Introduction à la modélisation et à la vérification

Application aux systèmes temporisés

Patricia Bouyer

LSV – CNRS & ENS de Cachan

Modélisation & Vérification

Introduction

Systèmes informatiques

- ordinateurs personnels, serveurs
- systèmes embarqués
  - téléphones mobiles
  - avions
  - fusées
  - Lego Mindstorms
  - automobiles (X-by-wire)
  - ...

Systèmes informatiques

- ordinateurs personnels, serveurs
- systèmes embarqués
  - téléphones mobiles
  - avions
  - fusées
  - Lego Mindstorms
  - automobiles (X-by-wire)
  - ...

Deux tendances: de plus en plus performants, de plus en plus miniaturisés

➜ de plus en plus complexes

Systèmes informatiques

- ordinateurs personnels, serveurs
- systèmes embarqués
  - téléphones mobiles
  - avions
  - fusées
  - Lego Mindstorms
  - automobiles (X-by-wire)
  - ...

Deux tendances: de plus en plus performants, de plus en plus miniaturisés

➜ de plus en plus complexes

Une caractéristique « nouvelle » : systèmes critiques

➜ fiabilité indispensable !

Quelques bogues célèbres

- **Therac 25**, traitement aux rayons X des tumeurs cancéreuses
  - « `x Up Edit e Enter` » en moins de 8s ⩾ radiation 125 x la dose normale
  - 6 décès aux États-Unis en 1986
  - erreur logicielle !

Quelques bogues célèbres

- **Therac 25**, traitement aux rayons X des tumeurs cancéreuses
  - « `x Up Edit e Enter` » en moins de 8s ⩾ radiation 125 x la dose normale
  - 6 décès aux États-Unis en 1986
  - erreur logicielle !
- **AT&T**
  - un patch non vérifié dans le système d'exploitation
  - une erreur dans un **`switch`** (en *C*)
  - le réseau téléphonique de la côte est des États-Unis a été bloqué pendant 9h !

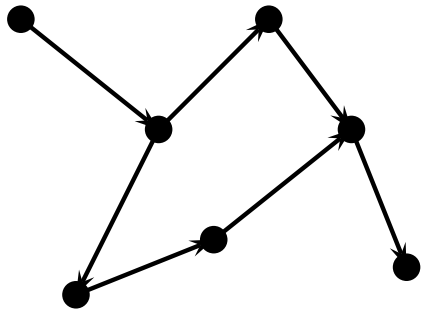
Quelques bogues célèbres

- **Therac 25**, traitement aux rayons X des tumeurs cancéreuses
  - « `x Up Edit e Enter` » en moins de 8s ⩾ radiation 125 x la dose normale
  - 6 décès aux États-Unis en 1986
  - erreur logicielle !
- **AT&T**
  - un patch non vérifié dans le système d'exploitation
  - une erreur dans un **`switch`** (en *C*)
  - le réseau téléphonique de la côte est des États-Unis a été bloqué pendant 9h !
- **Le bogue pentium**
  - une erreur dans la division flottante du processeur
  - heureusement, un chercheur en théorie des nombres veillait...
  - 470 millions de $

Une solution parmi d'autres

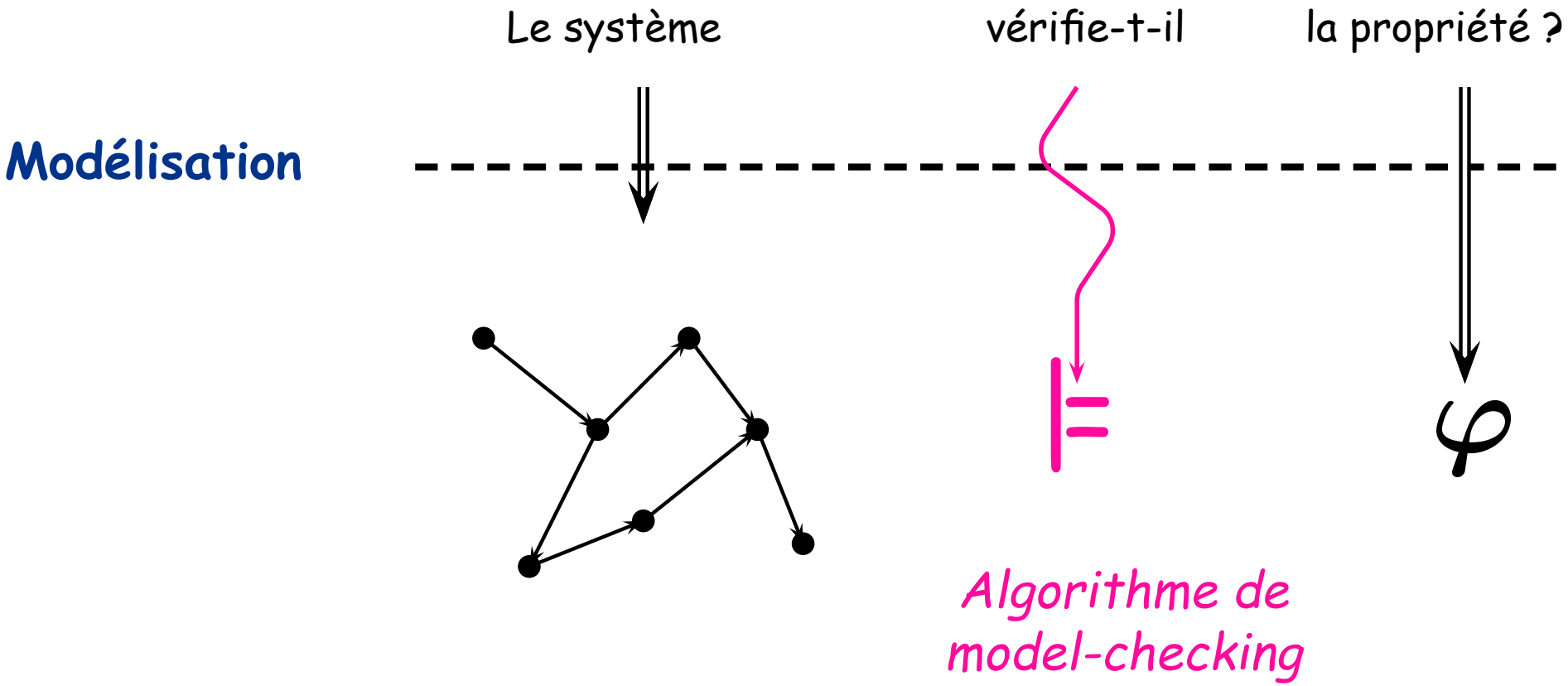
Le système vérifie-t-il la propriété ?

Modélisation



φ

Une solution parmi d'autres



Modélisation & Vérification

Modèles

Que souhaite-t-on modéliser ?

➜ **pour les systèmes à étudier**

- des structures de données (files...)
- des types de données (entiers...)
- des canaux de communication
- des changements discrets d'états
- des changements continus d'états

➜ **pour les propriétés à exprimer**

- des implications, des connections logiques
- des successions temporelles (l'action b a lieu après l'action a)
- des propriétés d'états, de chemins
- des invariances (la température est toujours inférieure à 28 degrés)
- la présence d'un état bloquant

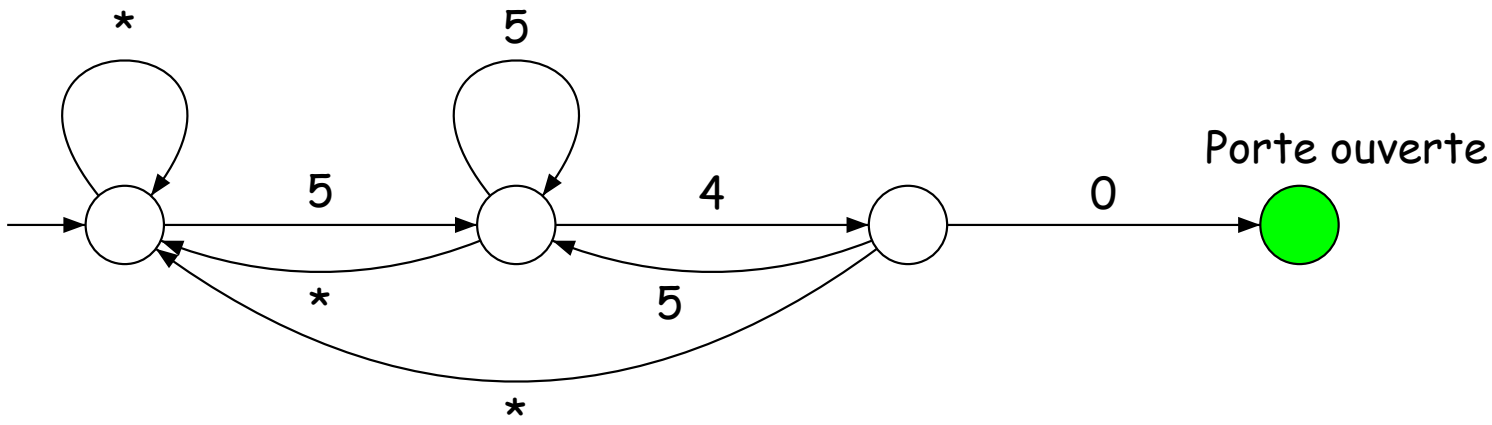
Quels modèles ? (1)

➜ **pour les systèmes à analyser :** des systèmes de transitions

- les automates finis
- les automates à pile
- les automates à compteur
- les automates temporisés
- les automates hybrides
- les réseaux de Petri
- les "message sequence charts"
- les algèbres de processus
- ...

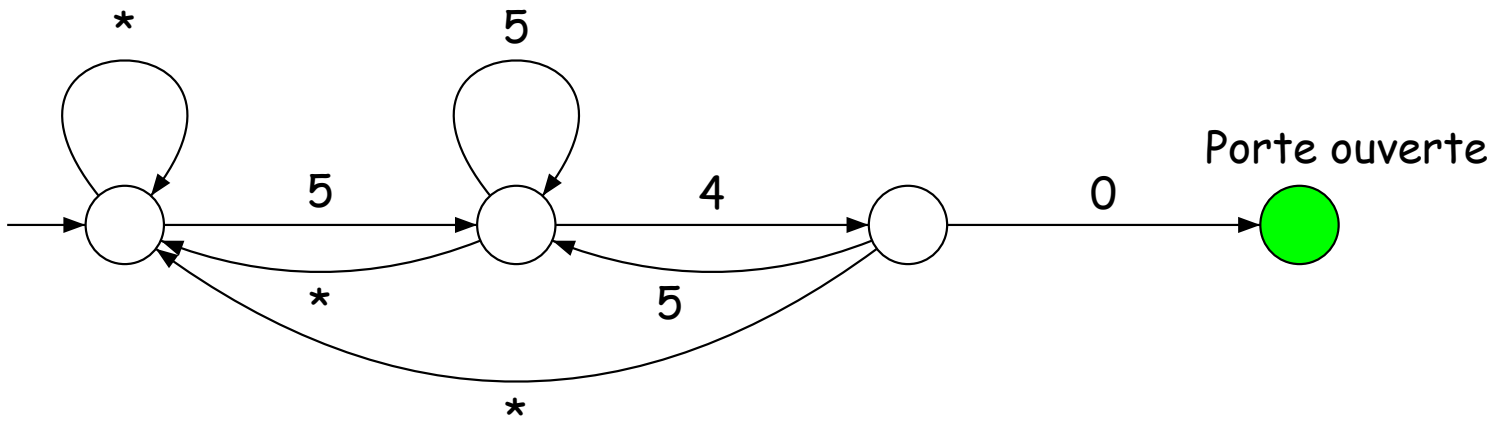
Quelques exemples

➜ **Le digicode :**

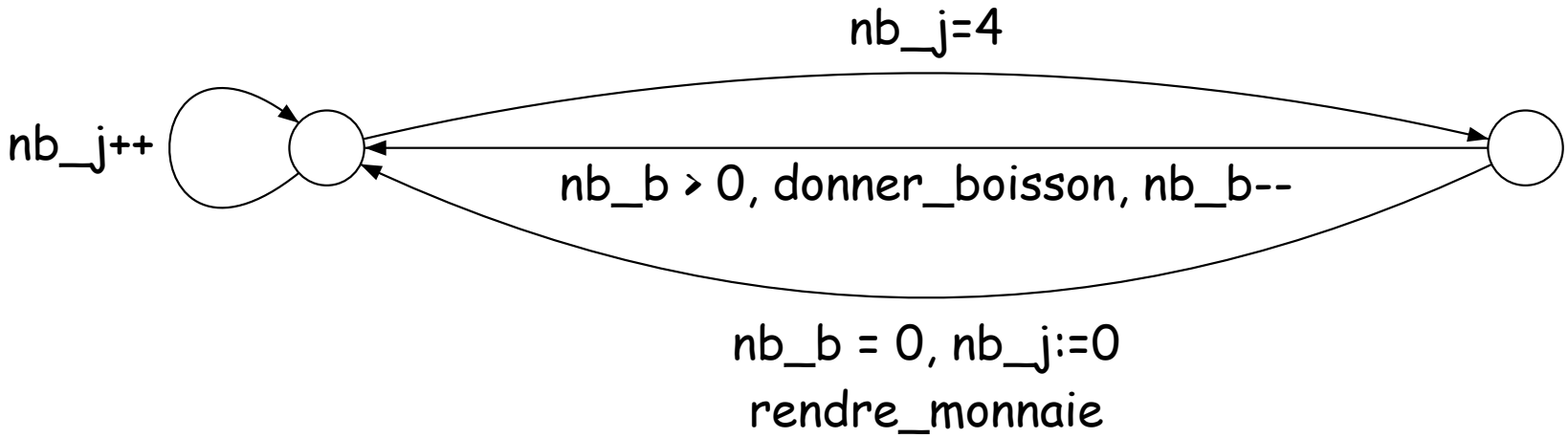


Quelques exemples

- **Le digicode :**

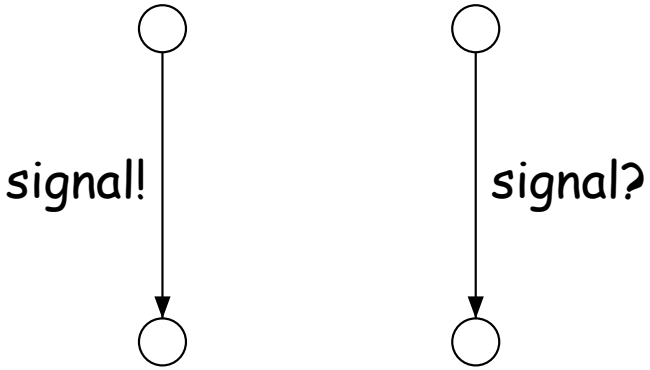


- **Un distributeur**



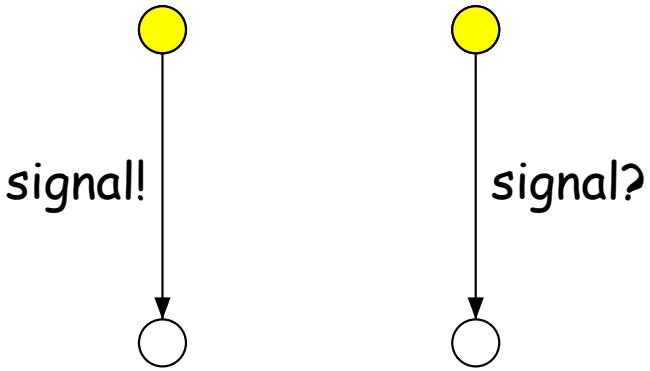
Composition de systèmes

Synchronisation binaire



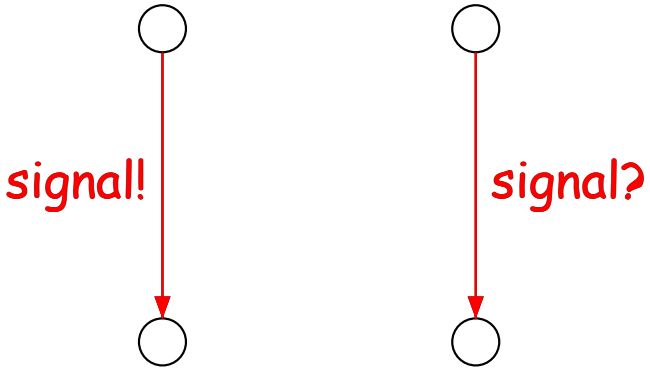
Composition de systèmes

Synchronisation binaire



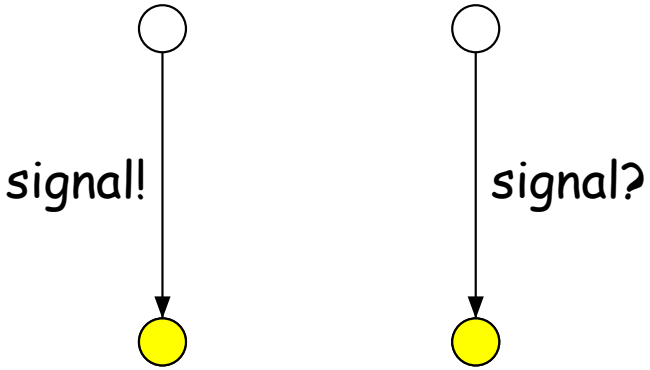
Composition de systèmes

Synchronisation binaire



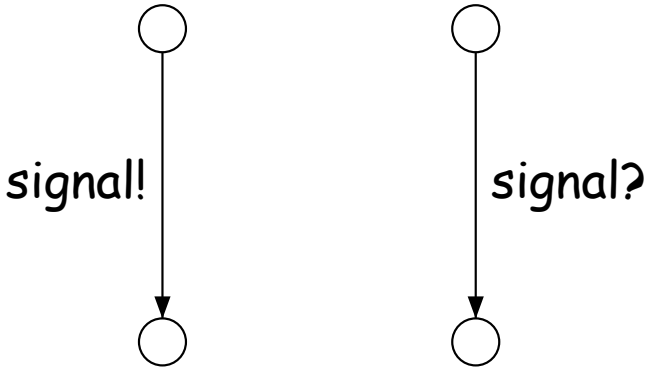
Composition de systèmes

Synchronisation binaire

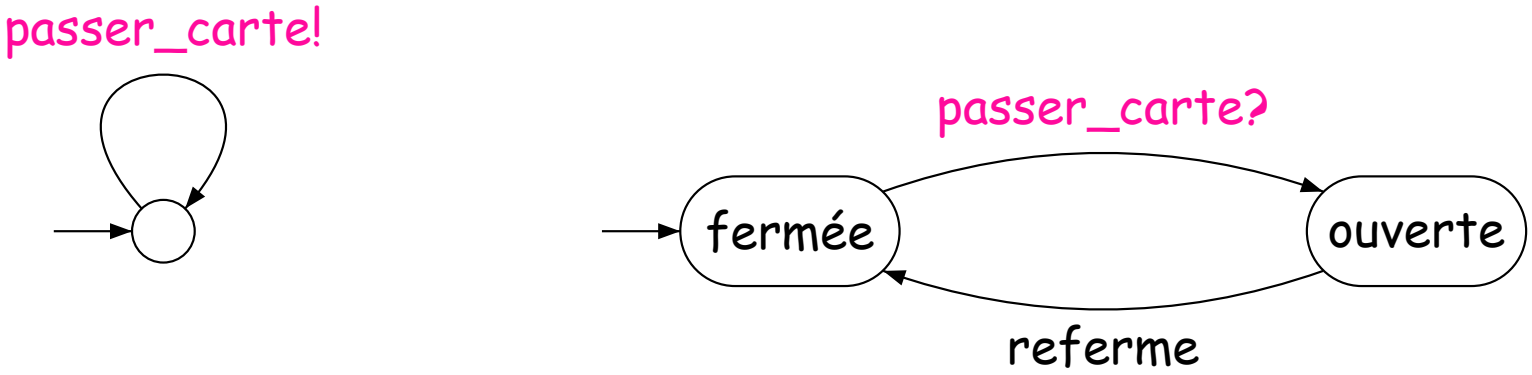


Composition de systèmes

Synchronisation binaire



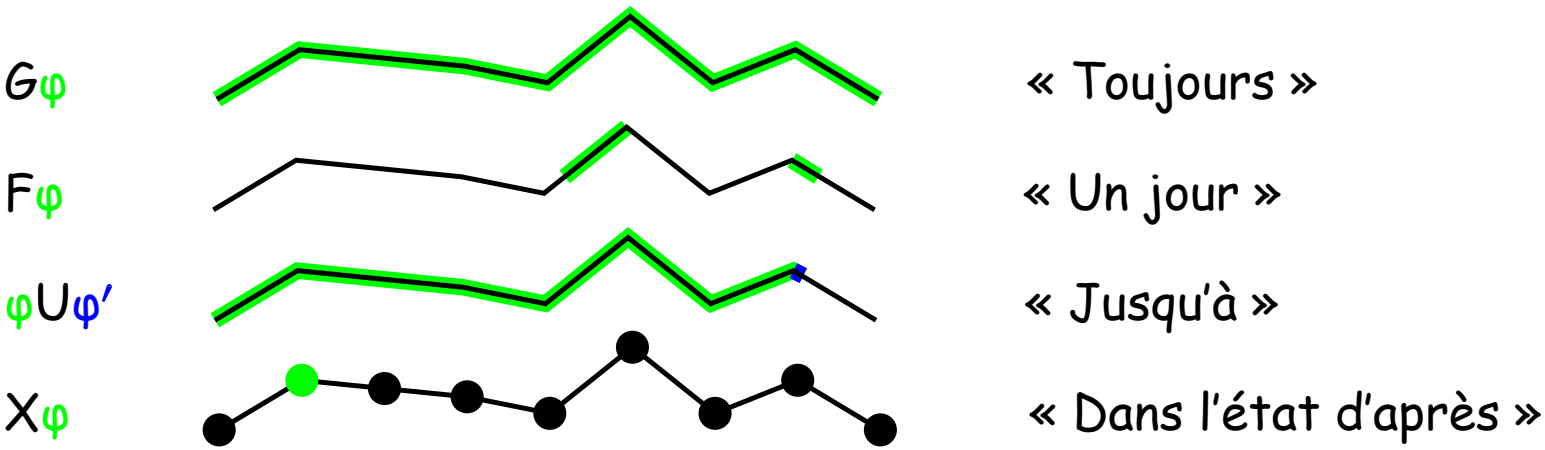
Exemple : entrée dans un bâtiment de l'ENS



Quels modèles ? (2)

➜ **pour les spécifications :** des langages logiques (ex : logique temporelle)

Formules de chemin :



Formules d'états :

[Pnueli 1977]

Quelques exemples

- Il est possible d'atteindre l'état critique.

Quelques exemples

- Il est possible d'atteindre l'état critique.

$$EF(\text{état\_critique})$$

Quelques exemples

- Il est possible d'atteindre l'état critique.

  EF(état_critique)

- Deux processus ne peuvent pas être simultanément en section critique.

Quelques exemples

- Il est possible d'atteindre l'état critique.

$$EF(\text{état\_critique})$$

- Deux processus ne peuvent pas être simultanément en section critique.

$$AG((\neg \text{section\_critique\_1}) \vee (\neg \text{section\_critique\_2}))$$

Quelques exemples

- Il est possible d'atteindre l'état critique.

$$EF(\text{état\_critique})$$

- Deux processus ne peuvent pas être simultanément en section critique.

$$AG((\neg \text{section\_critique\_1}) \vee (\neg \text{section\_critique\_2}))$$

- Le distributeur de billets ne donne pas d'argent tant que le code n'est pas correct.

Quelques exemples

- Il est possible d'atteindre l'état critique.

$$EF(\text{état\_critique})$$

- Deux processus ne peuvent pas être simultanément en section critique.

$$AG((\neg \text{section\_critique\_1}) \vee (\neg \text{section\_critique\_2}))$$

- Le distributeur de billets ne donne pas d'argent tant que le code n'est pas correct.

$$A((\neg \text{donne\_argent})U(\text{code\_correct}))$$

Quelques exemples

- Il est possible d'atteindre l'état critique.

$$EF(\text{état\_critique})$$

- Deux processus ne peuvent pas être simultanément en section critique.

$$AG((\neg \text{section\_critique\_1}) \vee (\neg \text{section\_critique\_2}))$$

- Le distributeur de billets ne donne pas d'argent tant que le code n'est pas correct.

$$A((\neg \text{donne\_argent})U(\text{code\_correct}))$$

- Si l'ascenseur est appelé au $6^{\text{ème}}$ étage, alors il s'y arrêtera.

Quelques exemples

- Il est possible d'atteindre l'état critique.

$$EF(\text{état\_critique})$$

- Deux processus ne peuvent pas être simultanément en section critique.

$$AG((\neg \text{section\_critique\_1}) \vee (\neg \text{section\_critique\_2}))$$

- Le distributeur de billets ne donne pas d'argent tant que le code n'est pas correct.

$$A((\neg \text{donne\_argent})U(\text{code\_correct}))$$

- Si l'ascenseur est appelé au $6^{\text{ème}}$ étage, alors il s'y arrêtera.

$$AG(\text{appel\_6} \Rightarrow (AF\ \text{arrêt\_6}))$$

Quelques exemples

- Il est possible d'atteindre l'état critique.

$$EF(\text{état\_critique})$$

- Deux processus ne peuvent pas être simultanément en section critique.

$$AG((\neg \text{section\_critique\_1}) \vee (\neg \text{section\_critique\_2}))$$

- Le distributeur de billets ne donne pas d'argent tant que le code n'est pas correct.

$$A((\neg \text{donne\_argent})U(\text{code\_correct}))$$

- Si l'ascenseur est appelé au $6^{\text{ème}}$ étage, alors il s'y arrêtera.

$$AG(\text{appel\_6} \Rightarrow (AF\ \text{arrêt\_6}))$$

- La barrière du passage à niveau est ouverte infiniment souvent.

Quelques exemples

- Il est possible d'atteindre l'état critique.

$$EF(\text{état\_critique})$$

- Deux processus ne peuvent pas être simultanément en section critique.

$$AG((\neg \text{section\_critique\_1}) \vee (\neg \text{section\_critique\_2}))$$

- Le distributeur de billets ne donne pas d'argent tant que le code n'est pas correct.

$$A((\neg \text{donne\_argent})U(\text{code\_correct}))$$

- Si l'ascenseur est appelé au $6^{\text{ème}}$ étage, alors il s'y arrêtera.

$$AG(\text{appel\_6} \Rightarrow (AF\ \text{arrêt\_6}))$$

- La barrière du passage à niveau est ouverte infiniment souvent.

$$AG\ AF(\text{barrière\_ouverte})$$

Quelques exemples (suite)

- Après la pluie, le beau temps.

Quelques exemples (suite)

- Après la pluie, le beau temps.

$$G(\text{pluie} \Rightarrow F\ \text{beau\_temps})$$

Quelques exemples (suite)

- Après la pluie, le beau temps.

$$G(\text{pluie} \Rightarrow F\ \text{beau\_temps})$$

- C'est en forgeant qu'on devient forgeron.

Quelques exemples (suite)

- Après la pluie, le beau temps.

$$G(\text{pluie} \Rightarrow F\ \text{beau\_temps})$$

- C'est en forgeant qu'on devient forgeron.

$$(F\ \text{forgeron}) \Rightarrow (\text{forger}\ U\ \text{forgeron})$$

Quelques exemples (suite)

- Après la pluie, le beau temps.

$$G(\text{pluie} \Rightarrow F\ \text{beau\_temps})$$

- C'est en forgeant qu'on devient forgeron.

$$(F\ \text{forgeron}) \Rightarrow (\text{forger}\ U\ \text{forgeron})$$

- Qui veut aller loin ménage sa monture.

Quelques exemples (suite)

- Après la pluie, le beau temps.

$$G(\text{pluie} \Rightarrow F\ \text{beau\_temps})$$

- C'est en forgeant qu'on devient forgeron.

$$(F\ \text{forgeron}) \Rightarrow (\text{forger}\ U\ \text{forgeron})$$

- Qui veut aller loin ménage sa monture.

$$G((F\ \text{aller\_loin}) \Rightarrow \text{ménager\_monture})$$

Quelques exemples (suite)

- Après la pluie, le beau temps.

$$G(\text{pluie} \Rightarrow F\ \text{beau\_temps})$$

- C'est en forgeant qu'on devient forgeron.

$$(F\ \text{forgeron}) \Rightarrow (\text{forger}\ U\ \text{forgeron})$$

- Qui veut aller loin ménage sa monture.

$$G((F\ \text{aller\_loin}) \Rightarrow \text{ménager\_monture})$$

- Tel qui rit vendredi, dimanche pleurera.

Quelques exemples (suite)

- Après la pluie, le beau temps.

$$G(\text{pluie} \Rightarrow F\ \text{beau\_temps})$$

- C'est en forgeant qu'on devient forgeron.

$$(F\ \text{forgeron}) \Rightarrow (\text{forger}\ U\ \text{forgeron})$$

- Qui veut aller loin ménage sa monture.

$$G((F\ \text{aller\_loin}) \Rightarrow \text{ménager\_monture})$$

- Tel qui rit vendredi, dimanche pleurera.

$$G((\text{rire} \wedge \text{vendredi}) \Rightarrow (\neg\text{dimanche}\ U\ (\text{dimanche} \wedge \text{pleurer})))$$

Comment choisir les modèles ?

Compromis expressivité / facilité d'analyse

➜ **Expressivité :**

- représenter de nombreux systèmes
- les représenter **succinctement**

➜ **Facilité d'analyse :**

- méthodes efficaces...

Comment choisir les modèles ?

Compromis expressivité / facilité d'analyse

➜ **Expressivité :**

- représenter de nombreux systèmes
- les représenter **succinctement**

➜ **Facilité d'analyse :**

- méthodes efficaces...
- ... si elles existent

Décidabilité ?

« Définition » : un problème est dit décidable s'il existe un algorithme pour le résoudre.

Propriété : il existe des problèmes non décidables

Exemple : un programme termine-t-il ? [Arrêt_programme]

- ➜ première réponse : on le lance et on regarde s'il s'arrête
  ➜ bof, pas très satisfaisant : on attend combien de temps ?
- ➜ considérons le programme suivant :

```
Program P
{
  Si P s'arrête, alors boucler
                 sinon s'arrêter
}
```

Décidabilité ?

```
Program P
{
  Si P s'arrête, alors boucler
                  sinon s'arrêter
}
```

Si le problème « Arrêt_programme » est décidable, alors :

- si **P** s'arrête, alors **P** boucle
- si **P** boucle, alors **P** s'arrête

Remarque : il y a même « plus » de problèmes non décidables que de problèmes décidables.

Un problème indécidable utile

La machine à deux compteurs, ou machine de Minsky.

ensemble fini d'instructions sur deux compteurs, x et y, du type suivant :

- Incrémentation :

```
(p):  x := x+1; goto (q)
```

- Décrémentation :

```
(p):  if x>0 then x := x-1; goto (q) else goto (r)
```

- Instruction spéciale « Arrêt »

Théorème [Minsky 67]. L'arrêt d'une machine à deux compteurs est indécidable.

Notion de réduction

Soient **P1** et **P2** deux problèmes à résoudre. Supposons que l'on ait :

```
Algo(P1)
{
  Quelques calculs;
  Algo(P2);
  Quelques calculs;
}
```

Si **P1** est indécidable, alors **P2** aussi est indécidable.

➜ **P1** a été réduit à **P2**

)

Modélisation & Vérification

Analyse des systèmes

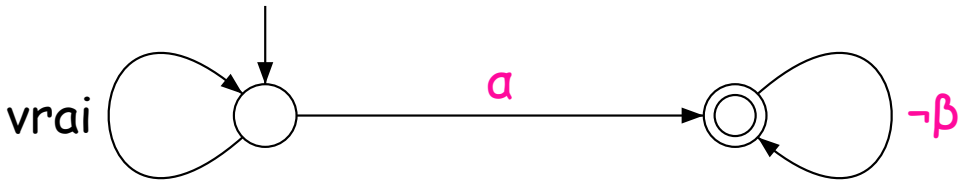
Un exemple simple

Considérons la formule

$$\varphi = G(\alpha \Rightarrow XF\beta)$$

« Toujours, si α est vrai, alors plus tard, il y aura un β »

Idée : exprimer $\neg\varphi$ à l'aide d'un automate, $B_{\neg\varphi}$

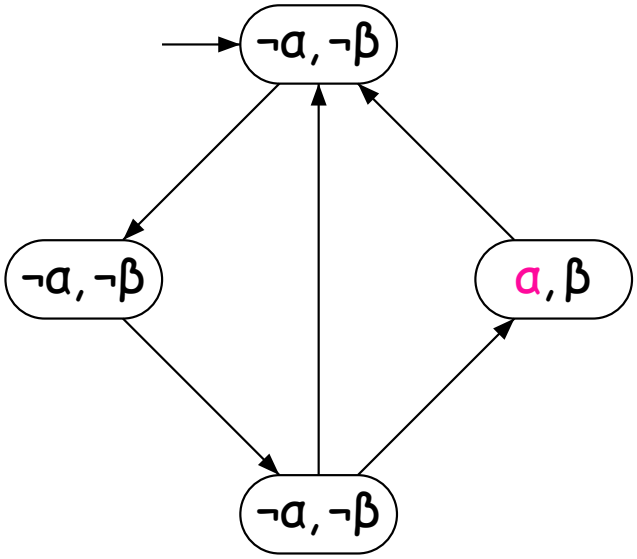


Propriété : si A est un automate fini,

A vérifie φ (équiv. tout chemin de A vérifie φ) ssi $A \otimes B_{\neg\varphi}$ n'accepte aucun mot

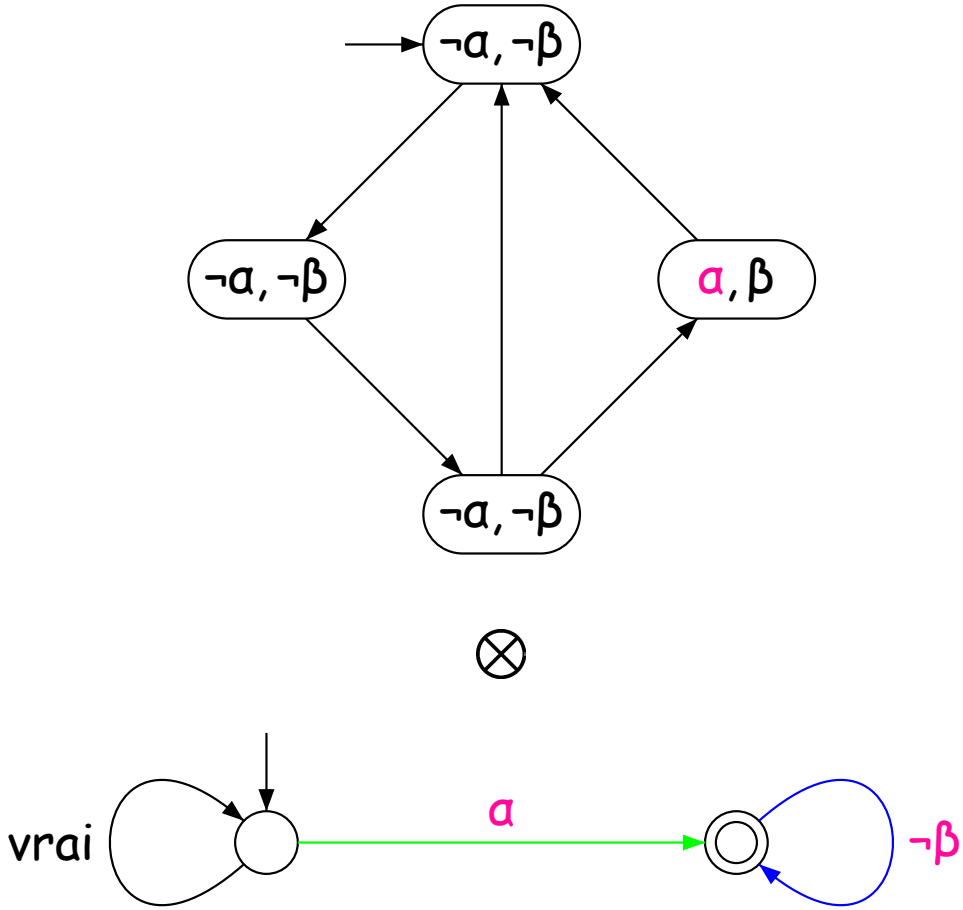
Un exemple simple (suite)

L'automate suivant vérifie-t-il φ ?



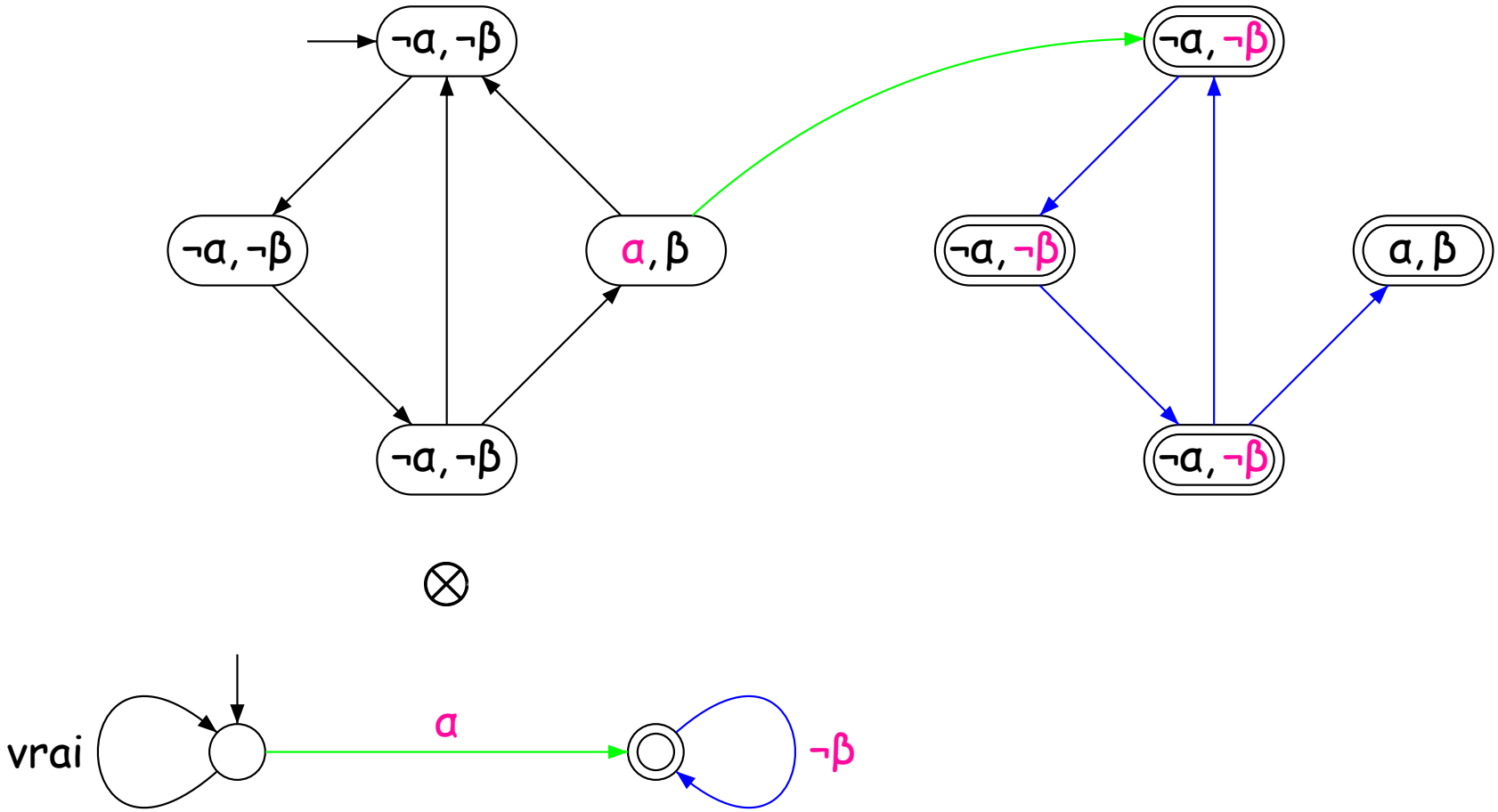
Un exemple simple (suite)

L'automate suivant vérifie-t-il φ ?



Un exemple simple (suite)

L'automate suivant vérifie-t-il φ ?



Ce n'est pas toujours si simple...

Le *problème élémentaire* est le calcul du Post*, i.e. tous les états accessibles à partir de l'état initial. On veut calculer tous les états s_n tels que

$$s_0 \to s_1 \to s_2 \cdots \to s_n \quad \text{(équivalent à } s_0 \to^* s_n\text{)}$$

si s_0 est l'état initial.

Ce n'est pas toujours si simple...

Le *problème élémentaire* est le calcul du Post*, i.e. tous les états accessibles à partir de l'état initial. On veut calculer tous les états s_n tels que

$$s_0 \to s_1 \to s_2 \cdots \to s_n \quad \text{(équivalent à } s_0 \to^* s_n\text{)}$$

si s_0 est l'état initial.

Problèmes potentiels :

- ➜ les systèmes souffrent d'une explosion combinatoire
- ➜ l'ensemble des états accessibles est (très souvent) infini

Ce n'est pas toujours si simple...

Le *problème élémentaire* est le calcul du Post*, i.e. tous les états accessibles à partir de l'état initial. On veut calculer tous les états s_n tels que

$$s_0 \to s_1 \to s_2 \cdots \to s_n \quad \text{(équivalent à } s_0 \to^* s_n\text{)}$$

si s_0 est l'état initial.

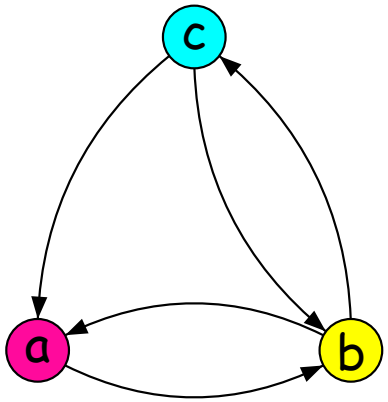
Problèmes potentiels :

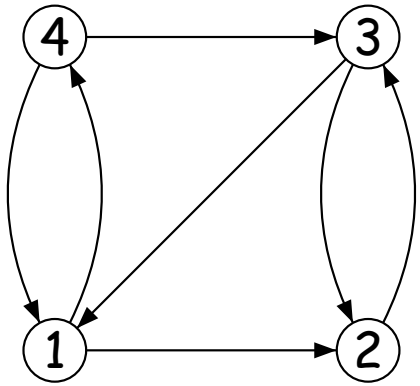
- ➜ les systèmes souffrent d'une explosion combinatoire
- ➜ l'ensemble des états accessibles est (très souvent) infini

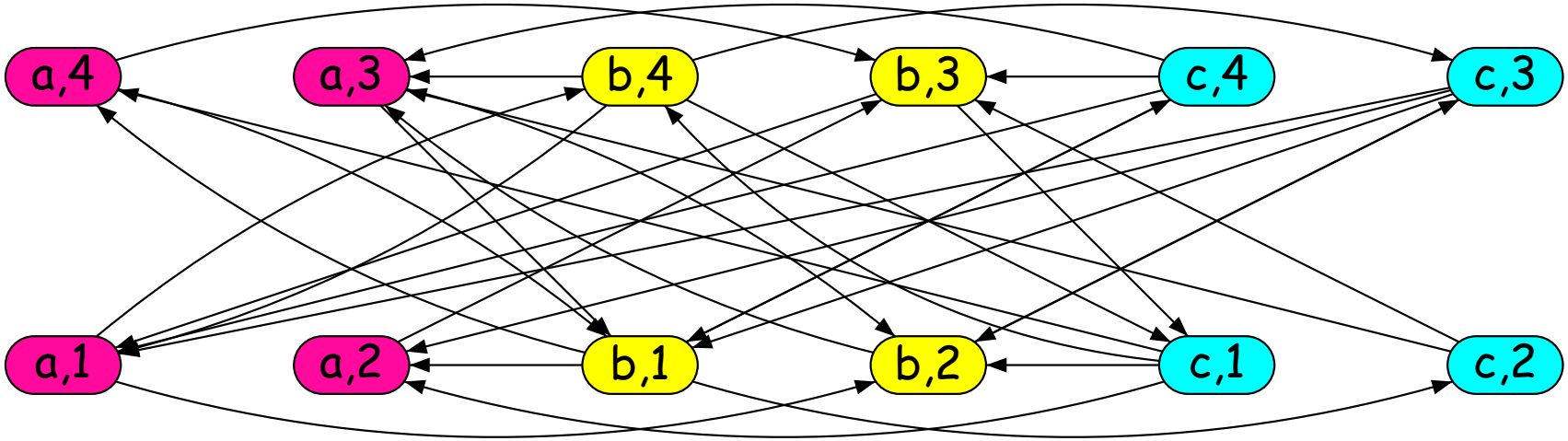
Conséquences :

- ➜ sur la décidabilité
- ➜ sur la complexité

Problème de l'explosion combinatoire







➜ taille du produit exponentielle en le nombre de composants

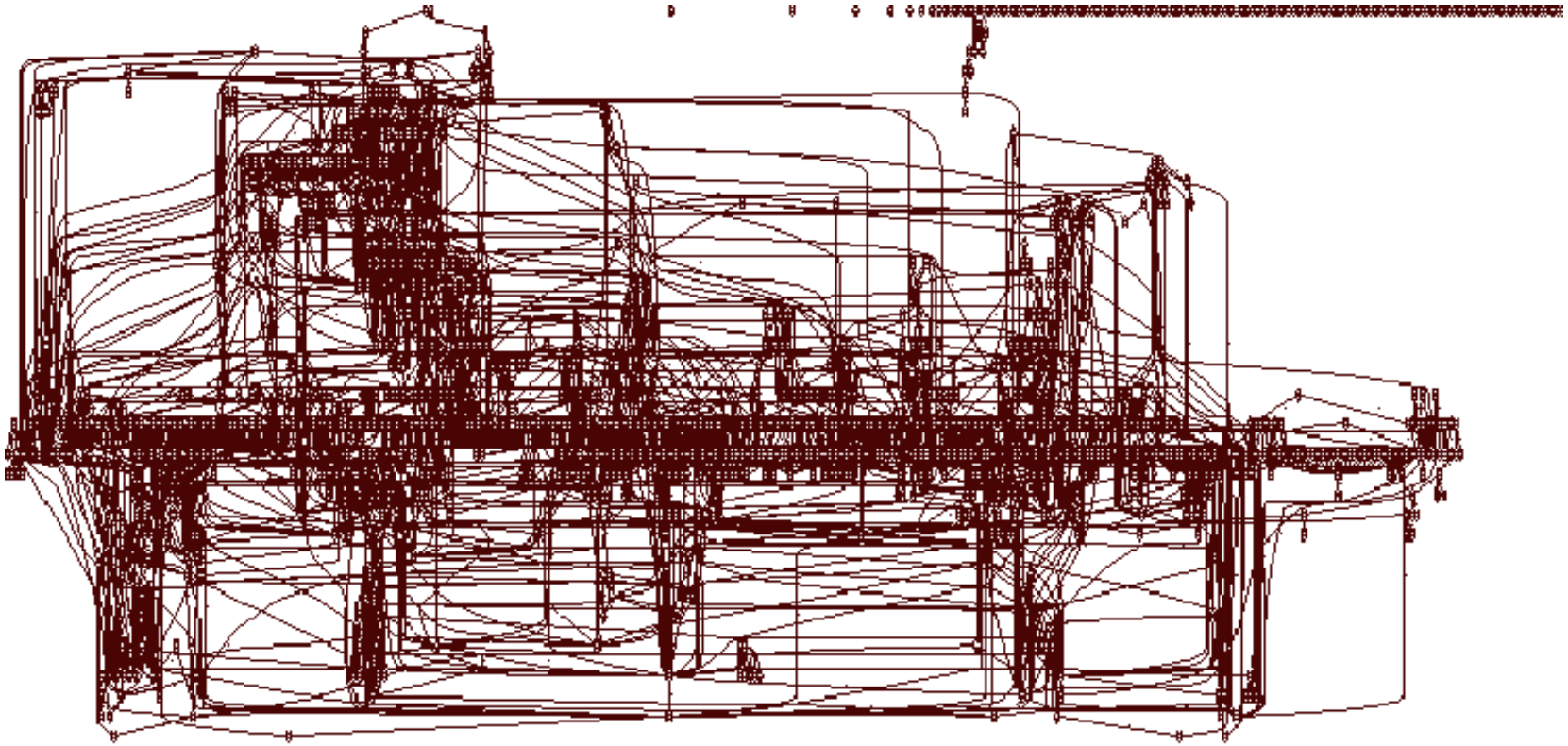
Un simulateur de train

1421 machines

11102 transitions

3204 états locaux

10^{476} états dans le produit synchronisé



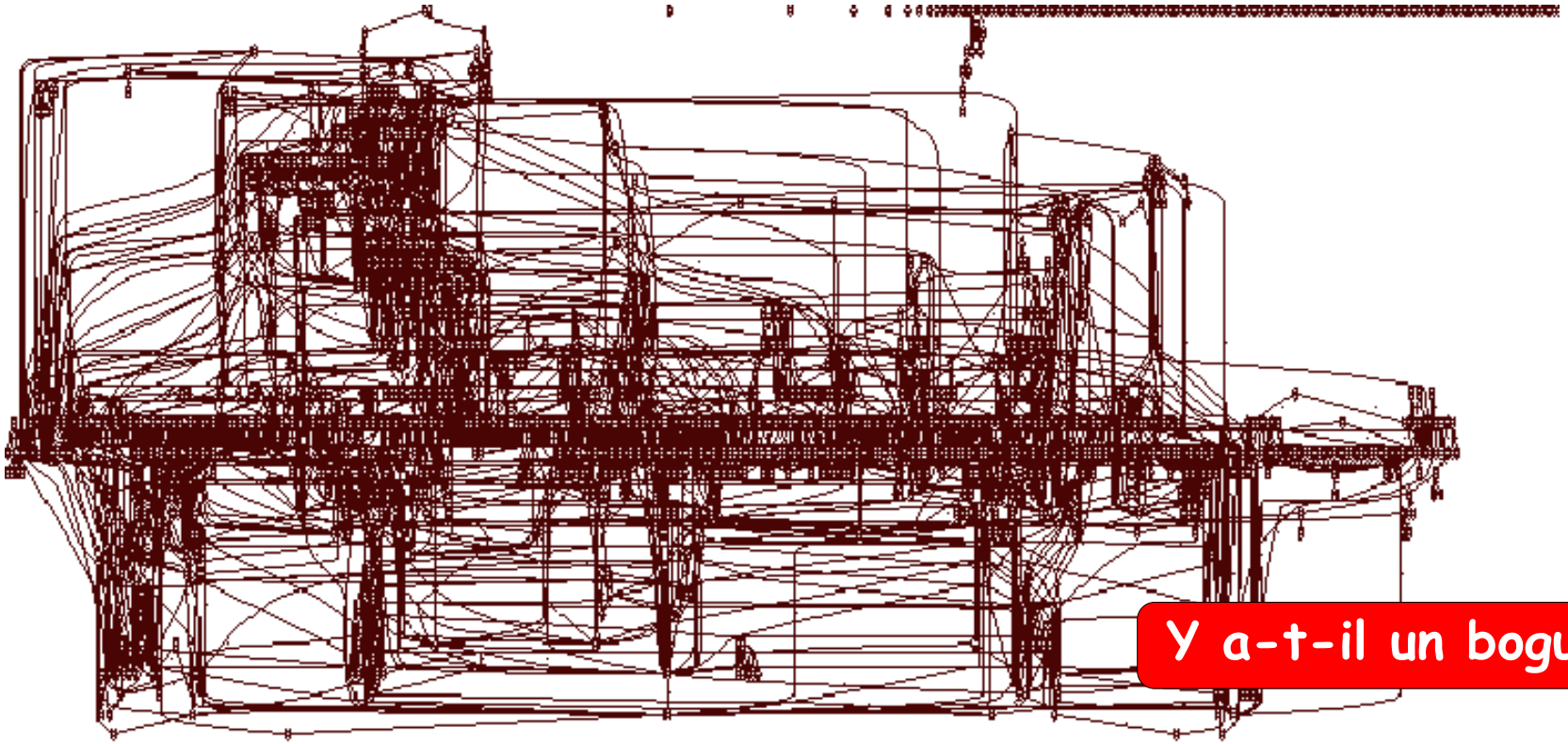
Un simulateur de train

1421 machines

11102 transitions

3204 états locaux

10^{476} états dans le produit synchronisé

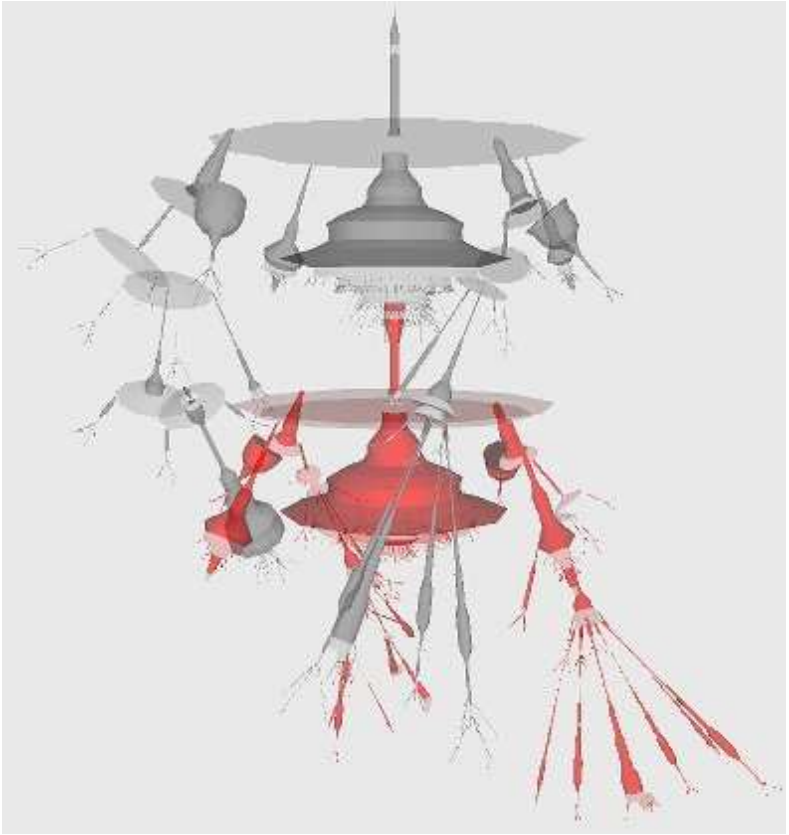


Y a-t-il un bogue ?

Solution : visualisation ?

Outil de visualisation de gros graphes, exemple de la norme IEEE1394

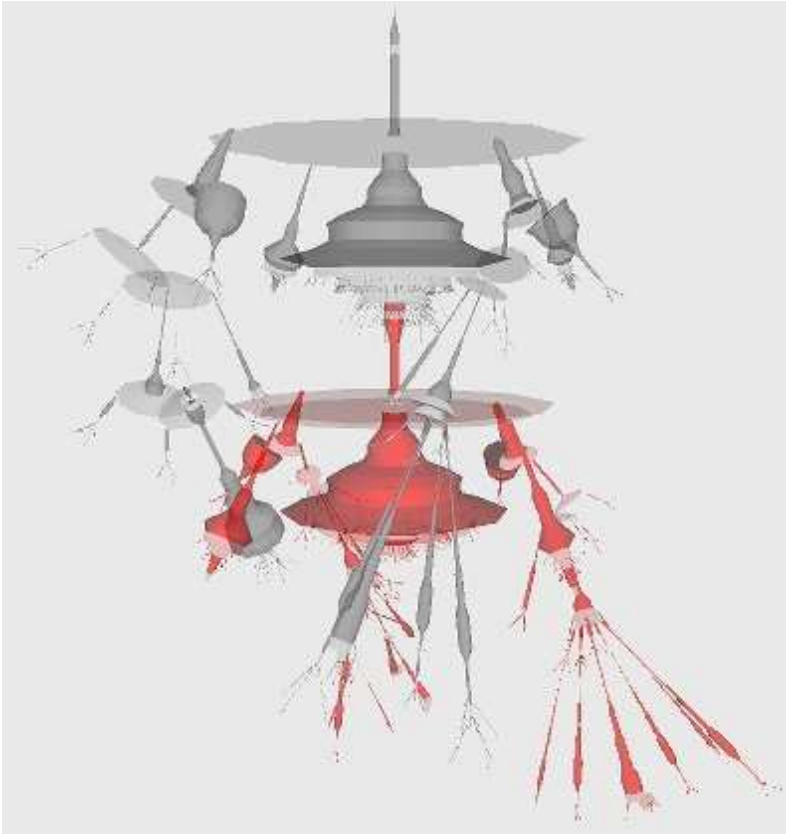
http://www.win.tue.nl/~fvham/fsm/



Solution : visualisation ?

Outil de visualisation de gros graphes, exemple de la norme IEEE1394

http://www.win.tue.nl/~fvham/fsm/

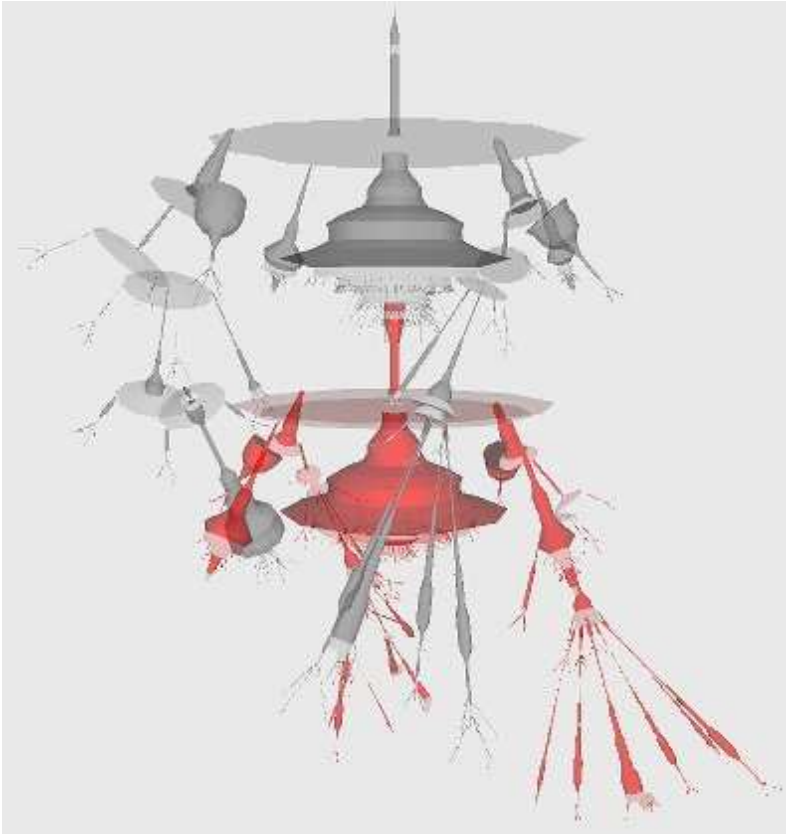


- ➜ intéressant
- ➜ sans doute pas suffisant

Solution : visualisation ?

Outil de visualisation de gros graphes, exemple de la norme IEEE1394

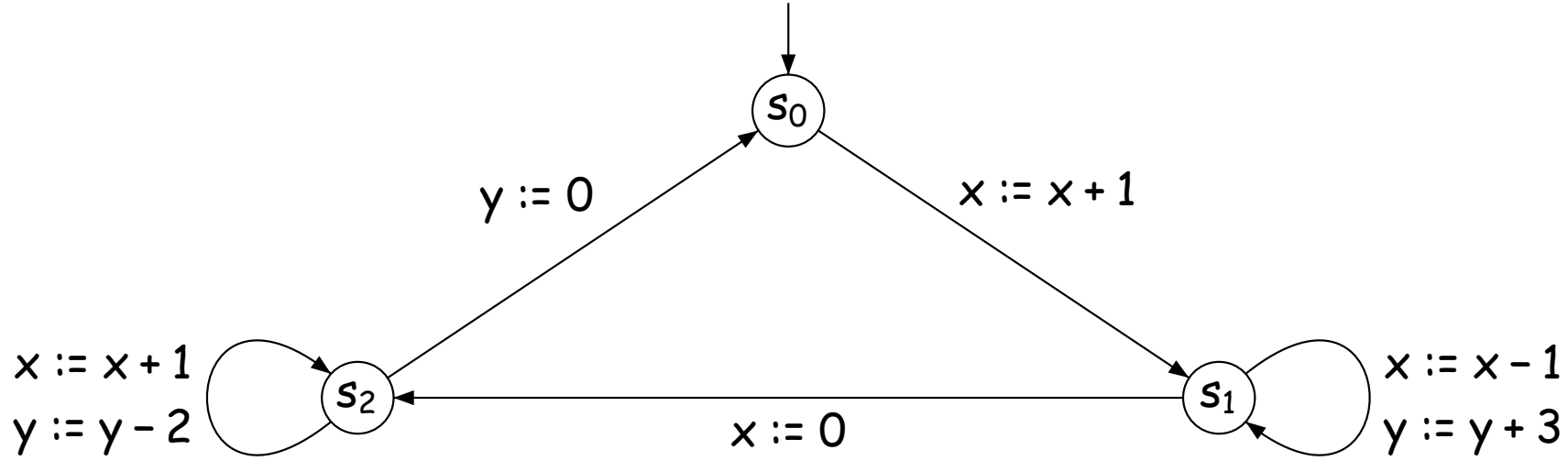
http://www.win.tue.nl/~fvham/fsm/



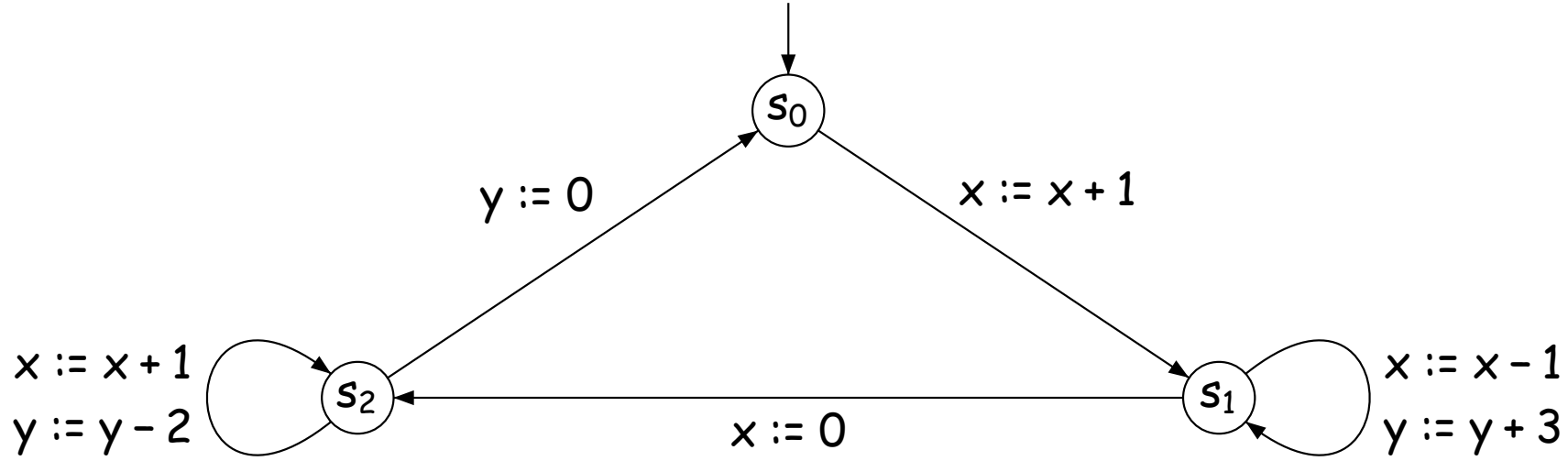
- ➜ intéressant
- ➜ sans doute pas suffisant

L'explosion combinatoire est un aspect qu'il ne faudra pas négliger.

L'infini...



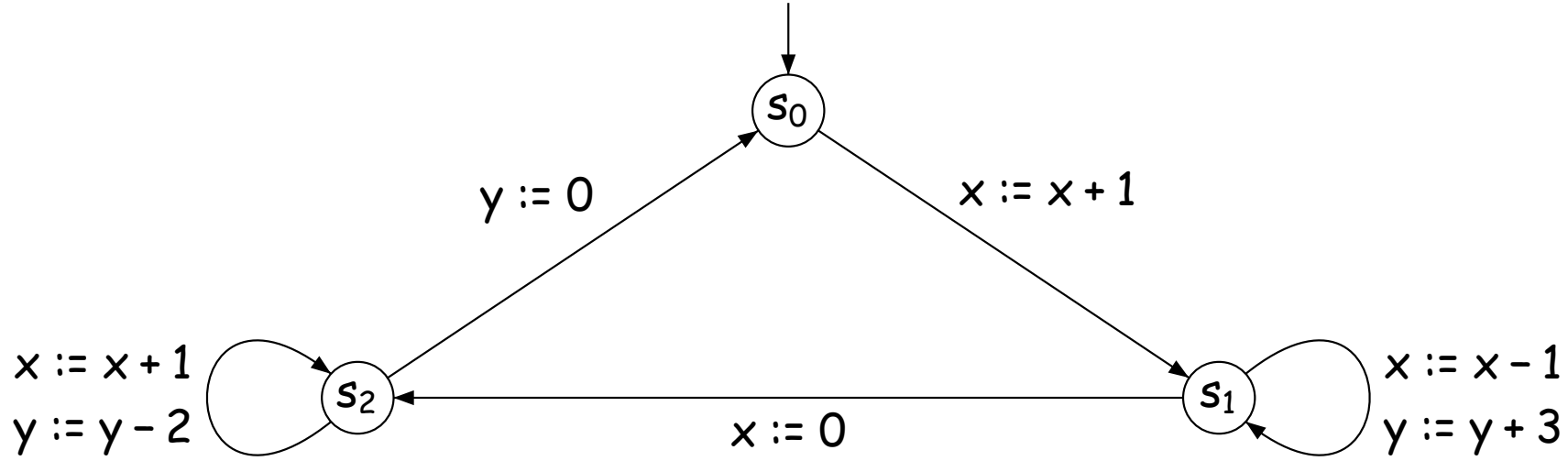
L'infini...



Une infinité de configurations est accessible à partir de s_0, par exemple :

$$(s_0,(0,0)) \to (s_1,(1,0)) \to (s_1,(0,-3)) \to (s_1,(-1,6)) \to (s_1,(-2,9))\ldots$$

L'infini...



Une infinité de configurations est accessible à partir de s_0, par exemple :

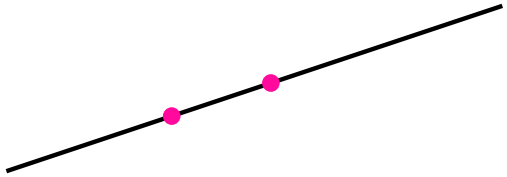
$$(s_0,(0,0)) \to (s_1,(1,0)) \to (s_1,(0,-3)) \to (s_1,(-1,6)) \to (s_1,(-2,9))\ldots$$

On peut cependant se passer de l'infini :

$$\begin{aligned} \mathrm{Post}^*(s_0,(0,0)) = \quad & s_0 \times ((0,0) + (1,0)^*) \\ \cup \ & s_1 \times (\{(1,0),(0,3)\} + \{(1,0),(0,3)\}^*) \\ \cup \ & s_2 \times (\{(0,0),(1,1),(2,2)\} + \{(3,0),(0,3)\}^*) \end{aligned}$$

Représentations symboliques

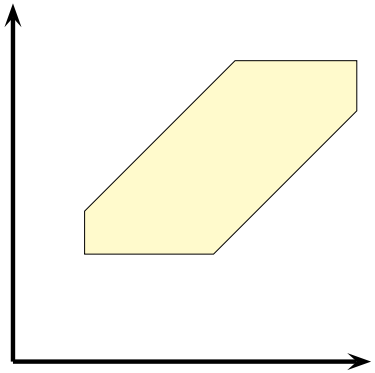
Ensemble infini / Représentation finie



- dans le plan, une droite
  représentée par deux points

- ensemble de mots aa, aaaa, aaaaaa...
  représenté par une expression rationnelle $aa(aa)^*$

- ensemble d'entiers, représentation par semi-linéaires
  $\rightarrow$ cf exemple précédent

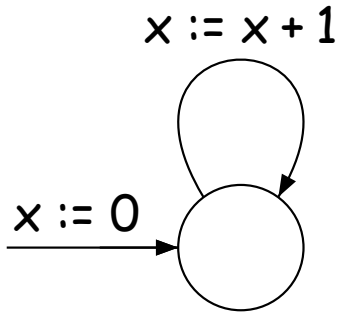


- ensemble de contraintes, polyèdres

Une technique : l'accélération

Idée : calculer en une étape plusieurs (voire une infinité d')étapes élémentaires de Post*

Exemple simple :



Au lieu de calculer

$$x = 0 \xrightarrow{x:=x+1} x = 1 \xrightarrow{x:=x+1} x = 2 \xrightarrow{x:=x+1} x = 3$$

on calcule en une seule étape

$$x = 0 \xrightarrow{x:=x+1}{}^{*} x \in \mathbb{N}$$

Une technique : l'accélération

Système producteur/consommateur

```
Variables entières : i, b, o1, o2
Condition initiale : i = 10, b=o1=o2=0
  ew  : si i>0 alors i:=i-1 et b:=b+1
  er1 : si b>0 alors o1:=o1+1 et b:=b-1
  er2 : si b>0 alors o2:=o2+1 et b:=b-1
```

$$S_0 \xrightarrow{ew}{}^* S_1 \xrightarrow{er1\ ew}{}^* S_2 \xrightarrow{er2\ ew}{}^* S_3$$

avec

- $S_1 = \{10 = \mathbf{i} + \mathbf{b}$ et $\mathbf{o1} = \mathbf{o2} = 0\}$
- $S_2 = \{10 = \mathbf{i} + \mathbf{b} + \mathbf{o1}$ et $\mathbf{o2} = 0\}$
- $S_3 = \{10 = \mathbf{i} + \mathbf{b} + \mathbf{o1} + \mathbf{o2}\}$

Donc $Post^*(S_0) = S_3$!

[Finkel,Leroux 2002]

Une technique : l'abstraction

Idée : « simplifier » le modèle M en $\alpha(M)$

[Cousot,Cousot 1977]

Une technique : l'abstraction

Idée : « simplifier » le modèle M en $a(M)$

➜ [propriété de correction]

si $a(M)$ vérifie la propriété fixée, alors M la vérifie aussi

[Cousot,Cousot 1977]

Une technique : l'abstraction

Idée : « simplifier » le modèle M en $a(M)$

➜ [propriété de correction]

si $a(M)$ vérifie la propriété fixée, alors M la vérifie aussi

➜ [propriété de complétude]

si $a(M)$ ne vérifie pas la propriété fixée, alors M ne la vérifie pas

[Cousot,Cousot 1977]

Une technique : l'abstraction

Idée : « simplifier » le modèle M en $a(M)$

➜ [propriété de correction]

si $a(M)$ vérifie la propriété fixée, alors M la vérifie aussi

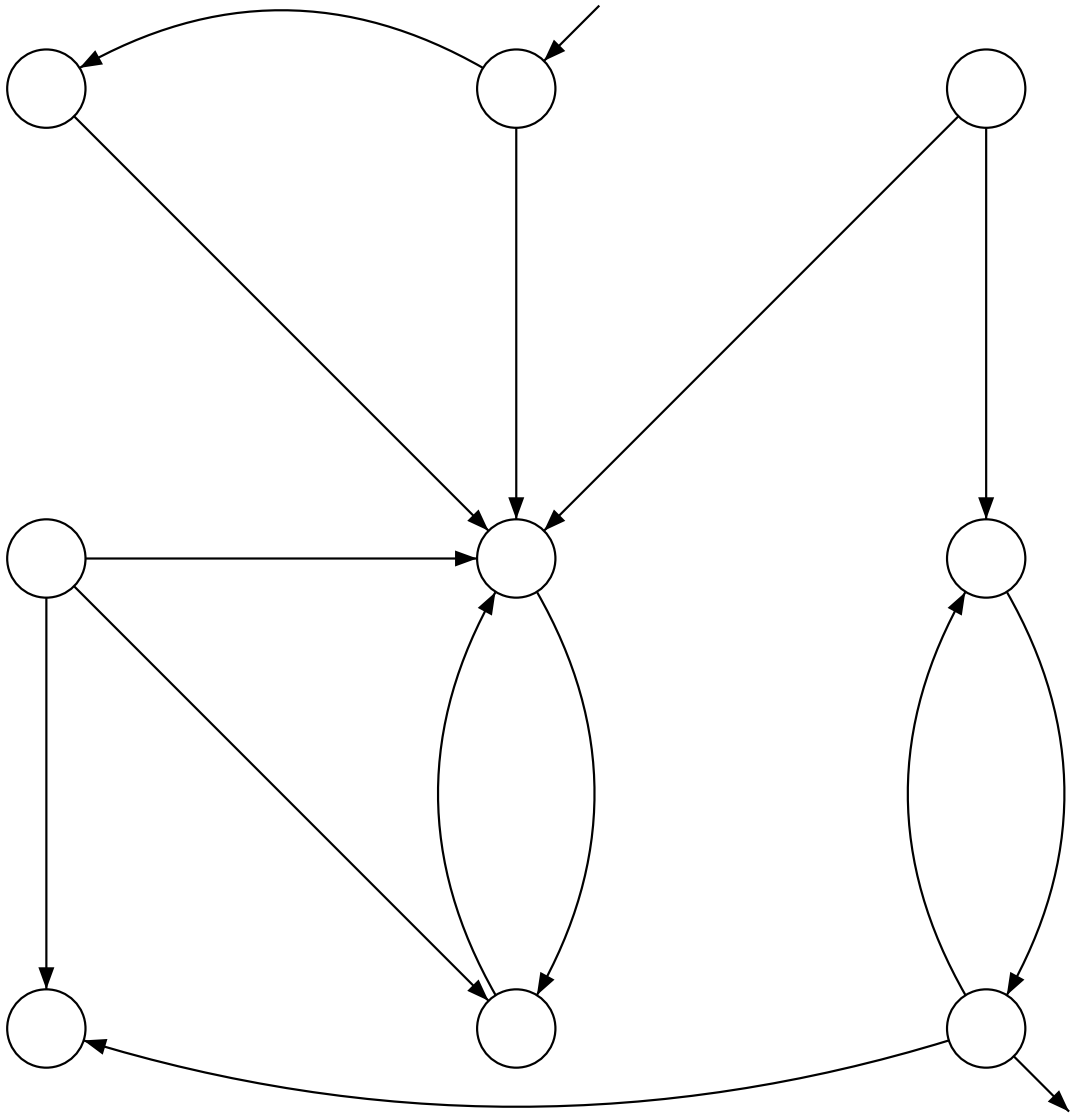
➜ [propriété de complétude]

si $a(M)$ ne vérifie pas la propriété fixée, alors M ne la vérifie pas

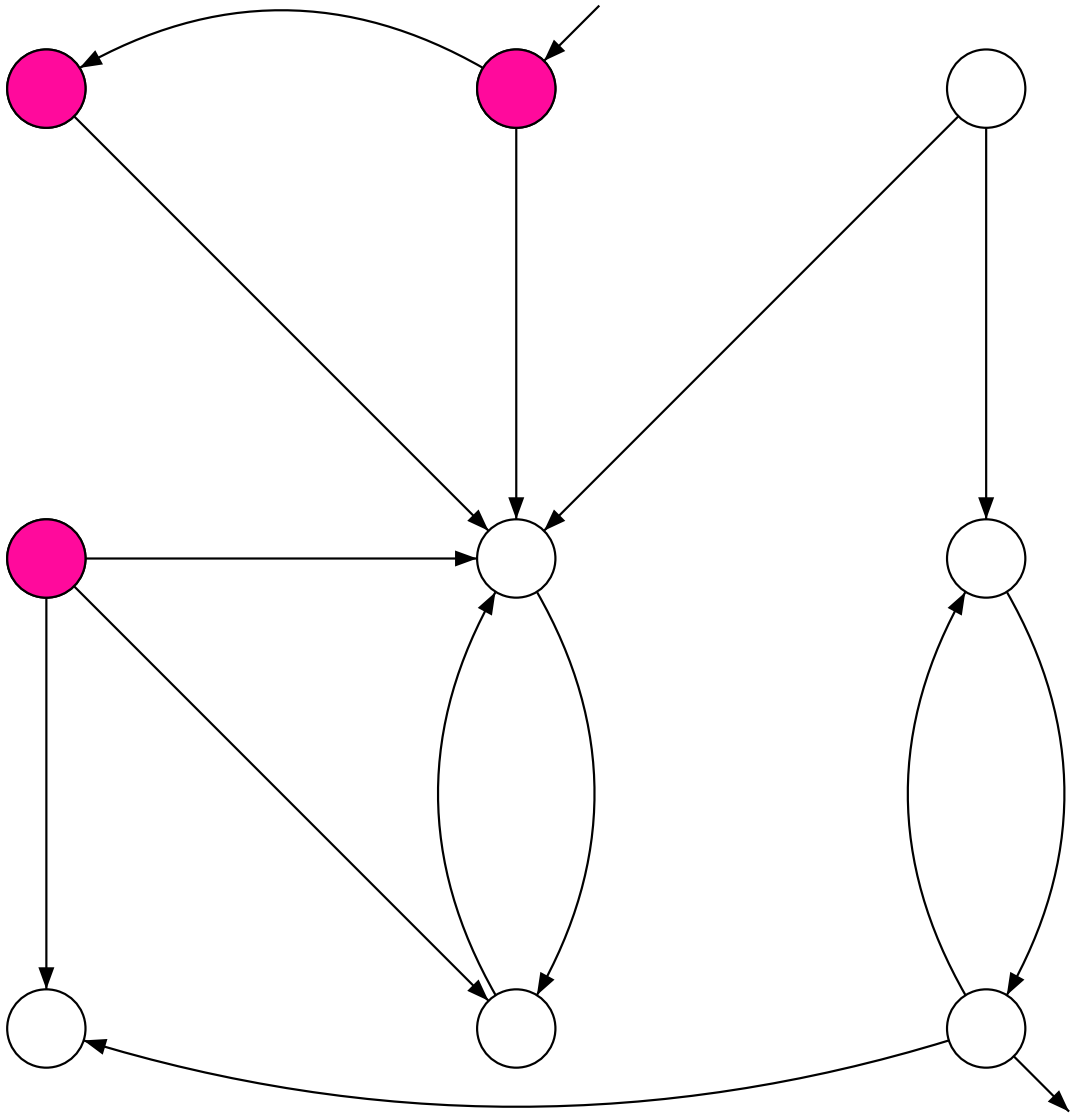
➜ bien choisir les parties du modèle que l'on abstrait

[Cousot,Cousot 1977]

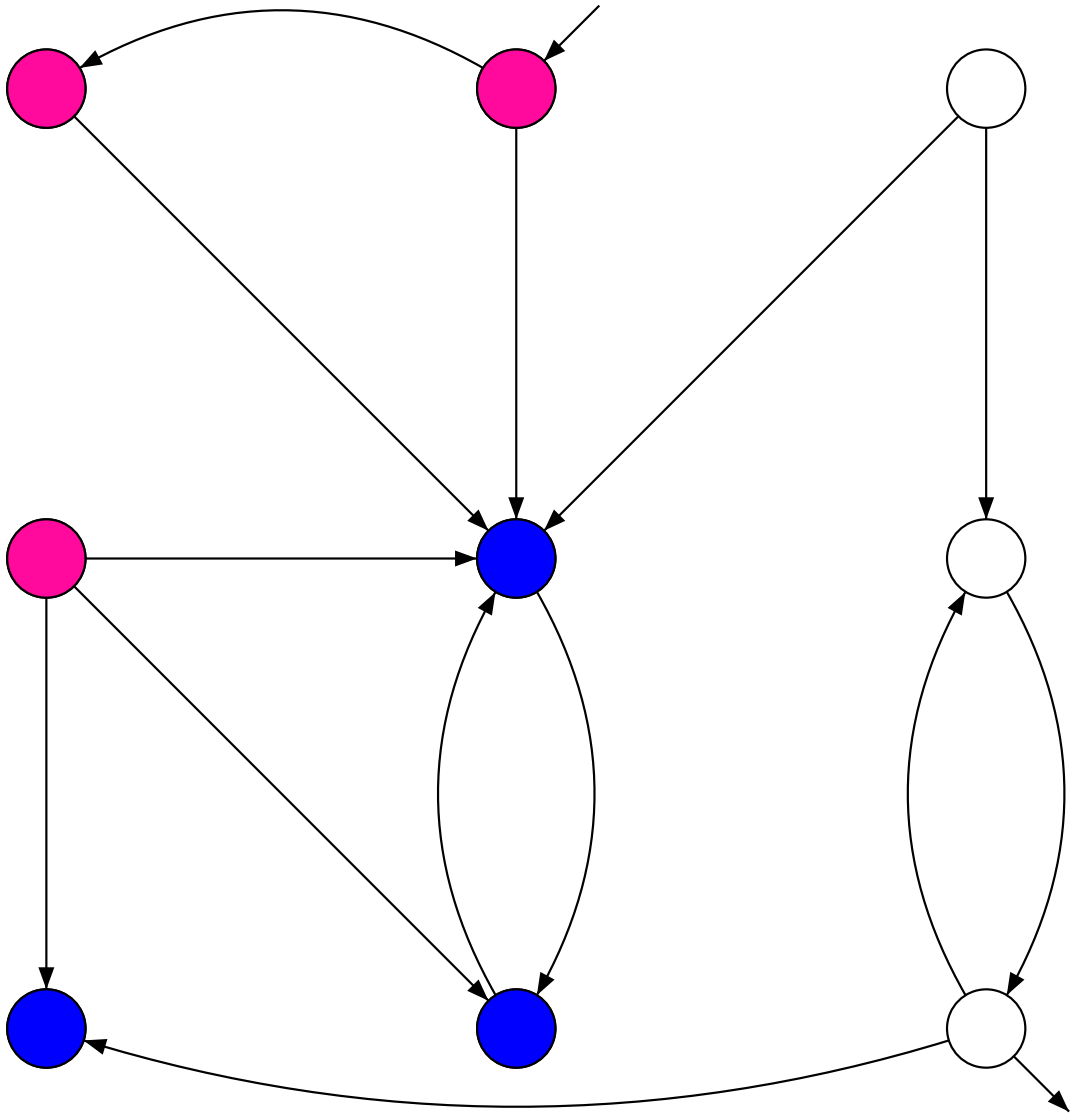
Un exemple d'abstraction correcte



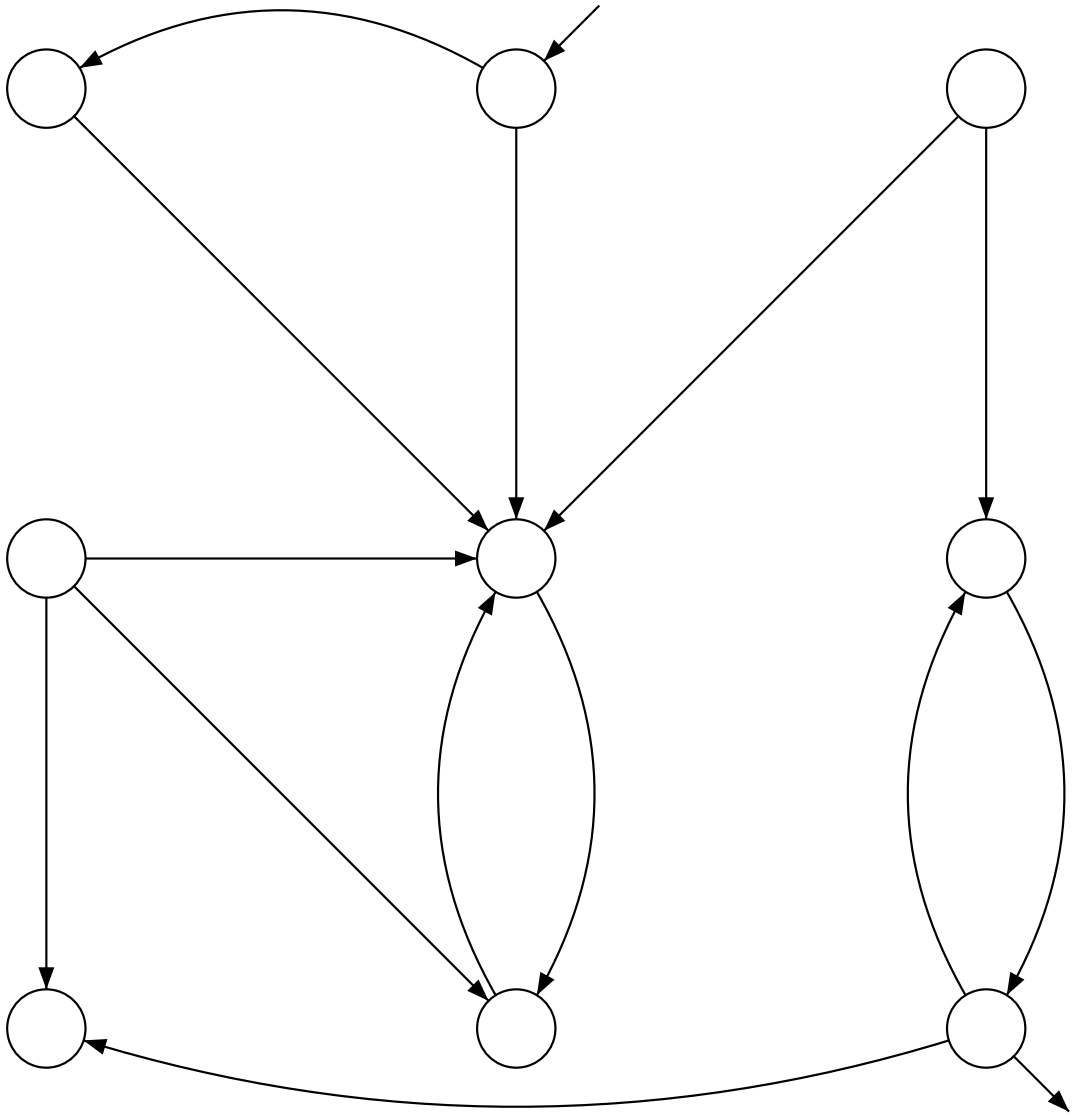
Un exemple d'abstraction correcte



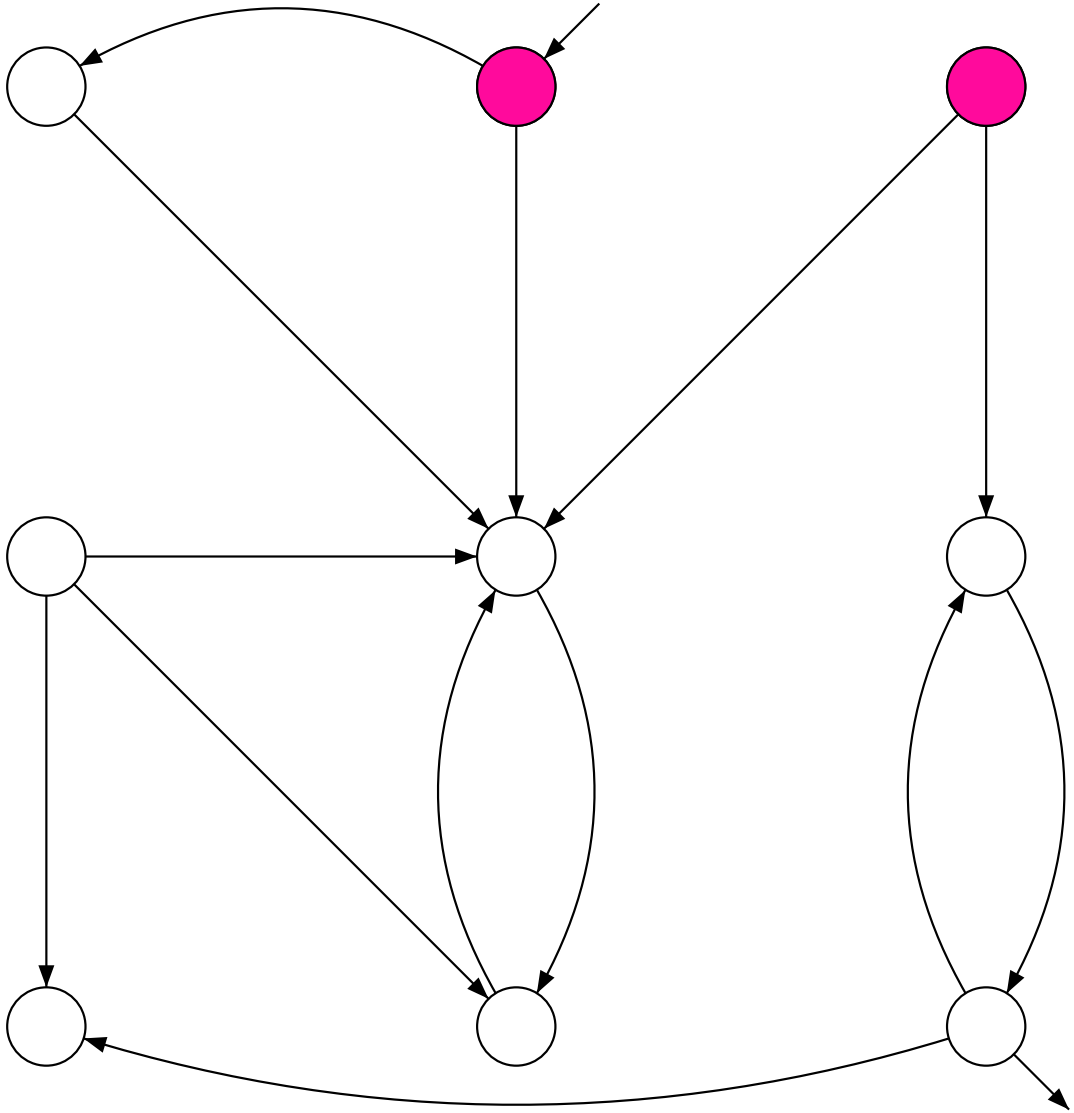
Un exemple d'abstraction correcte



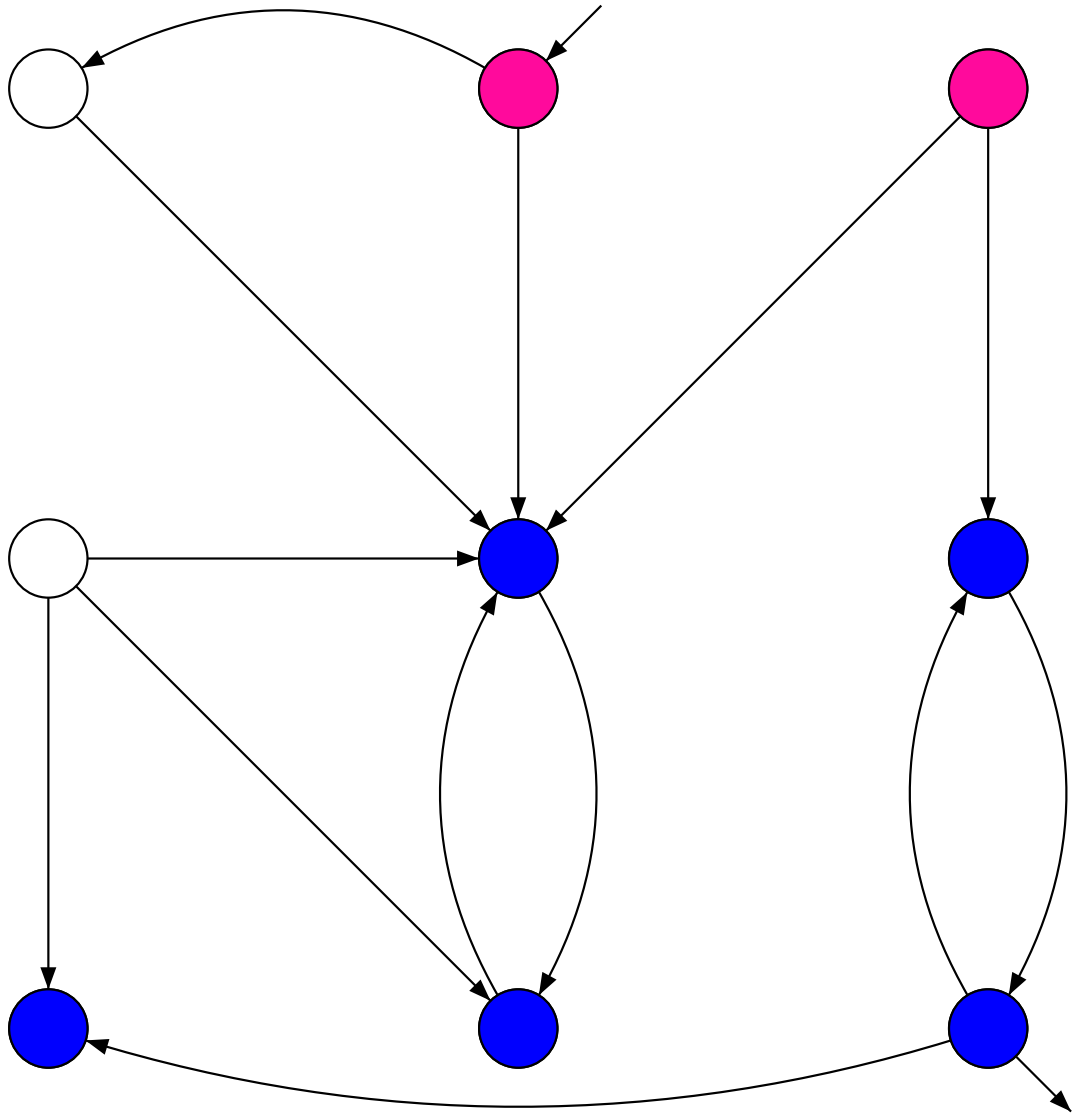
Un exemple d'abstraction non correcte



Un exemple d'abstraction non correcte



Un exemple d'abstraction non correcte



Problème !

Autres exemples en théorie des langages

- **Déterminisation :**

Autres exemples en théorie des langages

- **Déterminisation :**
  - correcte pour des propriétés telles que l'accessibilité, les langages acceptés

Autres exemples en théorie des langages

- **Déterminisation :**
  - correcte pour des propriétés telles que l'accessibilité, les langages acceptés
  - non correcte pour des propriétés telles que la longueur du plus petit cycle, pour des propriétés de blocage...

Autres exemples en théorie des langages

- **Déterminisation :**
  - correcte pour des propriétés telles que l'accessibilité, les langages acceptés
  - non correcte pour des propriétés telles que la longueur du plus petit cycle, pour des propriétés de blocage...
- **Minimisation :**

Autres exemples en théorie des langages

- **Déterminisation :**
  - correcte pour des propriétés telles que l'accessibilité, les langages acceptés
  - non correcte pour des propriétés telles que la longueur du plus petit cycle, pour des propriétés de blocage...
- **Minimisation :**
  - correcte pour des propriétés telles que l'accessibilité, etc...

Autres exemples en théorie des langages

- **Déterminisation :**
  - correcte pour des propriétés telles que l'accessibilité, les langages acceptés
  - non correcte pour des propriétés telles que la longueur du plus petit cycle, pour des propriétés de blocage...
- **Minimisation :**
  - correcte pour des propriétés telles que l'accessibilité, etc...
  - non correcte pour des propriétés de plus courts chemins, de blocage...

Exercice

Exercice : Peut-on abstraire (de manière effective) des automates communicants en remplaçant les canaux de communication par des canaux bornés tout en restant correct pour les propriétés d'accessibilité ?

Exercice

Exercice : Peut-on abstraire (de manière effective) des automates communicants en remplaçant les canaux de communication par des canaux bornés tout en restant correct pour les propriétés d'accessibilité ?

1.

automates communicants	$\xrightarrow{\text{abstraction } a}$	automates à compteurs
écrire a dans le canal c		$c := c + 1$
lire a dans le canal c		si $c > 0$ alors $c := c - 1$

Exercice

Exercice : Peut-on abstraire (de manière effective) des automates communicants en remplaçant les canaux de communication par des canaux bornés tout en restant correct pour les propriétés d'accessibilité ?

1.

automates communicants $\xrightarrow{\text{abstraction } \alpha}$ automates à compteurs

écrire a dans le canal c	$c := c + 1$
lire a dans le canal c	si $c > 0$ alors $c := c - 1$

L'abstraction α est complète pour les propriétés d'accessibilité !

Exercice

Exercice : Peut-on abstraire (de manière effective) des automates communicants en remplaçant les canaux de communication par des canaux bornés tout en restant correct pour les propriétés d'accessibilité ?

1.

automates communicants $\xrightarrow{\text{abstraction } \alpha}$ automates à compteurs

écrire a dans le canal c	c := c + 1
lire a dans le canal c	si c > 0 alors c := c − 1

L'abstraction α est complète pour les propriétés d'accessibilité !

➜ les automates communicants sont indécidables

Exercice

Exercice : Peut-on abstraire (de manière effective) des automates communicants en remplaçant les canaux de communication par des canaux bornés tout en restant correct pour les propriétés d'accessibilité ?

1.

automates communicants $\xrightarrow{\text{abstraction } \alpha}$ automates à compteurs

écrire a dans le canal c	c := c + 1
lire a dans le canal c	si c > 0 alors c := c − 1

L'abstraction α est complète pour les propriétés d'accessibilité !

➜ les automates communicants sont indécidables

2.

automates avec canaux bornés = automates finis

Exercice

Exercice : Peut-on abstraire (de manière effective) des automates communicants en remplaçant les canaux de communication par des canaux bornés tout en restant correct pour les propriétés d'accessibilité ?

1.

automates communicants $\xrightarrow{\text{abstraction } \alpha}$ automates à compteurs

écrire a dans le canal c	c := c + 1
lire a dans le canal c	si c > 0 alors c := c − 1

L'abstraction α est complète pour les propriétés d'accessibilité !

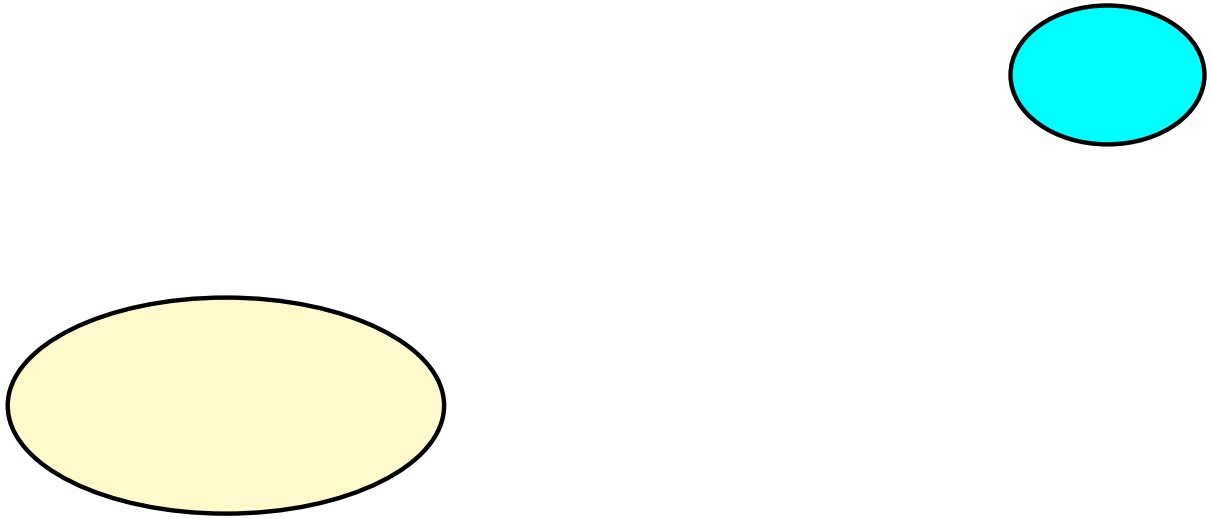
➜ les automates communicants sont indécidables

2.

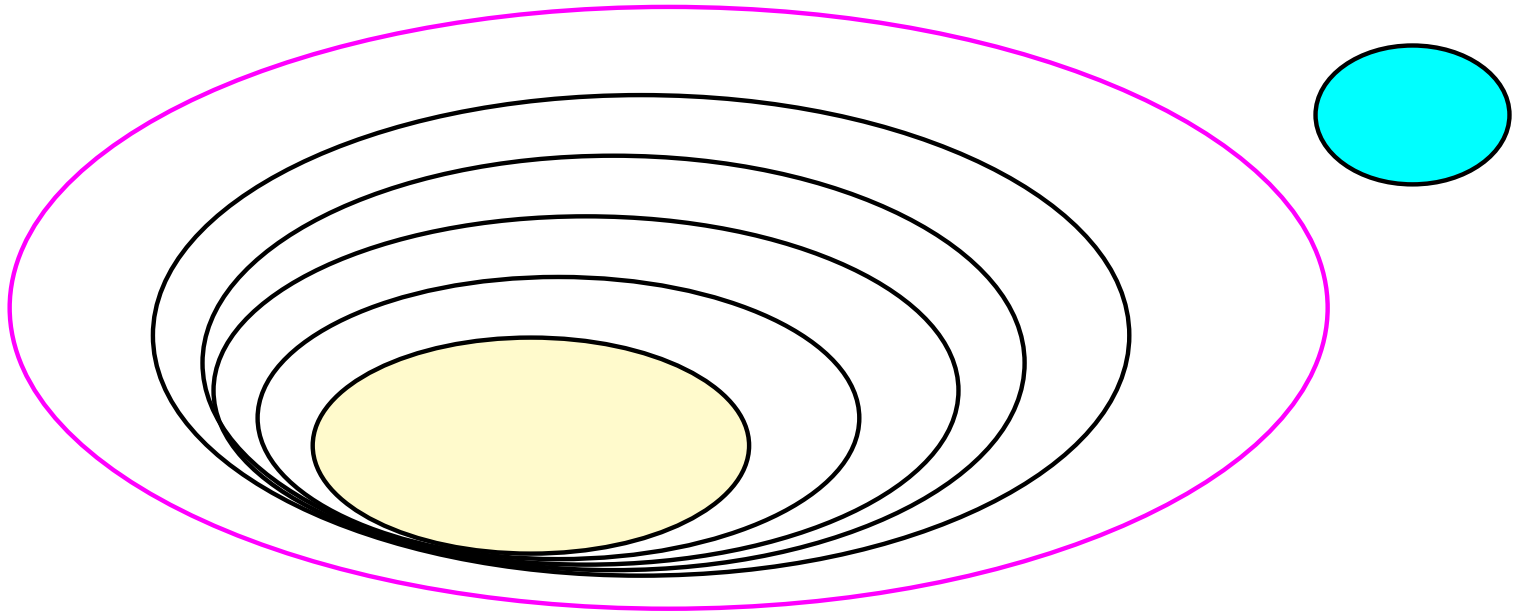
automates avec canaux bornés = automates finis

➜ il ne peut pas y avoir d'abstraction correcte (pour l'accessibilité) bornant la taille des canaux de communication

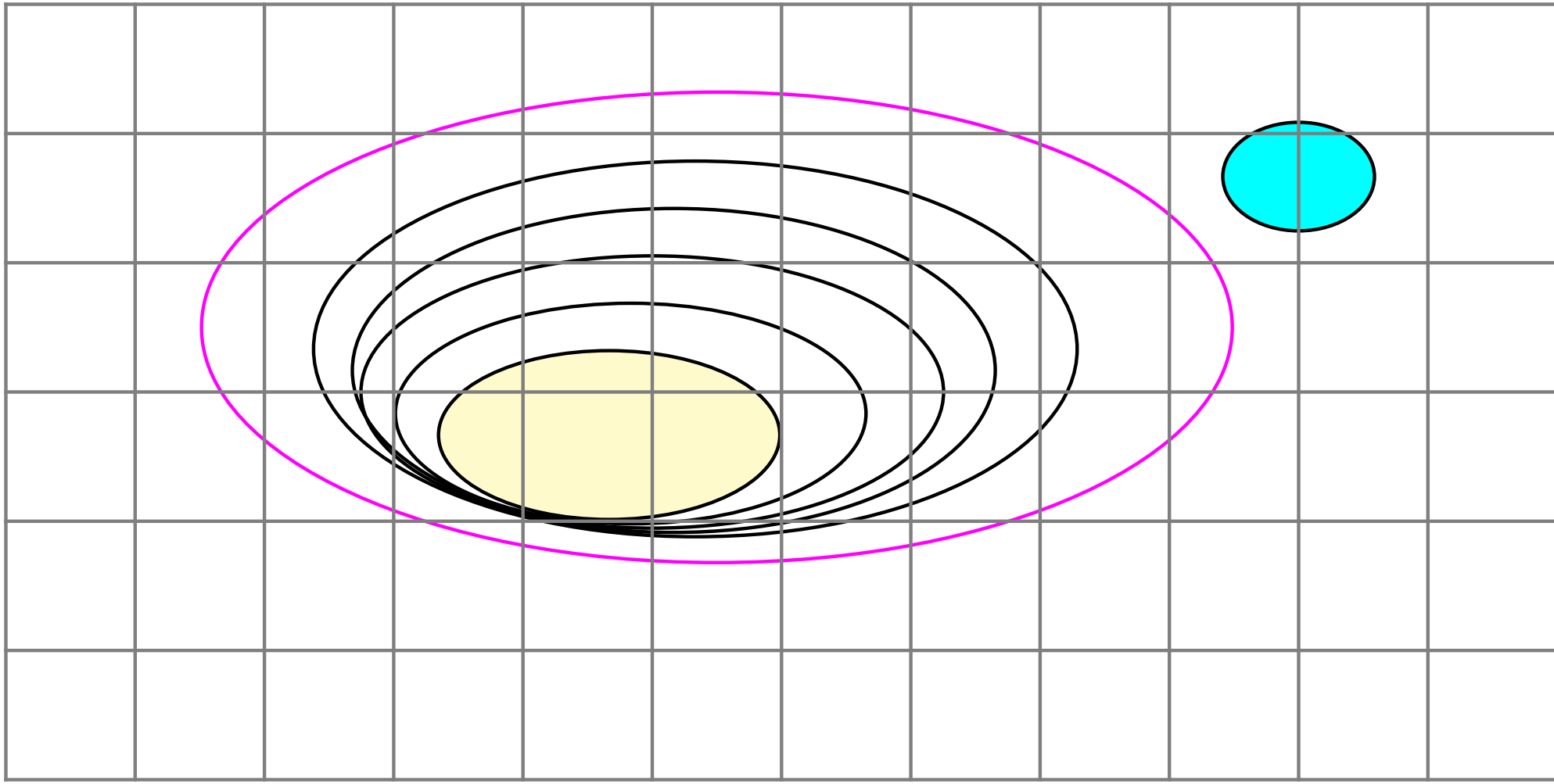
Une technique : l'abstraction par prédicats



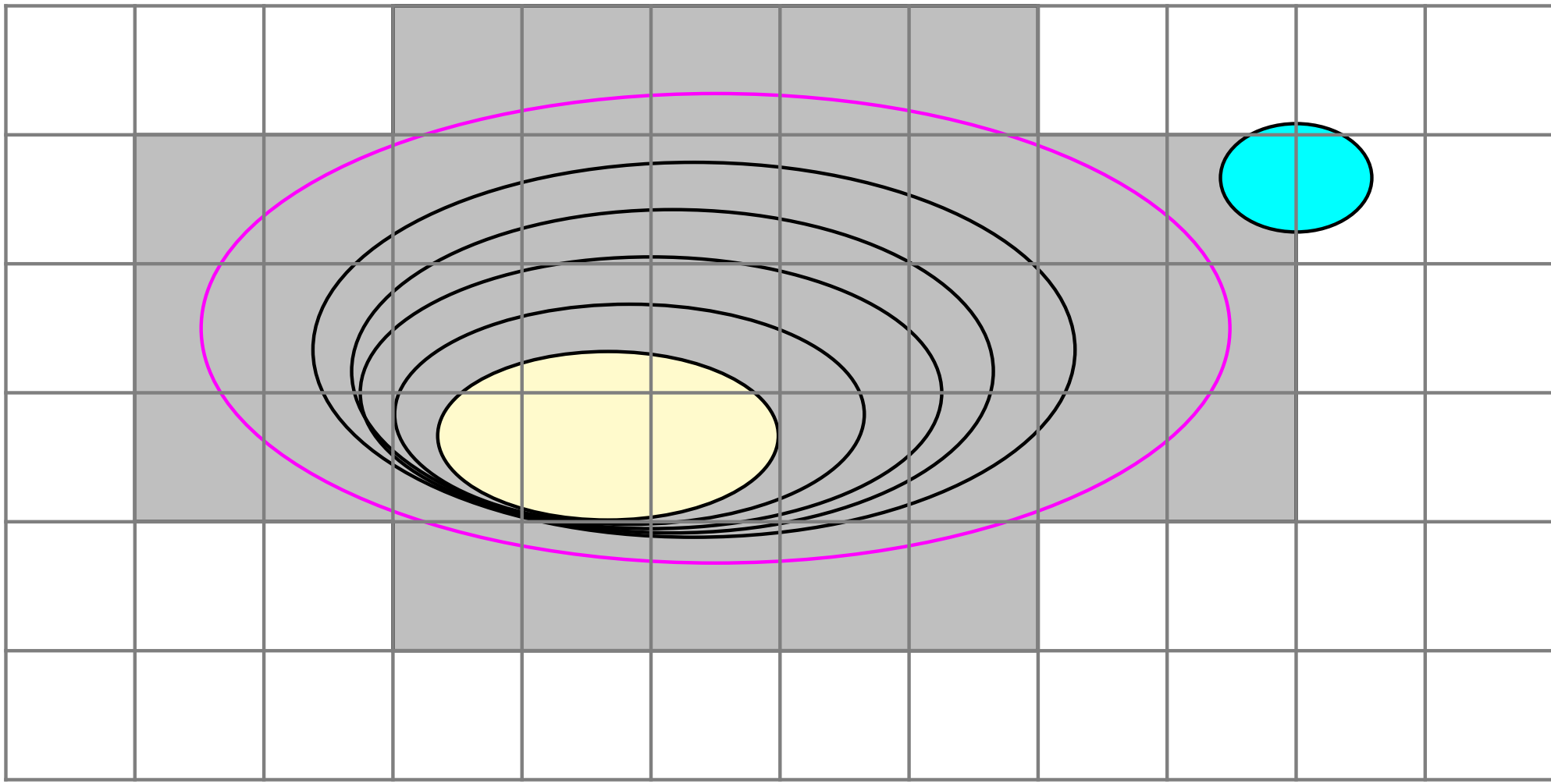
Une technique : l'abstraction par prédicats



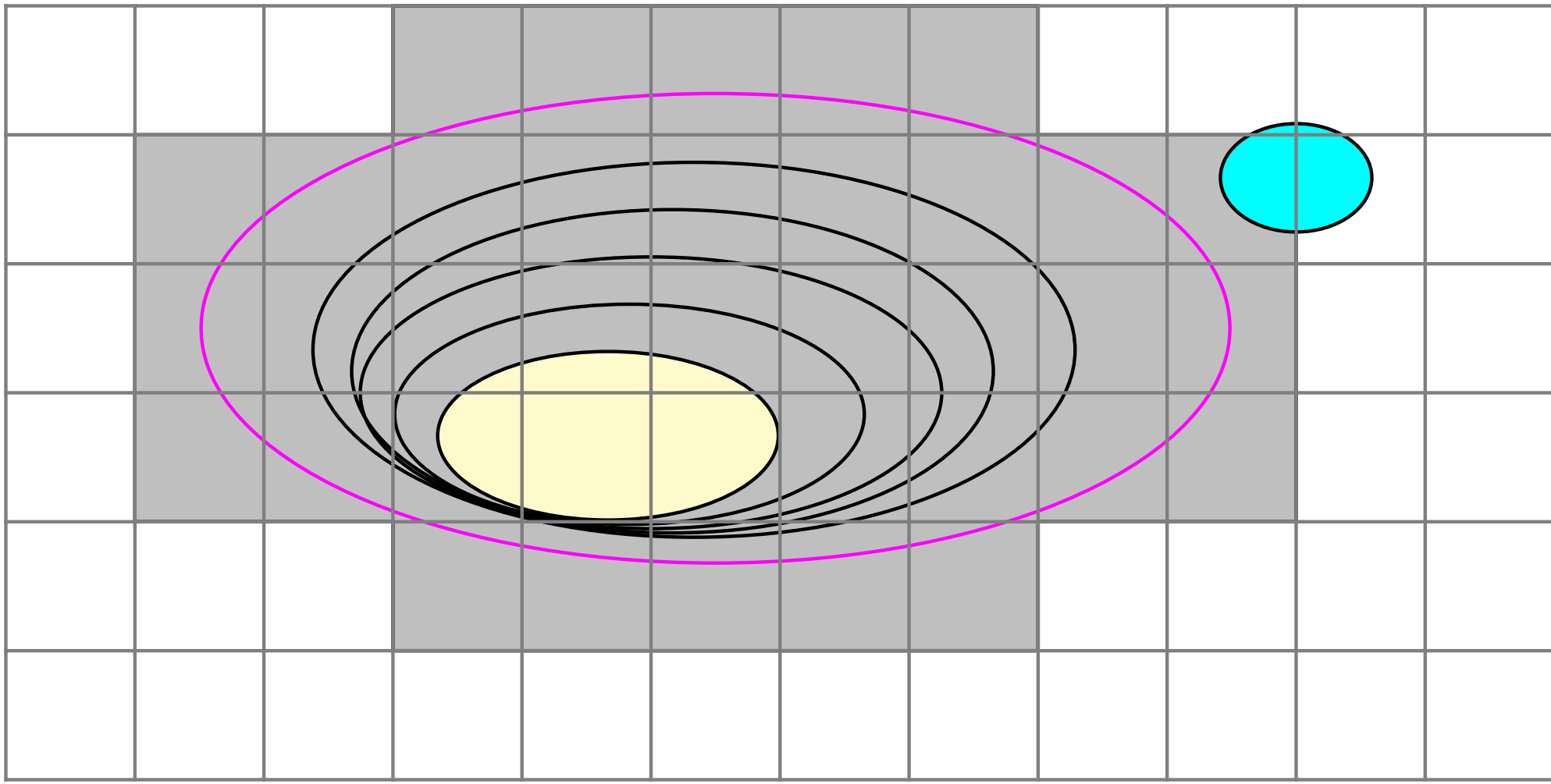
Une technique : l'abstraction par prédicats



Une technique : l'abstraction par prédicats

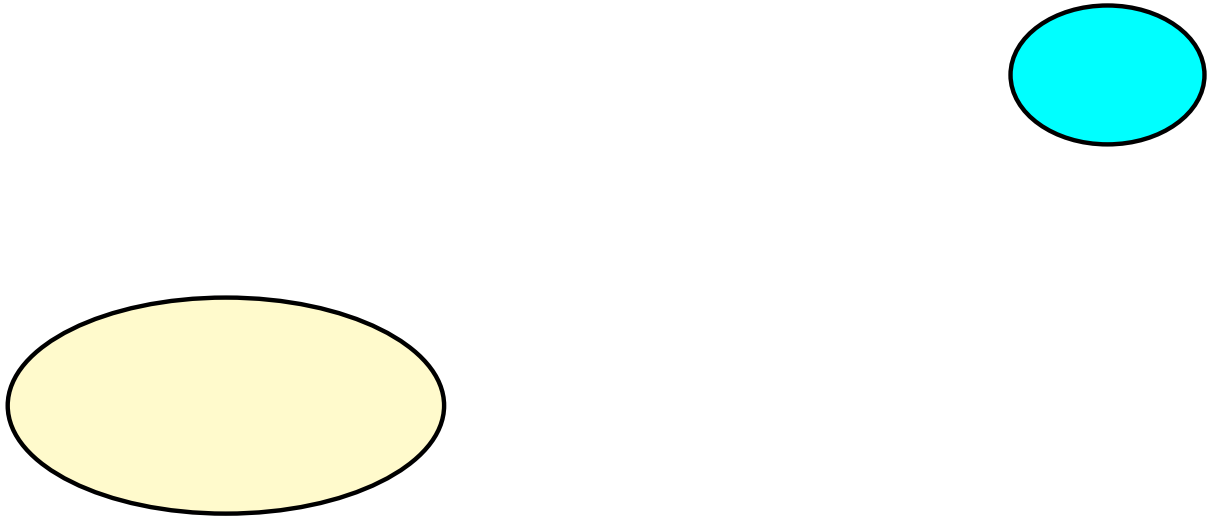


Une technique : l'abstraction par prédicats

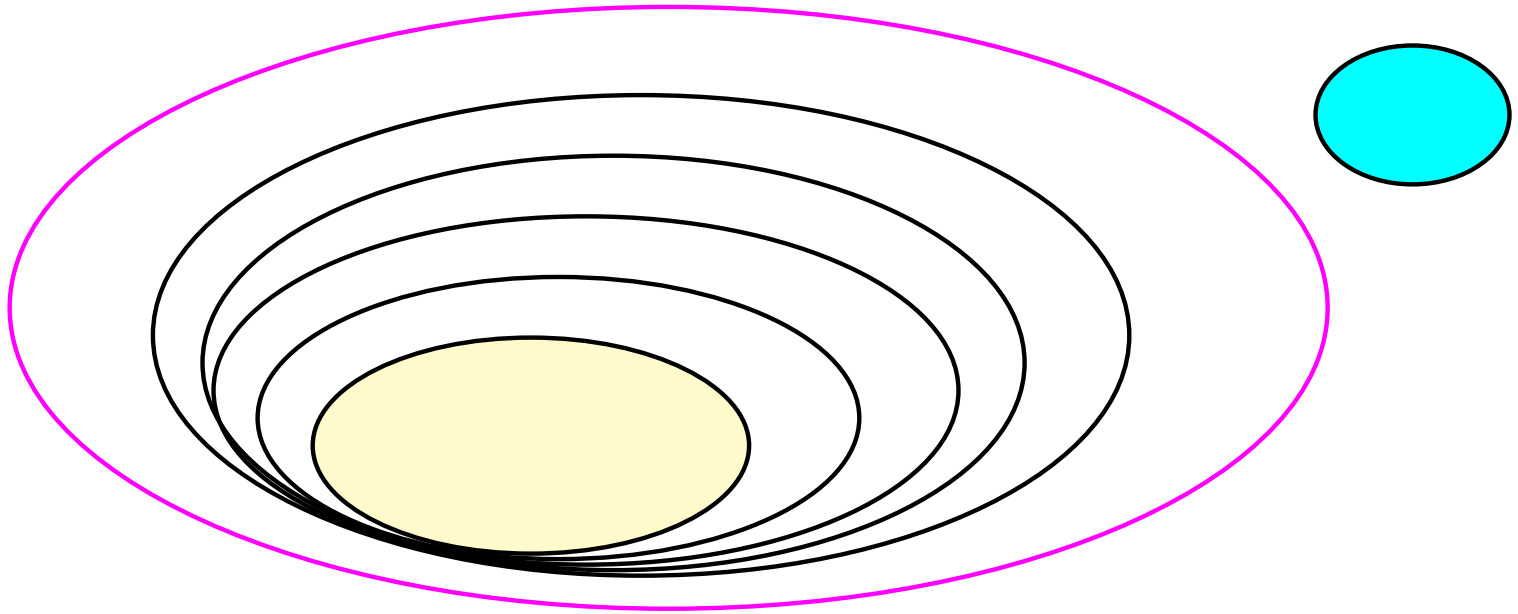


➜ abstraction trop grossière

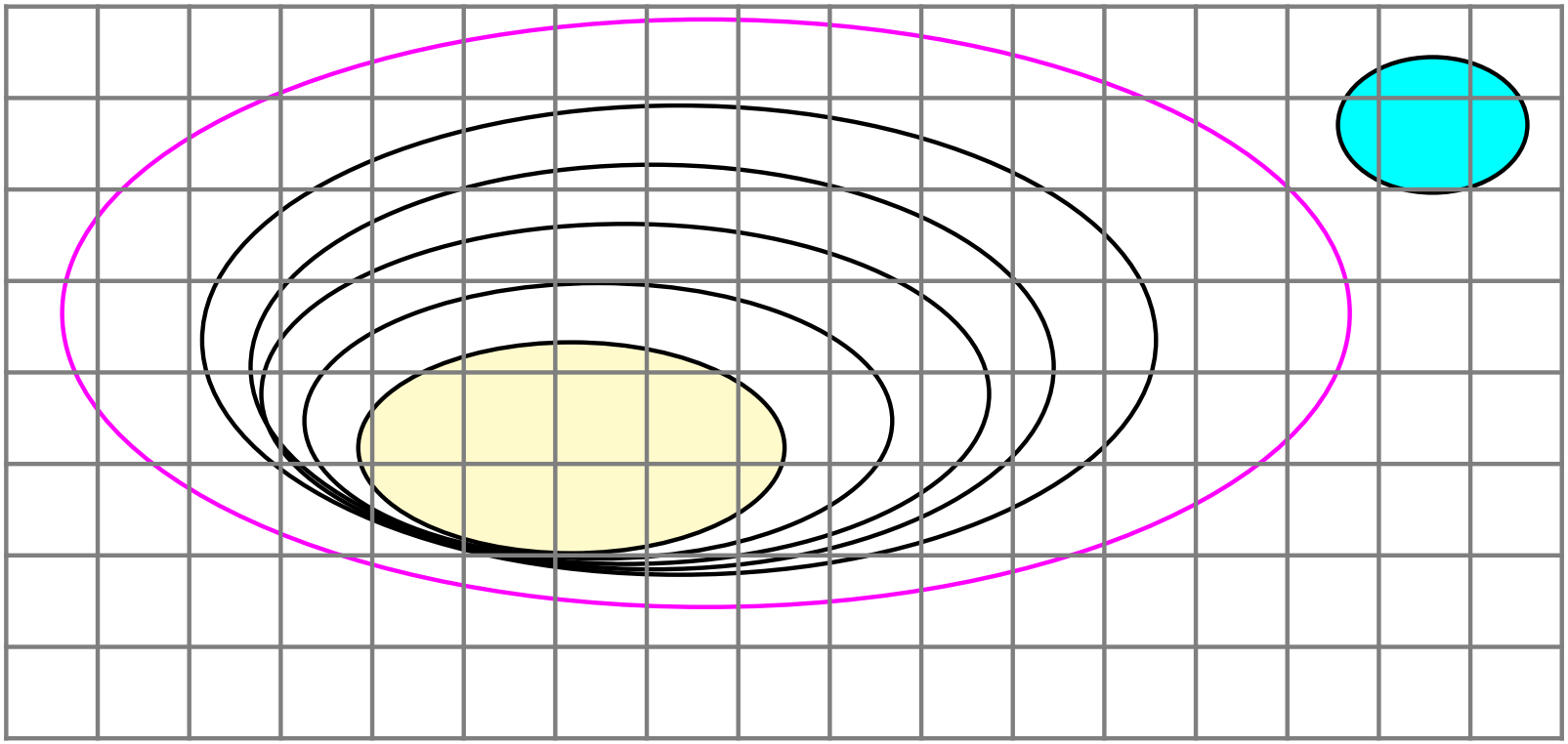
Une technique : l'abstraction par prédicats



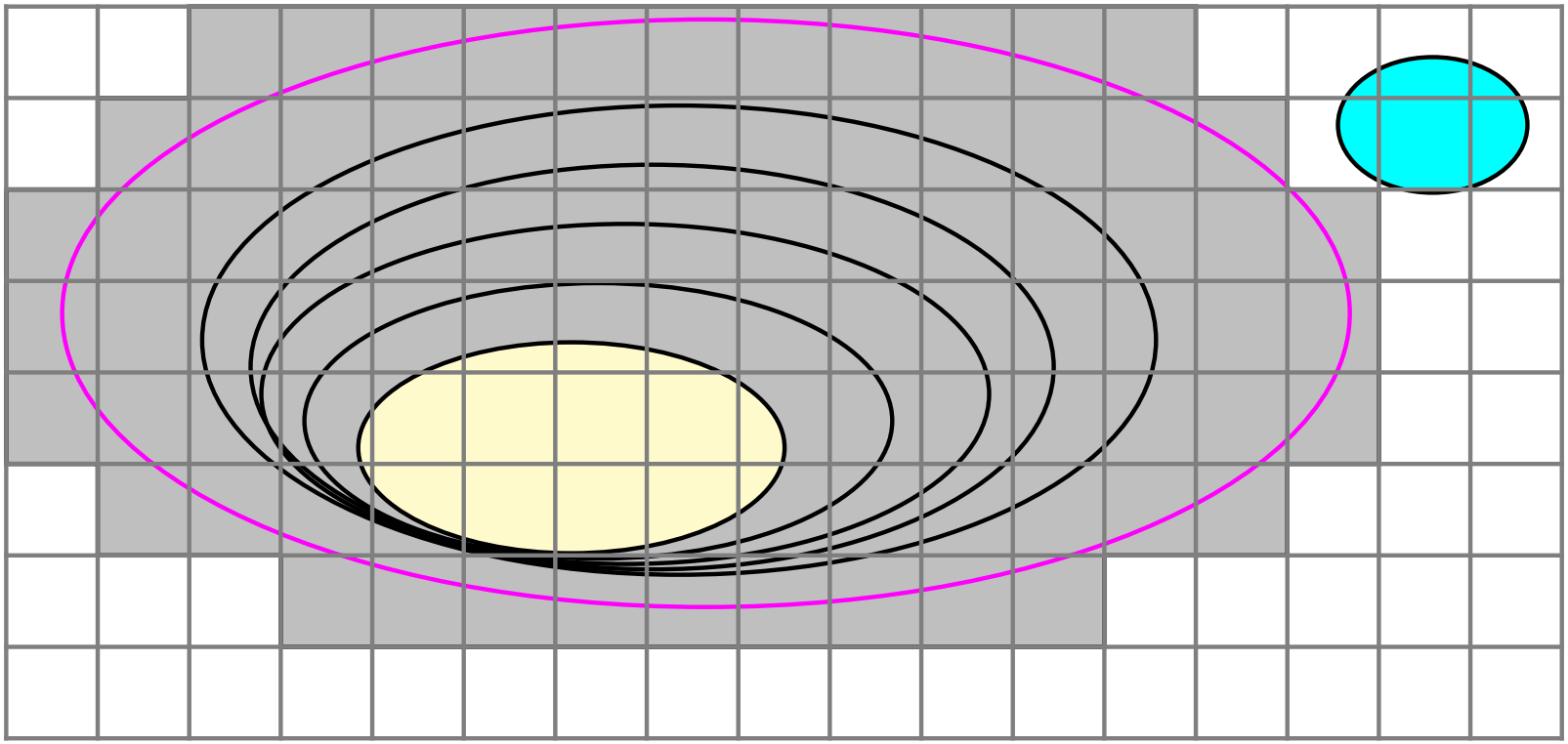
Une technique : l'abstraction par prédicats



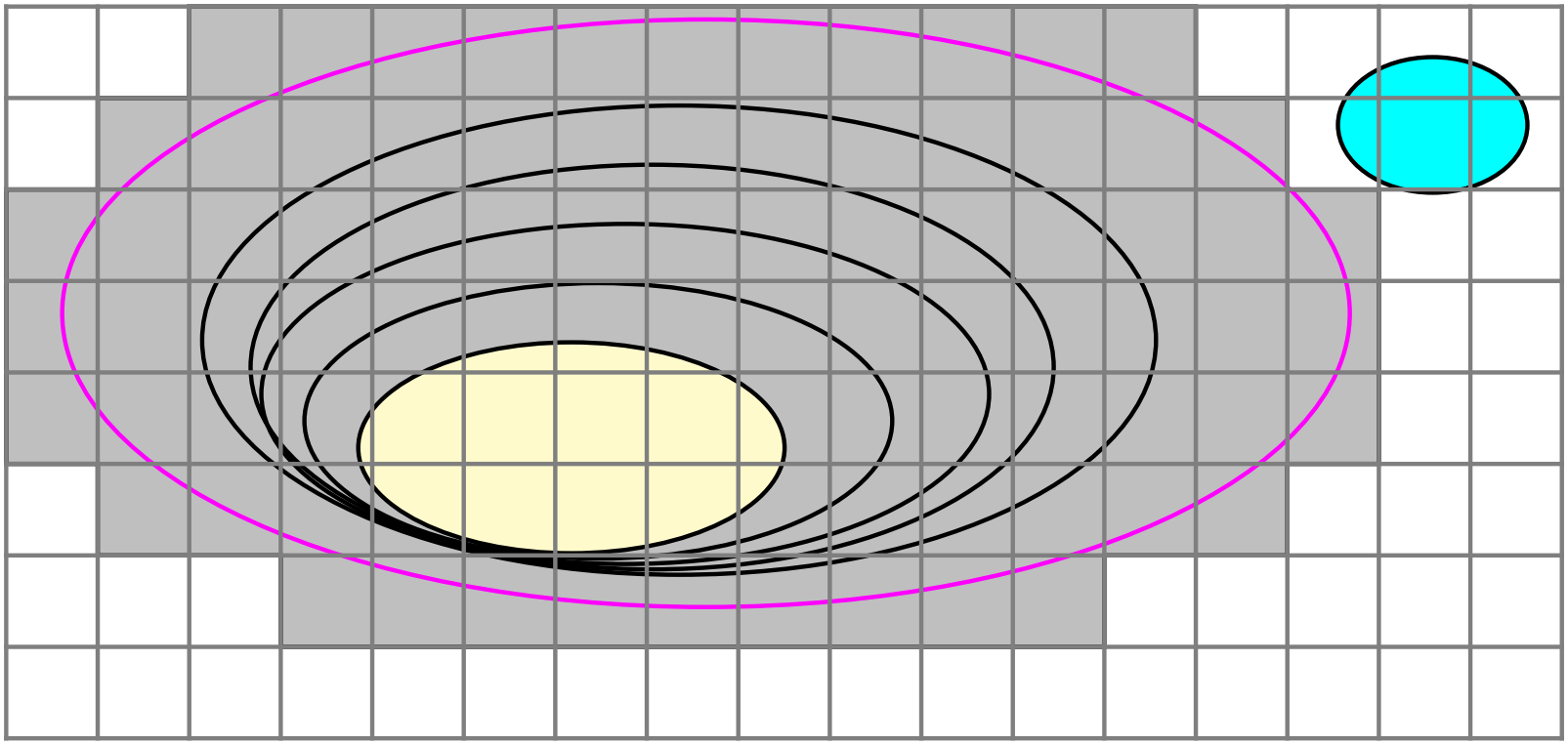
Une technique : l'abstraction par prédicats



Une technique : l'abstraction par prédicats



Une technique : l'abstraction par prédicats



➜ abstraction correcte

Un exemple, l'abstraction de programmes

```
 Example() {
1:  if (*){
7:     do {
            got_lock = 0;
8:          if (*){
9:             lock();
               got_lock++;
            }
10:         if (got_lock){
11:            unlock();
            }
12:    } while (*)
    }
2:  do {
         lock();
         old = new;
3:       if (*){
4:          unlock();
            new++;
         }
5:  } while (new != old);
6:  unlock();
    return;
}
```

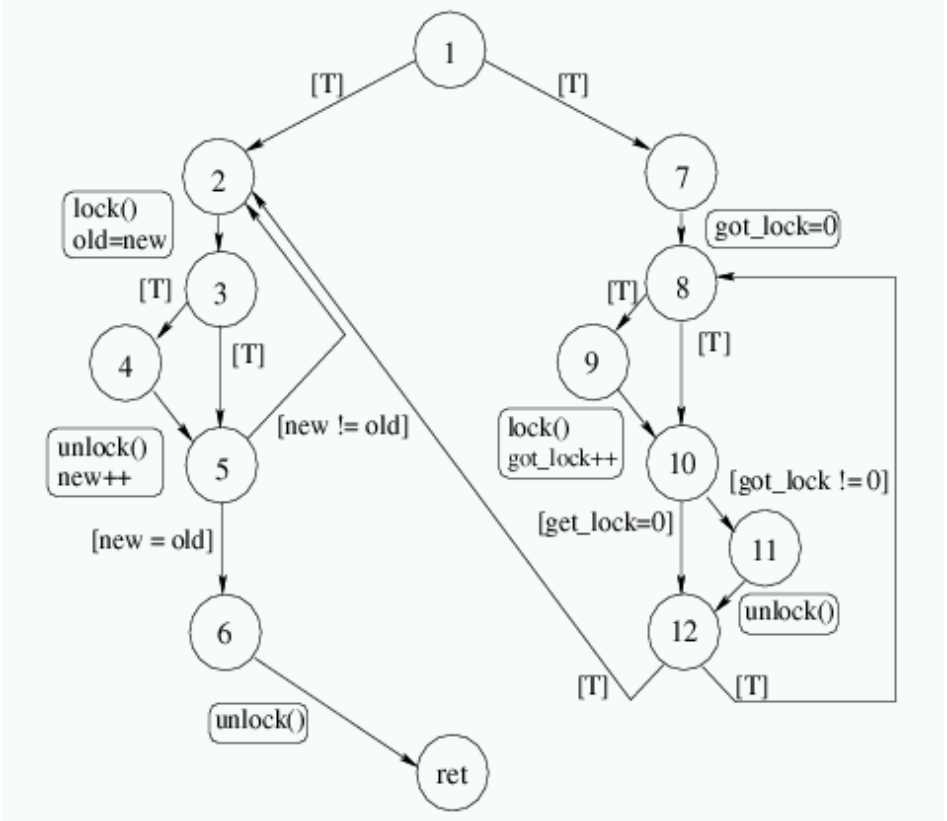
```
lock(){
   if (LOCK == 0){
      LOCK = 1;
   } else {
      ERROR
   }
}

unlock(){
   if (LOCK == 1){
      LOCK = 0;
   } else {
      ERROR
   }
}
```

[BLAST – Henzinger, Jhala, Majumdar, Sutre 2002]

Un exemple, l'abstraction de programmes

```
 Example() {
1:  if (*){
7:     do {
           got_lock = 0;
8:         if (*){
9:            lock();
              got_lock++;
           }
10:        if (got_lock){
11:           unlock();
           }
12:    } while (*)
    }
2:  do {
        lock();
        old = new;
3:      if (*){
4:         unlock();
           new++;
        }
5:  } while (new != old);
6:  unlock();
    return;
}
```

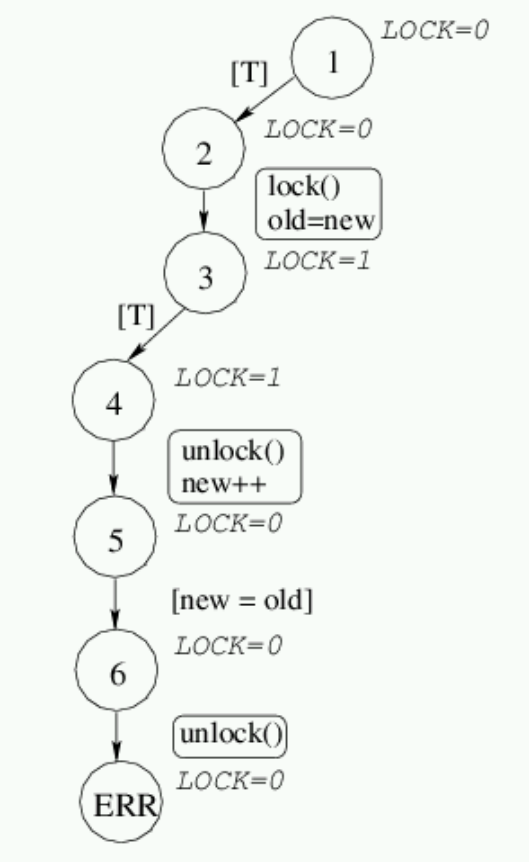
```
lock(){
   if (LOCK == 0){
      LOCK = 1;
   } else {
      ERROR
   }
}

unlock(){
   if (LOCK == 1){
      LOCK = 0;
   } else {
      ERROR
   }
}
```



[BLAST – Henzinger, Jhala, Majumdar, Sutre 2002]

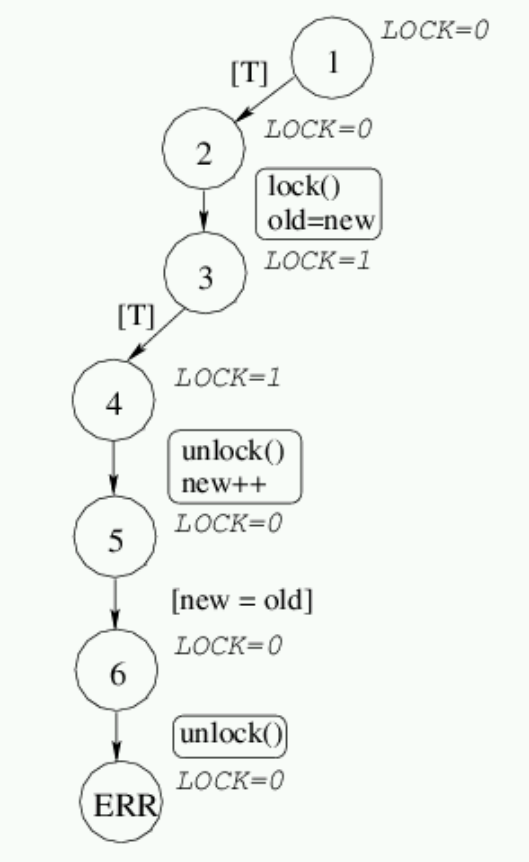
Un exemple, l'abstraction de programmes

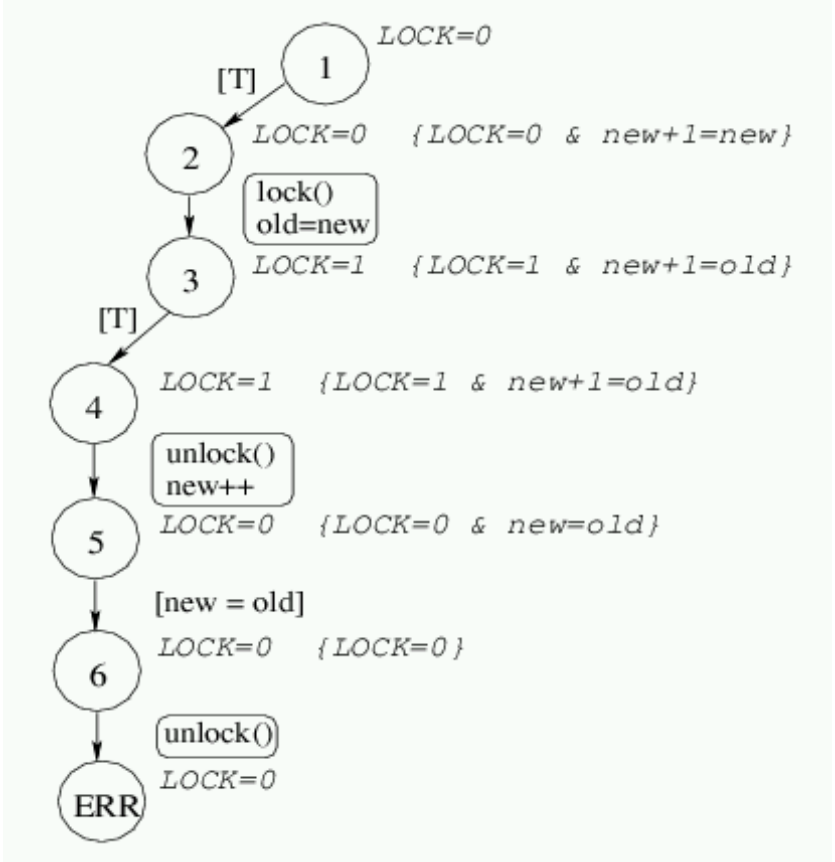
Prédicat de départ : "LOCK = 0" ou "LOCK = 1"



Un exemple, l'abstraction de programmes

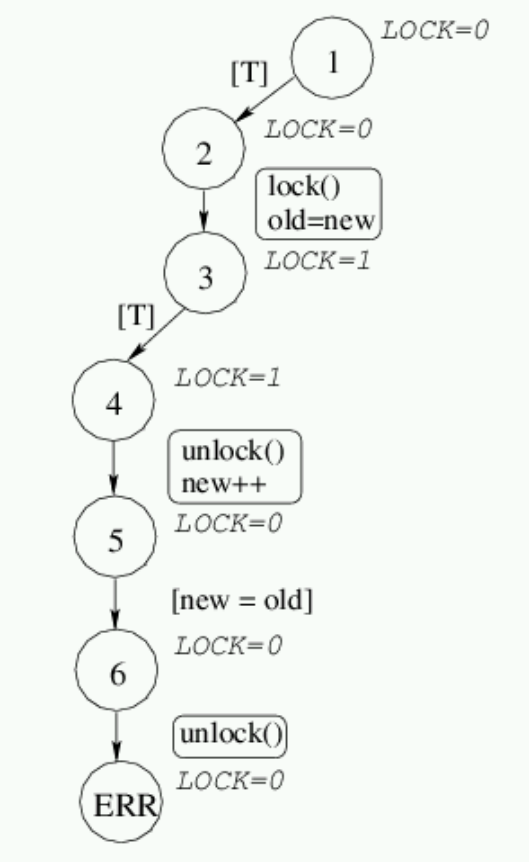
Prédicat de départ : "LOCK = 0" ou "LOCK = 1"

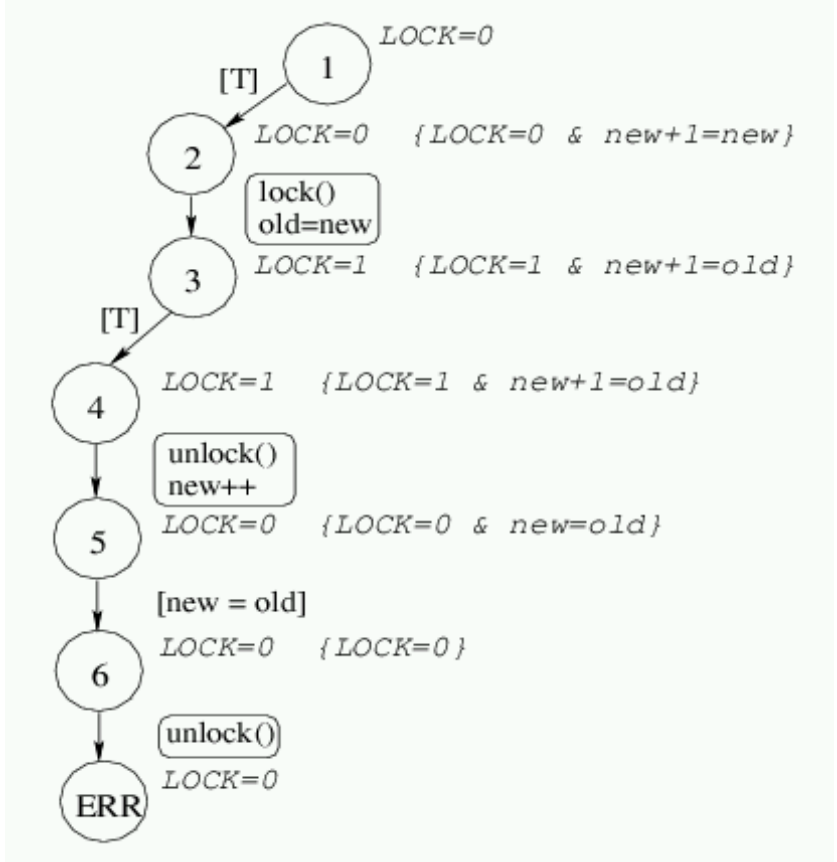




Un exemple, l'abstraction de programmes

Prédicat de départ : "LOCK = 0" ou "LOCK = 1"





On peut continuer en raffinant à l'aide du prédicat "new = old" ou "new != old"

➜ pas d'erreur

Un outil : les simulations

Ce que l'on souhaite : pouvoir comparer des systèmes entre eux, en particulier système concret / système abstrait

Un outil : les simulations

Ce que l'on souhaite : pouvoir comparer des systèmes entre eux, en particulier système concret / système abstrait

Langages : ok pour les langages rationnels, mais quid pour les langages avec des compteurs, des horloges, etc... ?

Un outil : les simulations

Ce que l'on souhaite : pouvoir comparer des systèmes entre eux, en particulier système concret / système abstrait

Langages : ok pour les langages rationnels, mais quid pour les langages avec des compteurs, des horloges, etc... ?

➜ bof

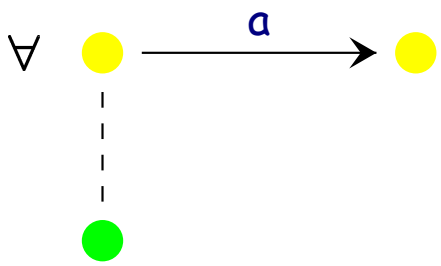
Un outil : les simulations

Ce que l'on souhaite : pouvoir comparer des systèmes entre eux, en particulier système concret / système abstrait

Langages : ok pour les langages rationnels, mais quid pour les langages avec des compteurs, des horloges, etc... ?

➜ bof

Comparaison plus structurelle (équivalence observationnelle) : une relation $\prec$ = {(●, ●)} est une simulation si



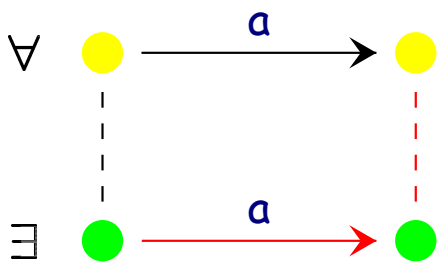
Un outil : les simulations

Ce que l'on souhaite : pouvoir comparer des systèmes entre eux, en particulier système concret / système abstrait

Langages : ok pour les langages rationnels, mais quid pour les langages avec des compteurs, des horloges, etc... ?

➜ bof

Comparaison plus structurelle (équivalence observationnelle) : une relation $\prec$ = {(●, ●)} est une simulation si



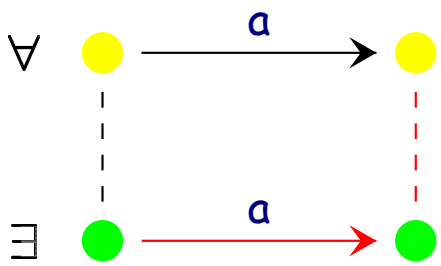
Un outil : les simulations

Ce que l'on souhaite : pouvoir comparer des systèmes entre eux, en particulier système concret / système abstrait

Langages : ok pour les langages rationnels, mais quid pour les langages avec des compteurs, des horloges, etc... ?

➜ bof

Comparaison plus structurelle (équivalence observationnelle) : une relation $\prec$ = {(●, ●)} est une simulation si



On dit alors que ● simule ●.

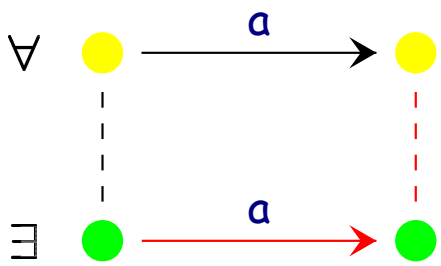
Un outil : les simulations

Ce que l'on souhaite : pouvoir comparer des systèmes entre eux, en particulier système concret / système abstrait

Langages : ok pour les langages rationnels, mais quid pour les langages avec des compteurs, des horloges, etc... ?

➜ bof

Comparaison plus structurelle (équivalence observationnelle) : une relation $\prec$ = {(●, ●)} est une simulation si



On dit alors que ● simule ●.

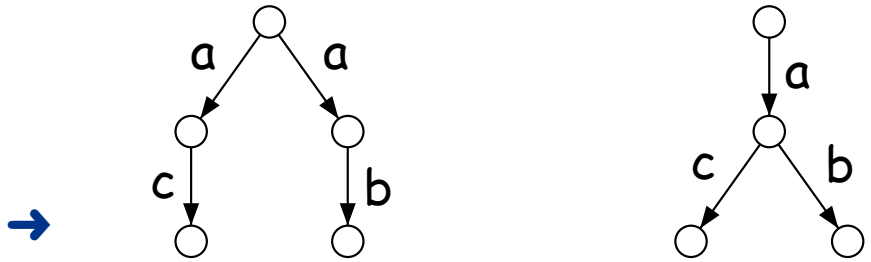
Une relation $\equiv$ est une bisimulation si $\equiv$ et $\equiv^{-1}$ sont des simulations.

Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?

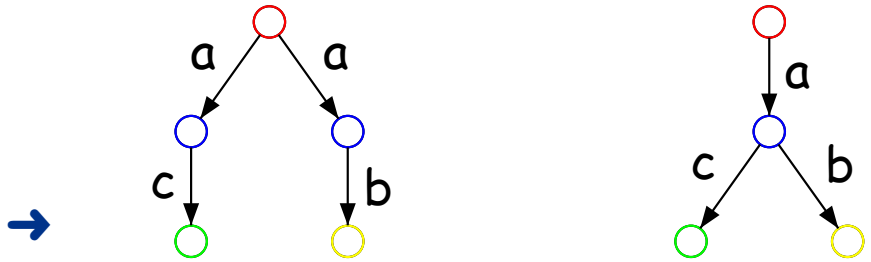
Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?



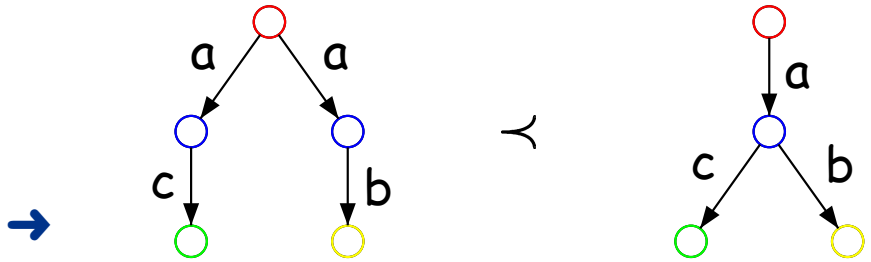
Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?



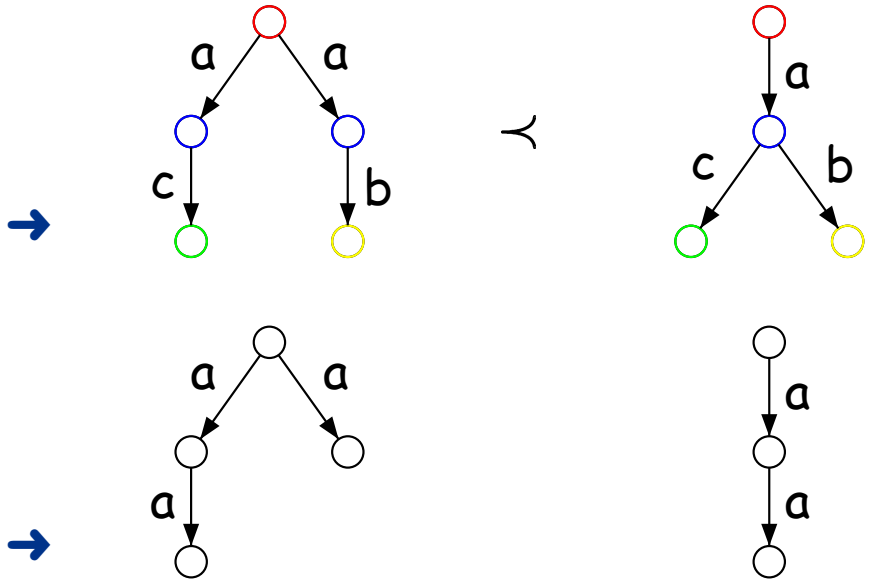
Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?



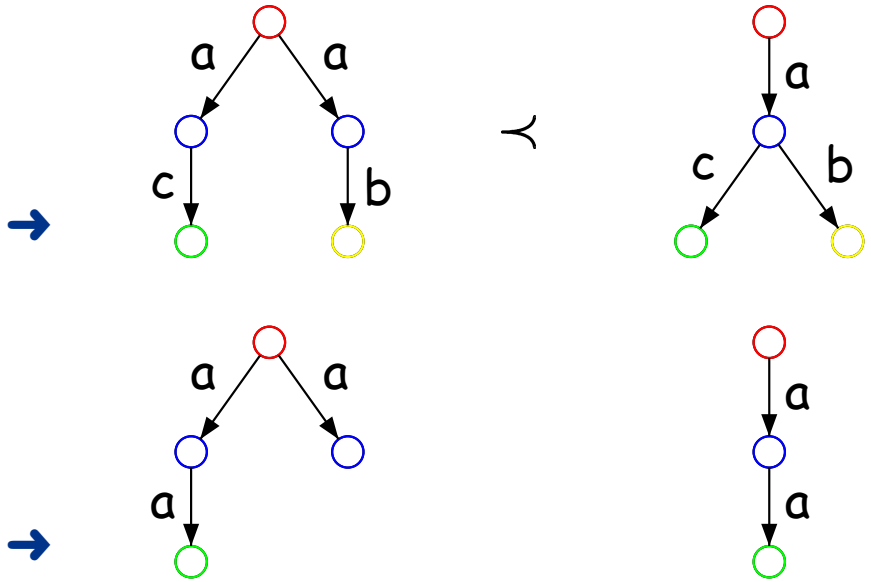
Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?



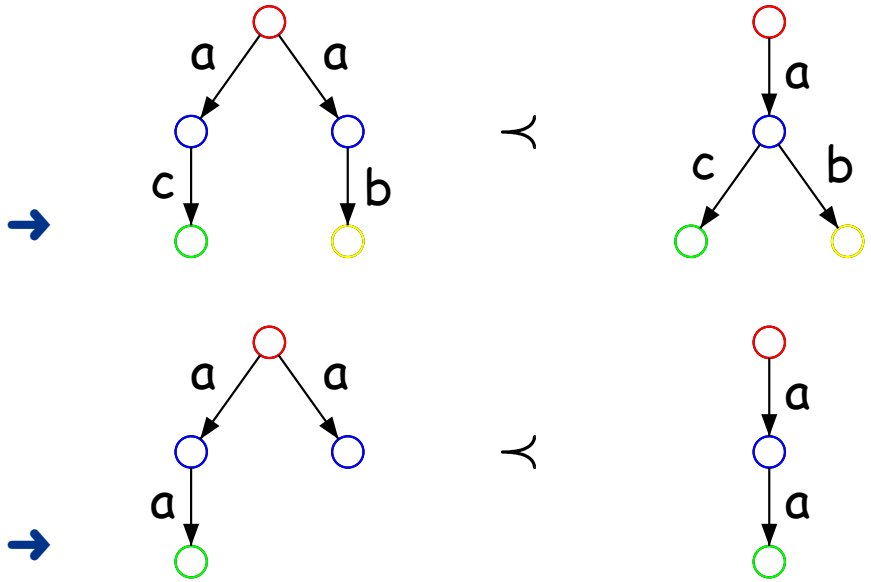
Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?



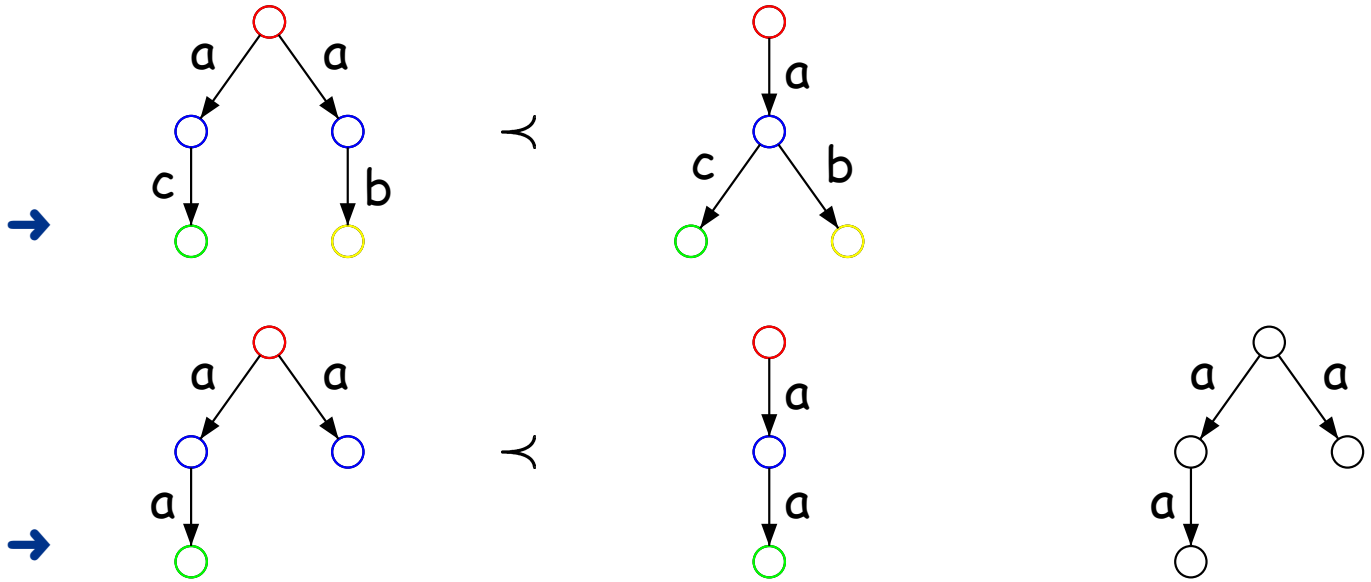
Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?



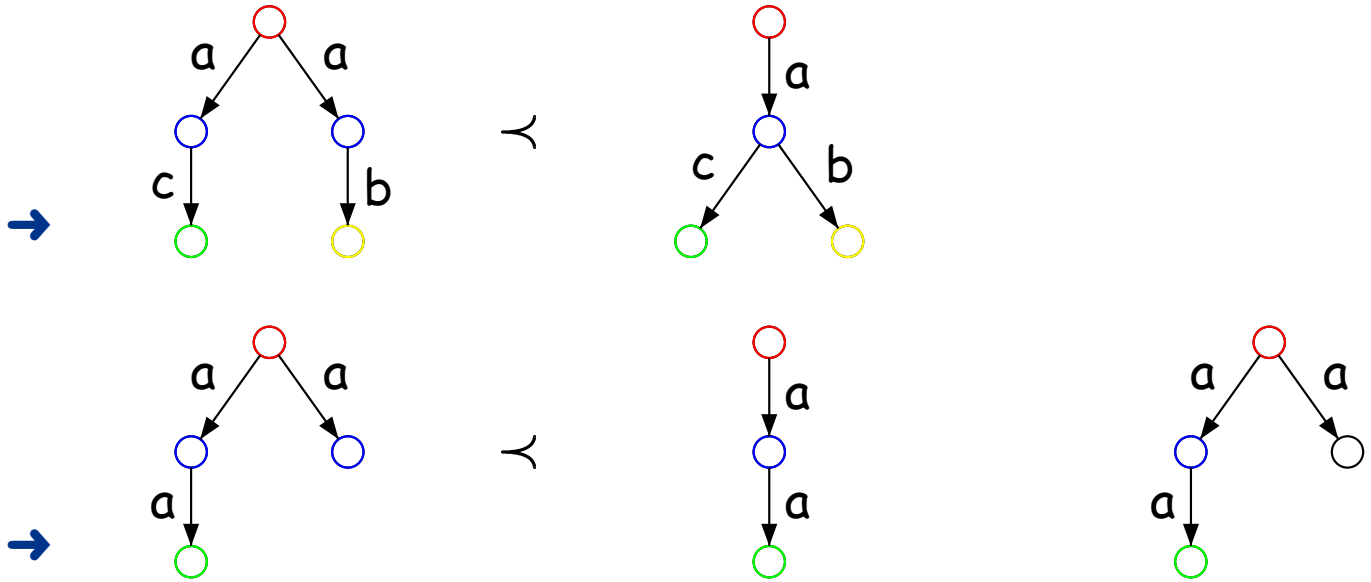
Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?



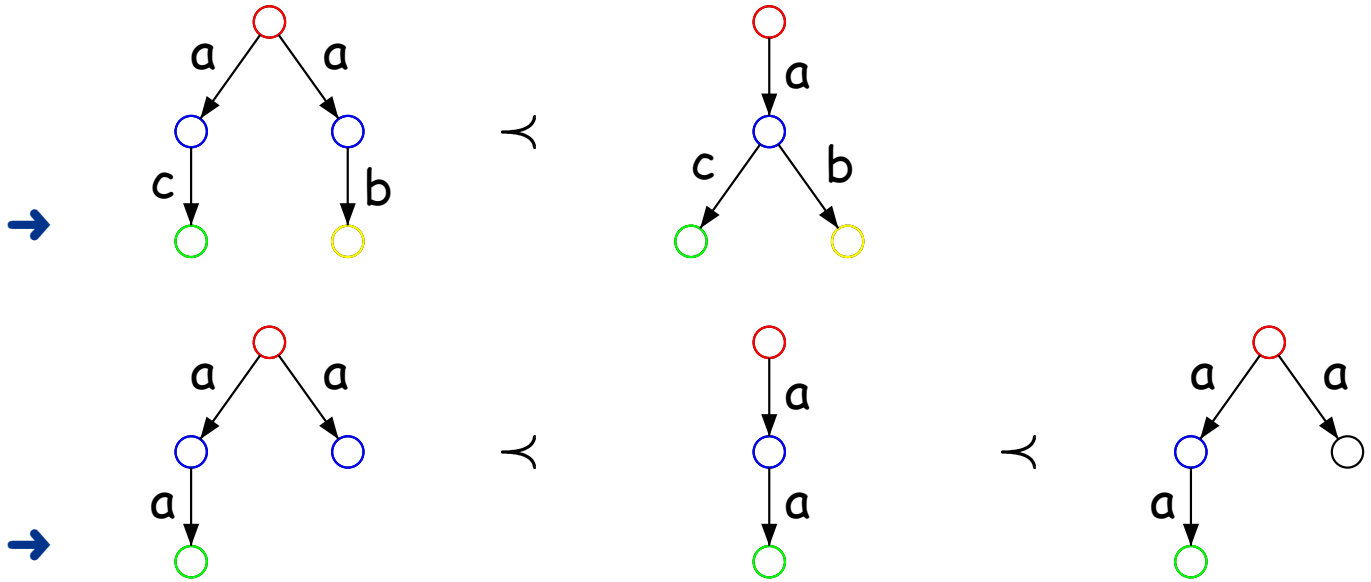
Exemples

Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?

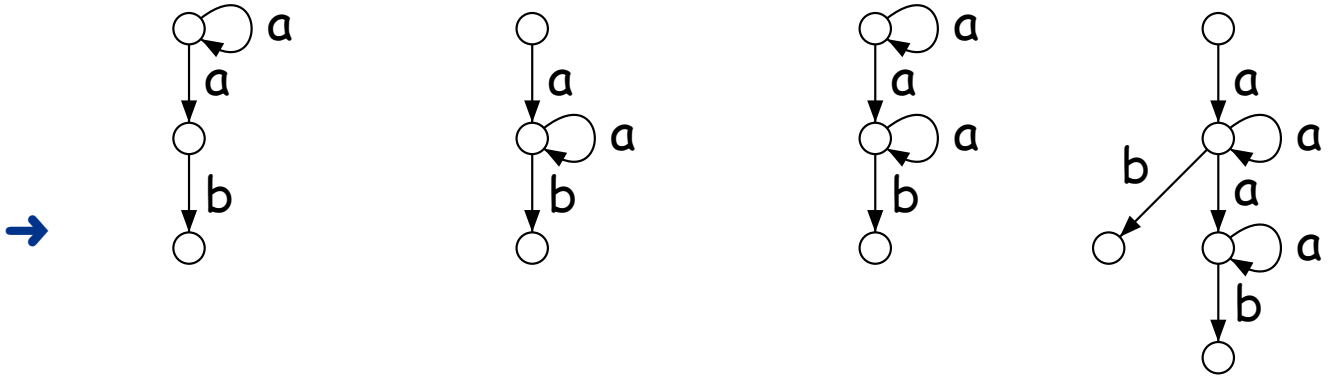


Exemples

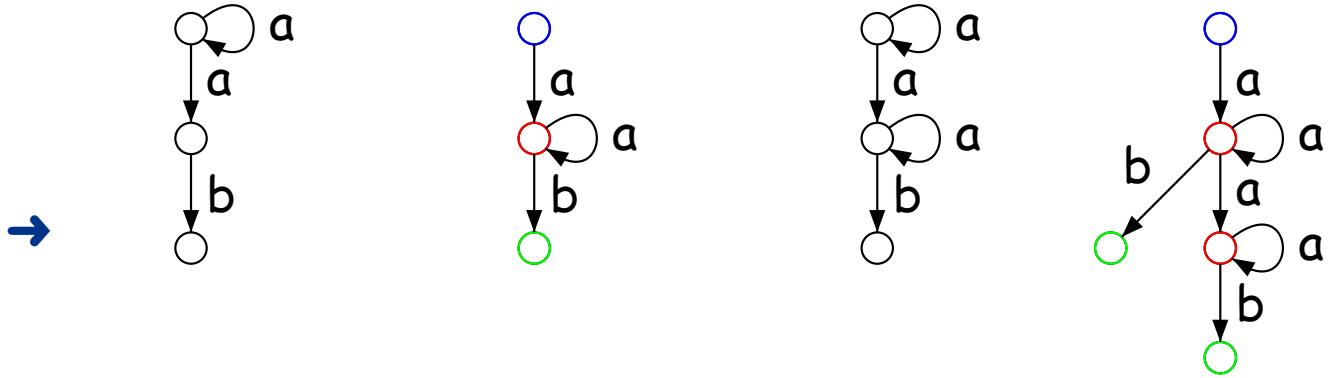
Les systèmes suivants sont-ils en bisimulation ? Si non, qui simule qui ?



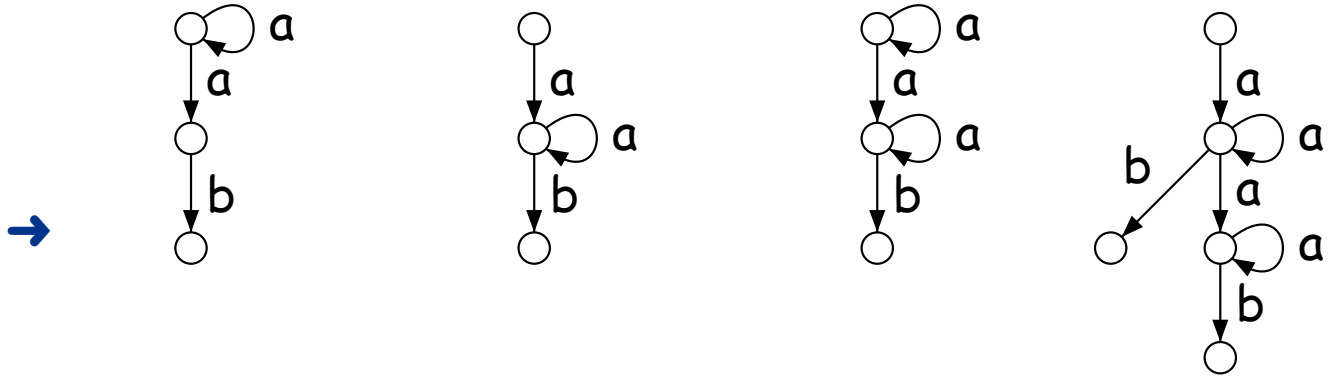
Exemples



Exemples

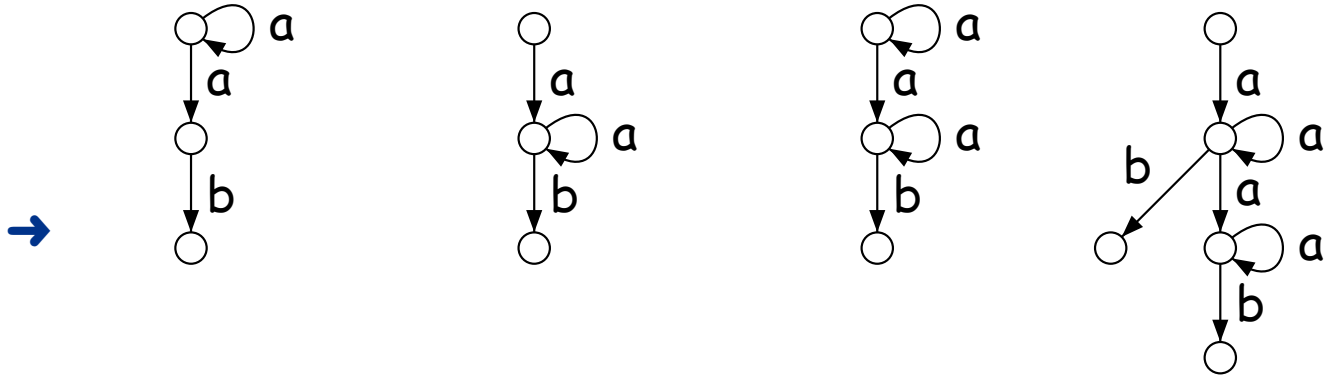


Exemples



Exercice : quelle(s) relation(s) y a-t-il entre un automate et son déterminisé ? et entre un automate et son automate minimal ?

Exemples



Exercice : quelle(s) relation(s) y a-t-il entre un automate et son déterminisé ? et entre un automate et son automate minimal ?

- le déterminisé simule l'automate original
- le minimal simule l'automate original
- aucun n'est en bisimulation avec l'automate original

Intérêt de la (bi)simulation

La bisimulation préserve les propriétés telles que :

- l'accessibilité
- le blocage
- la vivacité
- le langage reconnu
- ...
- bref, presque tout !

La simulation préserve moins de propriétés, par exemple elle ne préserve pas :

- le blocage
- la vivacité

Systèmes temporisés

Les systèmes temporisés

➜ systèmes dans lesquels le temps joue un rôle primordial

Exemple : l'alarme se déclenche moins de 2 minutes après le début de l'incendie

Les systèmes temporisés

➜ systèmes dans lesquels le temps joue un rôle primordial

Exemple : l'alarme se déclenche moins de 2 minutes après le début de l'incendie

$$AG\,(\text{incendie} \Rightarrow AF_{<2}\text{alarme})$$

Les systèmes temporisés

➜ systèmes dans lesquels le temps joue un rôle primordial

Exemple : l'alarme se déclenche moins de 2 minutes après le début de l'incendie

$$AG\ (\text{incendie} \Rightarrow AF_{<2}\text{alarme})$$

$$AG\ (\text{incendie} \Rightarrow x\ \text{in}\ AF(x < 2 \wedge \text{alarme}))$$

Les systèmes temporisés

➜ systèmes dans lesquels le temps joue un rôle primordial

Exemple : l'alarme se déclenche moins de 2 minutes après le début de l'incendie

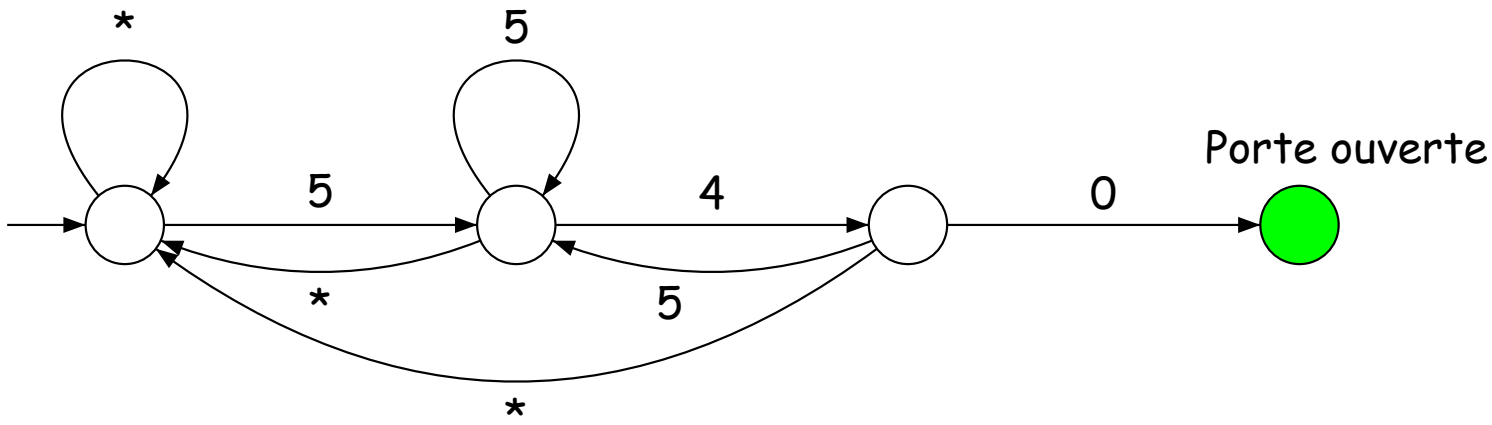
$$AG\ (\text{incendie} \Rightarrow AF_{<2}\text{alarme})$$

$$AG\ (\text{incendie} \Rightarrow x\ \text{in}\ AF(x < 2 \wedge \text{alarme}))$$

$$\max(X, \forall X \wedge [a]X \wedge (\text{incendie} \Rightarrow x\ \text{in}\ \max(Y, \text{alarme} \vee (x < 2 \vee \exists Y \vee \langle a \rangle Y))))$$

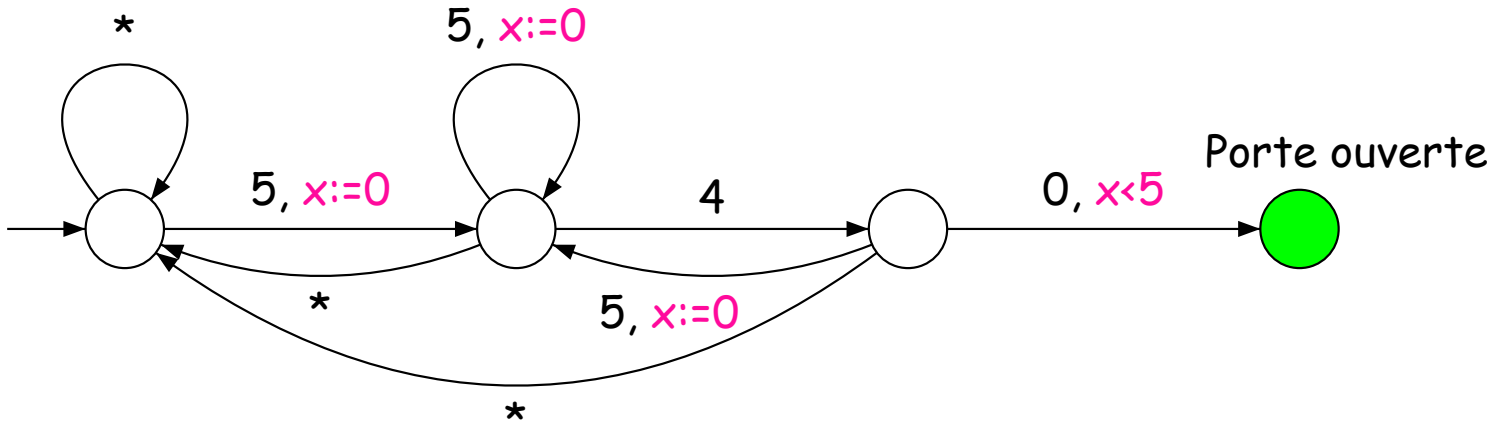
Le digicode

➜ **Le digicode :**



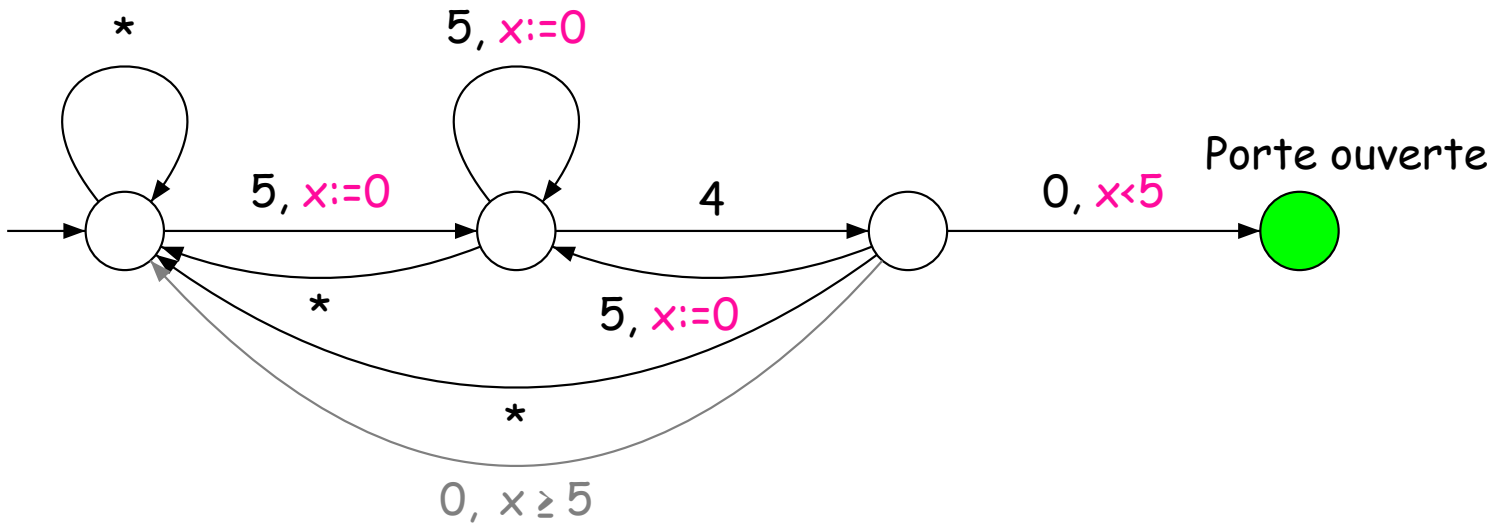
Le digicode

➜ **Le digicode :**



Le digicode

➜ **Le digicode :**



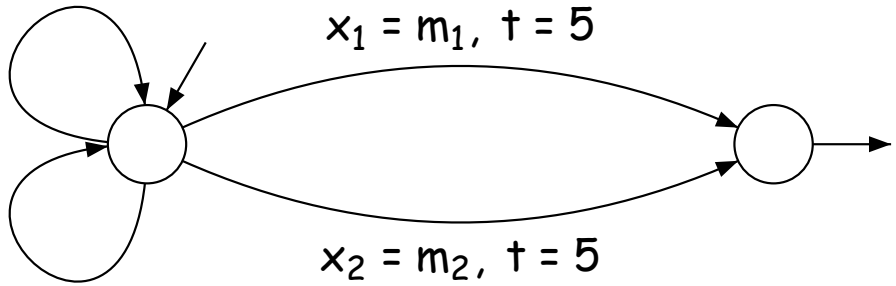
Le cuit-œuf

- **Le cuit-œuf** : on souhaite faire cuire un œuf dur (donc 5 minutes ☺) et on ne dispose que d'un sablier de 7 minutes et d'un sablier de 10 minutes. Comment faire ?

Le cuit-œuf

➜ **Le cuit-œuf** : on souhaite faire cuire un œuf dur (donc 5 minutes ☺) et on ne dispose que d'un sablier de 7 minutes et d'un sablier de 10 minutes. Comment faire ?

$x_i = m_i,\ x_j := 0,\ t := 0$



$x_i = m_i,\ x_j := 0$

x_1 horloge pour le minuteur de 7 minutes ($m_1 = 7$)

x_2 horloge pour le minuteur de 10 minutes ($m_2 = 10$)

t horloge pour mesurer les 5 minutes

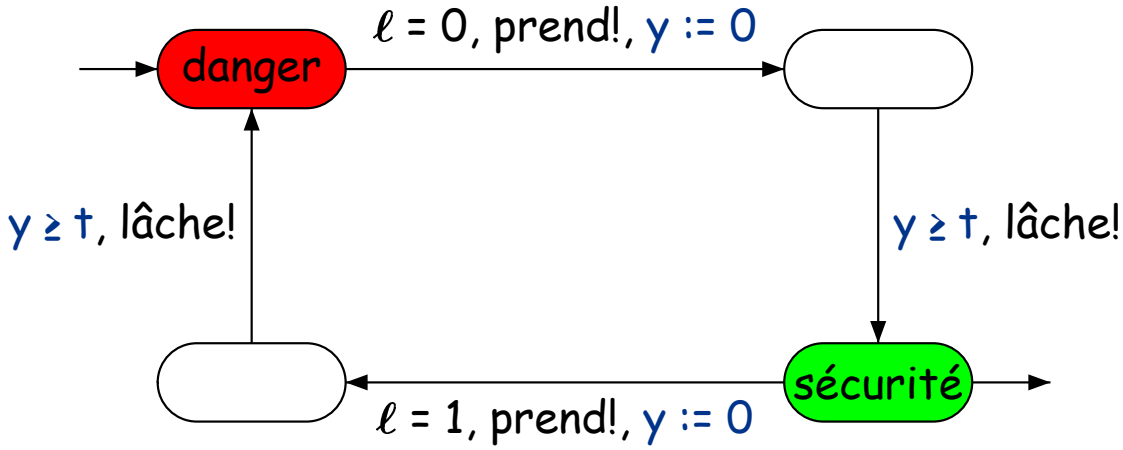
Les aventuriers

- Quatre aventuriers souhaitent traverser un pont en mauvais état. Ils ne mettent pas tous le même temps à faire la traversée. L'un met 5 minutes, un autre 10 minutes, un autre encore 20 minutes et le dernier met 25 minutes.

  Question : Peuvent-ils tous traverser le pont en moins de 60 minutes, sachant qu'il faut obligatoirement la torche pour traverser et que le pont ne peut pas supporter le poids de plus de deux personnes en même temps.

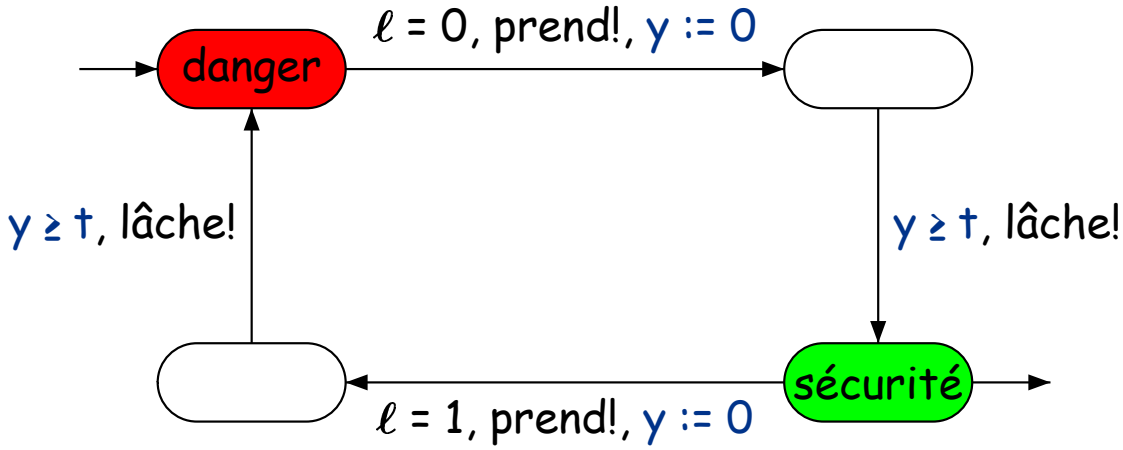
Les aventuriers

Un aventurier (t, temps de traversée) :

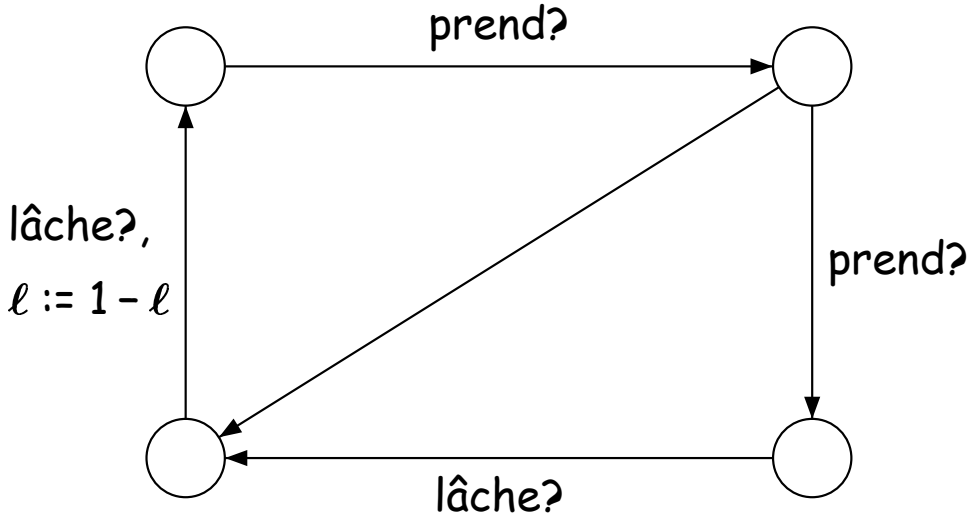


Les aventuriers

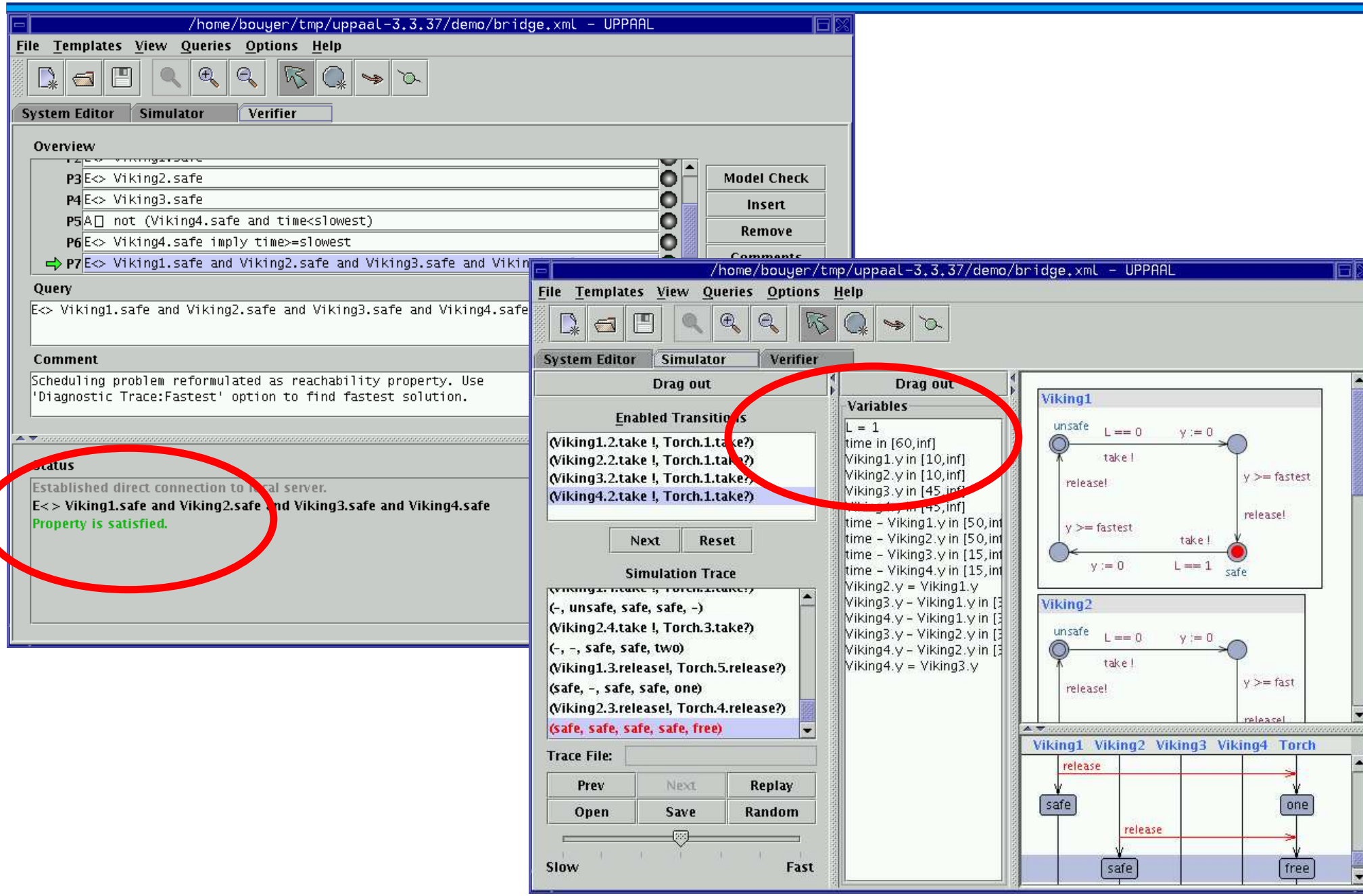
Un aventurier (t, temps de traversée) :



La torche :

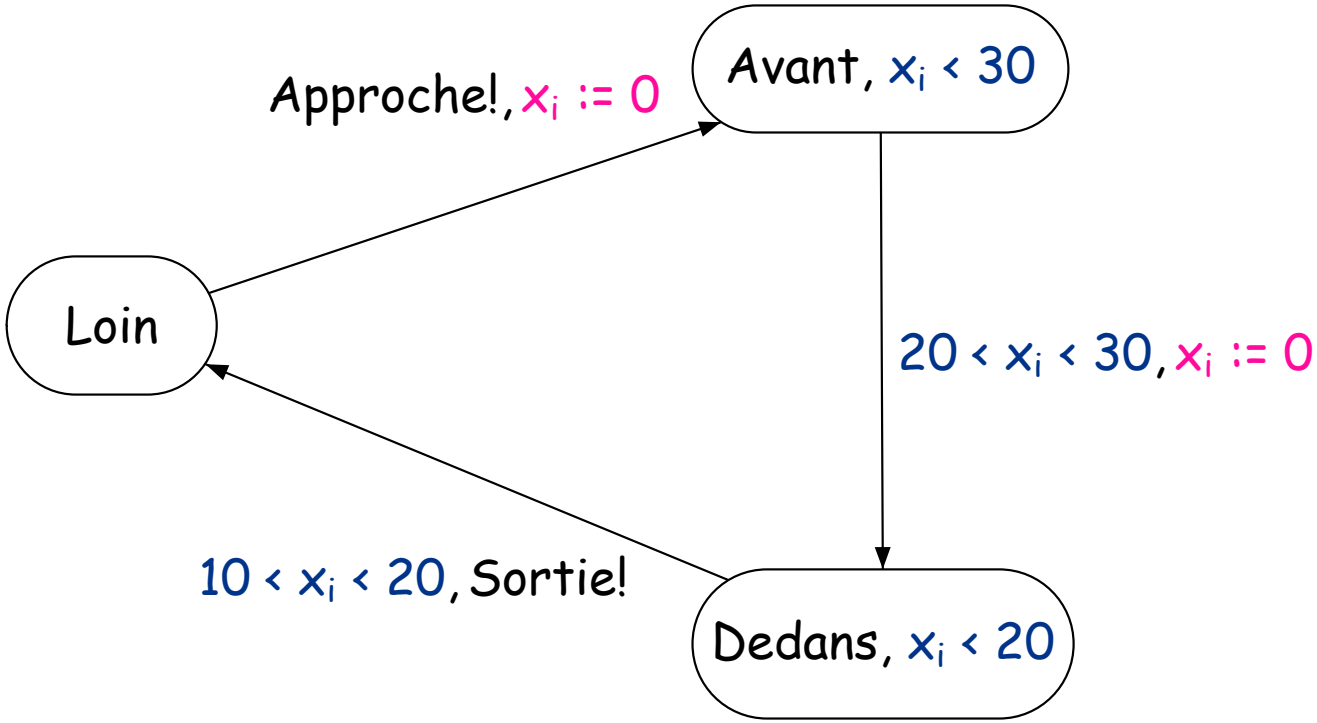


Les aventuriers



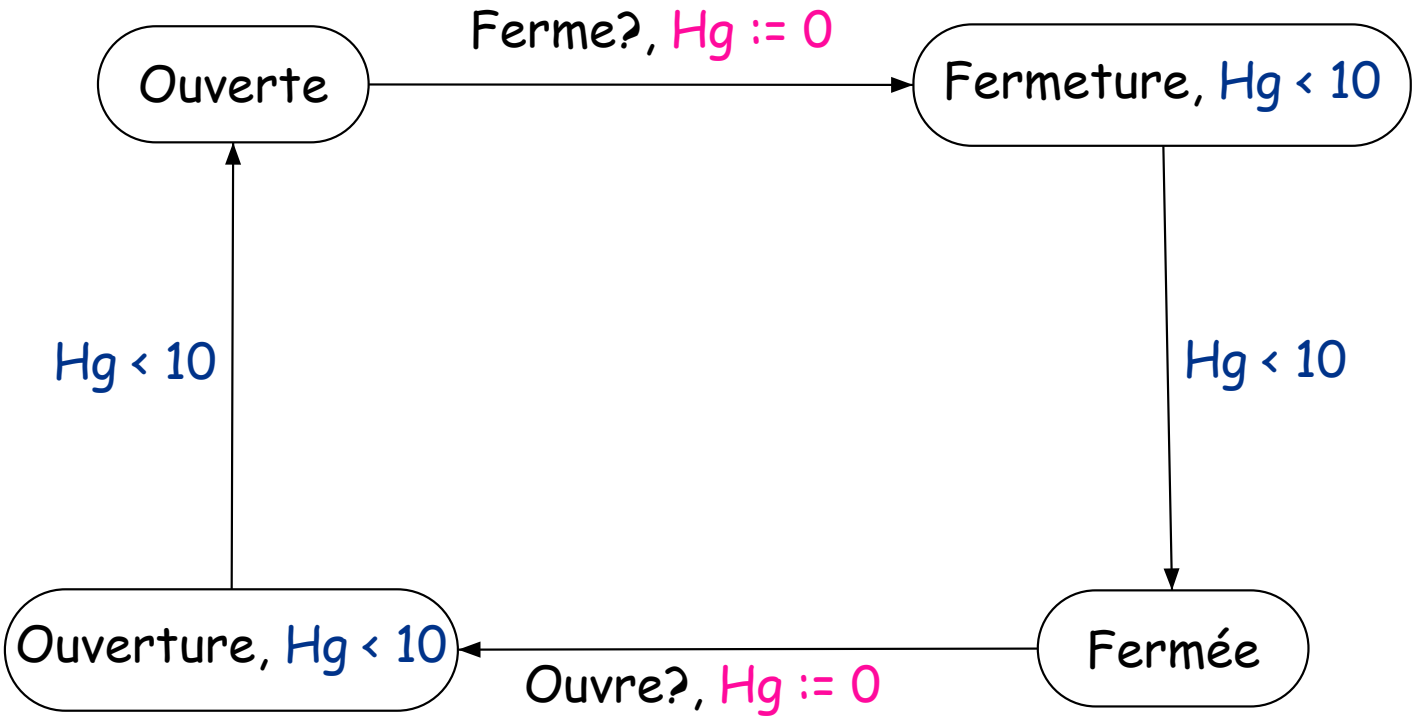
Le passage à niveau

➜ **Un train**



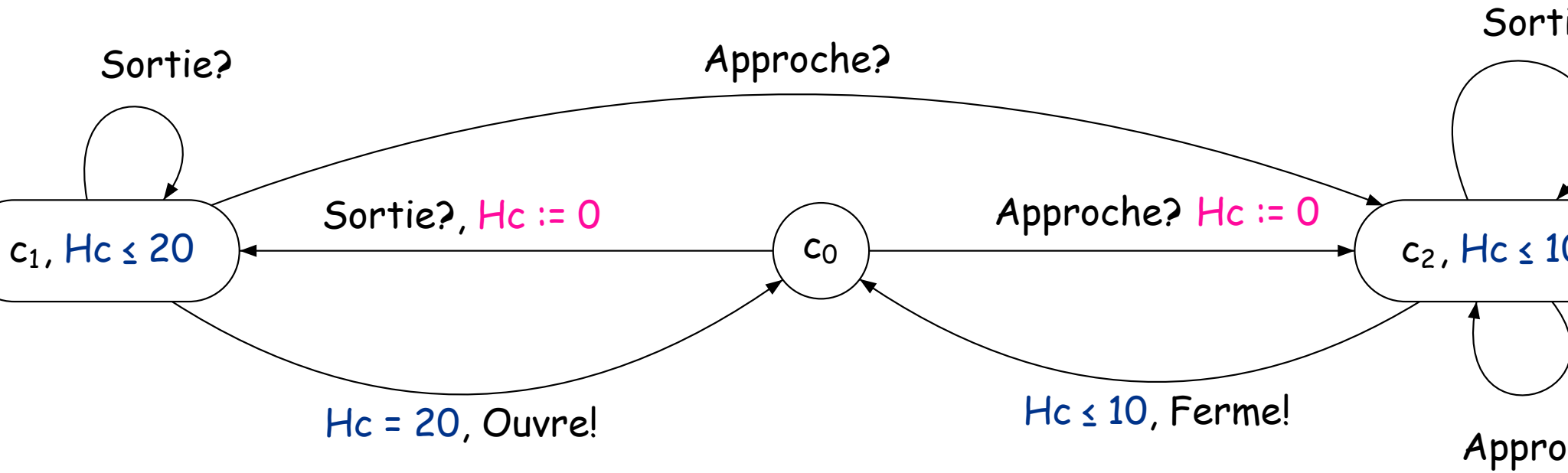
Le passage à niveau

- La barrière



Le passage à niveau

➜ **Le contrôleur**



Le passage à niveau

Exemples de propriétés à vérifier :

- ➜ La barrière est-elle fermée lorsque le train est dans le passage à niveau ?

Le passage à niveau

Exemples de propriétés à vérifier :

- La barrière est-elle fermée lorsque le train est dans le passage à niveau ?

$$AG(\text{train.Dedans} \Rightarrow \text{barrière.Fermée})$$

Le passage à niveau

Exemples de propriétés à vérifier :

- La barrière est-elle fermée lorsque le train est dans le passage à niveau ?

$$AG(\text{train.Dedans} \Rightarrow \text{barrière.Fermée})$$

- La barrière reste-t-elle toujours fermée moins de 5 minutes ?

Le passage à niveau

Exemples de propriétés à vérifier :

- La barrière est-elle fermée lorsque le train est dans le passage à niveau ?

$$AG(\text{train.Dedans} \Rightarrow \text{barrière.Fermée})$$

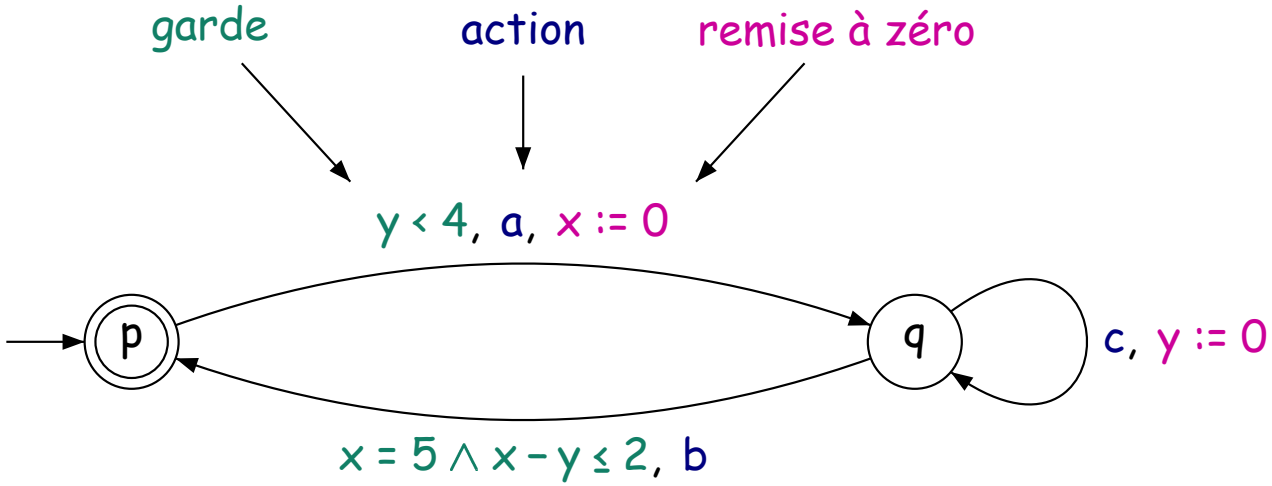
- La barrière reste-t-elle toujours fermée moins de 5 minutes ?

$$\neg EF(\text{barrière.Fermée} \wedge (\text{barrière.Fermée } U_{>5\ \text{min}}\ \neg\text{barrière.Fermée}))$$

Le modèle des automates temporisés

x, y : horloges

[Alur & Dill - 1990's]



	p	$\xrightarrow[3.2]{a}$	q	$\xrightarrow[5.1]{c}$	q	$\xrightarrow[8.2]{b}$	p	...
valeur de x	0		0		1.9		5	...
valeur de y	0		3.2		0		3.1	...

valuation des horloges

➜ mot temporisé $(a, 3.2)(c, 5.1)(b, 8.2)\ldots$

Quelques constatations

➜ une configuration du système, c'est :

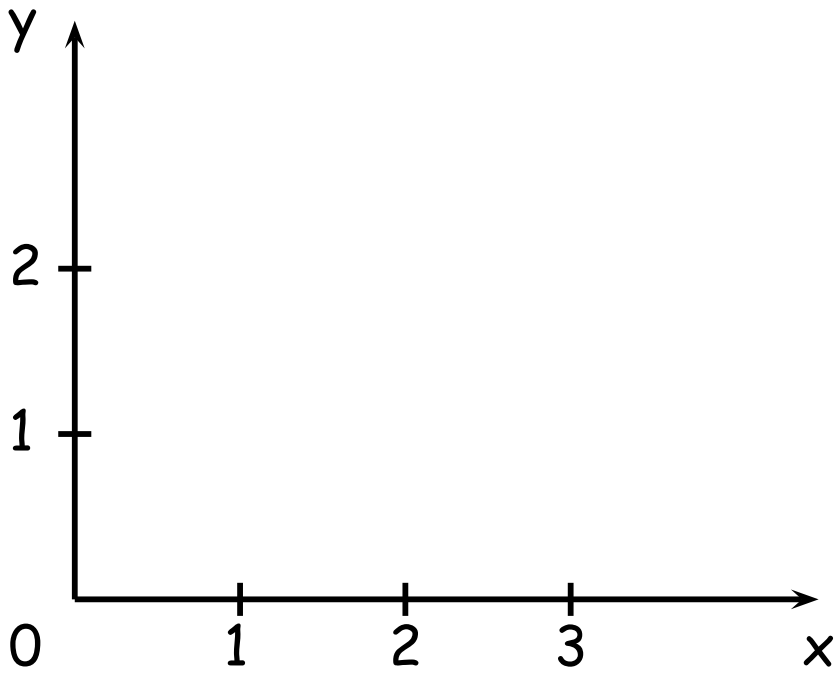
- un état de contrôle
- une valeur pour chaque horloge

➜ ensemble **infini** (non discret) de configurations

➜ toutes les variables avancent à la même vitesse

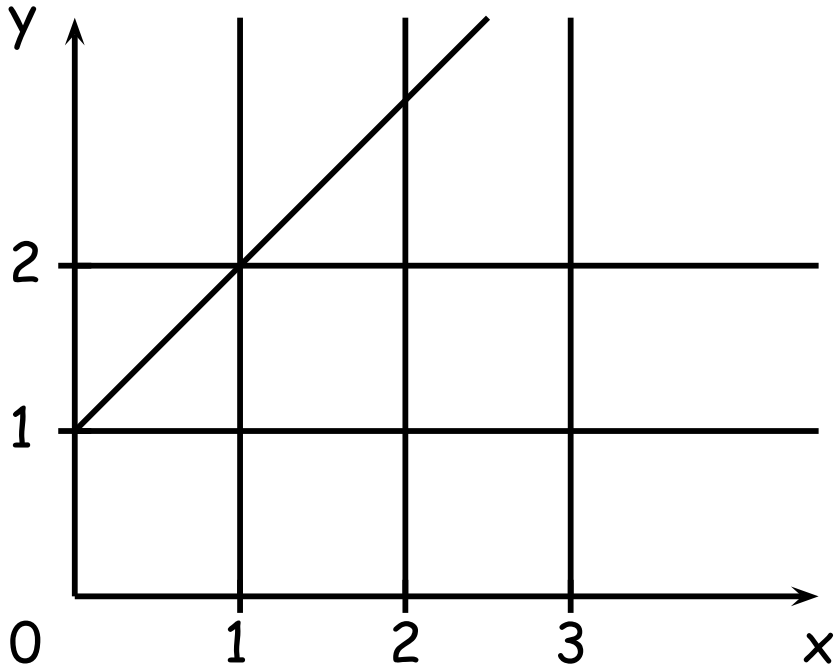
But : construire une abstraction **finie**, correcte et complète

L'abstraction des régions



Équivalence d'index fini

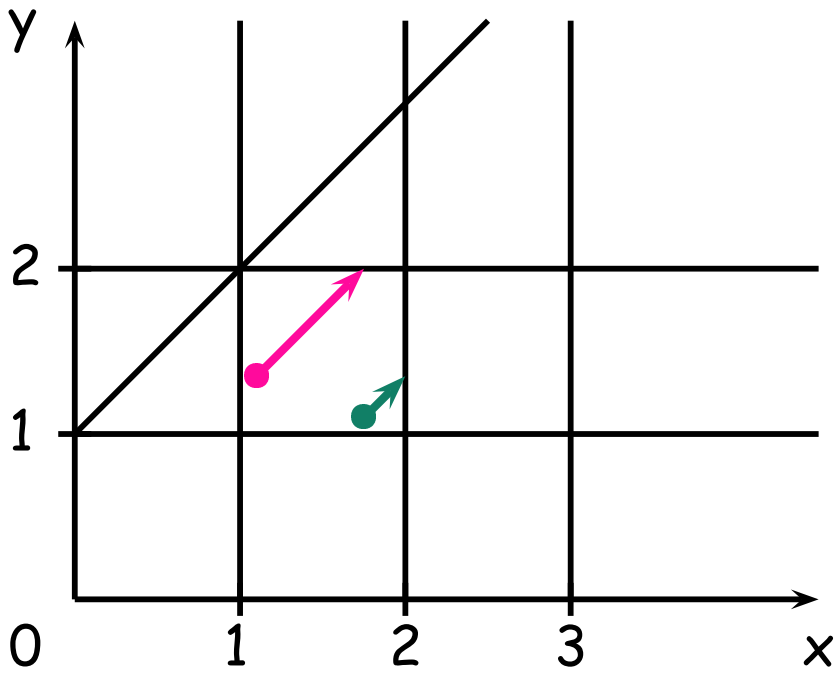
L'abstraction des régions



Équivalence d'index fini

➜ « compatibilité » entre les régions et les contraintes

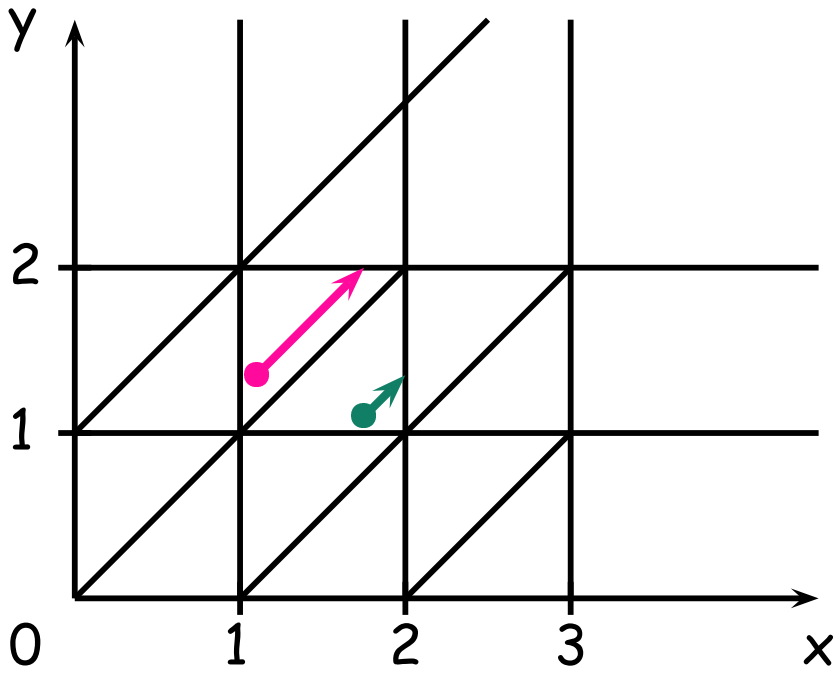
L'abstraction des régions



Équivalence d'index fini

- « compatibilité » entre les régions et les contraintes
- « compatibilité » entre les régions et l'écoulement du temps

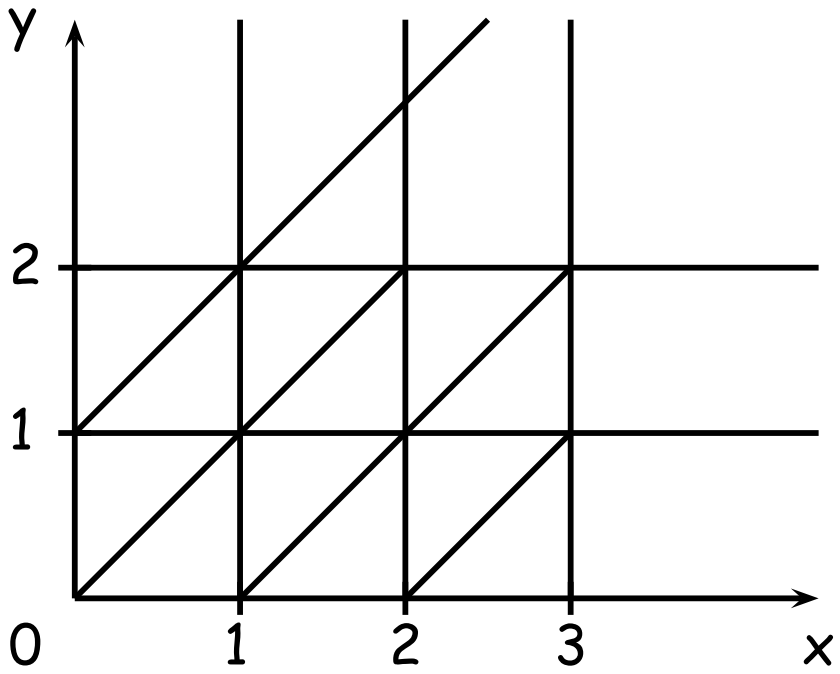
L'abstraction des régions



Équivalence d'index fini

- « compatibilité » entre les régions et les contraintes
- « compatibilité » entre les régions et l'écoulement du temps

L'abstraction des régions



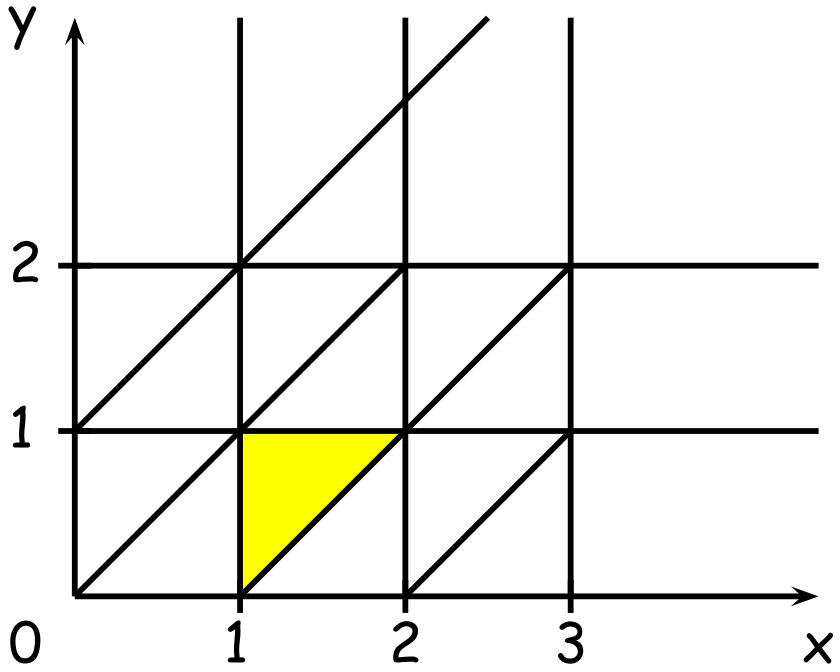
Équivalence d'index fini

➜ « compatibilité » entre les régions et les contraintes

➜ « compatibilité » entre les régions et l'écoulement du temps

➜ une propriété de bisimulation

L'abstraction des régions



Équivalence d'index fini

région définie par

$I_x =]1; 2[$, $I_y =]0; 1[$

$\{x\} < \{y\}$

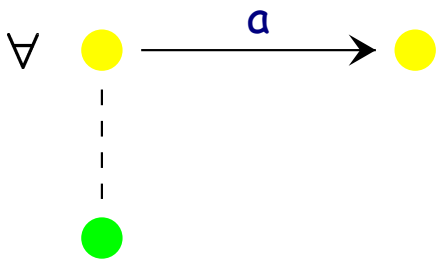
➜ « compatibilité » entre les régions et les contraintes

➜ « compatibilité » entre les régions et l'écoulement du temps

➜ une propriété de bisimulation

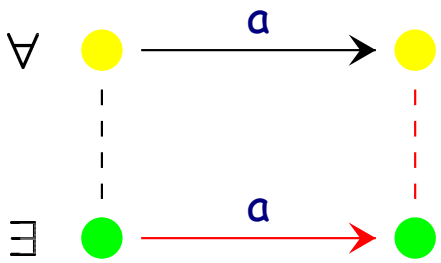
Quelques remarques

Bisimulation faible pour le temps



Quelques remarques

Bisimulation faible pour le temps



Quelques remarques

Bisimulation faible pour le temps

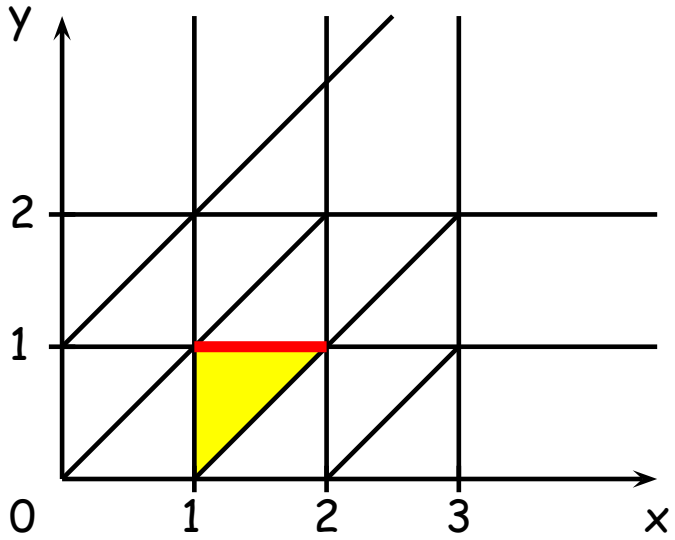
Quelques remarques

Bisimulation faible pour le temps

Quelques remarques

Bisimulation faible pour le temps

Exemple du calcul du successeur



région définie par

$I_x =]1; 2[$, $I_y =]0; 1[$

$\{x\} < \{y\}$

région successeur définie par

$I_x =]1; 2[$, $I_y = \{1\}$

Quelques remarques

Bisimulation faible pour le temps

Remarque : La bisimulation faible ne permet pas de vérifier des propriétés temporisées : nécessité d'introduire de nouvelles horloges.

L'automate des régions

automate temporisé $\otimes$ abstraction des régions

$q \xrightarrow{g,a,C:=0} q'$ est transformée en :

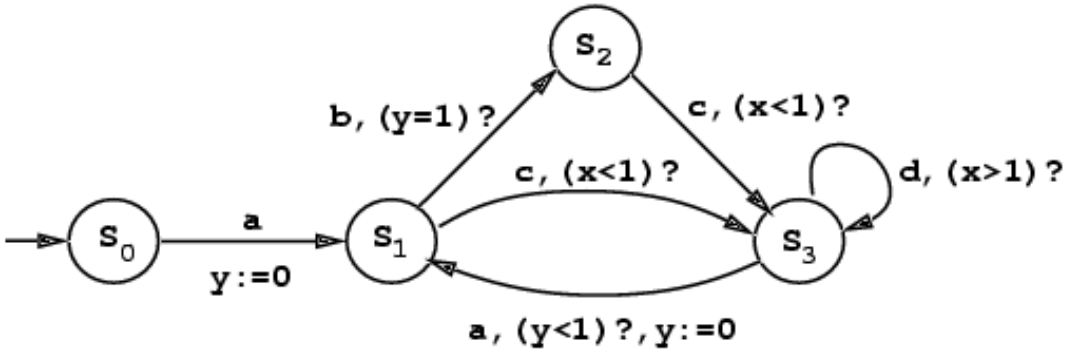
$(q,R) \xrightarrow{a} (q',R')$ s'il existe $R'' \in \mathrm{Succ}_t^*(R)$ t.q.

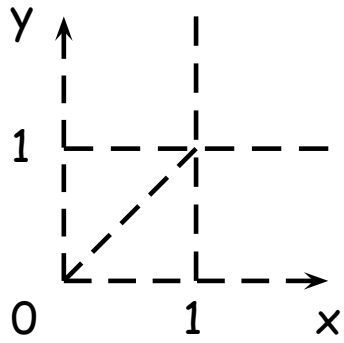
- $R'' \subseteq g$
- $[C \leftarrow 0]R'' \subseteq R'$

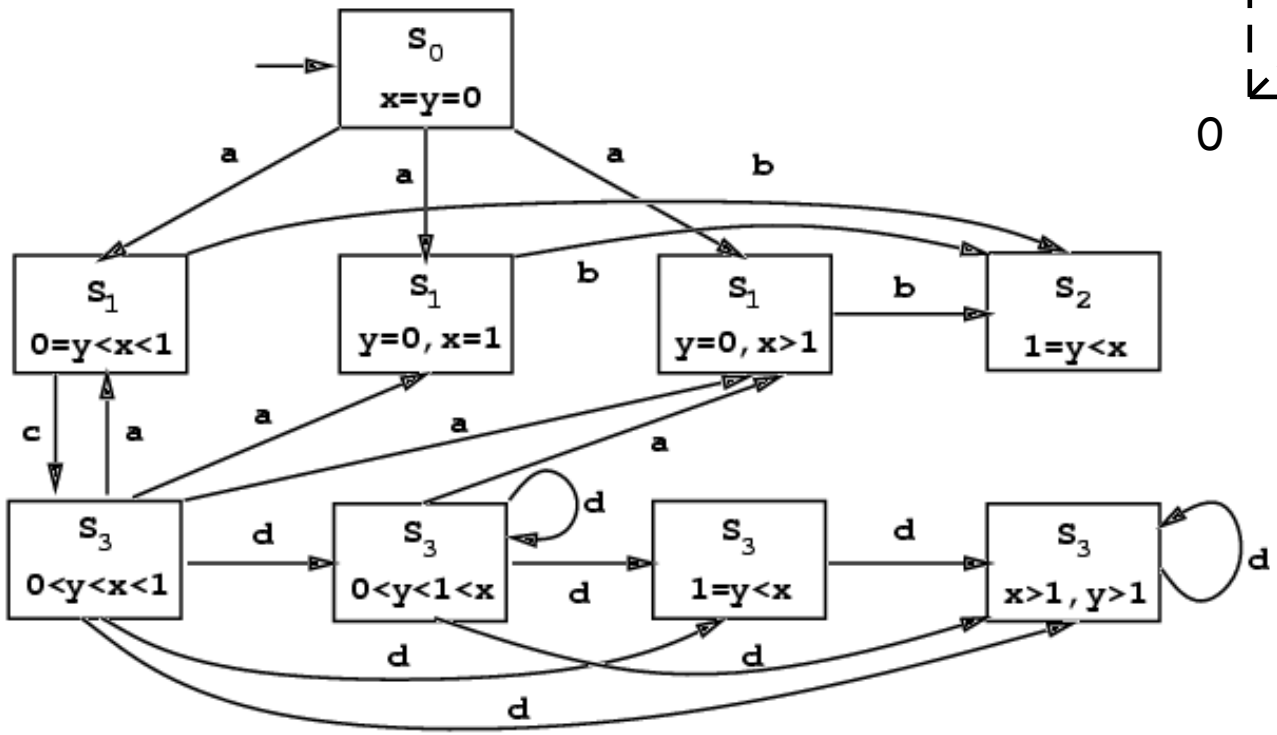
$\mathcal{L}$(automate des régions) = UNTIME($\mathcal{L}$(automate temporisé))

où $\mathrm{UNTIME}((a_1,t_1)(a_2,t_2)\ldots) = a_1a_2\ldots$

Un exemple [AD 90's]







Résultat fondamental

L'abstraction des régions est correcte vis-à-vis de l'accessibilité.

Le problème de l'accessibilité est décidable pour le modèle des automates temporisés

Cependant, c'est un problème PSPACE-complet.

Extension des automates temporisés

On peut imaginer :

- ajouter des actions silencieuses [BDGP98]
- autoriser plus de types de gardes (pour l'instant, on ne peut utiliser que $x - y \bowtie c$ et $x \bowtie c$) [BD00]
- effectuer plus d'opérations sur les horloges [BDFP00]
- autoriser des variables un peu étendues par rapport aux horloges

  ➜ systèmes hybrides

  [ACHH93,Hen96,HKPV98]

Pentes 0/1

Si l'on autorise d'arrêter les horloges, l'accessibilité devient indécidable

Pentes 0/1

Si l'on autorise d'arrêter les horloges, l'accessibilité devient indécidable

➜ par réduction d'une machine à deux compteurs

compteur c codé par deux horloges x_c et y_c, t.q. $c = x_c - y_c$

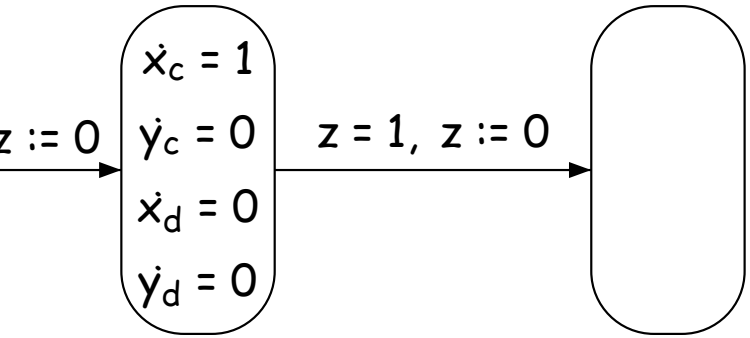
Pentes 0/1

Si l'on autorise d'arrêter les horloges, l'accessibilité devient indécidable

➜ par réduction d'une machine à deux compteurs

compteur c codé par deux horloges x_c et y_c, t.q. $c = x_c - y_c$

Incrémentation du compteur c



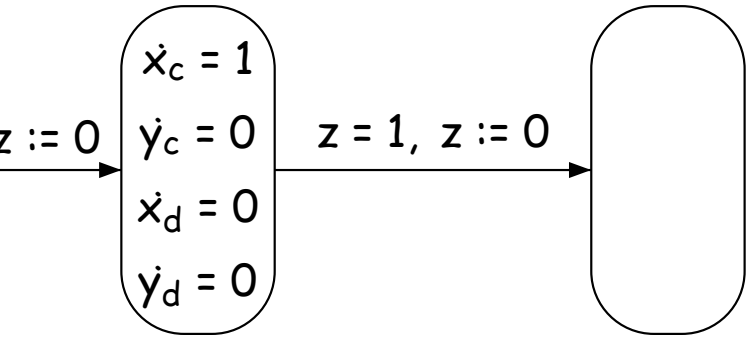
Pentes 0/1

Si l'on autorise d'arrêter les horloges, l'accessibilité devient indécidable

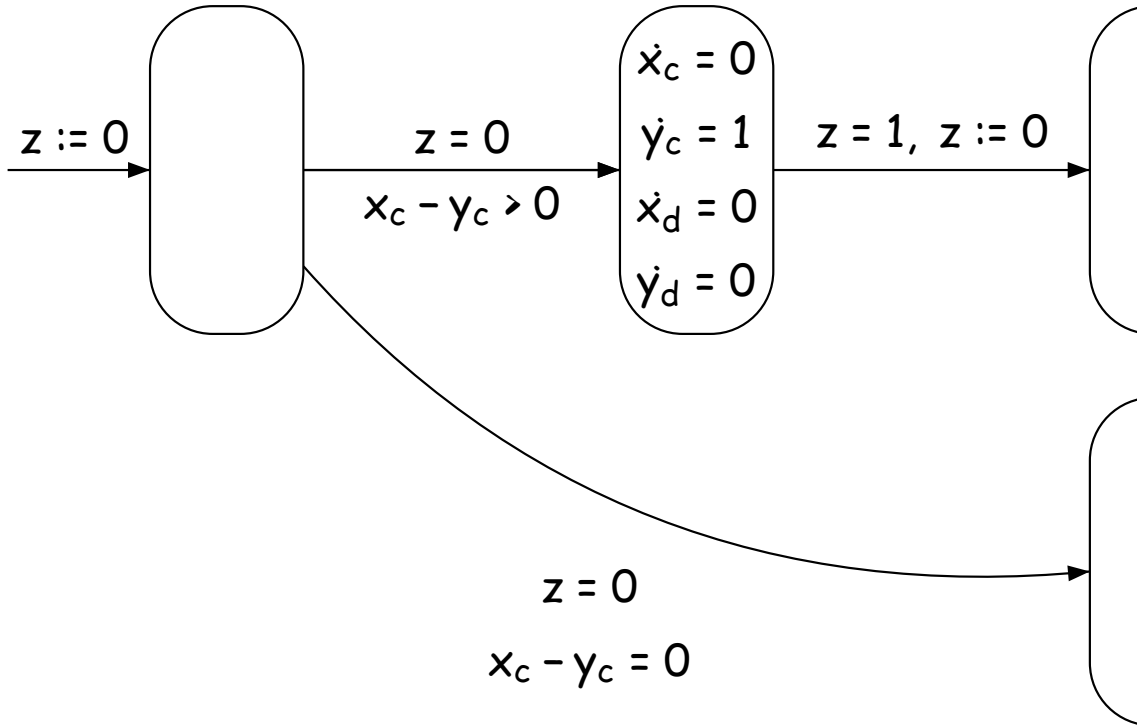
➜ par réduction d'une machine à deux compteurs

compteur c codé par deux horloges x_c et y_c, t.q. $c = x_c - y_c$

Incrémentation du compteur c



Décrémentation du compteur c



Utiliser des gardes $x + y \sim c$

$x + y \sim c$ et $x \sim c$

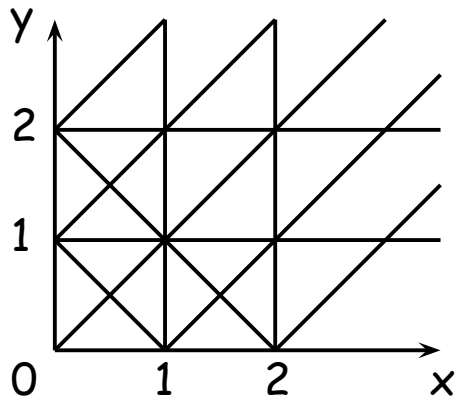
[Bérard,Dufourd 2000]

Utiliser des gardes $x + y \sim c$

$x + y \sim c$ et $x \sim c$

[Bérard,Dufourd 2000]

➜ **Décidabilité :** - avec deux horloges, décidable en utilisant l'abstraction basée sur les régions

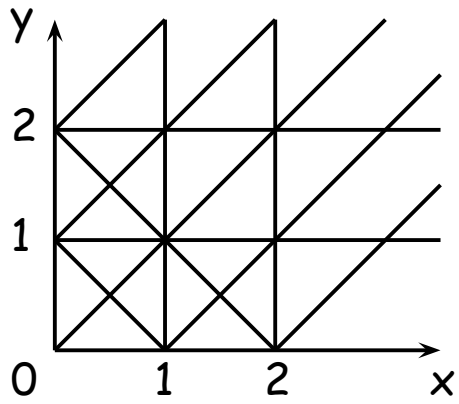


Utiliser des gardes $x + y \sim c$

$x + y \sim c$ et $x \sim c$

[Bérard,Dufourd 2000]

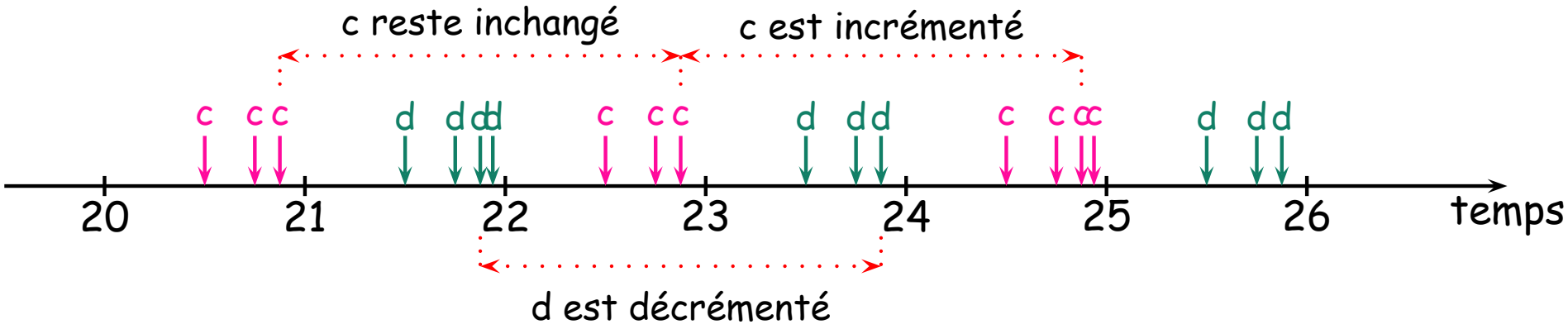
➜ **Décidabilité :** - avec deux horloges, décidable en utilisant l'abstraction basée sur les régions



- avec quatre horloges (ou plus), indécidable !

➜ preuve par réduction de l'arrêt d'une machine à deux compteurs

Preuve d'indécidabilité



➜ simulation de
- décrémentation de d
- incrémentation de c

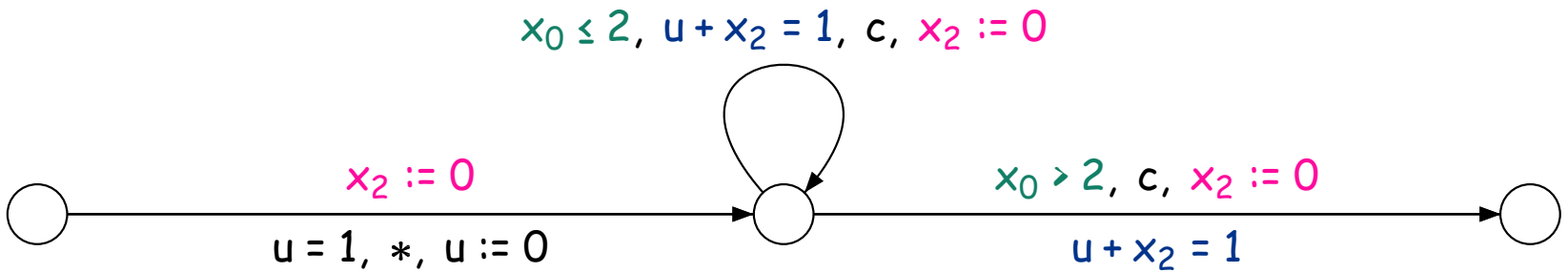
On va utiliser quatre horloges :
- u, horloge qui rythme le temps (chaque unité de temps)
- x_0, x_1, x_2 : horloges références pour les deux compteurs

"x_i référence pour c" ≡ "dernière fois où x_i a été remise à zéro = dernière fois que l'action c a été faite"

[Bérard,Dufourd 2000]

Preuve d'indécidabilité (suite)

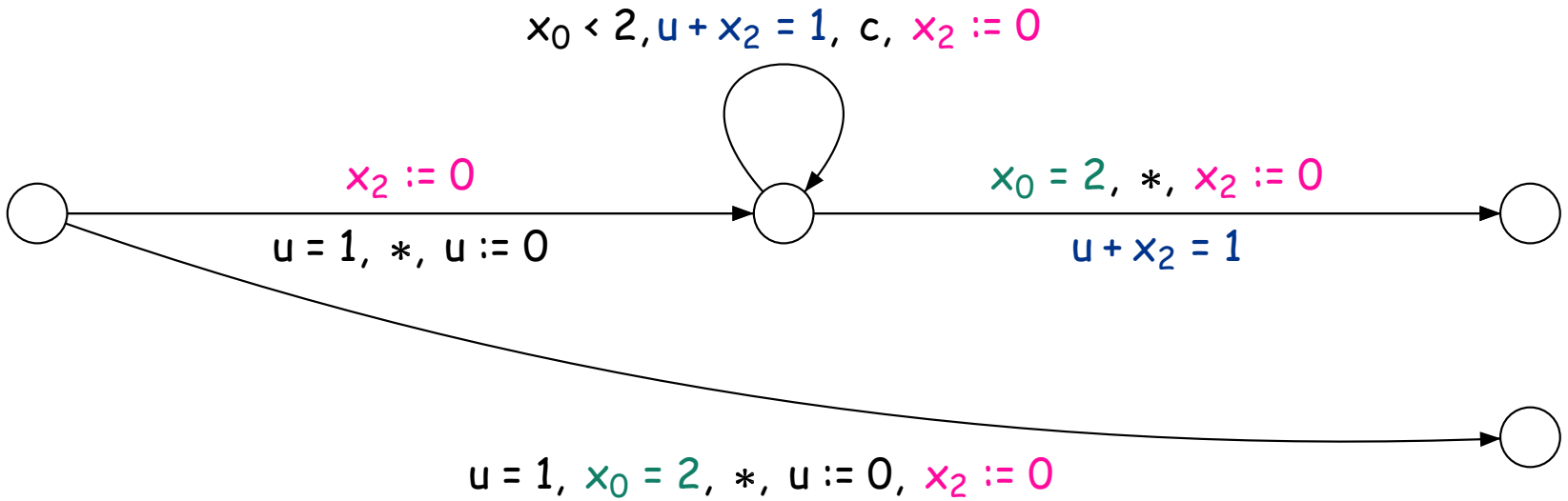
➜ **Incrément du compteur c:**



réf pour c est x_0

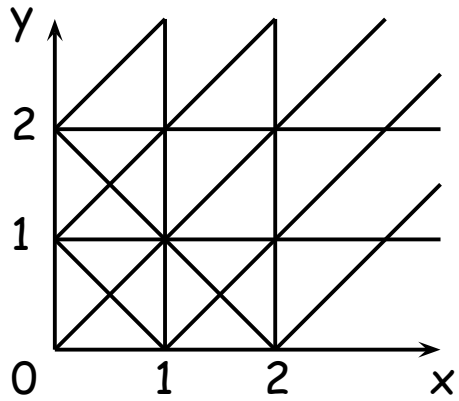
réf pour c est x_2

➜ **Décrément du compteur c :**



Récapitulatif

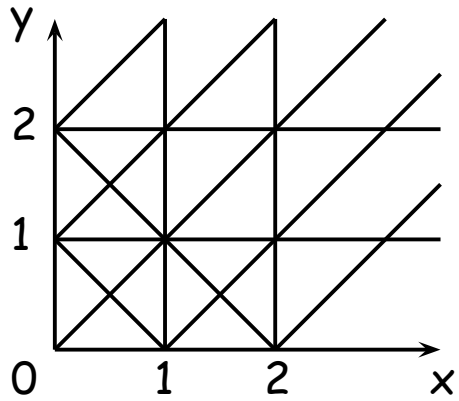
- Deux horloges : décidable !



- Quatre horloges et plus : indécidable !

Récapitulatif

- Deux horloges : décidable !



- Trois horloges : **question ouverte**
- Quatre horloges et plus : indécidable !

Implémentation des automates temporisés

La construction de l'automate des régions n'est pas utilisée en pratique.

- ➜ explosion combinatoire
  (le nombre de régions est exponentielle en le nombre d'horloges)
- ➜ pas de structures de données réellement adaptées

Implémentation des automates temporisés

La construction de l'automate des régions n'est pas utilisée en pratique.

- ➜ explosion combinatoire
  (le nombre de régions est exponentielle en le nombre d'horloges)
- ➜ pas de structures de données réellement adaptées

Des algorithmes de minimisation de l'automate des régions ont été proposés...
[Alur & Co 1992] [Tripakis,Yovine 2001]

Implémentation des automates temporisés

La construction de l'automate des régions n'est pas utilisée en pratique.

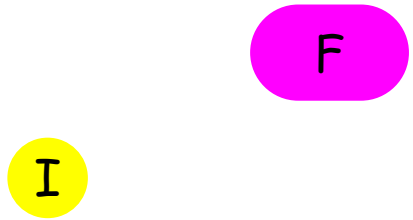
- ➜ explosion combinatoire
  (le nombre de régions est exponentielle en le nombre d'horloges)
- ➜ pas de structures de données réellement adaptées

Des algorithmes de minimisation de l'automate des régions ont été proposés...
[Alur & Co 1992] [Tripakis,Yovine 2001]

... mais **des techniques à la volée** sont préférées.

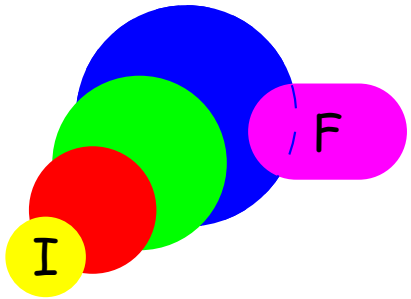
Analyse d'accessibilité

➜ **analyse en avant (Post*) :**
calculer les successeurs des configurations initiales



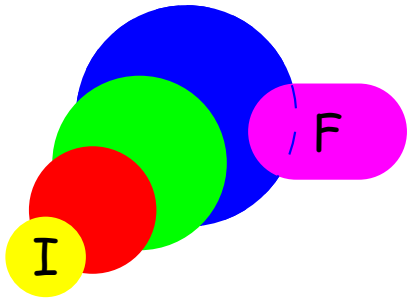
Analyse d'accessibilité

➜ **analyse en avant (Post*) :**
calculer les successeurs des configurations initiales



Analyse d'accessibilité

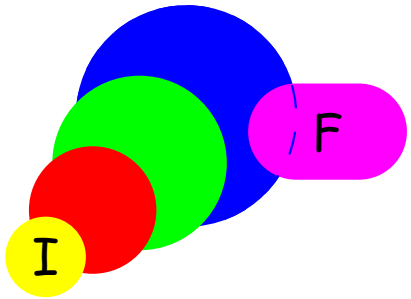
- **analyse en avant (Post*) :**
  calculer les successeurs des configurations initiales



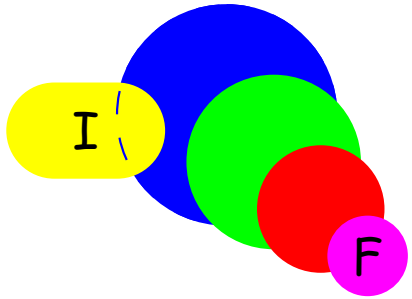
- **analyse en arrière (Pré*) :**
  calculer les prédécesseurs des configurations finales

Analyse d'accessibilité

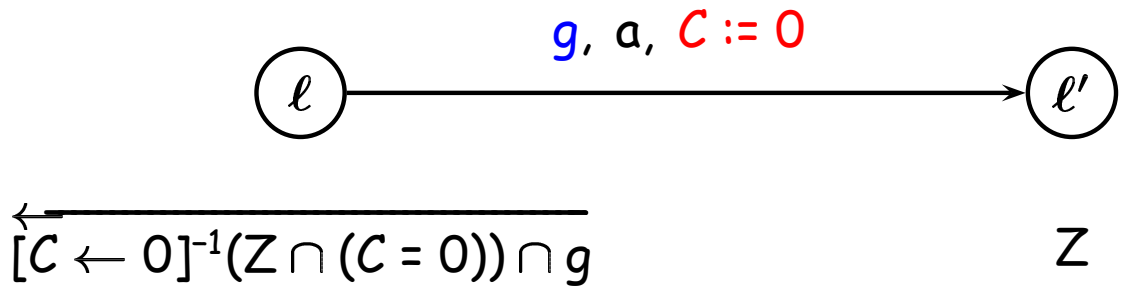
- **analyse en avant (Post*) :**
  calculer les successeurs des configurations initiales



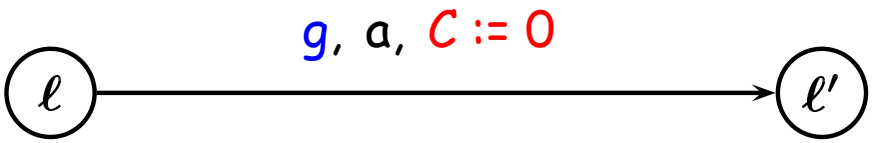
- **analyse en arrière (Pré*) :**
  calculer les prédécesseurs des configurations finales



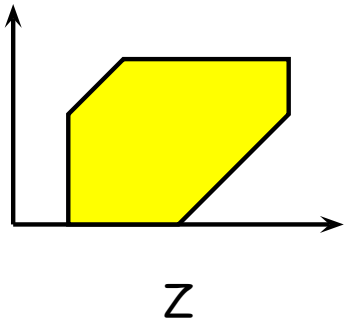
Analyse en arrière



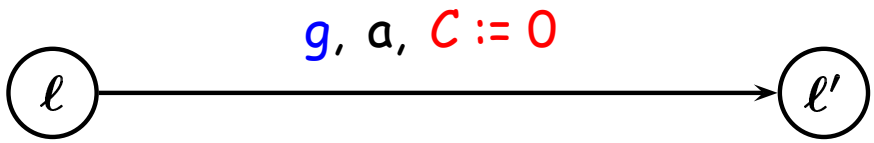
Analyse en arrière



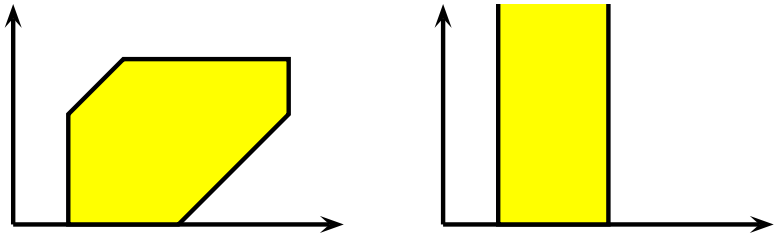
$$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g} \qquad Z$$



Analyse en arrière

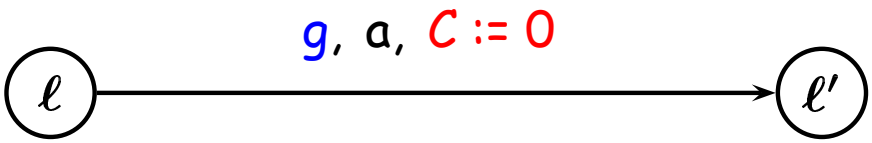


$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\quad Z$



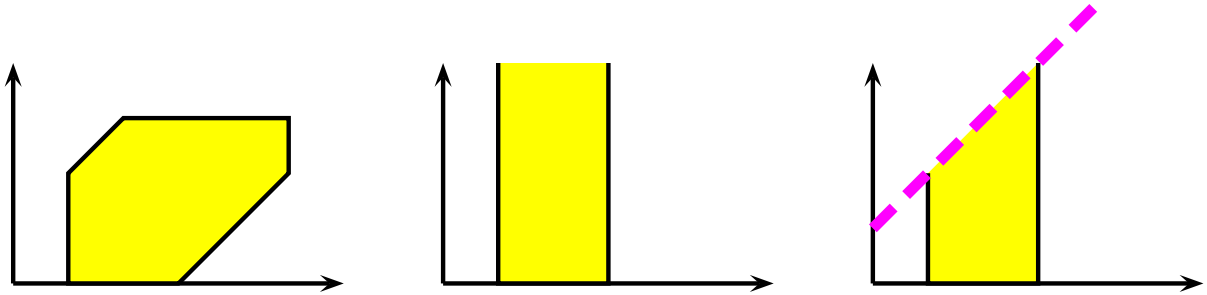
Z $\quad [C \leftarrow 0]^{-1}(Z \cap (C = 0))$

Analyse en arrière



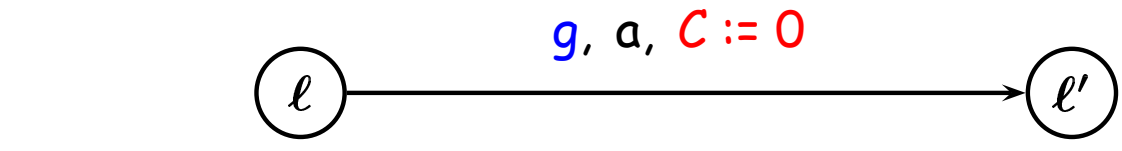
$$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$$

Z

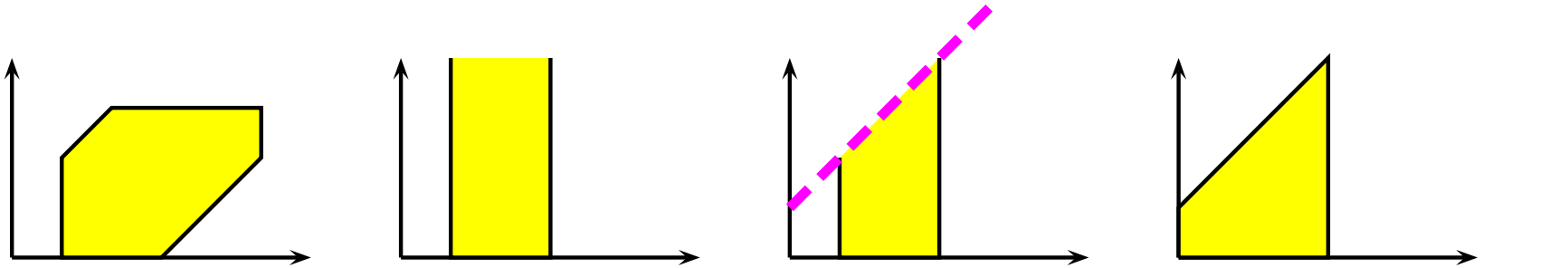


Z $\quad$ $[C \leftarrow 0]^{-1}(Z \cap (C = 0))$

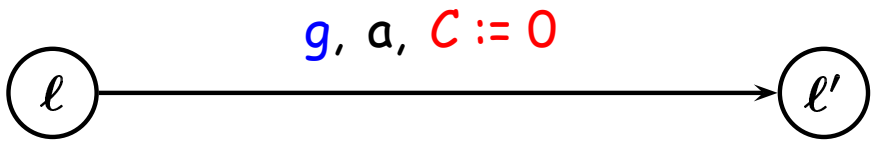
Analyse en arrière



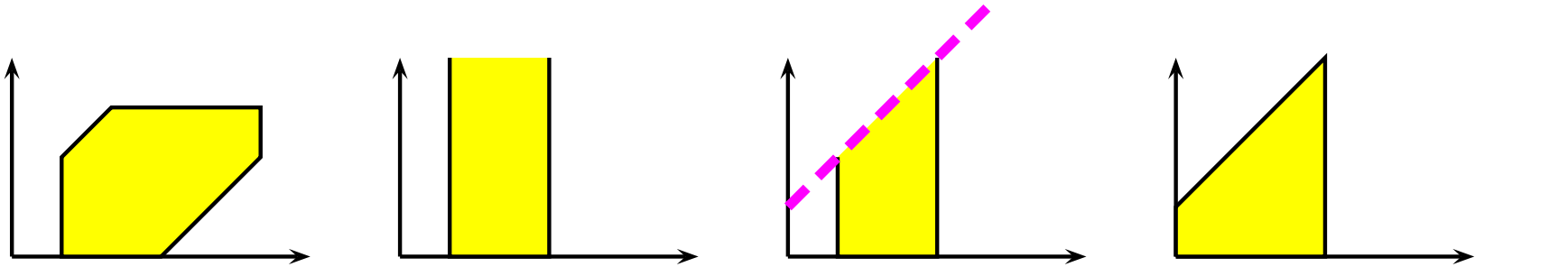
$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\quad Z$



Analyse en arrière



$\overleftarrow{[C \leftarrow 0]^{-1}(Z \cap (C = 0)) \cap g}$ $\quad Z$



Le calcul exact en arrière termine !

Analyse en arrière (suite)

Si $\mathcal{A}$ est un automate temporisé, on construit l'ensemble de régions associé.

Grâce à la propriété de bisimulation, on a :

"Chaque ensemble de valuations calculé par analyse en arrière est une union finie de régions"

Analyse en arrière (suite)

Si $\mathcal{A}$ est un automate temporisé, on construit l'ensemble de régions associé.

Grâce à la propriété de bisimulation, on a :

"Chaque ensemble de valuations calculé par analyse en arrière est une union finie de régions"

Soit R une région. Supposons :

- $v \in \overleftarrow{R}$ (par ex. $v + t \in R$)
- $v' \equiv_{reg.} v$

Il existe un t' t.q. $v' + t' \equiv_{reg.} v + t$, ce qui implique que $v' + t' \in R$ et donc $v' \in \overleftarrow{R}$.

Analyse en arrière (suite)

Si $\mathcal{A}$ est un automate temporisé, on construit l'ensemble de régions associé.

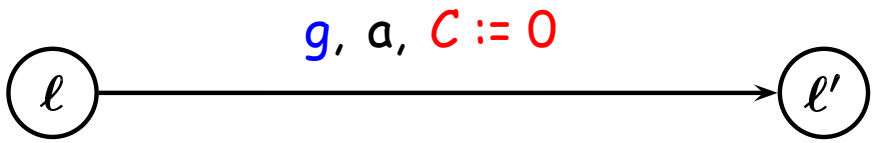
Grâce à la propriété de bisimulation, on a :

"Chaque ensemble de valuations calculé par analyse en arrière est une union finie de régions"

Cependant, le calcul en arrière ne se prête pas à l'utilisation de variables entières...

$$i := j.k + \ell.m$$

Analyse en avant

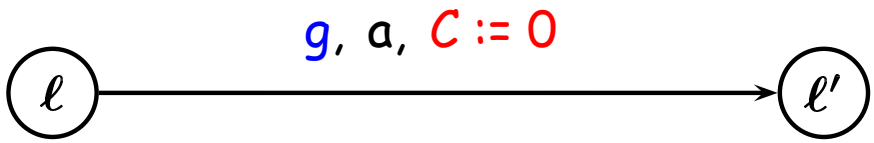


zones Z $[C \leftarrow 0](\overrightarrow{Z} \cap g)$

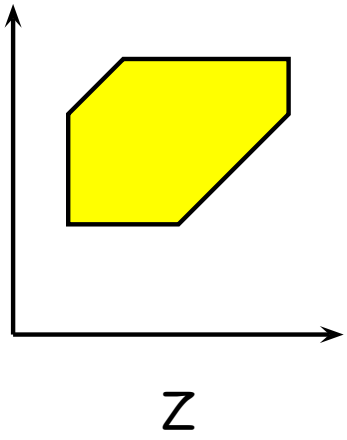
Une zone est un ensemble de valuations défini par une contrainte

$$\varphi ::= x \sim c \mid x - y \sim c \mid \varphi \wedge \varphi$$

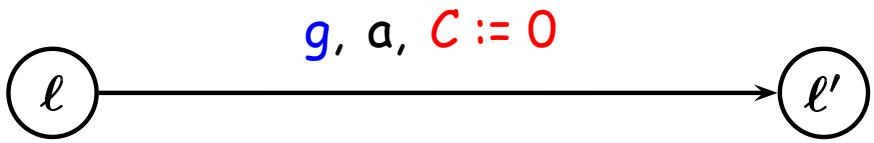
Analyse en avant



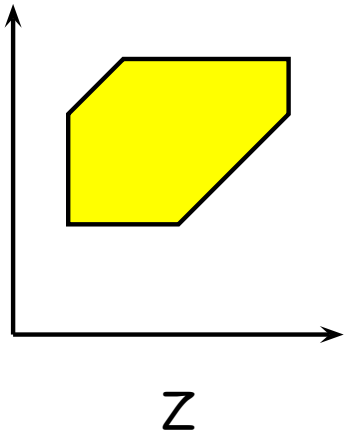
zones Z $[C \leftarrow 0](\overrightarrow{Z} \cap g)$

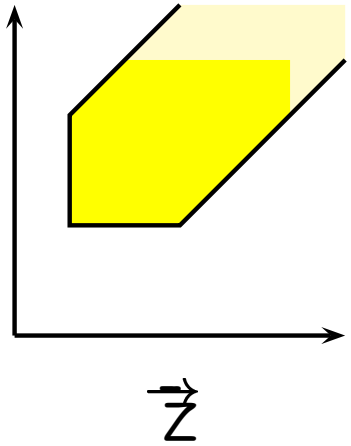


Analyse en avant

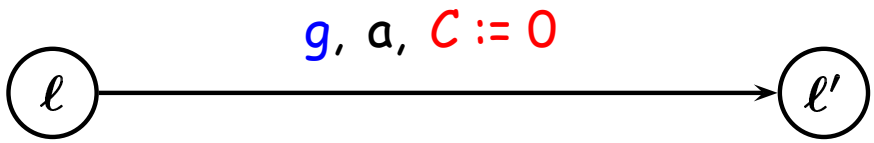


zones Z $[C \leftarrow 0](\overrightarrow{Z} \cap g)$

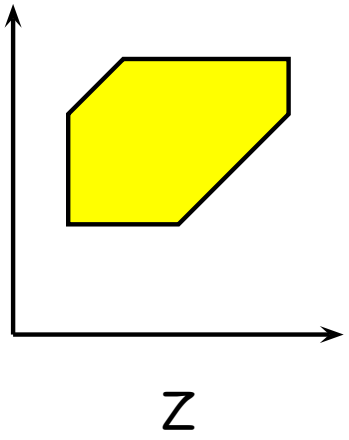


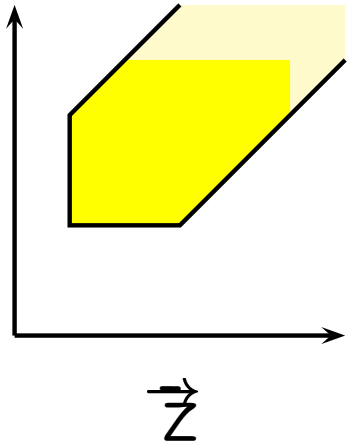


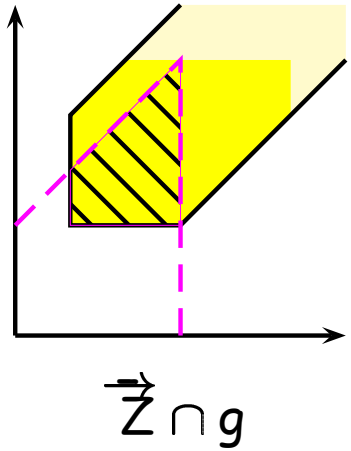
Analyse en avant



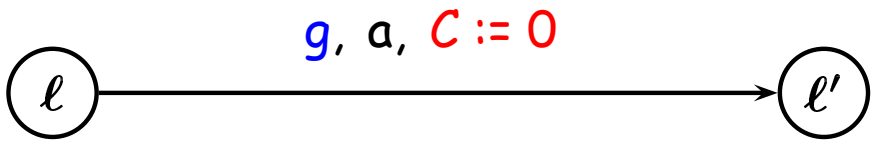
zones Z $[C \leftarrow 0](\overrightarrow{Z} \cap g)$



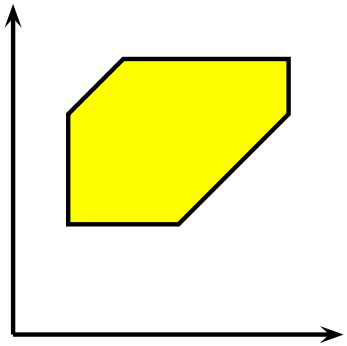


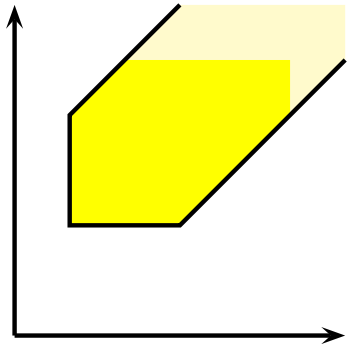


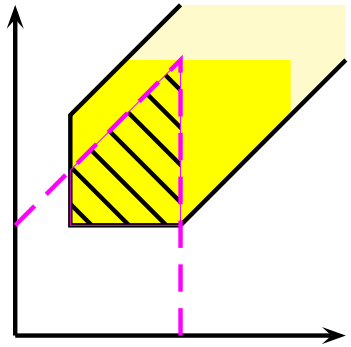
Analyse en avant

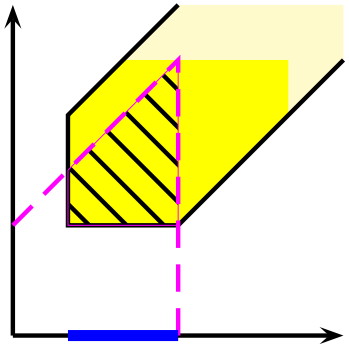


zones Z $[C \leftarrow 0](\overrightarrow{Z} \cap g)$

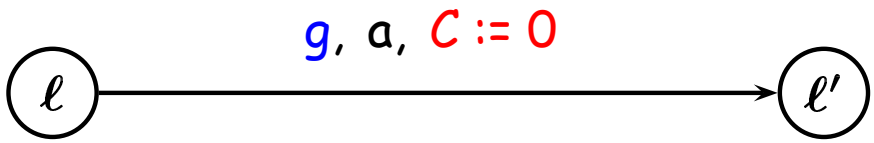




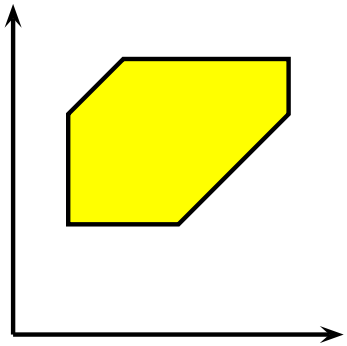


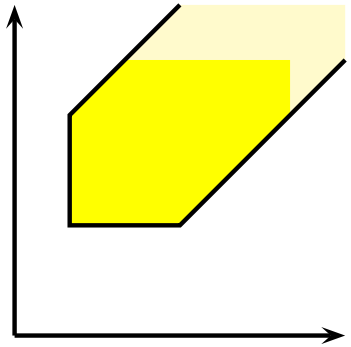


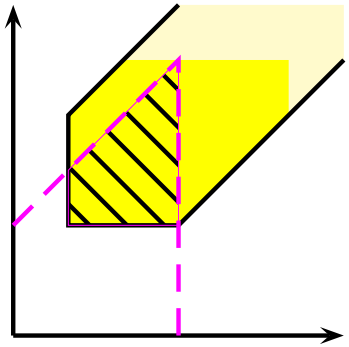
Analyse en avant

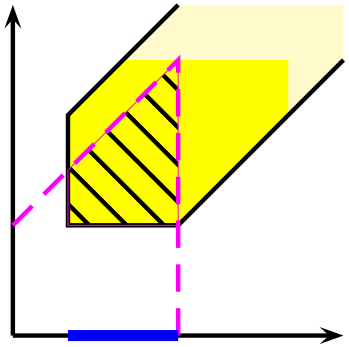


zones Z $[C \leftarrow 0](\overrightarrow{Z} \cap g)$


Z

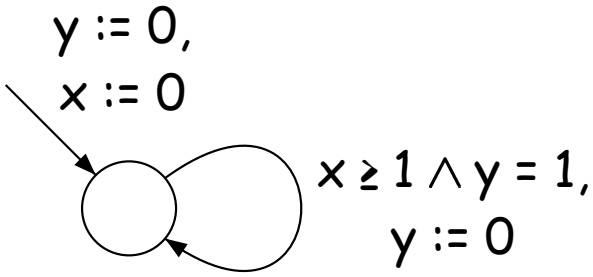

$\overrightarrow{Z}$

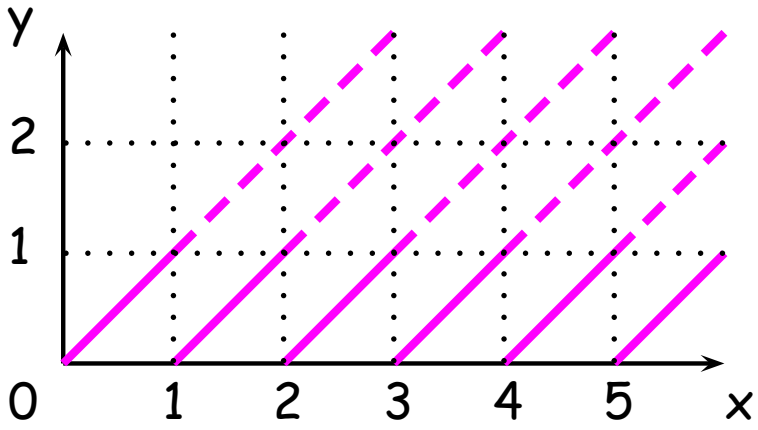

$\overrightarrow{Z} \cap g$


$[y \leftarrow 0](\overrightarrow{Z} \cap g)$

➜ un problème de terminaison

Non-terminaison de l'analyse en avant





➜ une infinité d'étapes...

« Solutions » à ce problème

(par ex. dans [Larsen,Pettersson,Yi 1997] ou dans [Daws,Tripakis 1998])

➜ **test d'inclusion** : si $Z \subseteq Z'$ et Z' déjà traité, alors on peut ne pas traiter Z

➜ correct pour l'accessibilité

...

« Solutions » à ce problème

(par ex. dans [Larsen,Pettersson,Yi 1997] ou dans [Daws,Tripakis 1998])

➜ **test d'inclusion** : si $Z \subseteq Z'$ et Z' déjà traité, alors on peut ne pas traiter Z

➜ correct pour l'accessibilité

➜ **activité**: éliminer les horloges redondantes [Daws,Yovine 1996]

➜ correct pour l'accessibilité

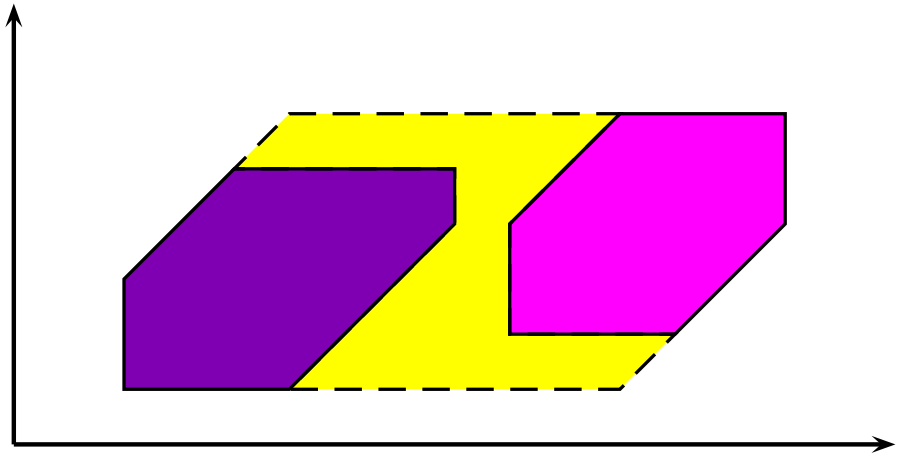
$$q \xrightarrow{g,a,C:=0} q' \Rightarrow Act(q) = horloges(g) \cup (Act(q') \setminus C)$$

...

« Solutions » à ce problème (suite)

➜ **approximation par enveloppe convexe** : si Z et Z' sont calculés, alors on surapproxime en prenant « $Z \sqcup Z'$ ».

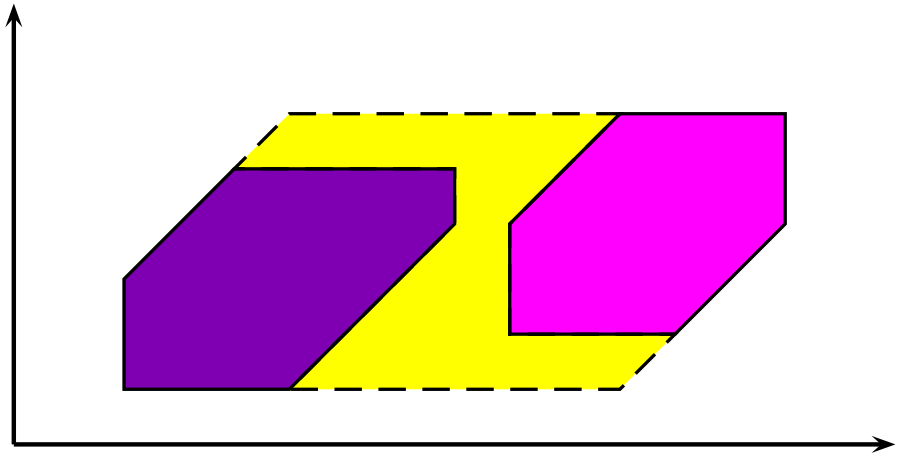
➜ "semi-correct" pour l'accessibilité



« Solutions » à ce problème (suite)

➜ **approximation par enveloppe convexe** : si Z et Z' sont calculés, alors on surapproxime en prenant « $Z \sqcup Z'$ ».

➜ "semi-correct" pour l'accessibilité



➜ **extrapolation**, un opérateur d'élargissement sur les zones

La structure de données DBM

DBM signifie « Difference Bounded Matrice » [Dill 1989]

$$(x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{bmatrix} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{bmatrix} \end{array}$$

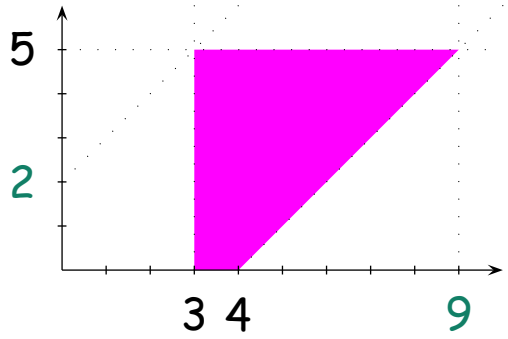
La structure de données DBM

DBM signifie « Difference Bounded Matrice » [Dill 1989]

$$(x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{bmatrix} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{bmatrix} \end{array}$$

➜ Existence d'une forme normale



$$\begin{bmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{bmatrix}$$

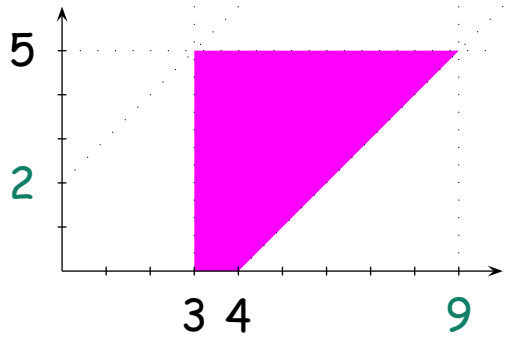
La structure de données DBM

DBM signifie « Difference Bounded Matrice » [Dill 1989]

$$(x_1 \geq 3) \wedge (x_2 \leq 5) \wedge (x_1 - x_2 \leq 4)$$

$$\begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{bmatrix} +\infty & -3 & +\infty \\ +\infty & +\infty & 4 \\ 5 & +\infty & +\infty \end{bmatrix} \end{array}$$

➜ Existence d'une forme normale

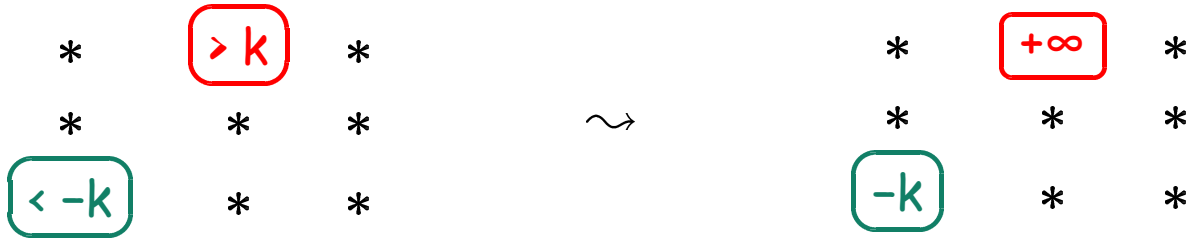


$$\begin{bmatrix} 0 & -3 & 0 \\ 9 & 0 & 4 \\ 5 & 2 & 0 \end{bmatrix}$$

➜ Toutes les opérations sur les zones peuvent être calculées avec les DBMs

L'opérateur d'extrapolation

Fixons un entier k ("$*$" représente un entier entre $-k$ et $+k$)



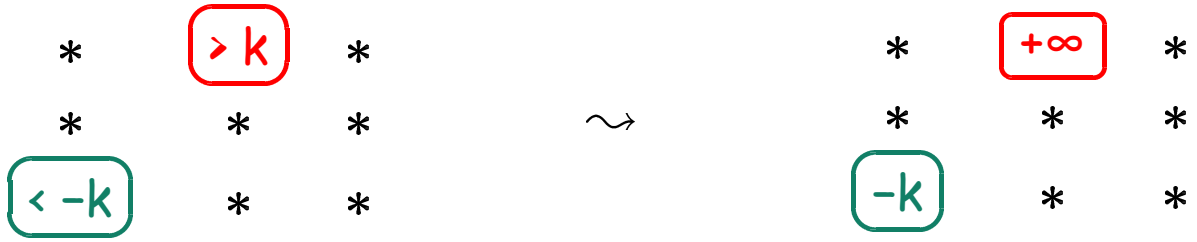
➜ "intuitivement", efface toutes les contraintes qui ne jouent pas un rôle primordial

➜ assure la terminaison

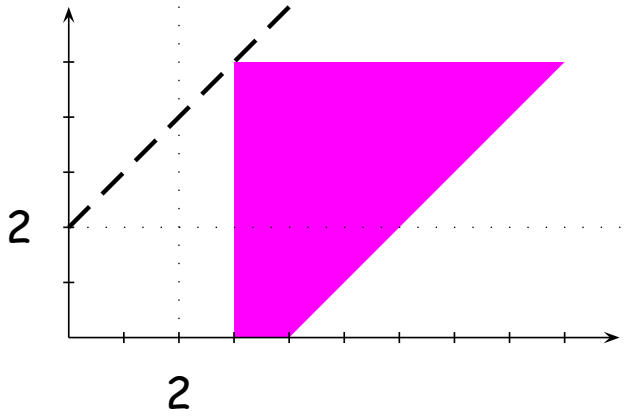
L'opérateur d'extrapolation

Fixons un entier k

("*" représente un entier entre –k et +k)



➜ "intuitivement", efface toutes les contraintes qui ne jouent pas un rôle primordial

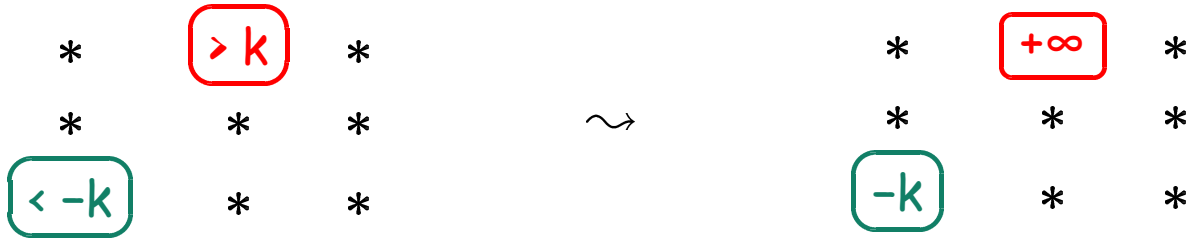


➜ assure la terminaison

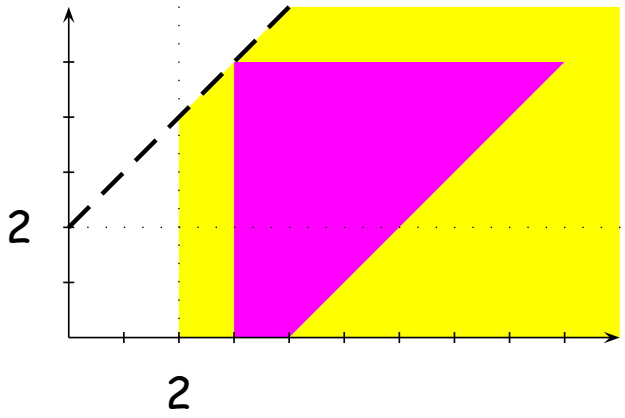
L'opérateur d'extrapolation

Fixons un entier k

("*" représente un entier entre –k et +k)



➜ "intuitivement", efface toutes les contraintes qui ne jouent pas un rôle primordial



➜ assure la terminaison

L'opérateur d'extrapolation (suite)

L'extrapolation est *parfois* une abstraction correcte des zones.

Conditions :

- ➜ Constante d'extrapolation = constante max. de l'automate
- ➜ Uniquement utilisation de gardes non diagonales $x \bowtie c$

Problème ouvert : trouver une abstraction correcte avec gardes diagonales

Et en pratique...

Depuis 1990 ...

- ➜ Des outils de vérifications ont été développés
  - UPPAAL (Uppsala, Suède et Aalborg, Danemark)
  - KRONOS (Grenoble)
  - HYTECH (Berkeley, USA)
  - CMC (Cachan)
- ➜ ...et appliqués avec succès à des études de cas !
  Exemples : Compréhension du bogue du protocole de Bang-Olufsen
  Vérification du protocole ABR.

➜ Des résultats importants mais il reste des problèmes...

Les défis actuels

- Les model checkers temporisés :
  - Explosion combinatoire
  - Représentation symbolique
    - problème DBM
    - mixer discret + continu
  - Abstractions plus rusées
  - Modèles plus adaptés
- Approches probabilistes
- Synthèse de systèmes

Pour en savoir plus

- partout dans le monde (USA, Danemark, Suède, Italie, France, Belgique, Pays-Bas, Inde, Chine...)
- au LSV 

N'hésitez pas à venir nous voir !

Quelques références

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger and Pei-Hsin Ho. Hybrid Automata: an Algorithmic Approach to Specification and Verification of Hybrid Systems. In Proc. Worksh. Hybrid Systems. VOlume 736 of LNCS, pp. 209-229. 1993.

[ACH+92] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, David Dill and Howard Wong-Toi. Minimization of Timed Transition Systems. In Proc. 3rd Int. Conf. Concurrency Theory (CONCUR'92). Volume 630 of LNCS, pp. 340-354. 1992.

[AD94] Rajeev Alur and David Dill. A Theory of Timed Automata. Theoretical Computer Science (TCS). Volume 126(2), pp.183-235. 1994.

[BBF+01] Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci and Philippe Schnoebelen. Model-Checking Techniques and Tools. Springer, 2001.

[BD00] Béatrice Bérard and Catherine Dufourd. Timed Automata with Additive Clock Constraints. Information Processing Letters (IPL). Volume 75(1-2), pp.1-7. 2000.

[BDGP98] Béatrice Bérard, Volker Diekert, Paul Gastin and Antoine Petit. Characterization of the Expressive Power of Silent Transitions in Timed Automata. Fundamenta Informaticae. Volume 36(2-3), pp. 145-182. 1998

Quelques références (suite)

[BDFP00] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury and Antoine Petit. Updatable Timed Automata. In Proc. 12th Int. Conf. Computer-Aided Verification (CAV'00). Volume 1855 of LNCS, pp. 464-479. 2000

[CC77] Patrick Cousot and Radhia Cousot. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In Proc. 4th ACM Symp. Principles of Programming Languages (POPL'77). pp. 238-252. 1977.

[Hen96] Thomas A. Henzinger. The Theory of Hybrid Automata. Proc. 11th IEEE Symp. Logic in Computer Science (LICS'96). pp. 278-292. 1996.

[HJMS02] Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar and Grégoire Sutre. Lazy Abstraction. In Proc. 29th ACM Symp. Principles of Programming Languages (POPL'02). ACM Press, 2002.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri and Pravin Varaiya. What's Decidable about Hybrid Automata? Journal of Computer and System Sciences. Volume 57(1), pp. 84-124. 1998.

[Pnu77] Amir Pnueli. The Temporal Logic of Programs. In Proc. 18th IEEE Symp. Foundations of Computer Science (FOCS'77), pp. 46-57. 1977.

Quelques références (suite)

[TY01] Stavros Tripakis and Sergio Yovine. Analysis of Timed Systems using Time-Abstracting Bisimulations. Formal Methods in System Design. Volume 18(1), pp. 25-68. 2001.