Modélisation et vérification

Yohan Boichut

(inspiré du cours de John Mullins, Ecole polytechnique de Montréal)

Cours Master IRAD – Semestre 3

Outline

Motivations

- Conception de systèmes fiables
  - Industrie des technologies de plus en plus tournées vers outils de spécification et vérification (Siemens, Thomson, Intel, ...)
  - Besoin de compétences pour savoir utiliser et raisonner dans un tel contexte
- Spécifier pour vérifier

Modélisation et vérification

Abstraction du système concret et simulation de ce modèle

- Modélisation de systèmes : automates (systèmes de transitions)
- Simulation du modèle : langages des automates (traces d'exécutions)
- Modélisation des propriétés attendues des systèmes (Logiques)

Vérification automatique par Model-checking

- Technique automatique
- Exploration complète des configurations des systèmes
- Retour de contre-exemple lorsqu'une propriété n'est pas vérifiée

Outline

Petits exemples de modélisation

Prenons une montre à affichage numérique hh:mm

Petits exemples de modélisation

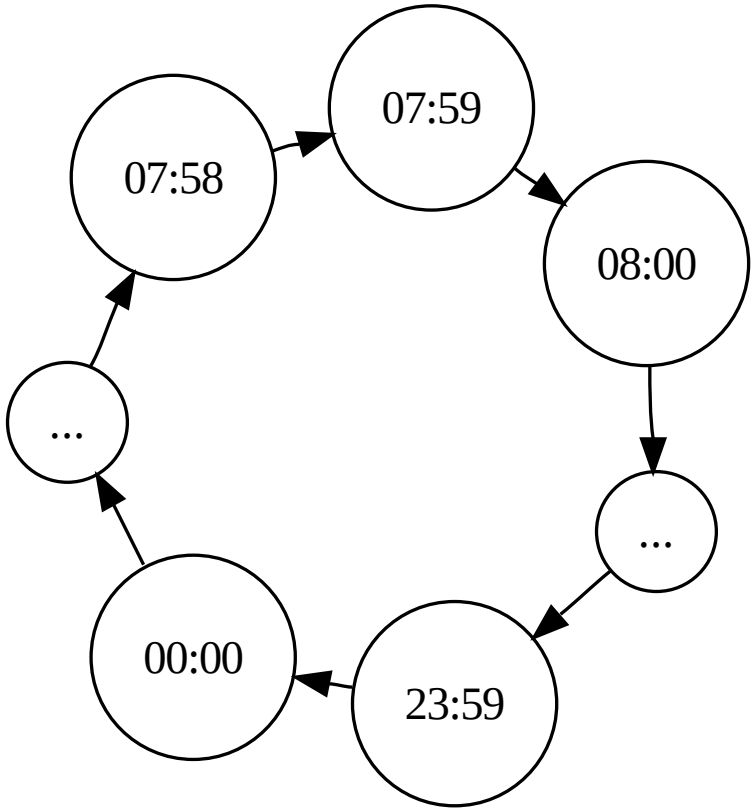
Prenons une montre à affichage numérique hh:mm

Avec $60 \times 24 = 1440$ états, nous pouvons représenter tous les états atteignables de notre montre

Petits exemples de modélisation

Prenons une montre à affichage numérique hh:mm

Avec $60 \times 24 = 1440$ états, nous pouvons représenter tous les états atteignables de notre montre



Petits exemples de modélisation

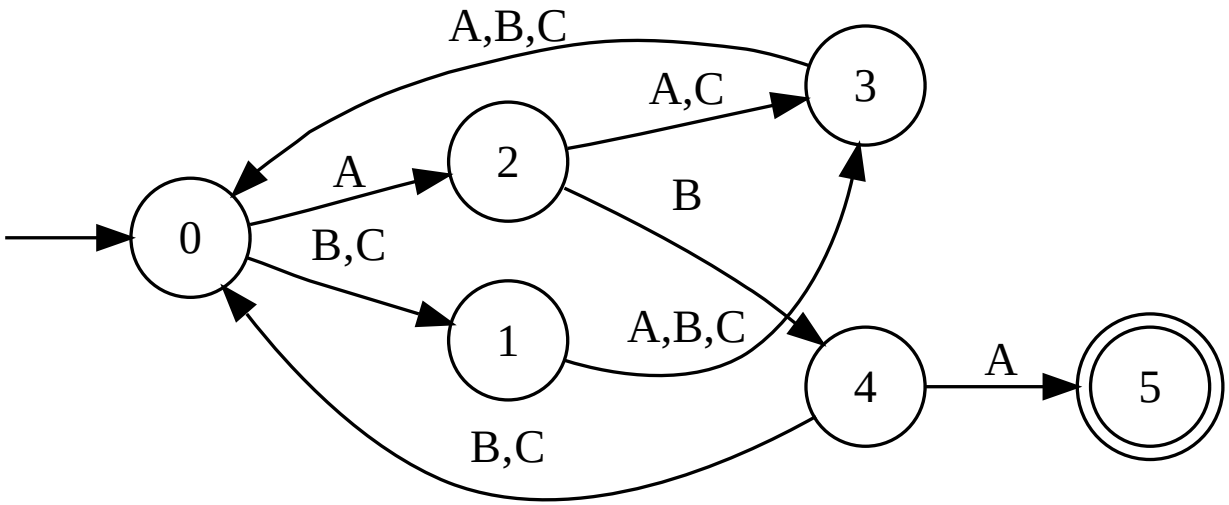
Prenons un digicode à trois touches A,B et C. La porte s'ouvre quand ABA est saisi

Le digicode est dans son état initial après saisie d'un mauvais code

Petits exemples de modélisation

Prenons un digicode à trois touches A,B et C. La porte s'ouvre quand ABA est saisi

Le digicode est dans son état initial après saisie d'un mauvais code



Petits exemples de modélisation

Prenons un compteur modulo 4

Petits exemples de modélisation

Prenons un compteur modulo 4

Opérations

- inc : incrémente de 1 le compteur
- dec : diminue de 1 le compteur

1 état par valeur du compteur : 4 états

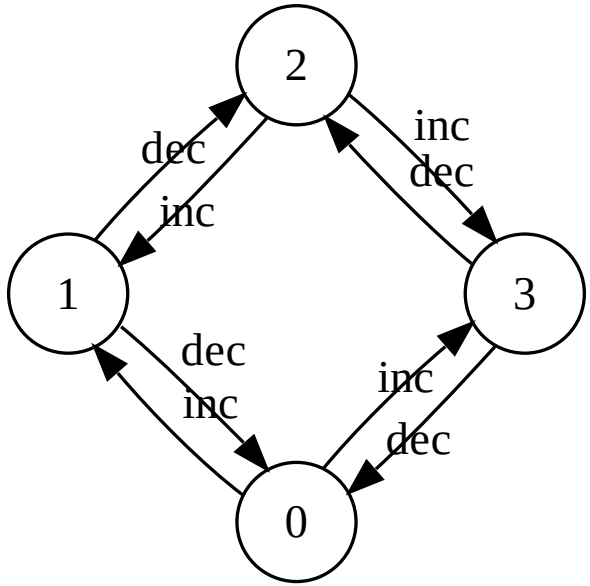
Petits exemples de modélisation

Prenons un compteur modulo 4

Opérations

- inc : incrémente de 1 le compteur
- dec : diminue de 1 le compteur

1 état par valeur du compteur : 4 états



Petits exemples de modélisation

Prenons un canal FIFO de capacité 2 sur l'alphabet $\{a, b\}$

Petits exemples de modélisation

Prenons un canal FIFO de capacité 2 sur l'alphabet $\{a, b\}$

Opérations

- in(x) : enfiler la lettre x si le canal n'est pas plein
- out(x) : défiler la lettre x si le canal n'est pas vide

1 état par configuration possible du canal : 7 états

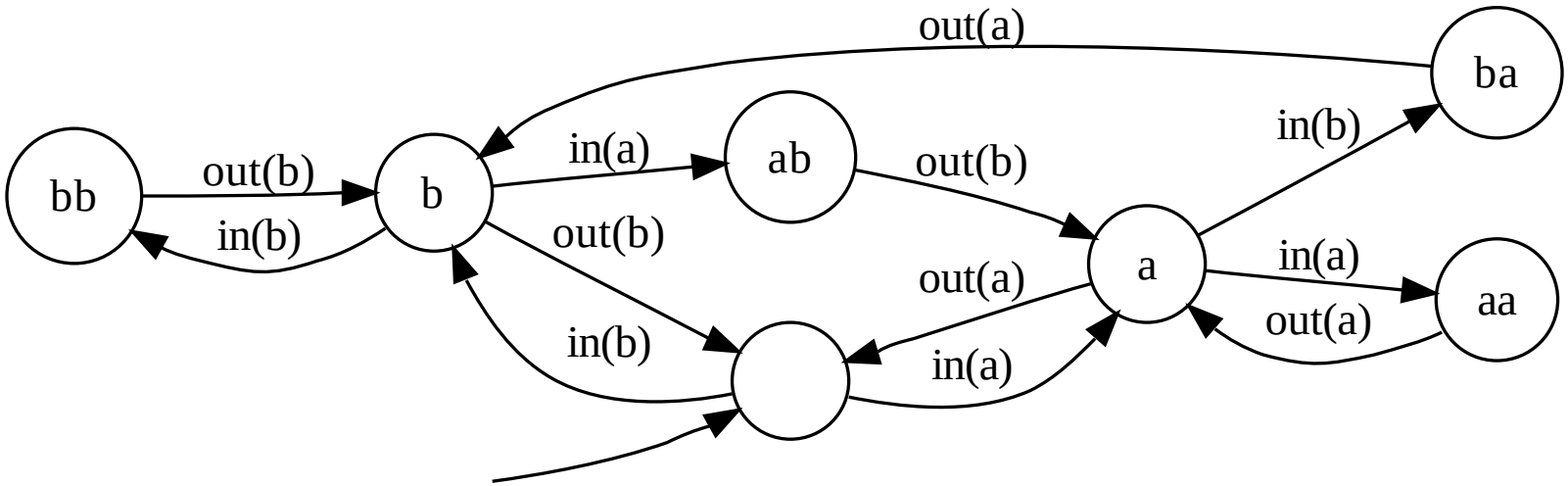
Petits exemples de modélisation

Prenons un canal FIFO de capacité 2 sur l'alphabet $\{a, b\}$

Opérations

- in(x) : enfiler la lettre x si le canal n'est pas plein
- out(x) : défiler la lettre x si le canal n'est pas vide

1 état par configuration possible du canal : 7 états



Petits exemples de modélisation

Prenons une variable booléenne

Petits exemples de modélisation

Prenons une variable booléenne

Opérations

- b = vrai, b = faux : test de la valeur de la variable b
- b := vrai, b := faux : affectation de la variable b

1 état par valeur

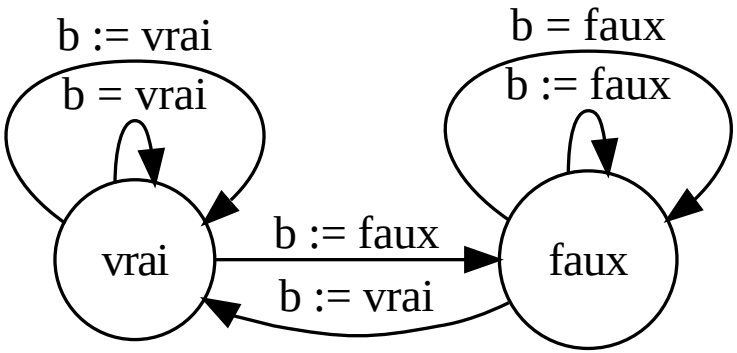
Petits exemples de modélisation

Prenons une variable booléenne

Opérations

- b = vrai, b = faux : test de la valeur de la variable b
- b := vrai, b := faux : affectation de la variable b

1 état par valeur



Petits exemples de modélisation

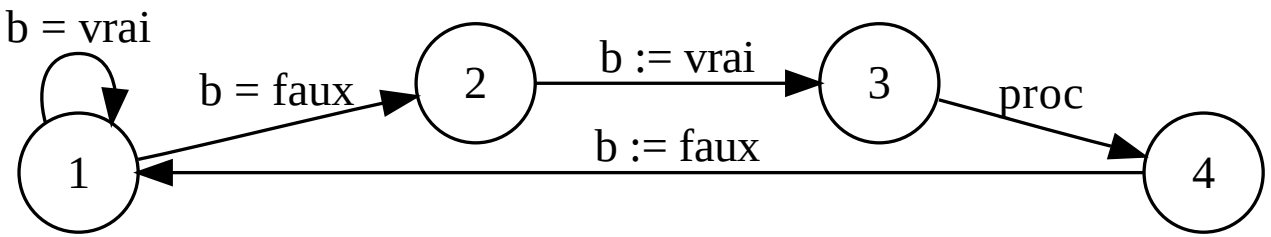
Prenons le programme séquentiel

```
1: While true do
     if not b then
     begin
       2: b:= true;
       3: proc ;
       4: b := false;
     end
   od
```

Petits exemples de modélisation

Prenons le programme séquentiel

```
1: While true do
     if not b then
     begin
       2: b:= true;
       3: proc ;
       4: b := false;
     end
   od
```



Outline

Systèmes de transitions

Définition

Un système de transitions est un couple $\mathcal{A} = \langle S, T \rangle$ *où*

- S *est un ensemble d'états fini ou infini*
- $T \subseteq S \times S$ *est un ensemble de transitions fini ou infini*

Systèmes de transitions

Définition

Un système de transitions est un couple $\mathcal{A} = \langle S, T \rangle$ *où*

- S *est un ensemble d'états fini ou infini*
- $T \subseteq S \times S$ *est un ensemble de transitions fini ou infini*

Définition

Un chemin c *de longueur* n *(noté* $|c| = n$*) est une suite de transitions* $s_1 \to s_2 \to s_3 \to \ldots \to s_n$*. Un chemin infini est une suite infinie de transitions*

Systèmes de transitions étiquetés

Définition

Un système de transitions étiqueté est un couple $\mathcal{A} = \langle S, T, \Sigma \rangle$ *où*

- S *est un ensemble d'états fini ou infini*
- $T \subseteq S \times \Sigma \times S$ *est un ensemble de transitions fini ou infini*
- Σ *est un ensemble d'actions*

Systèmes de transitions étiquetés

Définition

Un système de transitions étiqueté est un couple $\mathcal{A} = \langle S, T, \Sigma \rangle$ *où*

- S *est un ensemble d'états fini ou infini*
- $T \subseteq S \times \Sigma \times S$ *est un ensemble de transitions fini ou infini*
- Σ *est un ensemble d'actions*

Définition

Si c *est un chemin* $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \ldots$ *alors la suite* $a_1 a_2 a_3 \ldots$ *est une trace*

Systèmes de transitions

Définition

Un système de transitions paramétré est un tuple $\mathcal{A} = \langle S, T, S_{X_1}, \ldots, , S_{X_m}, T_{Y_1}, \ldots, T_{Y_n} \rangle$ *où*

- S *est un ensemble d'états*
- $T \subseteq S \times S$ *est l'ensemble des transitions*
- $S_{X_i} \subseteq S$ *sont des paramètres d'états*
- $T_{Y_i} \subseteq T$ *sont des paramètres de transitions*

En pratique, utilisation de systèmes de transitions étiquetés et paramétrés par un état initial et des états terminaux

Produit libre de systèmes de transitions

Systèmes composés de sous systèmes indépendants : aucune interaction entre les composants

Définition

Le produit libre $\mathcal{A}_1 \times \mathcal{A}_2$ *de deux systèmes de transitions* $\mathcal{A}_1 = \langle S_1, T_1, \Sigma_1 \rangle$ *et* $\mathcal{A}_2 = \langle S_2, T_2, \Sigma_2 \rangle$ *est le système de transition* $\mathcal{A} = \langle S, T, \Sigma \rangle$ *défini par*

- $S = S_1 \times S_2$
- $T = \{(s_1, s_2) \xrightarrow{(a_1, a_2)} (s'_1, s'_2) \mid ((s_1, a_1, s'_1) \in T_1 \wedge (s_2, a_2, s'_2) \in T_2) \vee (a_1 =' \_' \text{ et } s_1 = s'_1) \vee (a_2 =' \_' \text{ et } s_2 = s'_2)\}$
- $\Sigma = (\Sigma_1 \cup \{\_\}) \times (\Sigma_2 \cup \{\_\})$

Produit libre de systèmes de transitions

Exemple

Calculer le produit libre de $\mathcal{A}_1$ et $\mathcal{A}_2$ où

- *$\mathcal{A}_1$ représente un compteur modulo 2*
- *$\mathcal{A}_2$ représente un compteur modulo 3*

Produit synchronisé de systèmes de transitions

Définition

Le produit synchronisé $\mathcal{A}_1 \|_{Sync} \mathcal{A}_2$ *de* $\mathcal{A}_1$ *et* $\mathcal{A}_2$ *par rapport à* $Sync \subseteq \Sigma_1 \cup \{\_\}) \times (\Sigma_2 \cup \{\_\})$ *est le système de transitions* $\mathcal{A}_1 \times \mathcal{A}_2$ *restreint aux seules transitions présentes dans Sync*

Produit synchronisé de systèmes de transitions

Exemple

Calculer le produit synchronisé de $\mathcal{A}_1$ *et* $\mathcal{A}_2$ *avec* $Sync = \{(inc, inc), (dec, dec), (\_, \_))\}$ *où*

- $\mathcal{A}_1$ *représente un compteur modulo* 2
- $\mathcal{A}_2$ *représente un compteur modulo* 3

Synchronisation par messages

La relation de synchronisation couple les actions complémentaires : émission / réception de messages ($m!/m?$)

Exercice : Protocole de commerce électronique

But : manipuler de l'argent électronique sous forme de certificats monétaires

Données

- Un client, une banque et un marchand
- Cryptographie parfaite $\Longrightarrow$ validité des certificats monétaires

Scénario

Le client

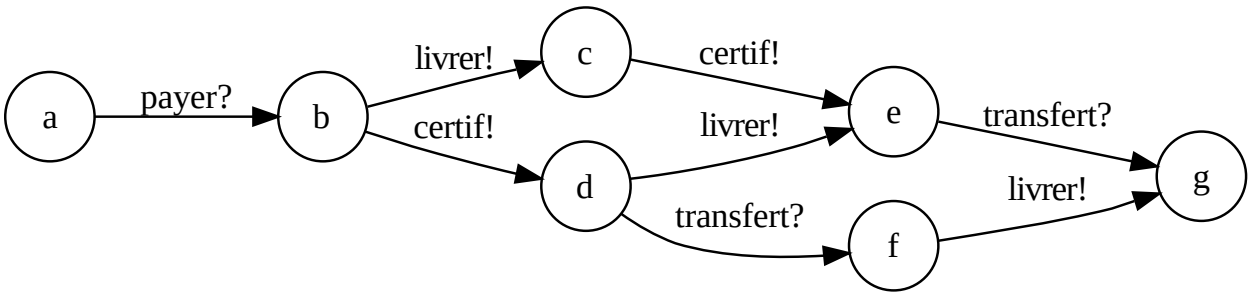
1. initie une action de paiement
   - 1.1 envoie au marchand son certificat électronique
   - 1.2 Sur présentation du certificat, le marchand demande à la banque l'émission d'un nouveau certificat monétaire et
   - 1.3 Le marchand livre la marchandise
2. émet une demande d'annulation dans quel cas, la banque après vérification du certificat retourne l'argent dans le compte client et annule sa validité

Exercice : Protocole de commerce électronique

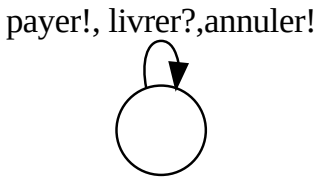
Modélisation

- Un client qui peut payer, attendre sa livraison et annuler
- Un marchand qui peut enregistrer le paiement, livrer / demander un nouveau certificat puis recevoir le transfert
- Une banque qui peut recevoir une demande d'annulation ou recevoir une demande de certificat puis réaliser le transfert

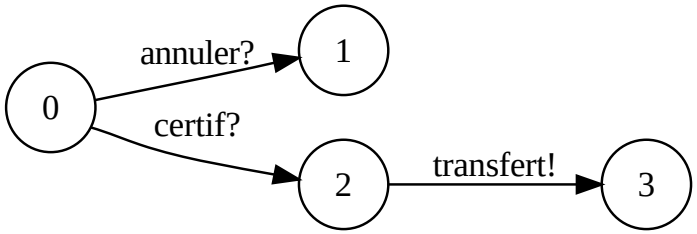
Exercice : Protocole de commerce électronique



- $\mathcal{A}_{marchand}$



- $\mathcal{A}_{client}$



- $\mathcal{A}_{banque}$

Calculer le produit synchronisé par message de ces trois automates !

Synchronisation par canal

P_0 et P_1 communiquent par messages avec un canal C

Nous pouvons ramener le problème à une synchronisation par messages entre les deux processus et un troisième modélisant un canal FIFO

Synchronisation par variables partagées

Algorithme de Peterson

P_0 execute

```
while true do
  begin
    {section non critique}
    d0 := true;
    tour := 0;
    attendre (d1=false ou tour=1);
    {section critique}
    d0 := false;
  end
```

P_1 execute

```
while true do
  begin
    {section non critique}
    d1 := true;
    tour := 1;
    attendre (d0=false ou tour=0);
    {section critique}
    d1 := false;
  end
```

Exercice

1. *Représentez les processus et les variables partagées par des systèmes de transitions*
2. *Estimez la taille du produit synchronisé de ces 5 composants !*

Outline

Motivation

- Complément de l'approche opérationnelle
- Descrition de propriétés que le système modélisé doit satisfaire

Motivation

- Complément de l'approche opérationnelle
- Descrition de propriétés que le système modélisé doit satisfaire

Exemple

Pour l'algorithme de Peterson

- *Il n'existe aucun état du produit synchronisé où les deux processus sont en même temps en section critique*

Motivation

- Complément de l'approche opérationnelle
- Descrition de propriétés que le système modélisé doit satisfaire

Exemple

Pour l'algorithme de Peterson

- *Il n'existe aucun état du produit synchronisé où les deux processus sont en même temps en section critique*
- *Il n'existe aucun deadlock*

Motivation

- Complément de l'approche opérationnelle
- Descrition de propriétés que le système modélisé doit satisfaire

Exemple

Pour l'algorithme de Peterson

- *Il n'existe aucun état du produit synchronisé où les deux processus sont en même temps en section critique*
- *Il n'existe aucun deadlock*
- *S'il existe un état où le processus essaie de rentrer en section critique alors il existe un état accessible de cet état où ce processus rentre effectivement en section critique*

Motivation

- Complément de l'approche opérationnelle
- Descrition de propriétés que le système modélisé doit satisfaire

Exemple

Pour l'algorithme de Peterson

- *Il n'existe aucun état du produit synchronisé où les deux processus sont en même temps en section critique*
- *Il n'existe aucun deadlock*
- *S'il existe un état où le processus essaie de rentrer en section critique alors il existe un état accessible de cet état où ce processus rentre effectivement en section critique*
- *Il n'existe pas de chemin infini constitué uniquement de transitions où les deux processus tentent de rentrer en section critique sans jamais y parvenir*

Et pour faire tout ceci...

- Utilisation de langages de spécification appelés logiques

Et pour faire tout ceci...

- Utilisation de langages de spécification appelés logiques
- Multitude de logiques

Et pour faire tout ceci...

- Utilisation de langages de spécification appelés logiques
- Multitude de logiques
  - Logique propositionnelle : propriétés purement locales

Et pour faire tout ceci...

- Utilisation de langages de spécification appelés logiques
- Multitude de logiques
  - Logique propositionnelle : propriétés purement locales
  - Logique temporelle linéaire : propriétés sur les systèmes en cours d'exécution

Logique propositionnelle

Formules construites inductivement à partir d'un ensemble fixé de propositions atomiques PA_V

$$PA_V = \{x = d \mid x \in V \wedge d \in D_x\}$$

avec V un ensemble de variables et pour une variable x, D_x représente l'ensemble des valeurs possibles de x

Logique propositionnelle

Formules construites inductivement à partir d'un ensemble fixé de propositions atomiques PA_V

$$PA_V = \{x = d \mid x \in V \wedge d \in D_x\}$$

avec V un ensemble de variables et pour une variable x, D_x représente l'ensemble des valeurs possibles de x

Syntaxe

L_0 est le plus petit ensemble de formules propositionnelles tel que

- $0, 1 \in L_0$
- $PA_V \subseteq L_0$
- Si $\phi, \psi \in L_0$ alors $\phi \wedge \psi \in L_0$ et $\phi \vee \psi \in L_0$
- Si $\phi \in L_0$ alors $\neg\phi \in L_0$

Pourquoi pas $\Rightarrow$ et $\Leftrightarrow$?

Pourquoi pas $\Rightarrow$ et $\Leftrightarrow$?

Parce que

$$\phi \Rightarrow \psi \equiv \neg\phi \vee \psi$$

$$\phi \Leftrightarrow \psi \equiv (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$$

Fonctions d'interprétation

Ces fonctions permettent de décorer des états d'un automate avec des propositions atomiques

Fonctions d'interprétation

Ces fonctions permettent de décorer des états d'un automate avec des propositions atomiques

Définition

Un automate non étiqueté sera un quadruplet $\langle S, S_0, T, \rho \rangle$ *où* $\rho : S \mapsto 2^{PA_V}$

Fonctions d'interprétation

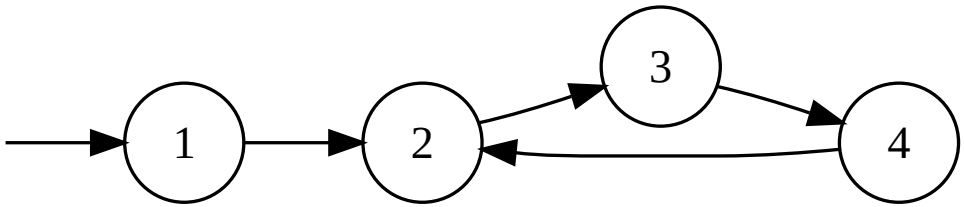
Ces fonctions permettent de décorer des états d'un automate avec des propositions atomiques

Définition

Un automate non étiqueté sera un quadruplet $\langle S, S_0, T, \rho \rangle$ *où* $\rho : S \mapsto 2^{PA_V}$

Exemple

A l'automate suivant, nous pouvons définir pour $PA_V = \{p, q, r, s, t\}$ *la décoration suivante :* $\rho(1) = \{p, q, t\}$, $\rho(2) = \{p, q, r\}$, $\rho(3) = \{p, s\}$ *et* $\rho(4) = \{p, r\}$



Satisfaction d'une formule

$$\mathcal{A}, s \models \phi$$

Satisfaction d'une formule

$$\mathcal{A}, s \vDash \phi$$

Inductivement sur la structure de la formule :

Satisfaction d'une formule

$$\mathcal{A}, s \vDash \phi$$

Inductivement sur la structure de la formule :

- $\mathcal{A}, s \nvDash 0$ et $\mathcal{A}, s \vDash 1$

Satisfaction d'une formule

$$\mathcal{A}, s \models \phi$$

Inductivement sur la structure de la formule :

- $\mathcal{A}, s \not\models 0$ et $\mathcal{A}, s \models 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, s \models \phi$ ssi $\phi \in \rho(s)$

Satisfaction d'une formule

$$\mathcal{A}, s \vDash \phi$$

Inductivement sur la structure de la formule :

- $\mathcal{A}, s \nvDash 0$ et $\mathcal{A}, s \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, s \vDash \phi$ ssi $\phi \in \rho(s)$
- $\mathcal{A}, s \vDash \phi \vee \psi$ ssi $\mathcal{A}, s \vDash \phi$ ou $\mathcal{A}, s \vDash \psi$

Satisfaction d'une formule

$$\mathcal{A}, s \vDash \phi$$

Inductivement sur la structure de la formule :

- $\mathcal{A}, s \nvDash 0$ et $\mathcal{A}, s \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, s \vDash \phi$ ssi $\phi \in \rho(s)$
- $\mathcal{A}, s \vDash \phi \vee \psi$ ssi $\mathcal{A}, s \vDash \phi$ ou $\mathcal{A}, s \vDash \psi$
- $\mathcal{A}, s \vDash \phi \wedge \psi$ ssi $\mathcal{A}, s \vDash \phi$ et $\mathcal{A}, s \vDash \psi$

Satisfaction d'une formule

$$\mathcal{A}, s \vDash \phi$$

Inductivement sur la structure de la formule :

- $\mathcal{A}, s \nvDash 0$ et $\mathcal{A}, s \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, s \vDash \phi$ ssi $\phi \in \rho(s)$
- $\mathcal{A}, s \vDash \phi \vee \psi$ ssi $\mathcal{A}, s \vDash \phi$ ou $\mathcal{A}, s \vDash \psi$
- $\mathcal{A}, s \vDash \phi \wedge \psi$ ssi $\mathcal{A}, s \vDash \phi$ et $\mathcal{A}, s \vDash \psi$
- $\mathcal{A}, s \vDash \neg\phi$ ssi $\mathcal{A}, s \nvDash \phi$

Satisfaction d'une formule

$$\mathcal{A}, s \vDash \phi$$

Inductivement sur la structure de la formule :

- $\mathcal{A}, s \nvDash 0$ et $\mathcal{A}, s \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, s \vDash \phi$ ssi $\phi \in \rho(s)$
- $\mathcal{A}, s \vDash \phi \vee \psi$ ssi $\mathcal{A}, s \vDash \phi$ ou $\mathcal{A}, s \vDash \psi$
- $\mathcal{A}, s \vDash \phi \wedge \psi$ ssi $\mathcal{A}, s \vDash \phi$ et $\mathcal{A}, s \vDash \psi$
- $\mathcal{A}, s \vDash \neg\phi$ ssi $\mathcal{A}, s \nvDash \phi$

ϕ est une tautologie si c'est une formule valide sur tous les états

Satisfaction d'une formule

$$\mathcal{A}, s \vDash \phi$$

Inductivement sur la structure de la formule :

- $\mathcal{A}, s \nvDash 0$ et $\mathcal{A}, s \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, s \vDash \phi$ ssi $\phi \in \rho(s)$
- $\mathcal{A}, s \vDash \phi \vee \psi$ ssi $\mathcal{A}, s \vDash \phi$ ou $\mathcal{A}, s \vDash \psi$
- $\mathcal{A}, s \vDash \phi \wedge \psi$ ssi $\mathcal{A}, s \vDash \phi$ et $\mathcal{A}, s \vDash \psi$
- $\mathcal{A}, s \vDash \neg\phi$ ssi $\mathcal{A}, s \nvDash \phi$

ϕ est une tautologie si c'est une formule valide sur tous les états

ϕ est une contradiction si c'est une formule qui n'est valide sur aucun état

Mais avant... petite paranthèse sur les ω–mots

$\omega - mots$

Expressions régulières

Pour un alphabet Σ,

- Si $a \in \Sigma$ alors a est une expression régulière
- Si e_1 et e_2 sont deux expressions régulières
  - $e_1.e_2$ est une expression régulière
  - $e_1 + e_2$ est une expression régulière
- Si e est une expression régulière alors e^* est une expression régulière

Expressions régulières

Exemple

Soit $\Sigma = \{a, b, c\}$ *alors l'expression régulière*

$$(ab^* + c)^*$$

dénote la concaténation de séquences de lettres de la forme

- *c ou*
- *a suivi d'un nombre nul ou fini de b*

Les expressions $\omega-$régulières

Définition

Les expressions $\omega-$régulières sont définies inductivement à partir des règles suivantes :

- *Si e_1, e_2 sont des expressions régulières (dénotant E_1, E_2) alors $e_1.e_2^\omega$ est une expression $\omega-$régulière (e_2^ω est une chaîne infinie composée de mots de E_2)*
- *Si e_1, e_2 sont des expressions $\omega-$régulières alors $e_1 + e_2^\omega$ est une expression $\omega-$régulière*
- *Σ est un ensemble d'actions*

Les expressions $\omega-$régulières

Définition

Les expressions $\omega-$régulières sont définies inductivement à partir des règles suivantes :

- *Si e_1, e_2 sont des expressions régulières (dénotant E_1, E_2) alors $e_1.e_2^\omega$ est une expression $\omega-$régulière (e_2^ω est une chaîne infinie composée de mots de E_2)*
- *Si e_1, e_2 sont des expressions $\omega-$régulières alors $e_1 + e_2^\omega$ est une expression $\omega-$régulière*
- *Σ est un ensemble d'actions*

Théorème

Une expression est $\omega-$régulière ssi elle est de la forme $\bigcup_{i=1}^{n}(e_i f_i^\omega)$ où e_i et f_i sont des expressions régulières

Expressions $\omega-$régulières

Exemple

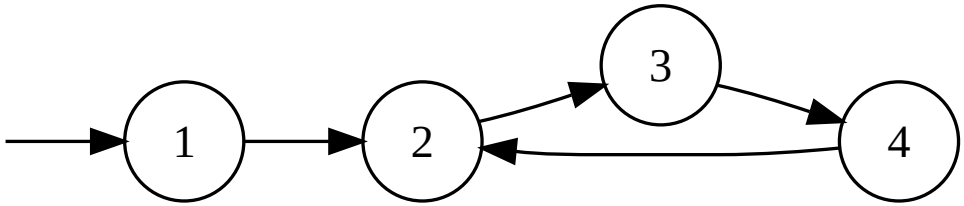
Soit $\Sigma = \{a, b, c\}$ *alors l'expression* $\omega-$*régulière*

$$(c^* a c^* b) c^\omega \cup (c^* a c^* b)^\omega$$

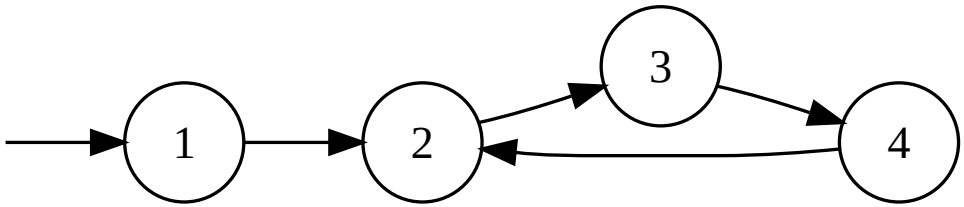
dénote l'ensemble des mots infinis pour lesquels

- *toute occurence de a doit être suivie de* $c^* b$ *et*
- *toute occurence de b doit être précédée de* ac^*

Le lien avec les systèmes de transitions ?



Le lien avec les systèmes de transitions ?



Exécution infinie se présente par l'expression suivante : $1(234)^{\omega}$

Cà, c'est fait...

$$)$$
$$\omega - mots$$

Donc... la logique temporelle linéaire

Syntaxe

L_1 est le plus petit ensemble de formules temporelles linéaires tel que

- $0, 1 \in L_1$
- $PA_V \subseteq L_1$
- Si $\phi, \psi \in L_1$ alors $\phi \wedge \psi \in L_1$ et $\phi \vee \psi \in L_1$
- Si $\phi \in L_1$ alors $\neg\phi \in L_1$

Donc... la logique temporelle linéaire

Syntaxe

L_1 est le plus petit ensemble de formules temporelles linéaires tel que

- $0, 1 \in L_1$
- $PA_V \subseteq L_1$
- Si $\phi, \psi \in L_1$ alors $\phi \wedge \psi \in L_1$ et $\phi \vee \psi \in L_1$
- Si $\phi \in L_1$ alors $\neg\phi \in L_1$
- Si $\phi, \psi \in L_1$ alors $\phi \wedge \psi \in L_1$ et $\phi U \psi \in L_1$

Donc... la logique temporelle linéaire

Syntaxe

L_1 est le plus petit ensemble de formules temporelles linéaires tel que

- $0, 1 \in L_1$
- $PA_V \subseteq L_1$
- Si $\phi, \psi \in L_1$ alors $\phi \wedge \psi \in L_1$ et $\phi \vee \psi \in L_1$
- Si $\phi \in L_1$ alors $\neg\phi \in L_1$
- Si $\phi, \psi \in L_1$ alors $\phi \wedge \psi \in L_1$ et $\phi U \psi \in L_1$
- Si $\phi \in L_1$ alors $N\phi \in L_1$

Satisfaction d'une formule de L_1

$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Satisfaction d'une formule de L_1

$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Inductivement sur la structure de la formule :

Satisfaction d'une formule de L_1

$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Inductivement sur la structure de la formule :

- $\mathcal{A}, c \nvDash 0$ et $\mathcal{A}, c \vDash 1$

Satisfaction d'une formule de L_1

$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Inductivement sur la structure de la formule :

- $\mathcal{A}, c \nvDash 0$ et $\mathcal{A}, c \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, c \vDash \phi$ ssi $\phi \in \rho(t_1)$

Satisfaction d'une formule de L_1

$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Inductivement sur la structure de la formule :

- $\mathcal{A}, c \nvDash 0$ et $\mathcal{A}, c \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, c \vDash \phi$ ssi $\phi \in \rho(t_1)$
- $\mathcal{A}, c \vDash \phi \vee \psi$ ssi $\mathcal{A}, c \vDash \phi$ ou $\mathcal{A}, c \vDash \psi$

Satisfaction d'une formule de L_1

$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Inductivement sur la structure de la formule :

- $\mathcal{A}, c \nvDash 0$ et $\mathcal{A}, c \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, c \vDash \phi$ ssi $\phi \in \rho(t_1)$
- $\mathcal{A}, c \vDash \phi \vee \psi$ ssi $\mathcal{A}, c \vDash \phi$ ou $\mathcal{A}, c \vDash \psi$
- $\mathcal{A}, c \vDash \phi \wedge \psi$ ssi $\mathcal{A}, c \vDash \phi$ et $\mathcal{A}, c \vDash \psi$

Satisfaction d'une formule de L_1

$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Inductivement sur la structure de la formule :

- $\mathcal{A}, c \nvDash 0$ et $\mathcal{A}, c \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, c \vDash \phi$ ssi $\phi \in \rho(t_1)$
- $\mathcal{A}, c \vDash \phi \vee \psi$ ssi $\mathcal{A}, c \vDash \phi$ ou $\mathcal{A}, c \vDash \psi$
- $\mathcal{A}, c \vDash \phi \wedge \psi$ ssi $\mathcal{A}, c \vDash \phi$ et $\mathcal{A}, c \vDash \psi$
- $\mathcal{A}, c \vDash \neg\phi$ ssi $\mathcal{A}, c \nvDash \phi$

Satisfaction d'une formule de L_1

$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Inductivement sur la structure de la formule :

- $\mathcal{A}, c \nvDash 0$ et $\mathcal{A}, c \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, c \vDash \phi$ ssi $\phi \in \rho(t_1)$
- $\mathcal{A}, c \vDash \phi \vee \psi$ ssi $\mathcal{A}, c \vDash \phi$ ou $\mathcal{A}, c \vDash \psi$
- $\mathcal{A}, c \vDash \phi \wedge \psi$ ssi $\mathcal{A}, c \vDash \phi$ et $\mathcal{A}, c \vDash \psi$
- $\mathcal{A}, c \vDash \neg\phi$ ssi $\mathcal{A}, c \nvDash \phi$
- $\mathcal{A}, c \vDash N\phi$ ssi $c = t.c'$ et $\mathcal{A}, c' \vDash \phi$

Satisfaction d'une formule de L_1

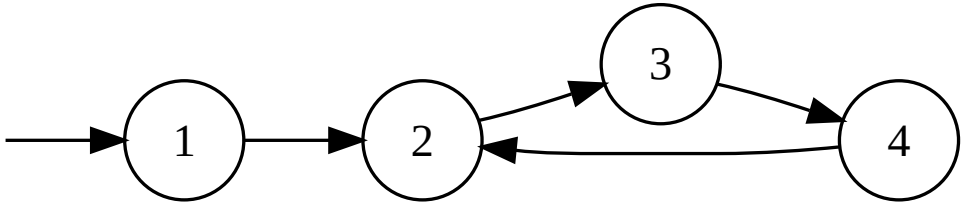
$$\mathcal{A}, c \vDash \phi$$

où $c = t_1 t_2 \ldots$

Inductivement sur la structure de la formule :

- $\mathcal{A}, c \nvDash 0$ et $\mathcal{A}, c \vDash 1$
- Si $\phi \in PA_V$ alors $\mathcal{A}, c \vDash \phi$ ssi $\phi \in \rho(t_1)$
- $\mathcal{A}, c \vDash \phi \vee \psi$ ssi $\mathcal{A}, c \vDash \phi$ ou $\mathcal{A}, c \vDash \psi$
- $\mathcal{A}, c \vDash \phi \wedge \psi$ ssi $\mathcal{A}, c \vDash \phi$ et $\mathcal{A}, c \vDash \psi$
- $\mathcal{A}, c \vDash \neg\phi$ ssi $\mathcal{A}, c \nvDash \phi$
- $\mathcal{A}, c \vDash N\phi$ ssi $c = t.c'$ et $\mathcal{A}, c' \vDash \phi$
- $\mathcal{A}, c \vDash \phi U \psi$ ssi
  - $c = t_1 \ldots t_n c'$ avec $\mathcal{A}, c' \vDash \psi$ et $\forall i \in \{1 \ldots n\}$, $\mathcal{A}, t_i \ldots t_n c' \vDash \phi$ ou
  - $\mathcal{A}, c \vDash \psi$

Petit exemple



avec $\rho(1) = \{p, q, t\}$, $\rho(2) = \{p, q, r\}$, $\rho(3) = \{p, s\}$ et $\rho(4) = \{p, r\}$

- $\mathcal{A}, 1(234)^\omega \stackrel{?}{\models} p \wedge \neg r$
- $\mathcal{A}, 1(234)^\omega \stackrel{?}{\models} r \Rightarrow s$
- $\mathcal{A}, 1(234)^\omega \stackrel{?}{\models} N(p \Leftrightarrow s)$
- $\mathcal{A}, 1(234)^\omega \stackrel{?}{\models} NNs$
- $\mathcal{A}, 1(234)^\omega \stackrel{?}{\models} qUs$

Opérateurs de temps $\Box$ et $\Diamond$

- $\Diamond \phi \stackrel{Def}{=} 1 U \phi$
- $\Box \phi \stackrel{Def}{=} \neg \Diamond \neg \phi$

Opérateurs de temps $\Box$ et $\Diamond$

- $\Diamond\phi \stackrel{Def}{=} 1U\phi$
- $\Box\phi \stackrel{Def}{=} \neg \Diamond \neg\phi$

D'un point de vue sémantique

1. $\mathcal{A}, c \vDash \Diamond\phi$ ssi il existe un suffixe c' de c tel que $\mathcal{A}, c' \vDash \phi$
2. $\mathcal{A}, c \vDash \Box\phi$ ssi pour tout suffixe c' de c, $\mathcal{A}, c' \vDash \phi$

Propriétés temporelles utiles

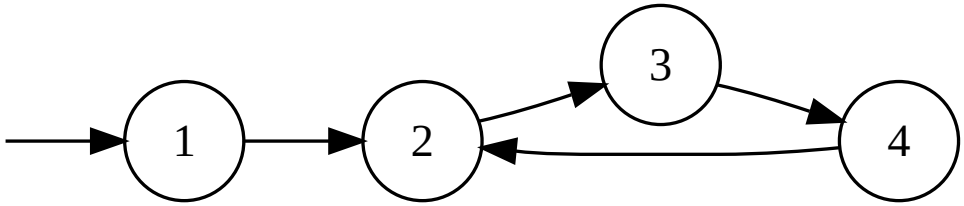
Soit c un chemin infini de $\mathcal{A}$, $\mathcal{A}, c \vDash \Box \Diamond \phi$ ssi il existe une infinité de suffixes c_2 tels que

$$\mathcal{A}, c_2 \vDash \phi$$

Soit c un chemin infini de $\mathcal{A}$, $\mathcal{A}, c \vDash \Diamond \Box \neg \phi$ ssi il existe un nombre fini de suffixes c' tels que

$$\mathcal{A}, c' \vDash \phi$$

Petit exemple



avec $\rho(1) = \{p, q, t\}$, $\rho(2) = \{p, q, r\}$, $\rho(3) = \{p, s\}$ et $\rho(4) = \{p, r\}$

- $\mathcal{A}, 1(234)^\omega \stackrel{?}{\models} \Box \Diamond r$
- $\mathcal{A}, 1(234)^\omega \stackrel{?}{\models} \Box \Diamond (qUs)$
- $\mathcal{A}, 1(234)^\omega \stackrel{?}{\models} \Diamond \Box \neg t$

Model-checking

Définition

Soit $\mathcal{A} = \langle S, S_0, T, \rho \rangle$, un automate et ϕ, une formule de L_1 alors :

1. *ϕ est réalisable d'un état s si pour tout chemin de $\mathcal{A}$ s.c issu de s on a $\mathcal{A}, s.c \models \phi$. On note alors $\mathcal{A}, s \models \phi$*
2. *ϕ est réalisable si elle est réalisable de tout état $s \in S_0$. On note alors $\mathcal{A} \models \phi$*
3. *ϕ est valide si tout automate est un de modèle de ϕ. On note alors $\models \phi$*

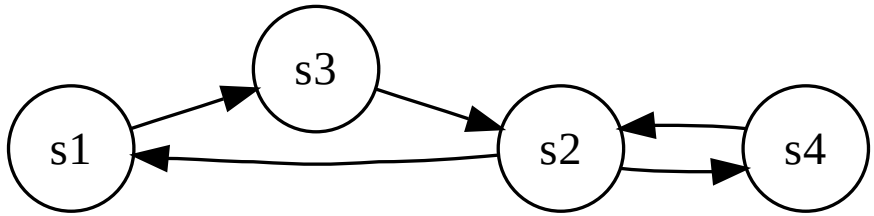
Model-checking : étant donné un automate $\mathcal{A}$ et une propriété spécifiée par une formule ϕ, on vérifie :

$$\mathcal{A} \models \phi$$

Quelques petits exercices

Exercice

L'automate ci-dessous modélise un feux de circulation avec comme fonction d'interprétation : $\rho(s_1) = v, \rho(s_2) = o, \rho(s_3) = r$ *et* $\rho(s_4) = i$.



1. $\mathcal{A}, s_1 \stackrel{?}{\models} NN(o)$
2. $\mathcal{A}, (s_1 s_3 s_2)^* (s_4 s_2)^\omega \stackrel{?}{\models} (\neg i) U i$
3. $\mathcal{A}, (s_1 s_3 s_2)^* (s_4 s_2)^\omega \stackrel{?}{\models} \Diamond \Box \neg r$
4. $\mathcal{A}, s_4 \stackrel{?}{\models} (\neg i) U i$
6. $\mathcal{A} \stackrel{?}{\models} \Diamond o$
7. $\mathcal{A}, (s_1 s_3 s_2)^\omega \stackrel{?}{\models} \Box \neg i$
8. $\mathcal{A} \stackrel{?}{\models} \Box \Diamond o$
9. $\mathcal{A}, s_1 \stackrel{?}{\models} (\neg i) U i$

Encore un petit pour la route

Exercice

- *Canal unidirectionnel parfait entre un émetteur S et un récepteur R*
- *S et R sont munis d'un tampon de capacité infinie parfait $S.out$ et $R.in$*
- *Un message m envoyé par S est inséré dans $S.out$, puis acheminé par le canal et enfin présent dans $R.in$*
- *M est un ensemble de messages fini*
- $PA_V = \{m \in S.out : m \in M\} \cup \{m \in R.in : m \in M\}$

Encore un petit pour la route

$PA_V = \{m \in S.out : m \in M\} \cup \{m \in R.in : m \in M\}$
Exprimer par des formules de L_1 *étendu* les propriétés suivantes :

1. *Un message ne peut pas être dans les deux tampons en même temps*
2. *Le canal ne perd pas de message*
3. *Le canal préserve à la sortie l'ordre d'entrée des messages*
4. *Le canal ne génère pas spontanément de messages*

Outline

Les automates de Büchi

Structure reconnaissant des $\omega-$mots

Définition

Un automate de Büchi est un quintuplet $\mathcal{B} = \langle S, T, S_0, F, \Sigma \rangle$ *où*

- S *est un ensemble d'états*
- $S_0 \subseteq S$ *est l'ensemble des états initiaux*
- $T \subseteq S \times \Sigma \times S$ *est l'ensemble des transitions*
- $F \subseteq S$ *est l'ensemble des états finaux*
- Σ *est un alphabet*

Les automates de Büchi

Structure reconnaissant des $\omega-$mots

Définition

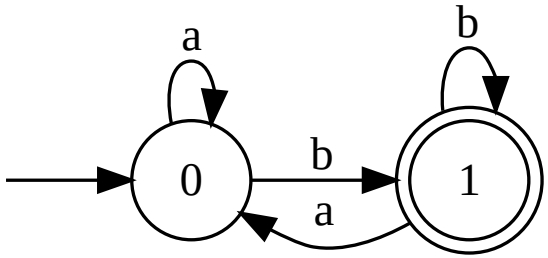
Un automate de Büchi est un quintuplet $\mathcal{B} = \langle S, T, S_0, F, \Sigma \rangle$ *où*

- S *est un ensemble d'états*
- $S_0 \subseteq S$ *est l'ensemble des états initiaux*
- $T \subseteq S \times \Sigma \times S$ *est l'ensemble des transitions*
- $F \subseteq S$ *est l'ensemble des états finaux*
- Σ *est un alphabet*

Critère d'acceptation de Büchi

Un $\omega-$mot sur Σ est reconnaissable par $\mathcal{B}$ si la chaîne des états visités par l'automate passe infiniment souvent par des états de F

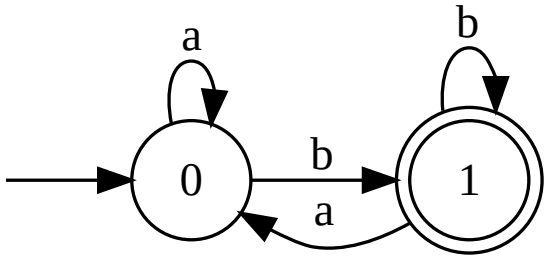
Automate de Büchi



L'ensemble des $\omega-$mots reconnaissable par l'automate est

$$(a^* bb * a)^\omega + (a^* bb * a)^* a^* b(b)^\omega$$

Automate de Büchi



L'ensemble des $\omega-$mots reconnaissable par l'automate est

$$(a^* bb * a)^\omega + (a^* bb * a)^* a^* b(b)^\omega$$

Théorème

Un $\omega-$langage est reconnaissable par un automate de Büchi ssi il est $\omega-$régulier

Automate de Büchi

Le vide est décidable pour les automates de Büchi

Automate de Büchi

Le vide est décidable pour les automates de Büchi

Algorithme de Tarjan-Paige

- Enumération des composantes fortement connexes atteignables à partir de S_0
- Langage est vide si toutes les CFC ne passent pas par un état final

Automate de Büchi

Le vide est décidable pour les automates de Büchi

Algorithme de Tarjan-Paige

- Enumération des composantes fortement connexes atteignables à partir de S_0
- Langage est vide si toutes les CFC ne passent pas par un état final

Model-checking se réduit au problème du vide

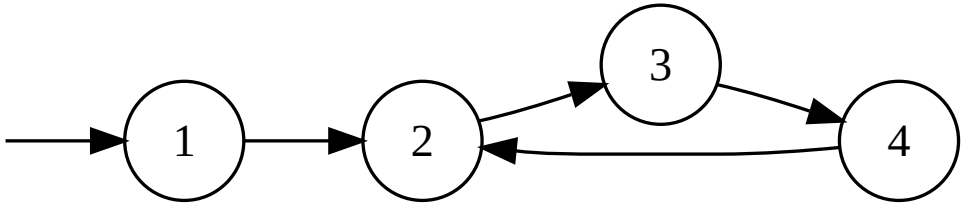
Transformation d'un système en automate de Büchi

Avec $PA_V = \{p, q, r, s, t\}$, on peut construire 2^5 valuations possibles représentables par 2^5 vecteurs de dimension 5

Exemple

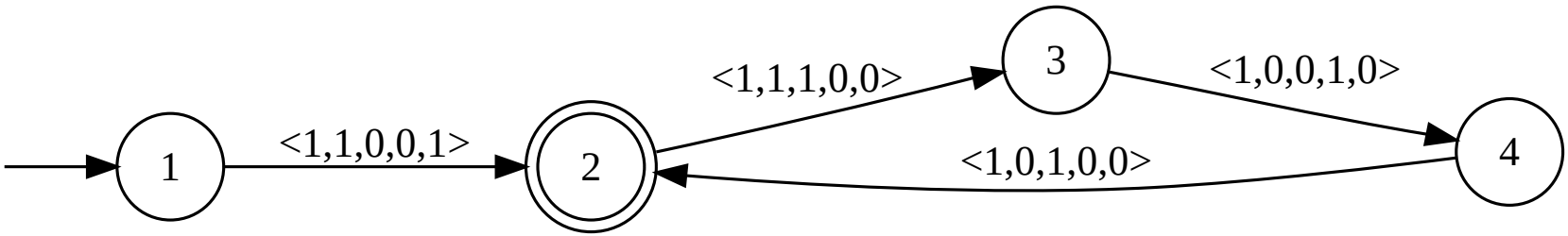
- $\langle 0, 0, 1, 0, 1 \rangle$ *représente* $\{r, t\}$
- $\langle 1, 1, 1, 0, 0 \rangle$ *représente* $\{p, q, r\}$
- $\langle 0, 0, 0, 0, 0 \rangle$ *représente* $\varnothing$

Transformation d'un système en automate de Büchi



avec $\rho(1) = \{p, q, t\}$, $\rho(2) = \{p, q, r\}$, $\rho(3) = \{p, s\}$ et $\rho(4) = \{p, r\}$

En considérant les vecteurs $\langle p, q, r, s, t \rangle$, le chemin $1(234)^\omega$ et les propositions mises en jeu peuvent être représentées par



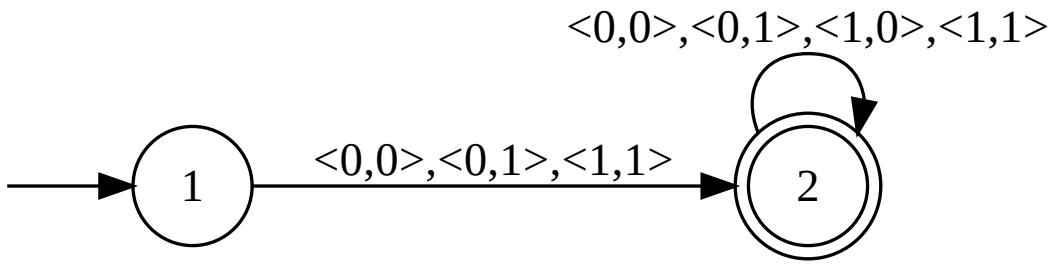
Lien entre logique temporelle linéaire / automates de Büchi

Théorème

(Wolper-Vardi,Sistla, 1983) Etant donné une formule ϕ de la LTL, on peut construire un automate de Büchi $\mathcal{A}_\phi = \langle S, T, S_0, F, \Sigma \rangle$ avec $\Sigma = 2^{PA}$ et $|S| \leq 2^{O(|\phi|)}$ tel que $\mathcal{L}(\mathcal{A})$ est exactement l'ensemble des modèles de ϕ

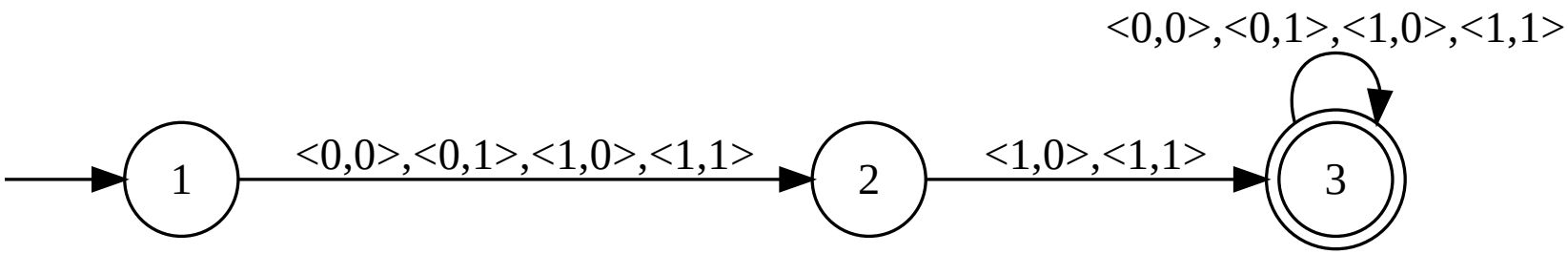
$p_1 \Rightarrow p_2$

$p_1 \Rightarrow p_2$



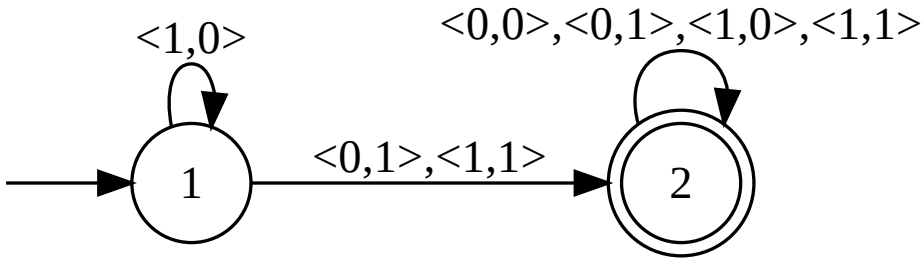
Np_1

Np_1



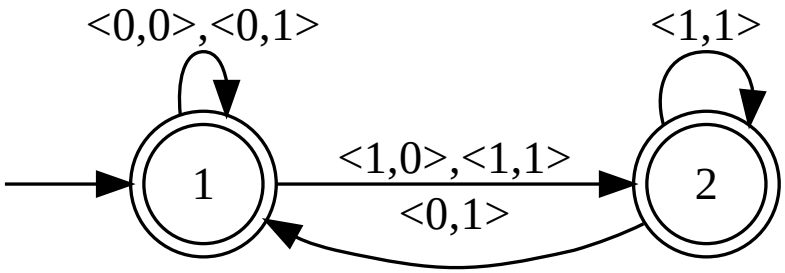
$p_1 U p_2$

$p_1 U p_2$



$$\Box(p_1 \Rightarrow Np_2)$$

$\Box(p_1 \Rightarrow Np_2)$



Model-Checking LTL

$\mathcal{A} \vDash \phi$ revient à

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}_\phi) \equiv \mathcal{L}(\mathcal{A}) \cap (\mathcal{L}(\mathcal{B}_\phi))^c \equiv \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}_{\neg\phi}) = \varnothing$$

Model-Checking LTL

$\mathcal{A} \vDash \phi$ revient à

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}_\phi) \equiv \mathcal{L}(\mathcal{A}) \cap (\mathcal{L}(\mathcal{B}_\phi))^c \equiv \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}_{\neg\phi}) = \varnothing$$

- Calcul de l'intersection plus simple que l'inclusion

Model-Checking LTL

$\mathcal{A} \vDash \phi$ revient à

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}_\phi) \equiv \mathcal{L}(\mathcal{A}) \cap (\mathcal{L}(\mathcal{B}_\phi))^c \equiv \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}_{\neg\phi}) = \varnothing$$

- Calcul de l'intersection plus simple que l'inclusion
- Calcul du complément est difficile

Model-Checking LTL

$\mathcal{A} \vDash \phi$ revient à

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}_\phi) \equiv \mathcal{L}(\mathcal{A}) \cap (\mathcal{L}(\mathcal{B}_\phi))^c \equiv \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}_{\neg\phi}) = \varnothing$$

- Calcul de l'intersection plus simple que l'inclusion
- Calcul du complément est difficile

Exemple

$$\phi = \Box(p_1 \Rightarrow N \Diamond p_2)$$

Exprimer $\neg\phi$

Exemple

$$\phi = \Box(p_1 \Rightarrow N \Diamond p_2)$$

Exprimer $\neg\phi$

$$\neg\phi = \Diamond(p_1 \wedge N \Box \neg p_2)$$

Exemple

$$\phi = \Box(p_1 \Rightarrow N \Diamond p_2)$$

Exprimer $\neg\phi$

$$\neg\phi = \Diamond(p_1 \wedge N \Box \neg p_2)$$

Exprimer cette formule sous forme d'automate

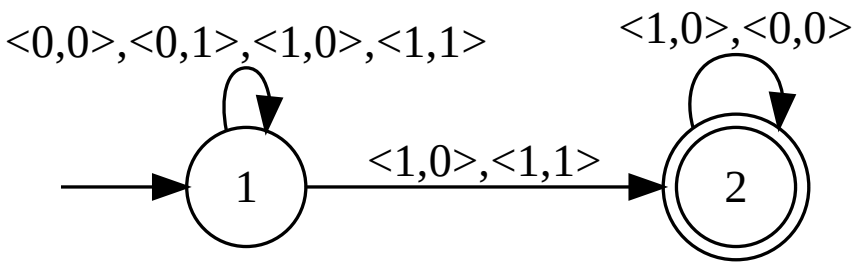
Exemple

$$\phi = \Box(p_1 \Rightarrow N \Diamond p_2)$$

Exprimer $\neg\phi$

$$\neg\phi = \Diamond(p_1 \wedge N \Box \neg p_2)$$

Exprimer cette formule sous forme d'automate



Intersection automates

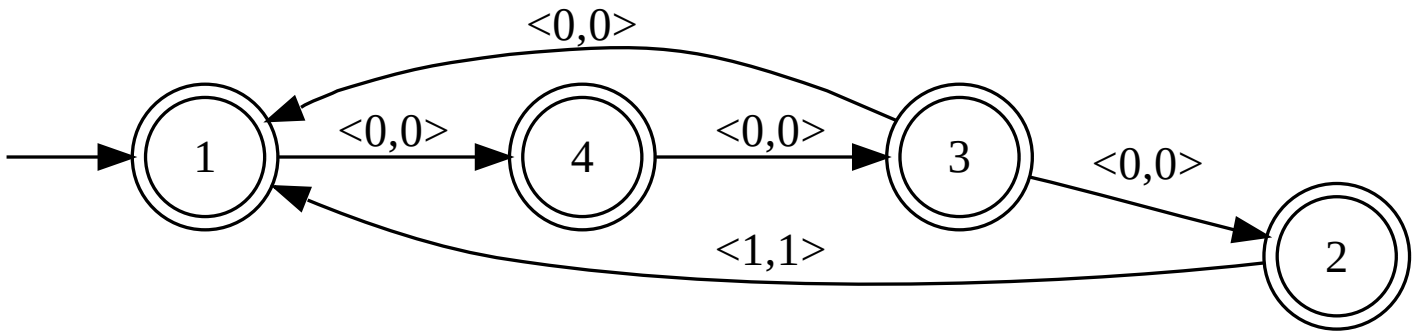
Définition

Soit $\mathcal{A} = \langle S_1, S_{01}, T_1, F_1, \Sigma \rangle$ *et* $\mathcal{B}_{\neg\phi} = \langle S_2, S_{02}, T_2, F_2, \Sigma \rangle$ *alors* $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}_{\neg\phi})$ *est l'ensemble des* $\omega-$*mots reconnaissables par l'automate de Büchi* $\mathcal{A} \otimes \mathcal{B}_{\neg\phi} = \langle S, S_0, T, F, \Sigma \rangle$ *où*

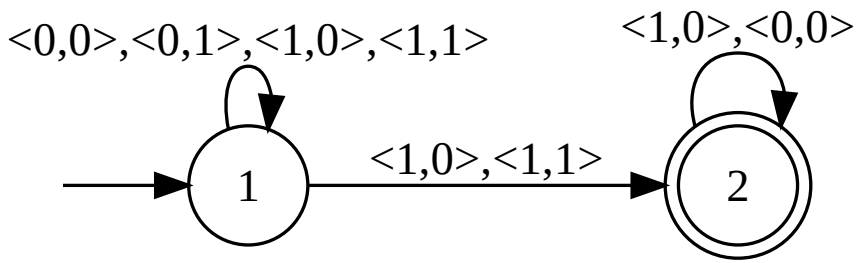
- $S = S_1 \times S_2$
- $S_0 = S_{01} \times S_{02}$
- T *est la synchronisation de* T_1 *et* T_2 *sur les actions identiques*
- $F = F_1 \times F_2$

Vérification de ϕ par Model-Checking

Soit le système spécifié par l'automate $\mathcal{A}$ suivant

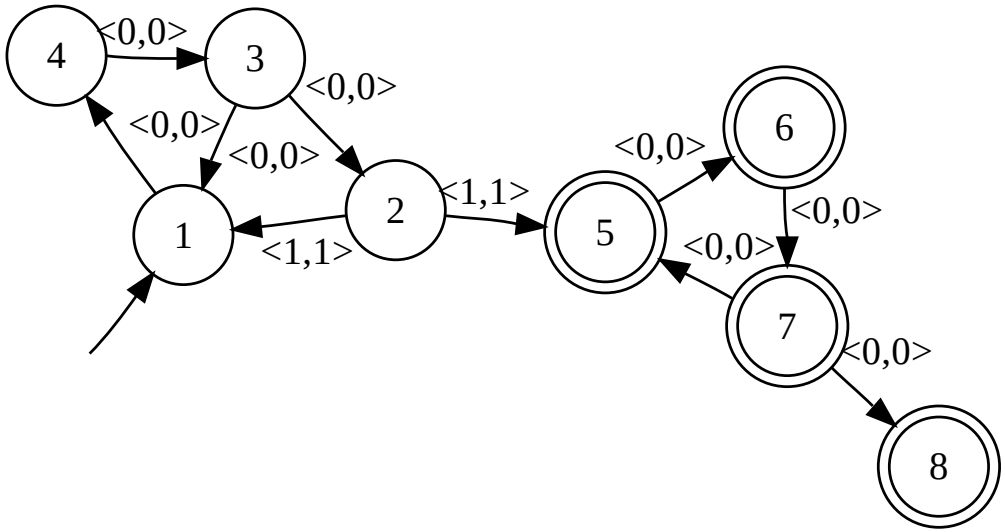


Pour rappel, $\mathcal{B}_{\neg(\Box(p_1 \Rightarrow N \Diamond p_2))}$ est l'automate suivant

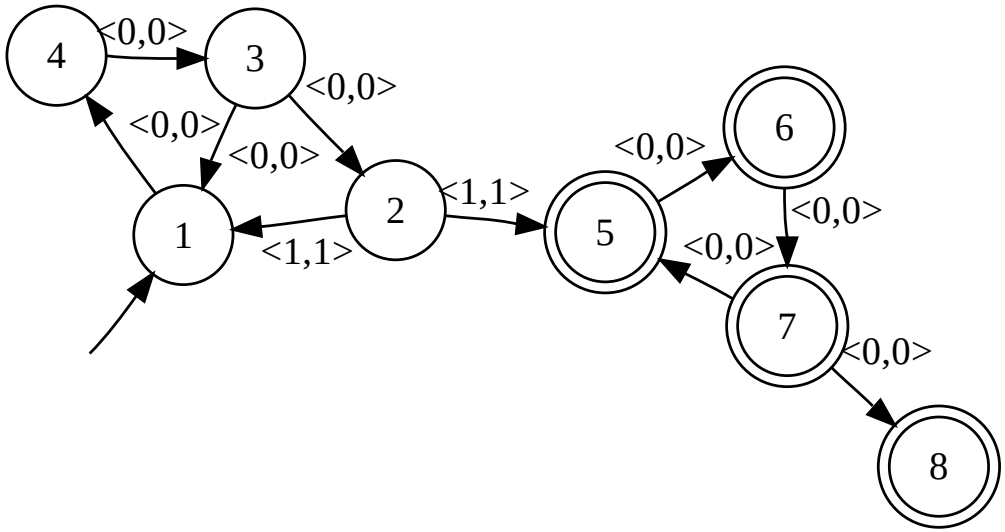


Calculer le produit synchronisé $\mathcal{A} \|_{Sync} \mathcal{B}_{\neg(\Box(p_1 \Rightarrow N \Diamond p_2))}$

Résultat du produit

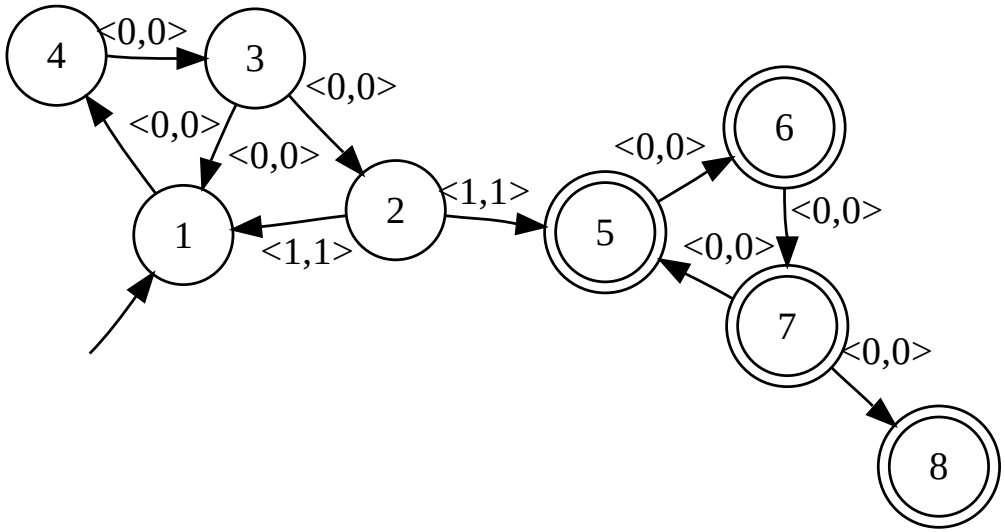


Résultat du produit



$$\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}_{\neg\phi}) \neq \varnothing$$

Résultat du produit



$$\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}_{\neg\phi}) \neq \varnothing$$

Par conséquent,

$$\mathcal{A} \not\models \Box(p_1 \Rightarrow N \Diamond p_2)$$

Outline

Et alors ?

- C'est bien beau de spécifier des systèmes de transitions par des automates. . .

Et alors ?

- C'est bien beau de spécifier des systèmes de transitions par des automates...
  - Définition de langages de haut-niveau : Promela, CASPER, ...

Et alors ?

- C'est bien beau de spécifier des systèmes de transitions par des automates...
  - Définition de langages de haut-niveau : Promela, CASPER, ...
- Y a t'il des outils automatiques performants ?

Et alors ?

- C'est bien beau de spécifier des systèmes de transitions par des automates...
  - Définition de langages de haut-niveau : Promela, CASPER, ...
- Y a t'il des outils automatiques performants ? Of course there are

Et alors ?

- C'est bien beau de spécifier des systèmes de transitions par des automates...
  - Définition de langages de haut-niveau : Promela, CASPER, ...
- Y a t'il des outils automatiques performants ? Of course there are
  - SPIN

Et alors ?

- C'est bien beau de spécifier des systèmes de transitions par des automates...
  - Définition de langages de haut-niveau : Promela, CASPER, ...
- Y a t'il des outils automatiques performants ? Of course there are
  - SPIN
  - FDR

Et alors ?

- C'est bien beau de spécifier des systèmes de transitions par des automates...
  - Définition de langages de haut-niveau : Promela, CASPER, ...
- Y a t'il des outils automatiques performants ? Of course there are
  - SPIN
  - FDR
  - CADP

Et alors ?

- C'est bien beau de spécifier des systèmes de transitions par des automates...
  - Définition de langages de haut-niveau : Promela, CASPER, ...
- Y a t'il des outils automatiques performants ? Of course there are
  - SPIN
  - FDR
  - CADP
  - ...

Architecture de SPIN