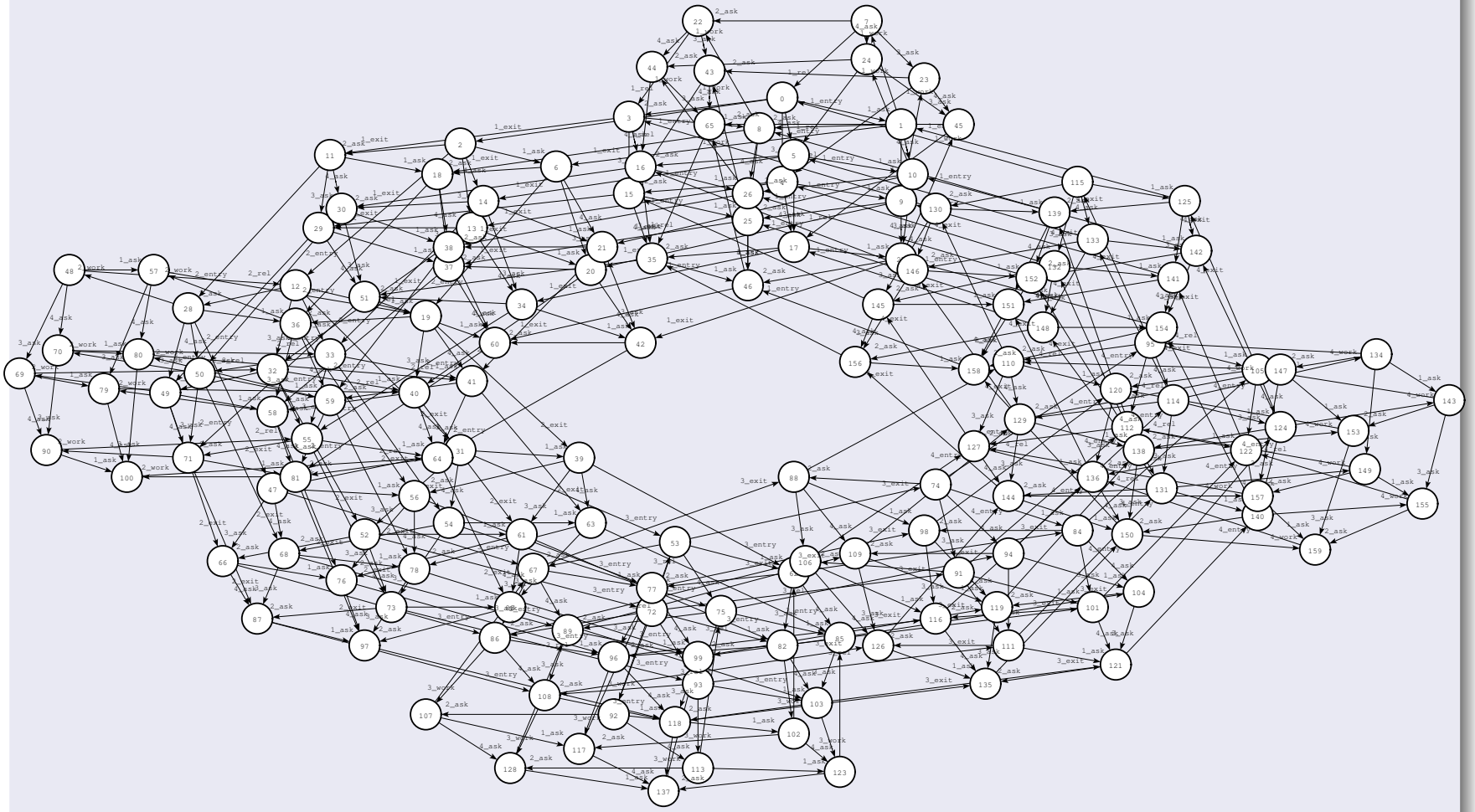


Anneau à 4 sites



Génération du LTS

Anneau à 4 sites : 160 états / 496 transitions



Besoin d'abstraction !

Outil CADP
Construction and Analysis of Distributed Processes

CADP

CADP : Software Tools for Designing Reliable Protocols and Systems
INRIA-Rhone Alpes CONVECS
(Construction of Verified Concurrent Systems)
http ://cadp.inria.fr/
TP INSA : Caesar (TDF) / BCG (VP)

BCG : (Binary-Coded Graphs)

bcg_io : conversion `.aut` $\leftrightarrow$ `.bcg`

bcg_labels : renommage (`-rename`) / masquage (`-hide`)

bcg_min : calcul d'équivalence/quotient (observationnelle, . . .)
livelocks (τ-cycles)

aldebaran : calcul de la safety-équivalence, quotient (langage+blocage)

Exposé tutorial : Model-Checking & CADP (16/10/2014)

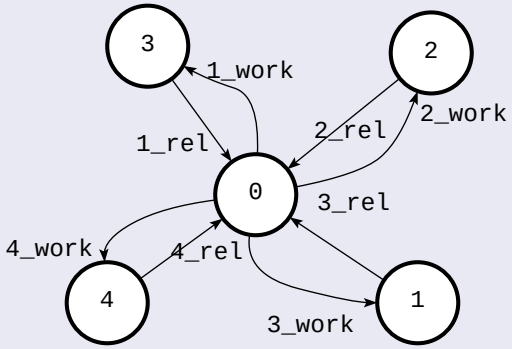
Slides & vidéo : http ://projects.laas.fr/IFSE/FMF/J4/

Vérification avec CADP de l'exclusion mutuelle

Propriété d'exclusion mutuelle : *Entre deux entrées consécutives en section critique, il y a forcément la sortie du premier entré.*

Spécification comportementale : LTS + Equivalence

LTS attendu



Safety Equivalence
(plus fine que langage)

$\Sigma_{Obs} = \bigcup_{i \in [1,4]} \{i_rel, i_work\}$

```
mutex.hide

hide all but
 "1_rel"
 "1_work"
 "2_rel"
 "2_work"
  .../...
```

Mise en oeuvre en CADP

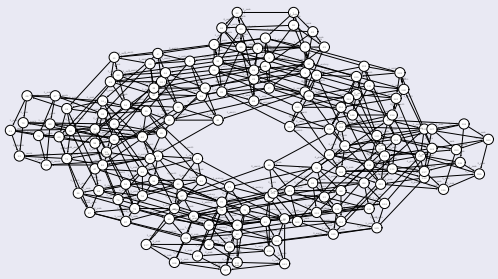
```
bcg_io tk4.aut tk4.bcg        -- conversion du .aut en .bcg
bcg_labels -hide mutex.hide  tk4.bcg tk4mutex.bcg
                -- masquage des événements not in Sigma_Obs
bcg_io tk4mutex.bcg tk4mutex.aut          -- retour en .aut
aldebaran -smin tk4mutex.aut > tk4mutexsmin.aut
                          -- calcul de la projection safety
nd tk4mutexsmin.aut                  -- visualisation avec nd
```

Vérification exclusion mutuelle

Vérification par projection/introspection

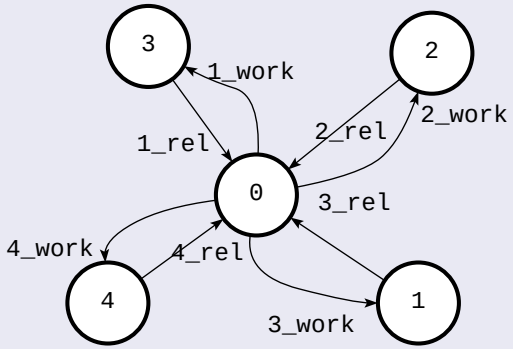
1. Calcul équivalence wrt Σ_{obs}
2. Calcul du quotient (projection) du LTS initial
   Synthèse du Service offert
3. Comparaison avec le service attendu

LTS concret



Projection / Σ_{obs}

Service **offert**

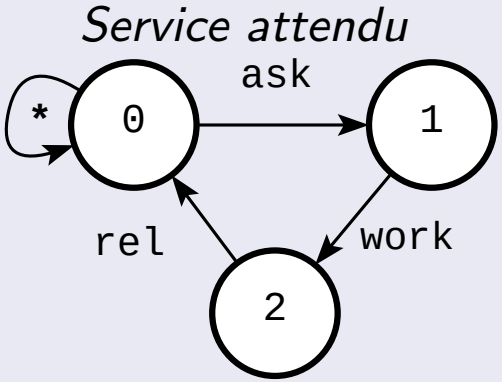


Identique au Service **attendu**

$\mapsto$ **Exclusion mutuelle satisfaite**

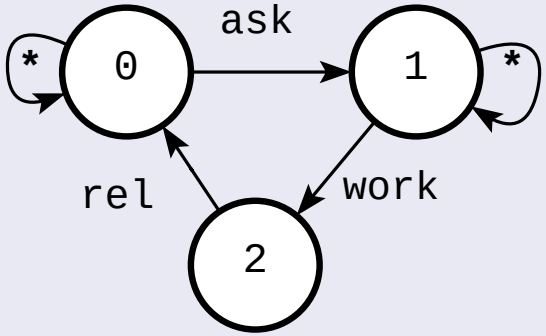
Vérification de l'absence de famine

Spécification via un LTS de l'absence de famine (Rappel)



Absence de blocage
+ **Séquencement :** $Ask! - > Work? - > Rel!$
+ **Divergence :**
requise en 0 : "repos éternel possible"
prohibée en 1 : "pas d'attente infinie"
prohibée en 2 : "pas de travail à perpétuité"

Service **offert**



Differe du
Service **attendu**

Service Attendu Vs Offert

OK : Blocage, Séquencement
KO : Divergence
Presence de τ-cycle dans la classe 1
$\mapsto$ **Famine possible**

s
Vérification : Non satisfaction de la propriété $\mapsto$ Correction du modèle

De la vérification à l'aide au diagnostic

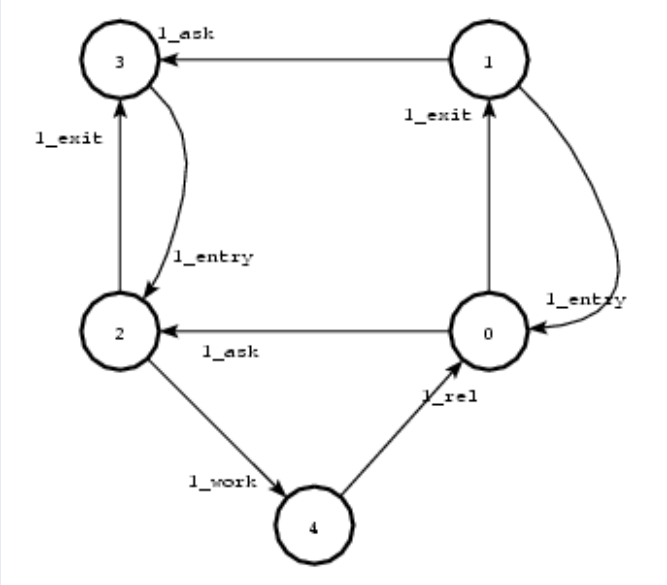
Principe :

Re-utiliser les techniques de projection/introspection pour "comprendre" pourquoi une propriété n'est pas satisfaite et aider ainsi à la mise en place de corrections.

1. "localiser les comportements/composants" potentiellement fautifs
   composant isolé ou "coalition" de composants ?
2. "Observer" leur comportement "effectif" dans le système

Mise en oeuvre sur le modèle de Token-Ring

- Composant potentiellement fautif : station(s)
- Observation de tous les événements (communication ou non) de la station
- Synthèse du comportement effectif $\Rightarrow$

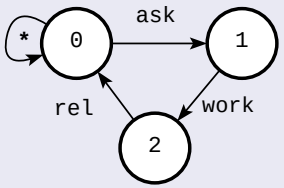


De la vérification à l'aide au diagnostic

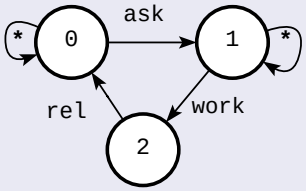
Service attendu Vs Comportements offerts

$$\Sigma_{2obs} = \Sigma_{1obs} \cup \{entry, exit\}$$

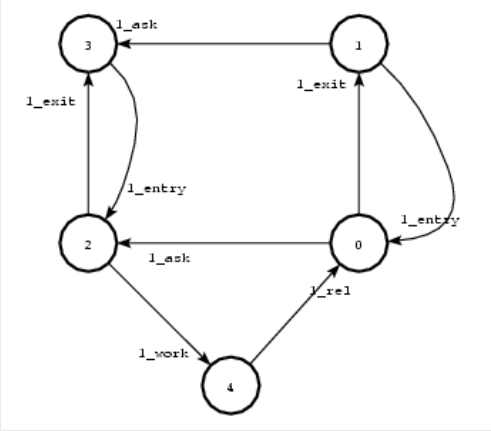
$$\Sigma_{1obs} = \{ask, work, rel\}$$



Attendu



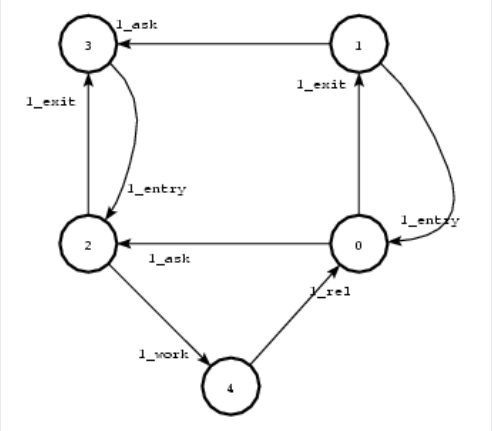
Offert



Offert

Remarque

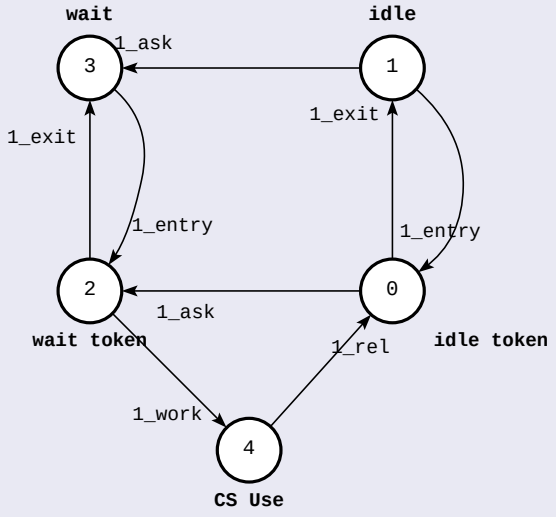
Σ_{2obs} prend en compte tous les événements du site (client $\times$ station), on est capable d'associer à chaque classe de comportement l'état du site.

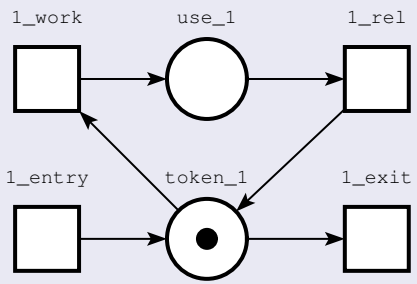


Classe	*Client*	*Station*
0	*idle*	
1	*idle*	*token*
2	*wait*	*token*
3	*wait*	
4	*cs*	*use*

Analyse du comportement offert par le site

Analyse des cycles : Identification des pbs



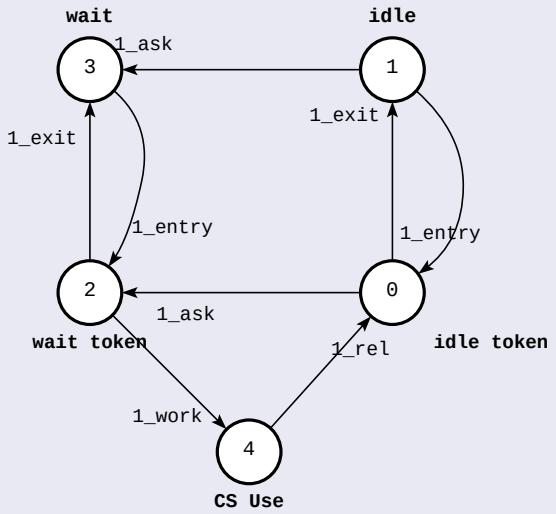


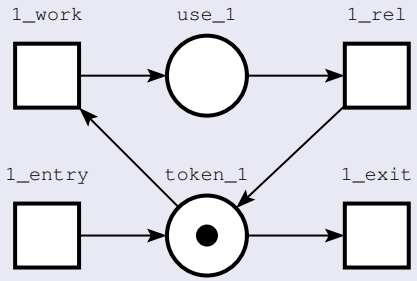
Comportement de la Station

- $0 - exit - > 1 - entry - > 0$: *correspond à la divergence attendue dans "0" ; Pour les états de cette classe le client est idle*
- $2 - exit - > 3 - entry - > 2$: *correspond à la divergence inattendue mais constatée dans "1". Pour les états de cette classe le client est wait, la station devrait prioritairement servir son site au lieu de transmettre le jeton à la station suivante* $\mapsto$ *premier pb*
- $0 - ask - > 2 - work - > 4 - rel - > 0$: *correspond à une monopilisation du jeton par le site* <u>*observé*</u> *qui peut provoquer une famine chez les sites* <u>*non observés*</u>*. Un site ne doit pas pouvoir monopoliser le jeton* $\mapsto$ *sd pb*

Correction de la station

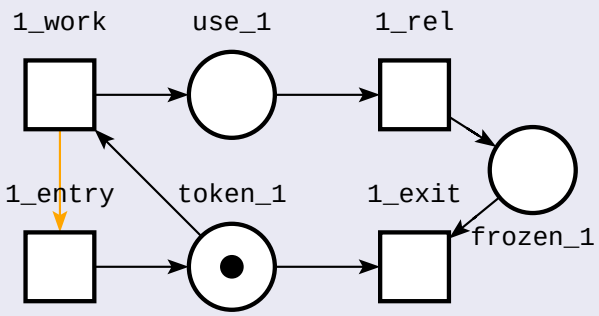
Prise en compte des corrections





Comportement de la Station

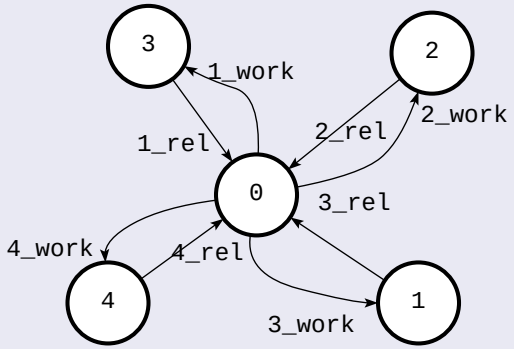
- $2 - exit - > 3 - entry - > 2$: *La station devrait prioritairement servir son site au lieu de transmettre le jeton à la station suivante* $\mapsto$ *rendre* `work` *prioritaire sur* `exit`
- $0 - ask - > 2 - work - > 4 - rel - > 0$: *Un site ne doit pas pouvoir monopoliser le jeton* $\mapsto$ *forcer à rendre le jeton après usage*



Corrections de la Station

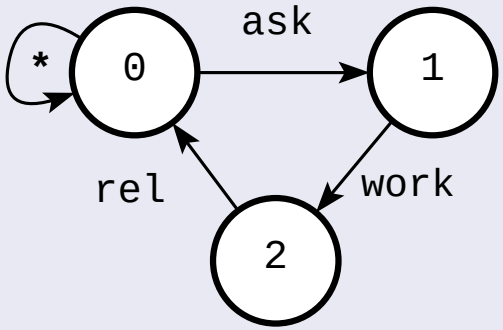
Vérification après correction

Exclusion mutuelle



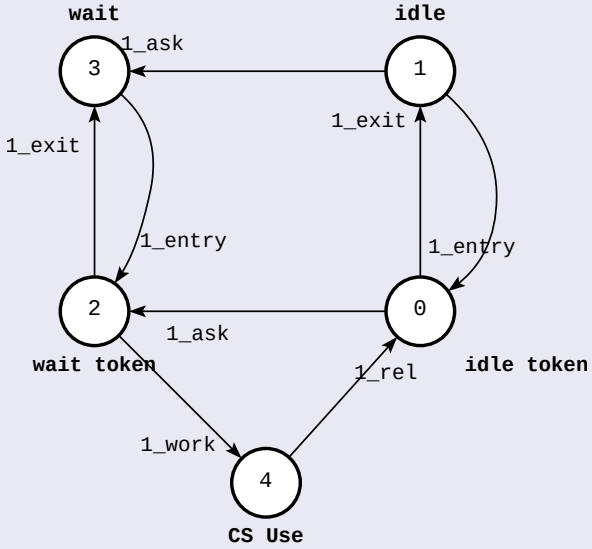
OK

Absence de Famine

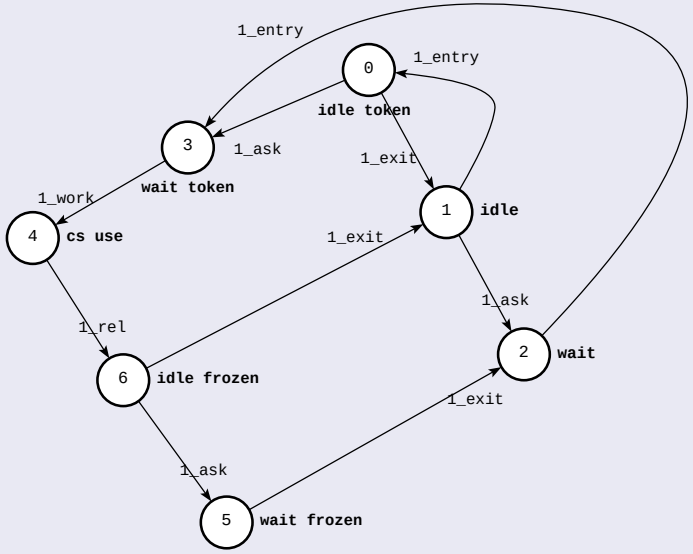


OK

Impact des modifications sur le comportement effectif du site



Avant



Après

Outline

6. Expressivité des approches comportementales
   - Bisimulation, Simulation, Co-Simulation
     - Bisimulation
     - Simulation
     - Simulation
   - Logique d'Hennessy-Milner
     - Calcul Propositionnel (rappels)
     - Logiques Modales (Propositionnelles)
     - Structures de Kripke
     - Logique d'Hennessy-Milner
   - Caractérisation modale des Equivalences de comportement
     - Théorème d'Hennessy-Milner
     - Bisimulation et Divergence

Expressivité des approches comportementales

Objectifs

- Etudier l'expressivité des différentes relations d'équivalence vues précédemment et par suite l'expressivité des approches comportementales.
- Comparer/rapprocher les techniques de vérification comportementales et assertionnelles.

Principe/Moyens : Donner une caractérisation en logique modale des équivalences de comportement.

Ingrédients

1. Simulation / Bisimulation [Park 81] - cadre abstrait permettant de généraliser les équivalences déjà rencontrées.
2. $\mathcal{HML}$ - [Hennesy-Milner 85] - une logique *modale* "simple" permettant de spécifier formellement les pouvoirs de vérification respectifs des différentes équivalences de comportement.

$\mapsto$ Caractérisation modale des approches comportementales

Bisimulation [Park 1981]

Bisimulation

$\mathcal{K} =< S, s_0, \Sigma, \rightarrow >$ et $\mathcal{K}' =< S', s'_0, \Sigma, \rightarrow >$ deux LTS

Une relation binaire R ($R \subset S \times S'$) est une bisimulation ssi
$\forall (p, p') \in R$ et $\forall t \in \Sigma$:

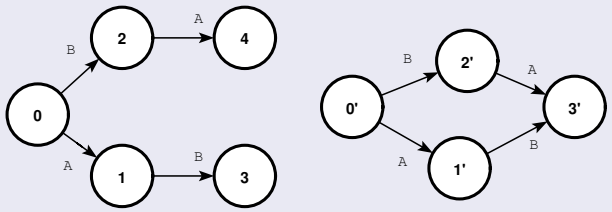
[Si $p \xrightarrow{t} q$ alors $\exists q' \in S' : p' \xrightarrow{t} q'$ et $(q, q') \in R$]

ET

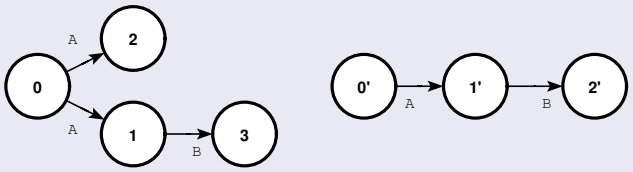
[Si $p' \xrightarrow{t} q'$ alors $\exists q \in S : p \xrightarrow{t} q$ et $(q, q') \in R$]

$\mathcal{K}$ et $\mathcal{K}'$ sont bisimilaires ssi $\exists$ bisimulation R : $(s_0, s'_0) \in R$

Exemples :



Deux LTS bisimilaires
R = {(0,0'),(1,1'),(2,2'),(3,3'),(4,3')}
est une bisimulation avec $(0, 0') \in R$



Deux LTS non bisimilaires
Par l'absure, soit B une bisimulation $\ni (0, 0')$ on aurait forcément $(2, 1') \in B$ or $1' \xrightarrow{B}$ tandis que $2 \not\xrightarrow{B}$

Propriétés des bisimulations

L'inverse d'une bisimulation est une bisimulation.
La composée de 2 bisimulations est une bisimulation.
La réunion de 2 bisimulations est une bisimulation.

Equivalence Forte ($\sim$)

$p \sim q$ *ssi* $\exists$ une bisimulation B telle que $(p, q) \in B$

$\sim$ *est réflexive car l'identité est une bisimulation. Symétrie et transitivité proviennent resp. que l'ensemble des bisimulations est stable respectivement par inversion et par composition.*

Prop : L'équivalence forte est la plus grande bisimulation.

Bisimulation engendrée par une relation R

$E : 2^{S \times S'} \mapsto 2^{S \times S'}$ définie par $(s, s') \in E(R)$ *ssi* $(1) \wedge (2) \wedge (3)$ où

(1) $(s, s') \in R$

(2) $\forall t \in T, \forall q \in S : s \xrightarrow{t} q \Rightarrow \exists q' \in S' : s' \xrightarrow{t} q'$ avec $(q, q') \in R$

(3) $\forall t \in T, \forall q' \in S' : s' \xrightarrow{t} q' \Rightarrow \exists q \in S : s \xrightarrow{t} q$ avec $(q', q) \in R$

E est une fonction décroissante dans un treillis complet, elle admet donc un plus grand point fixe : i.e, $\exists R^\omega$ telle que $E(R^\omega) = R^\omega$

Prop : R^ω est la plus grande bisimulation dans R.

Si $R = S \times S'$ alors R^ω égale l'équivalence forte (i.e., observationnelle)

Simulation

Généralités

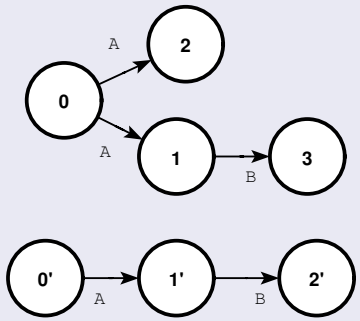
La notion de simulation - une demi bisimulation - permet de définir un pré-ordre (une relation binaire réflexive et transitive) sur les STE. La simulation est définie en rompant la symétrie de la bisimulation

Définition

$R \subset S \times S'$ est une Simulation ssi
$\forall (p, p') \in R$ et $\forall t \in T$: [Si $p \xrightarrow{t} q$ alors $\exists q' \in S'$: $p' \xrightarrow{t} q'$ et $(q, q') \in R$]

Σ Simule Σ' ssi $\exists$ Simulation R : $(s_0, s'_0) \in R$

Exemple (suite)



$R_{EE'} = \{(0, 0'), (1, 1'), (2, 1'), (3, 2')\}$ est une relation de simulation entre E et E'.
$R_{E'E} = \{(0', 0), (1', 1), (2', 3)\}$ est une relation de simulation entre E' et E.

Co-simulation

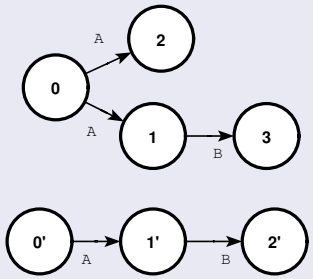
Généralités

La **co-simulation** *permet d'obtenir une relation d'équivalence à partir du pré-ordre de simulation.*

Définition

Σ co-simule $\Sigma' ssi \Sigma$ simule Σ' et Σ' simule Σ.

Exemple (suite)



E simule E' et E' simule E
E et E' sont co-similaires
Mais E et E' ne sont pas bisimilaires

Co-simulation Vs Bisimulation

On peut être co-similaire sans être bisimilaire (cf ex précédent)

Prop : Si R est une simulation de E sur E' et R^{-1}, la relation inverse, est une simulation de E' sur E alors R est une bisimulation.
nb : La derniere section précise les différences entre co-simulation et bisimulation.

$\mathcal{HML}$: Logique d'Hennessy-Milner

Logique d'Hennessy-Milner (1985)

Logique Modale (temporelle) propostitionnelle
i.e., Extension modale du calcul propositionnel $\mathcal{CP}(P)$
Formules évaluées sur une structure de Kripke étiquetée

Historique Logiques Modales

Aristote (322 A J.C) / Clarence Lewis ($\approx$ 1910) Forme Actuelle

Logiques Temporelles ($\geq$ 1980)
- Manna-Pnuelli (WI)
- Clark-Emerson (CMU)
- Sifakis (Verimag)

Logiques Epistemologiques ($\geq$ 1970)

Logiques Déontiques, etc, etc

1996 ACM Turing : Pnuelli – Logique temporelle

2007 ACM Turing : Clarke, Emerson, & Sifakis – Model-Checking

Logique : Terminologie

Logique = Syntaxe + Axiomatique + Sémantique

- Syntaxe $\rightsquigarrow$ Définition du langage
- Axiomatique $\rightsquigarrow$ Mécanisme déductif
  (Axiomes + Règles d'Inférence, $\vdash$)
- Sémantique $\rightsquigarrow$ Validité des formules
  (Modèle, Relation de Satisfaisabilité $\models$)

Plan

- Calcul propositionnel (déjà connu)
- Logique Modale
- $\mathcal{HML}$

Calcul Propositionnel (1/2)

Ingrédients

$\mathcal{P}$, Un ensemble de variables propositionnelles
$\mathcal{P} = \{p_1, p_2 \dots p_n\}$

c, Un ensemble de connecteurs
$c = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$

$C_{Prop}(\mathcal{P})$ **Calcul propositionnel** construit sur $\mathcal{P}$

Syntaxe

- $p \in C_{Prop}(\mathcal{P}) \quad \forall p \in \mathcal{P}$
- Si $f \in C_{Prop}(\mathcal{P})$ Alors $\neg f \in C_{Prop}(\mathcal{P})$
- Si $f, g \in C_{Prop}(\mathcal{P})$ Alors $f \# g \in C_{Prop}(\mathcal{P})$
  où $\# \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$

Calcul Propositionnel (2/2)

Sémantique

1. $\mathcal{M}$, **Modèle**
   pour le calcul propositionnel, $\mathcal{M}$ est un sous-ensemble de $\mathcal{P}$
2. $\models$, **Relation de satisfaction**,
   définie par induction sur le structure des formules
   Pour $p \in \mathcal{P}, f$ et $g \in C_{Prop}(\mathcal{P})$
   - $\mathcal{M} \models p$ ssi $p \in \mathcal{M}$ $(\forall p \in \mathcal{P})$
   - $\mathcal{M} \models \neg f$ ssi Non $(\mathcal{M} \models f)$
   - $\mathcal{M} \models f \wedge g$ ssi $(\mathcal{M} \models f$ et $\mathcal{M} \models f)$

   Définition : Formule A est **valide** (notée $\models A$) ssi $\mathcal{M} \models A \quad \forall M$

Sucre

$A \vee B \equiv \neg(\neg A \wedge \neg B)$
$A \Rightarrow B \equiv \neg A \vee B$
$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (A \Leftarrow B)$

Logiques Modales

Ingrédients : Les mêmes augmentées par deux **Modalités : $\Box, \Diamond$**

$\mathcal{P}$, Un ensemble de variables propositionnelles $\mathcal{P} = \{p_1, p_2 \dots p_n\}$

c, Un ensemble de connecteurs $c = \{\neg, \vee, \Box, \Diamond\}$

$LM_{Prop}(\mathcal{P})$ **Logique Modale** construite sur $\mathcal{P}$

Syntaxe

- $p \in LM_{Prop}(\mathcal{P}) \quad \forall p \in \mathcal{P}$
- Si $f \in LM_{Prop}(\mathcal{P})$ Alors $\neg f, \Box f, \Diamond f \in LM_{Prop}(\mathcal{P})$
- Si $f, g \in LM_{Prop}(\mathcal{P})$ Alors $f \ \wedge \ g \in LM_{Prop}(\mathcal{P})$

Sémantique

A définir : : Un modèle $\mathcal{M}$ et une relation de satisfaction $\models$
.... 1960 !

Logiques Modales

Ingrédients : Les mêmes augmentées par deux **Modalités : $\Box, \Diamond$**

$\mathcal{P}$, Un ensemble de variables propositionnelles $\mathcal{P} = \{p_1, p_2 \dots p_n\}$

c, Un ensemble de connecteurs $c = \{\neg, \vee, \Box, \Diamond\}$

$LM_{Prop}(\mathcal{P})$ **Logique Modale** construite sur $\mathcal{P}$

Syntaxe

- $p \in LM_{Prop}(\mathcal{P}) \quad \forall p \in \mathcal{P}$
- Si $f \in LM_{Prop}(\mathcal{P})$ Alors $\neg f, \Box f, \Diamond f \in LM_{Prop}(\mathcal{P})$
- Si $f, g \in LM_{Prop}(\mathcal{P})$ Alors $f \ \wedge \ g \in LM_{Prop}(\mathcal{P})$

Sémantique

A définir : : Un modèle $\mathcal{M}$ et une relation de satisfaction $\models$
.... 1960 !

Modèle pour la logique modale

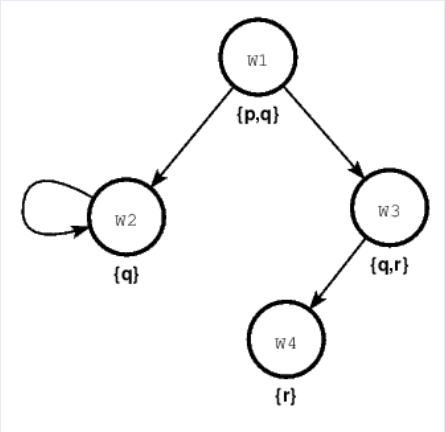
Sémantique des Mondes Possibles (S. Kripke 1960)

Structure de Kripke : $\mathcal{M} =< W, R, \nu >$ où
W est un ensemble de mondes
R est une relation entre ces mondes ($R \subset W \times W$)
ν est une valuation $\nu : W \mapsto \mathcal{P}(P)$

Exemple de Structure de Kripke

$\mathcal{M} =< W, R, \nu >$
$W = \{w_1, w_2, w_3, w_4\}$
$R = \{(w_1, w_2), (w_1, w_3), (w_2, w_2), (w_3, w_4)\}$

$$\nu : \quad w_1 \mapsto \{p, q\} \qquad w_3 \mapsto \{r, q\}$$
$$\quad\;\; w_2 \mapsto \{q\} \qquad w_4 \mapsto \{r\}$$



Notations : $w_i \; R \; w_j$ ssi $(w_i, w_j) \in R$ **&** $R(w_i) =_{Def} \{w_j \in M : w_i R w_j\}$

$\models$: relation de satisfaction

Notations

$\mathcal{M}, w \models A$ signifie que la formule A est satisfaite dans le monde w du modèle $\mathcal{M}$.

$\mathcal{M} \models A$ signifie A est vraie dans tout monde de $\mathcal{M}$

Définition de $\models$

Pour $p \in \mathcal{P}$ & $f, g \in LM(\mathcal{P})$

$\mathcal{M}, w \models p$ ssi $p \in \nu(w)$ $(\forall p \in \mathcal{P})$

$\mathcal{M}, w \models \neg f$ ssi Non $(\mathcal{M}, w \models f)$

$\mathcal{M}, w \models f \wedge g$ ssi $\mathcal{M}, w \models f$ et $\mathcal{M}, w \models g$

$\mathcal{M}, w \models \Box f$ ssi $\forall w' \in \mathcal{M} : w \ R \ w' \Rightarrow \mathcal{M}, w' \models f$

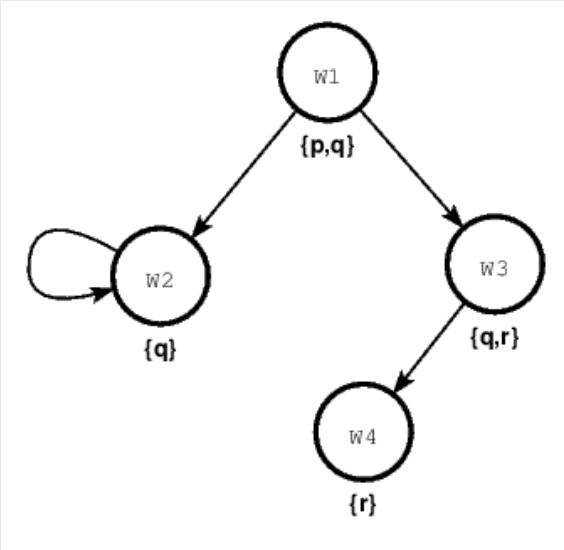
$\mathcal{M}, w \models \Diamond f$ ssi $\exists w' \in \mathcal{M} : w \ R \ w'$ et $\mathcal{M}, w' \models f$

Conventions : $\Box$ *Nécessaire* & $\Diamond$ *Possible*

A manier avec prudence (cf exo #2)

Sémantique Logique Modale

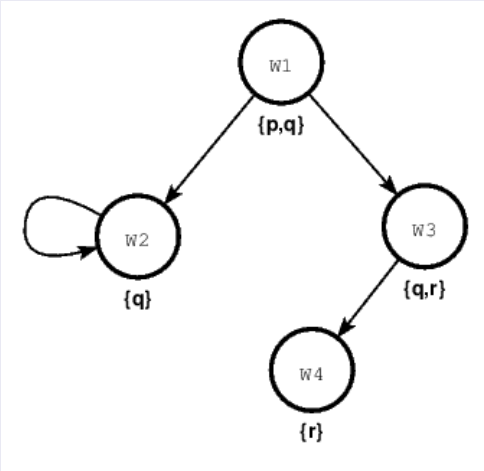
Exemple



1. $\mathcal{M}, w_1 \models \Box q$
   car $R(w_1) = \{w_2, w_3\}$,
   $q \in \nu(w_2)$ & $q \in \nu(w_3)$
   donc $w_2 \models q$ <u>et</u> $w_3 \models q$
2. $\mathcal{M}, w_1 \models \Diamond r$
   car $w_1 \; R \; w_3$ & $w_3 \models r$
3. $\mathcal{M}, w_1 \models \Diamond \neg r$
   car $w_1 \; R \; w_2$ & $w_2 \not\models r$
   (i.e. $r \notin \nu(w_2)$)
4. $\mathcal{M}, w_1 \models \neg \Box r$
   car $w_2 \in R(W_1)$ & $w_2 \not\models r$
5. $\mathcal{M}, w_1 \models \Diamond \Diamond r$
   car $w_1 \; R \; w_3$ & $w_3 \models \Diamond r$

Exercices

Le Modèle



Exo #1

	w_1	w_2	w_3	w_4
q				
$\Diamond q$				
$\Diamond \neg q$				
$\Box q$				
$\Box \neg q$				

Exo #2

1. $\mathcal{M}, w_1 \models \Diamond \Box q$
2. $\mathcal{M}, w_2 \models (\Box q) \wedge (\Diamond q) \wedge q$
3. $\mathcal{M}, w_4 \models (\Box q) \wedge (\Box \neg q) \wedge (\neg \Diamond q) \wedge (\neg \Diamond \neg q)$

Exo #3 : Montrer que pour toute SK $\mathcal{M} =< W, R, \nu >$

$\forall w \in W$ Si $R(w) = \emptyset$ alors $\quad \mathcal{M}, w \models \Box f \qquad (\forall f)$

$\qquad\qquad\qquad\qquad\qquad\qquad \mathcal{M}, w \models \neg \Diamond f \qquad (\forall f)$

$\mathcal{HML}$: Logique d'Hennessy-Milner (1985)

$\mathcal{HML}$ en deux lignes ....

Logique Modale (temporelle) propostitionnelle basée événements (et non états)
Formules évaluées sur une structure de Kripke étiquetée

Structure de Kripke étiquetée

$\mathcal{M} =< W, R, \Sigma, \nu >$ où
W est un ensemble de mondes
R est une relation entre ces mondes ($R \subset W \times \Sigma \times W$)
ν est une valuation $\nu : W \mapsto \mathcal{P}(P)$

Syntaxe

HML est le plus petit ensemble tel que

$$True \in \mathcal{HML}$$
$$A, B \in \mathcal{HML} \Rightarrow A \wedge B, \neg A \in \mathcal{HML}$$
$$A, \in \mathcal{HML}\ , o \in \Sigma \Rightarrow < o > \in \mathcal{HML}$$

Sucre

$False \equiv \neg True$ $\qquad$ $A \vee B \equiv \neg(\neg A \wedge \neg B)$
$< w > A \equiv < i_1 > < i_2 > \ldots < i_n > A$ $\qquad$ pour $w = i_1.i_2.\ldots.i_n$
$[i]A \equiv \neg < i > \neg A$

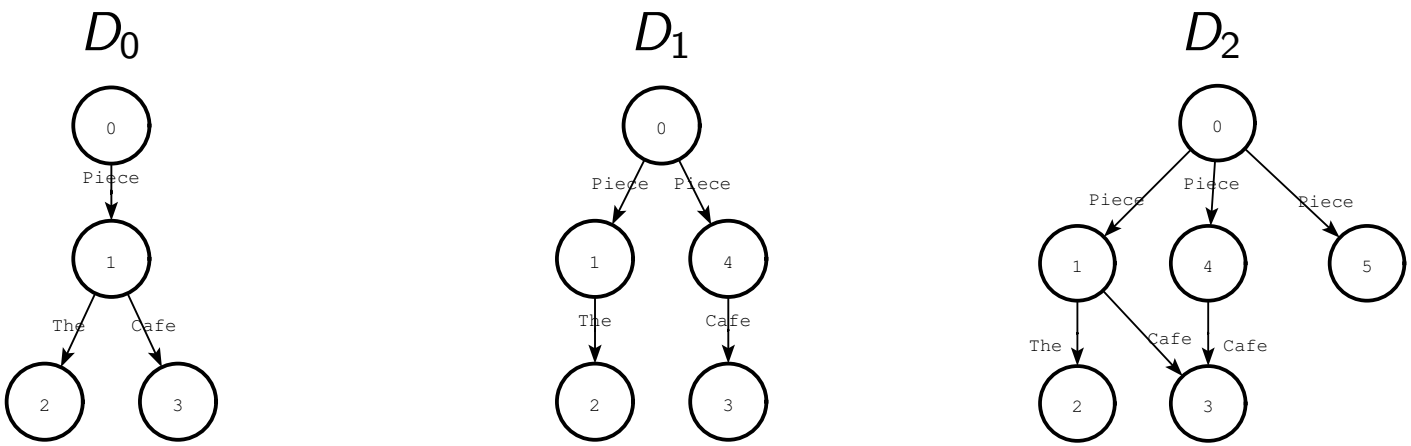
Sémantique $\models$

$W, s \models T \quad \forall s \in S$
$W, s \models A \wedge B$ ssi $W, s \models A$ et $W, s \models B$
$W, s \models \neg A$ ssi Non $(W, s \models \neg A)$
$W, s \models < i > A$ ssi $s \xrightarrow{i} s'$ et $W, s' \models A$

Exemples

- $W, p \models < a > True$
  Une a-expérimentation est possible à partir de p
- $W, p \models < a > (< b > True \wedge < c > True)$
  A partir de p, une a-expérimentation est possible menant dans un état pour lequel à la fois b **et** *c sont possibles.*
- $W, p \models [a] False$
  Aucune a-expérimentation n'est possible partant de p
  $([s] False \equiv \neg < s > \neg False \equiv \neg < s > True)$
- $W, p \models [a](< b > [c] False)$
  Toute a-expérimentation partant de p mène dans un état pour lequel une b-expérimentation est possible menant dans un état bloqué vis-à-vis des c-expérimentations

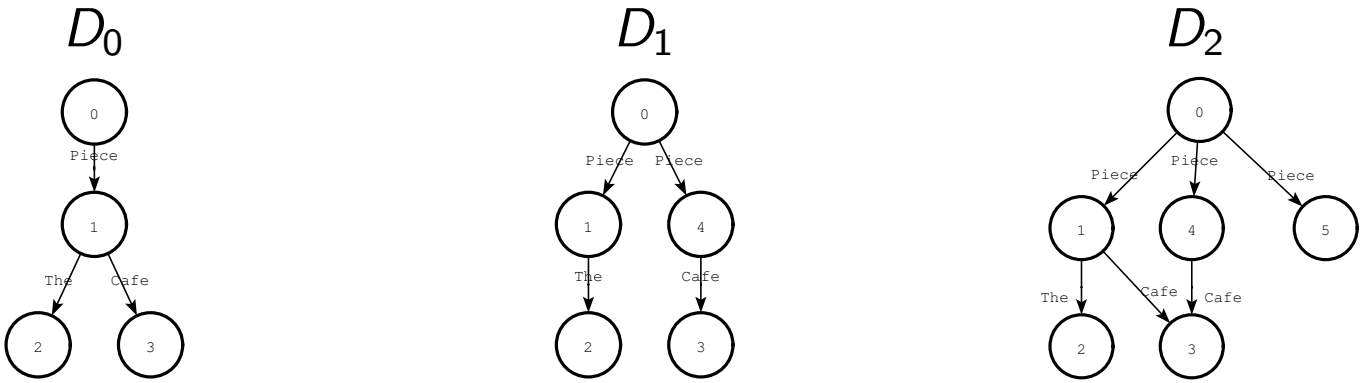
Tea Time (bis)



Exercice

En notant $[[\phi]] = \{w \in W : W, w \models \phi\}$, complétez le tableau ci-dessous :

$[[\ ]]$	D_0	D_1	D_2
$< Cafe > T \wedge < The > T$			
$[Cafe]F$			
$< Cafe > T \wedge [The]F$			
$< Piece >< Cafe > T$			
$[Piece] < Cafe > T$			
$[Piece]T$			



Exercice (suite)

$F_1 \equiv < Piece > [Cafe]F$
$F_2 \equiv < Piece > (< Cafe > T \wedge < The > T)$
$F_3 \equiv [Piece](< Cafe > T \wedge < The > T)$
$F_4 \equiv < Piece > ((< Cafe > T \wedge [The]F)$
$\vee (< The > T \wedge [Cafe]F))$

$\models$	F_1	F_2	F_3	F_4
$D_0, 0$				
$D_1, 0$				
$D_2, 0$				

Exercice

Rappel

Equivalence langage : pas de distinction entre les 3 spécifications
Traces maximales : D_2 est ditinguée de D_0 et de D_1
Equivalence Observationelle : 3 spécifications différentes

$\mathcal{HML}$ permet-elle aussi de les distinguer ?

Principe :

Donner une caractérisation logique des équivalences de comportement : deux états sont équivalents s'ils satisfont les mêmes énoncés d'une logique donnée (la logique adéquate avec la relation d'équivalence)

Notation : Pour une logique L et une structure de Kripke K, on note
$TH_L : S \mapsto 2^L$ où $TH_L(s) = \{f \in L : K, s \models f\}$

Equivalence Observationnelle

Prop : $s \equiv_{Obs} q$ ssi $TH_{\mathcal{HML}}(s) = TH_{\mathcal{HML}}(q)$
$\mathcal{HML}$ caractérise **Modalement** l'Eq Observationnelle

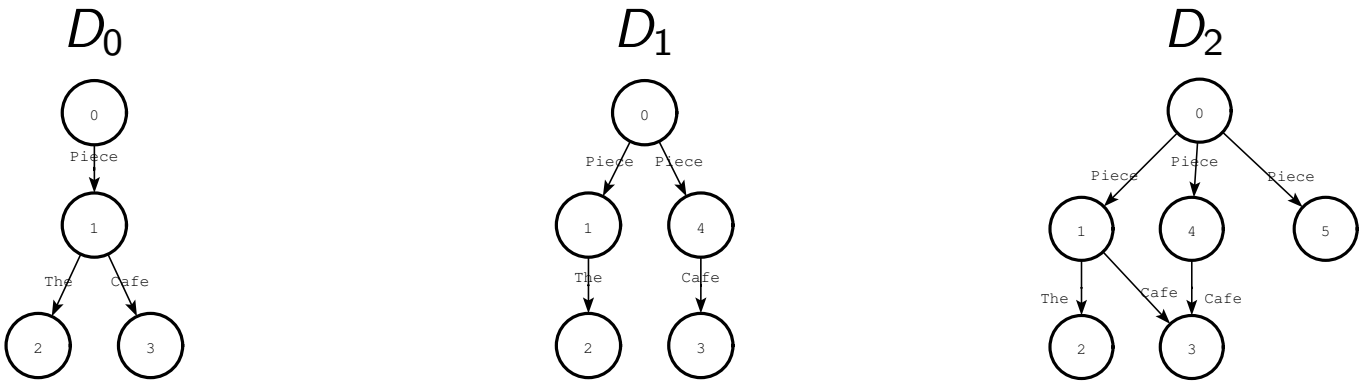
Co-Simulation

$\mathcal{M} =_{Def} \{f \in \mathcal{HML}, f$ ne contenant pas de $\wedge\}$
Prop : $s \equiv_{Cosim} q$ ssi $TH_{\mathcal{M}}(s) = TH_{\mathcal{M}}(q)$
$\mathcal{M}$ caractérise **modalement** la relation de co-simulation

Equivalence Langage

$\mathcal{N} =_{Def} \{f \in \mathcal{M}, f$ ne contenant pas de $\neg\}$
Prop : $s \equiv_{Lang} q$ ssi $TH_{\mathcal{N}}(s) = TH_{\mathcal{N}}(q)$
$\mathcal{N}$ caractérise **modalement** l'équivalence langage

Tea-time (fin)



Rappel

Equivalence langage : pas de distinction entre les 3 spécifications
Traces maximales : D_2 est ditinguée de D_0 et de D_1
Equivalence Observationelle : 3 spécifications différentes

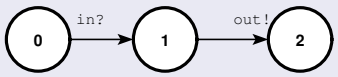
Exercice

- Trouver une formule de $\mathcal{HML}$ caractéristique de chacun de ces 3 LTS i.e., trouver ϕ_i ($i \in [1,3]$) tq $D_j, 0 \models \phi_i$ ssi $i = j$
- Montrer que l' on ne peut pas trouver de formule de $\mathcal{N}$ permettant de distinguer l'un de ces distributeurs.

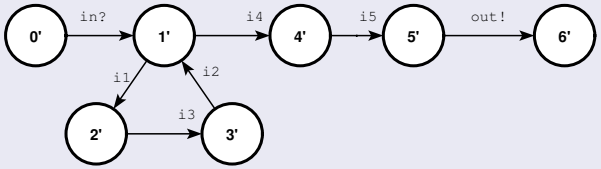
etc ....

$\mathcal{HML}$ et Divergence

2 medium de communication : $\Sigma_{Obs} = \{in?, out!\}$



$$S/\sim = \{\{0\},\{1\},\{2\}\}$$



$$S'/\sim = \{\{0'\},\{1',2',3',4',5'\},\{6'\}\}$$

$$(S \cup S')/\sim = \{\{0,0'\},\{1,1',2',3',4',5'\},\{2,6'\}\}$$

Divergence (définition)

Etant donné un LTS K et $\Sigma_{Obs} \subset \Sigma$

Un état s de S est un état de divergence ssi $\forall k \in, \exists \delta \in (\Sigma \setminus \Sigma_{Obs})^{\mathbf{k}} : s \xrightarrow{\delta}$

On note $Div(S)$, l'ensemble des états de divergence de S

Exercice : La divergence n'est pas exprimable par $\mathcal{HML}$

Par l'absurde : on suppose l'existence de $Div \in \mathcal{HML}$ tq

$K, s \models Div$ ssi s est un état de divergence

En reprenant les medium ci-dessus : on a $1' \sim 4'$

avec $1'$, un état de divergence (i.e., $1' \models Div$),

et $4'$ qui n'est pas un état de divergence (i.e., $4' \not\models Div$),

et on obtient la contradiction avec TH HM - $Div \notin \mathcal{HML}$

Bisimulation et Divergence

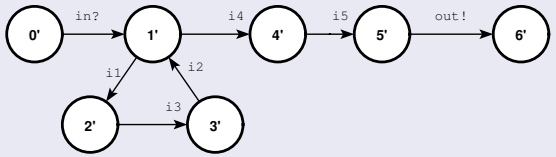
Bisimulation sensible à la divergence ($\sim_{Div}$)

On note $\asymp_{Div} \subset S \times S$, la relation d'équivalence définie par

$$S/\asymp_{Div} = \{Div(S), (S \setminus Div(S)\}$$

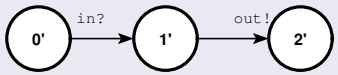
On note $\sim_{Div}$ la **bisimulation engendrée** par $\asymp_{Div}$ (cf def slide ...)

Application

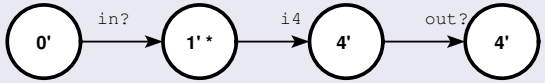


$$\begin{aligned} S'/\sim &= \{\{0'\},\{1',2',3',4',5'\},\{6'\}\} \\ S'/\asymp_{Div} &= \{\{1',2',3'\},\{0',4',5,6'\}\} \\ S'/\sim_{Div} &= \{\{0'\},\{1',2',3'\},\{4',5,\},\{6'\}\} \end{aligned}$$

Quotients associés



$\sim$ (Eq. Obs)



$\sim_{Div}$ (Eq. Obs + Divergence)

$\mathcal{HML}_{Div}$ loqique adéquate avec $\sim_{Div}$

$\mathcal{HML}_{Div}$ analogue à $\mathcal{HML}$ sauf

Syntaxe : $Div \in \mathcal{HML}_{Div}$

Sémantique ($\models$) :$W, s \models Div$ ssi $s \in Div(W)$

Prop : $s \sim_{Div} q$ ssi $TH_{\mathcal{HML}_{Div}}(s) = TH_{\mathcal{HML}_{Div}}(q)$