Modélisation des systèmes: les fondements

Dominique Méry

December 16, 2004

Formalisation

- Observation
- Modélisation: données, opérations, modalités
- Validation
- Expérimentation

Systèmes

- systèmes répartis: réseaux, télécommunications . . .
- systèmes critiques: systèmes embarqués . . .
- systèmes tolérants aux fautes: réseaux, télécommunications . . .
- systèmes domestiques: un magnétoscope, une zapette, un téléphone mobile, une machine à laver . . .
- systèmes logiciels/matériels

Démarche du cours

- Développer des modèles réalistes de systèmes.
- Introduire les concepts nécessaires ou les rappeler.
- Expérimenter la démarche dans un outil.
- Gérer au mieux les abstractions et les raffinements.

Présentation

- Utiliser une technique de modélisation aussi générale que possible
- Un système réparti comprend des nœuds ou des sites répartis physiquement
- Chaque site comprend des actions ou des événements qui ont une action locale ou de communication.
- Un système réparti est modélisé par une relation de transition non-déterministe et avec événetuellement de l'équité
- Les systèmes de transitions sont de bons outils pour modélsier de tels systèmes.

Systèmes de transition

- Un ensemble fini ou infini d'états Σ
- Un ensemble d'états initiaux $\mathcal{I} \subseteq \Sigma$
- Une relation de transition sur Σ, notée $\longrightarrow \subseteq \Sigma \times \Sigma$
- $(\Sigma, \mathcal{I}, \longrightarrow)$ désignera un système de transition.

Systèmes de transitions étiquettées

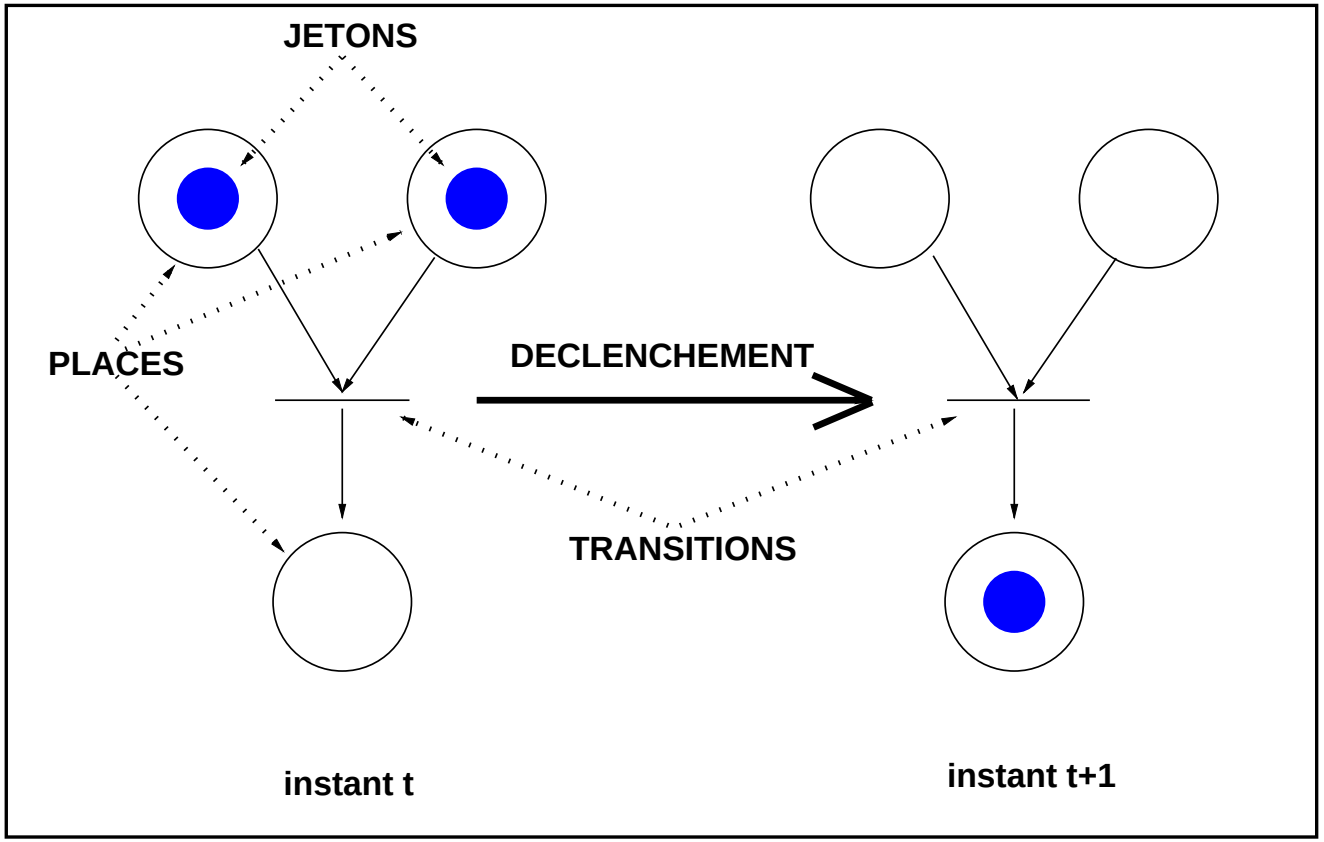
- Un ensemble fini ou infini d'états Σ
- Un ensemble d'états initiaux $\mathcal{I} \subseteq \Sigma$
- Un ensemble d'états terminaux $\mathcal{T} \subseteq \Sigma$
- Un ensemble d'événements $\mathcal{E}$
- Une relation de transition sur Σ, notée $\longrightarrow \subseteq \Sigma \times \mathcal{E} \times \Sigma$
- $(\Sigma, \mathcal{I}, \mathcal{T}, \mathcal{E}, \longrightarrow)$ désignera un système de transition étiquettée.

Exemples de systèmes de transition

- Une grammaire (N, T, P, S) permet de construire un système de transition sur l'ensemble des configurations $(N \cup T)^\star$.
- Une machine de Turing $(Q, \Sigma, \Gamma, B, \delta)$ permet de construire un système de transition sur l'ensemble des configurations $(\Sigma \cup \Gamma)^\star$.
- Un réseau de Petri
- Un programme

Réseaux de Petri

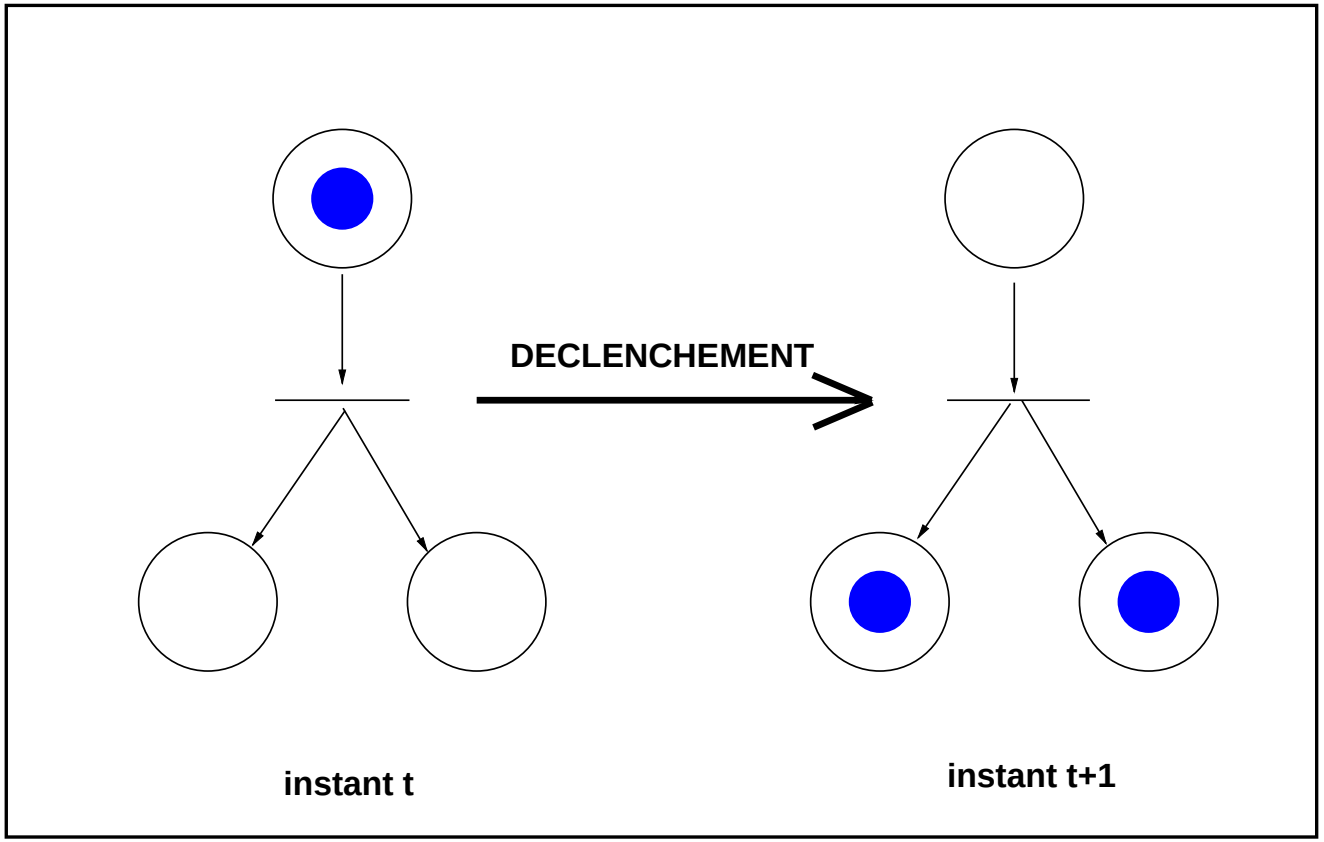
- Un réseau de Petri est un graphe dirigé biparti ayant des jetons constituant la marquage.
- Le réseau est caractérisé par son marquage qui évolue au cours de l'exécution des transitions
- Le déclenchement ou l'activation des transitions est fonction de conditions de ressources sur les places avant la transition et après la transition.


JETONS
PLACES
DECLENCHEMENT
TRANSITIONS
instant t
instant t+1

Réseaux de Petri

- Les transitions peuvent soit consommer des jetons (synchronisation) soit produire de jetons (activités concurrentes):

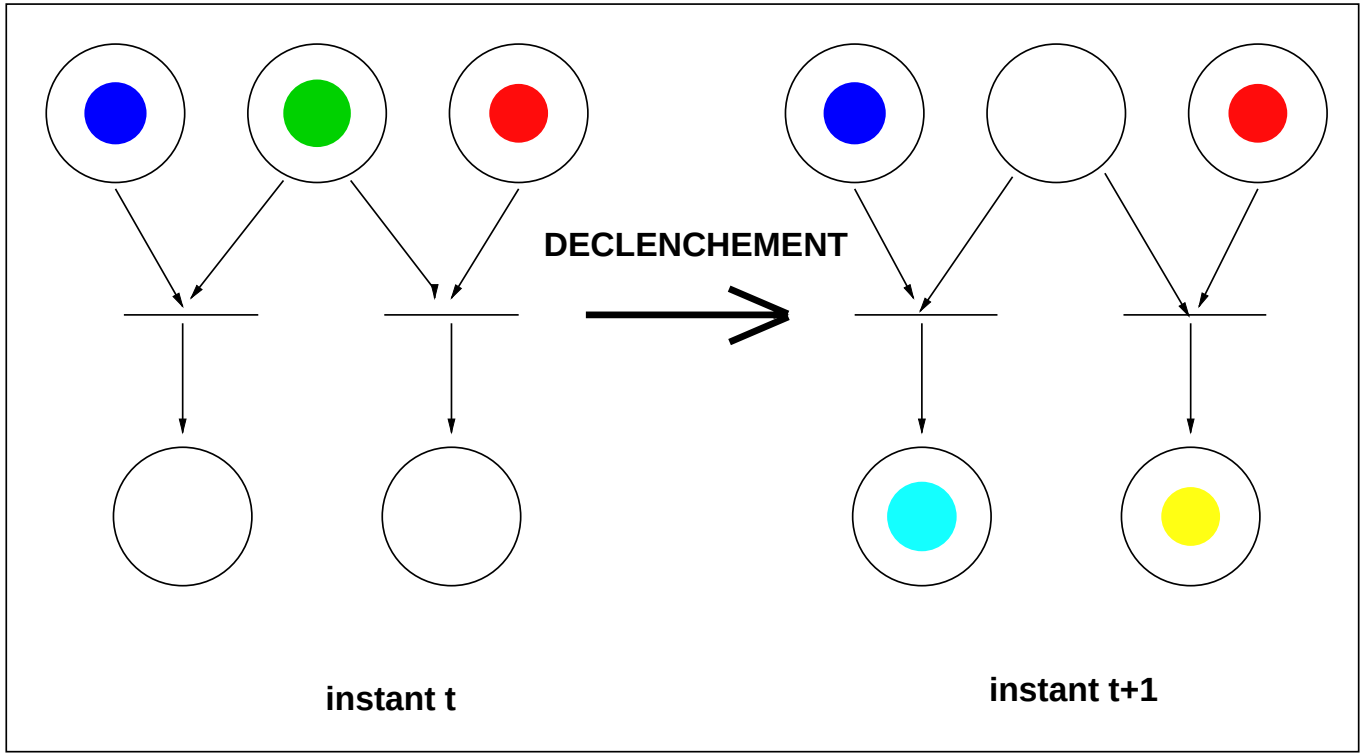
- Les ressources sont modélisées par les jetons présents t il peut y avoir une limitation de la capacité des places.


DECLENCHEMENT
instant t
instant t+1

Réseaux de Petri

- Le partage d'une ressource est modélisé par le partage d'un jeton requis pour l'une ou l'autre des transitions possibles c'est-à-dire activable.

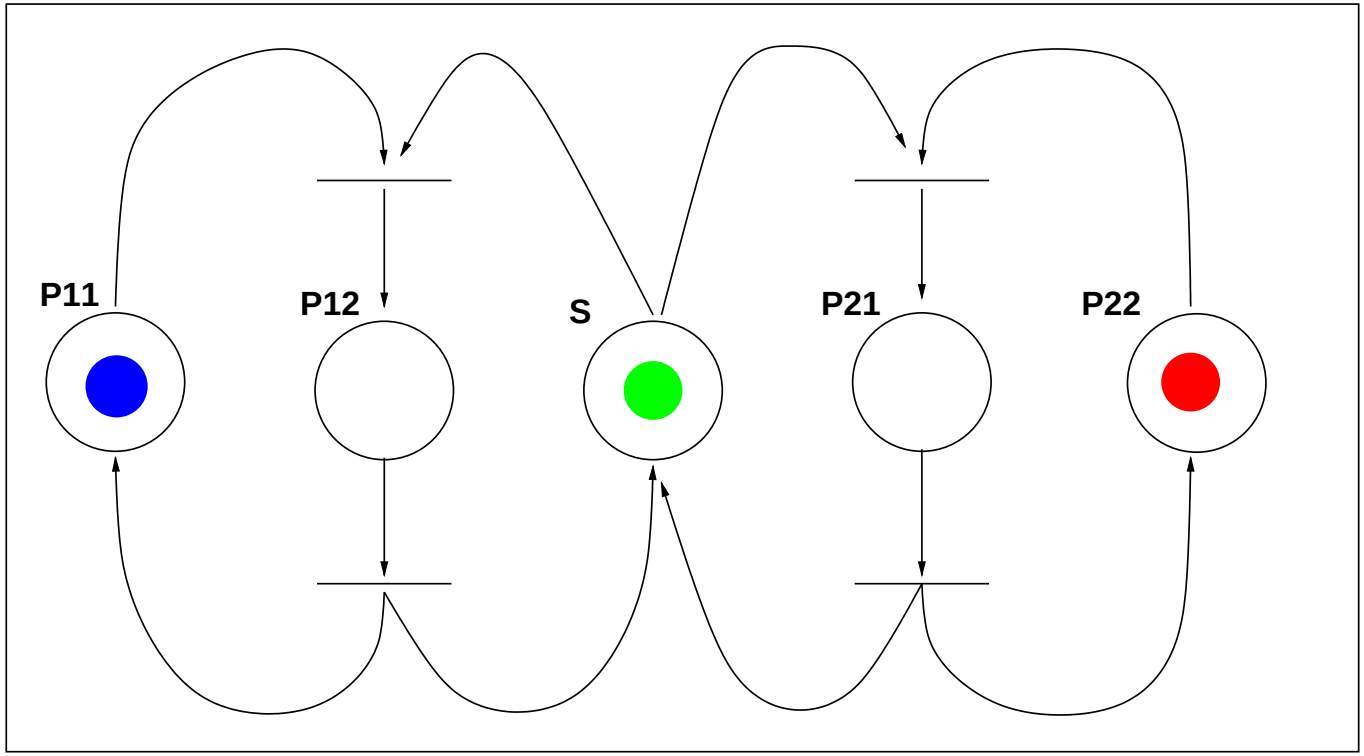
- Le jeton vert est consommé par l'une ou l'autre des deux transitions possibles.


DECLENCHEMENT
instant t
instant t+1

Réseaux de Petri

- La synchronisation de processus est réalisée par une place S qui est partagée par deux processus P1 et P2:

- La propriété d'exclusion mutuelle est garantie par l'utilisation exclusive du jeton de la place S par les processus P1 et P2.

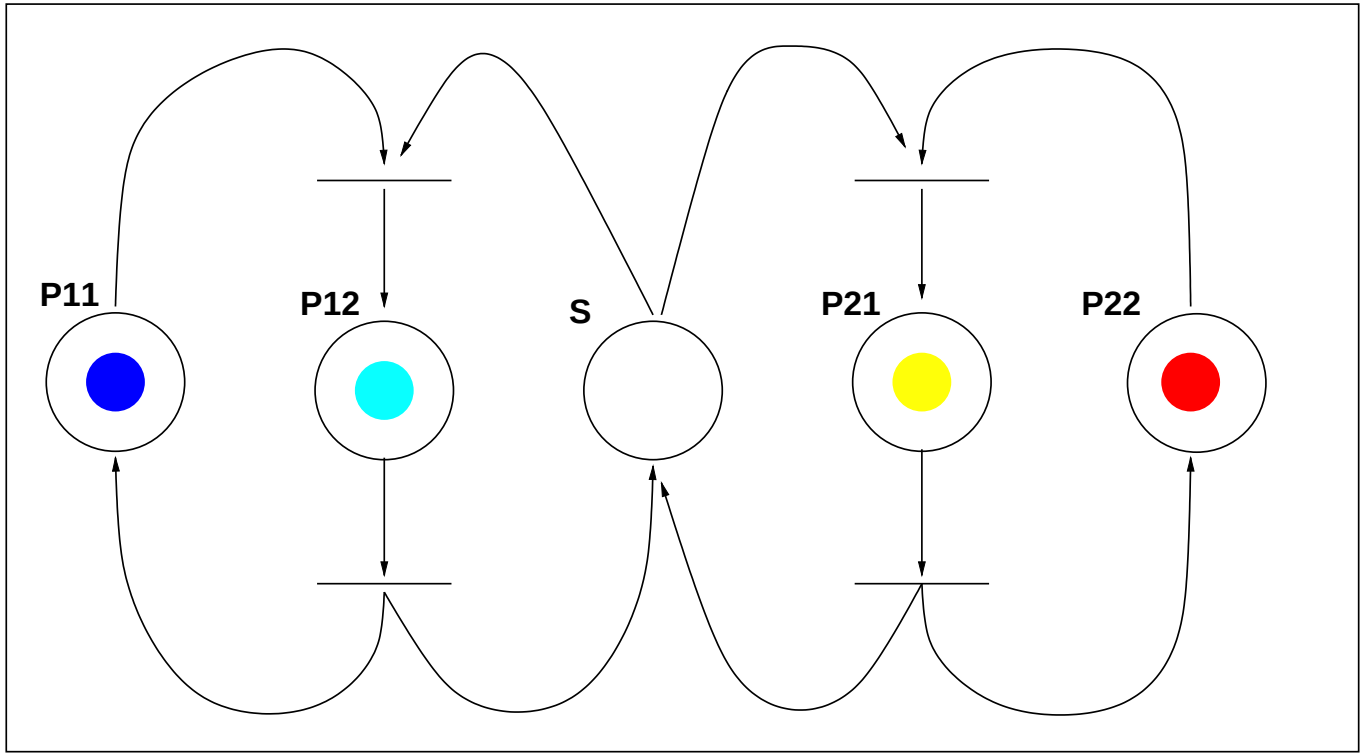

P11
P12
S
P21
P22

Réseaux de Petri

- Le déclenchement de l'une des deux transitions est possible quand le jeton vert est en place mais une seule est activée.

- Les réseaux de Petri (1962) ont été créés par **Carl Adam Petri** (avec un C et pas un K) et ont été largement utilisés par la communauté informatique et automatique.

- Des extensions ont été proposées notamment en colorant les jetons ou en ajoutant des probablités aux transitions.


P11
P12
S
P21
P22

Réseaux de Petri

Un réseau de Petri est un uple R=(S,T,F,K,M,W)

- S est l'ensemble (fini) des places.
- T est l'ensemble (fini) des transitions.
- $S \cap T = \emptyset$
- F est la relation du flôt d'exécution:

  $F \subseteq S \times T \cup T \times S$

- K représente la capacité de chaque place:

  $K \in S \rightarrow Nat \cup \{\omega\}$

Réseaux de Petri

- M représente le initial marquage chaque place:

  $M \in S \rightarrow Nat \cup \{\omega\}$ et vérifie la condition $\forall\ s \in S : M(s) \leq K(s)$.

- W représente le poids de chaque arc:

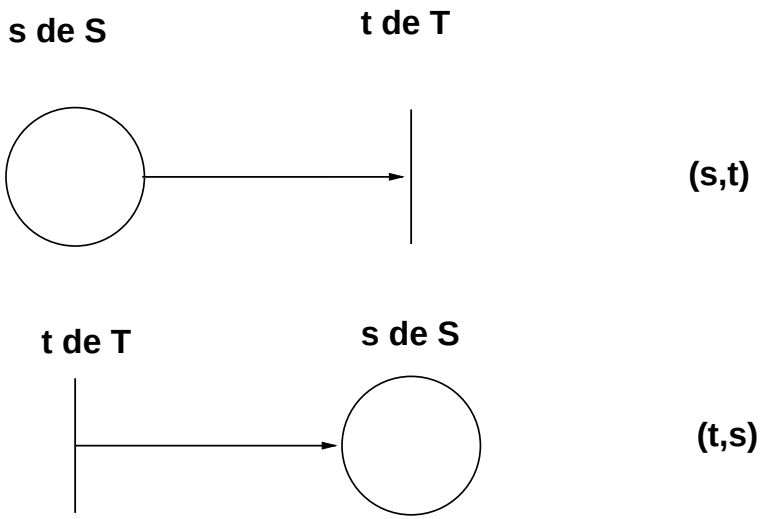
  $W \in F \rightarrow Nat \cup \{\omega\}$

- relation entre la représentation graphique et la définition textuelle:

- un marquage M pour R est une fonction de S dans $Nat \cup \{\omega\}$:

  $M \in S \rightarrow Nat \cup \{\omega\}$

  et respectant la condition $\forall\ s \in S : M(s) \leq K(s)$.


s de S
t de T
(s,t)
t de T
s de S
(t,s)

Réseaux de Petri

- une transition t de T est activable à partir de M un marquage de R si

  1. $\forall$ s $\in$ { s' $\in$ S | (s',t) $\in$ F } :

     M(s) $\geq$ W(s,t).

  2. $\forall$ s $\in$ { s' $\in$ S | (t,s') $\in$ F } :

     M(s) $\leq$ K(s) - W(s,t).

- Pour chaque transition t de T, Pre(t) est l'ensemble des places conduisant à t et Post(t) est l'ensemble des places pointées par un lien depuis t:

  Pre(t) = { s' $\in$ S : (s',t) $\in$ F }

  Post(t) = { s' $\in$ S : (t,s') $\in$ F }

Réseaux de Petri

- Soit une transition t de T activable à partir de M un marquage de R :

  1. $\forall$ s $\in$ { s' $\in$ S | (s',t) $\in$ F } :

     M(s) $\geq$ W(s,t).

  2. $\forall$ s $\in$ { s' $\in$ S | (t,s') $\in$ F } :

     M(s) $\leq$ K(s) - W(s,t).

- un nouveau marquage M' est défini à partir de M par: ' $\forall$ s $\in$ S,

$$M'(s) = \begin{cases} M(S) - W(S,T), \text{ SI } S \in PRE(T) - POST(T) \\ M(S) + W(T,S), \text{ SI } S \in POST(T) - PRE(T) \\ M(S) - W(S,T) + W(T,S), \text{ SI } S \in PRE(T) \cap POST(T) \\ M(S), \text{ SINON} \end{cases}$$

Réseaux de Petri

- Une relation de transition sur l'ensemble des marquages possibles modélise l'activité du réseau :

$$M_0 \xrightarrow{T_0} M_1 \xrightarrow{T_1} M_2 \xrightarrow{T_2} M_3 \xrightarrow{T_3} M_4 \xrightarrow{T_4} \ldots M_l \xrightarrow{T_l} M_{l+1} \xrightarrow{T_{l+1}} \ldots$$

- Un réseau est bloqué, si aucune de ses transitions n'est activable.

- Un réseau est non bloqué en permanence ou vif, si initialement et pour tout marquage atteint au cours du calcul, au moins une transition est activable.

Réseaux de Petri

- Un invariant de marquage pour un réseau de Petri est un expression de la forme suivante:

  $\exists$ p_1,... $p_n \in$ P:

  $\exists$ q_1,... $q_n \in$ Entiers:

  $\exists$ C $\in$ Entiers:

  $\forall$ M $\in \mathcal{M}$: q_1 M(p_1) + q_n M(p_n) = C

- Les réseaux de Petri sont aussi représentés à l'aide de matrices pour leur flôt et cela défintion une algèbre sur les réseaux de Petri: $M_K = M_I + W . S$ est l'équation fondamentale.

- Les réseaux de Petri permettent d'exprimer des contraintes de synchronisation

Réseaux de Petri

- Le modèle est aussi puissant que les machines de Turing
- Le modèle permet de modéliser les activités concurrentes et non déterministes.
- Le Graphcet est une forme proche des réseaux de Petri et est utilisé pour la modélisation des systèmes.
- La notion sous-jacente est celle des systèmes de transition discrets.

Variables flexibles

- Une variable flexible x est caractérisée par plusieurs valeurs:
  - sa valeur initiale ix ou $\underline{x}$.
  - sa valeur courante vx.
  - sa valeur courante vx'.
  - sa valeur finale fx ou $\overline{x}$.
- Une variable flexible x admet des valeurs observées à des instants sucessifs: $x_0 \ x_1 \ x_2 \ x_3 \ \ldots \ x_i \ \ldots$
- Un état s est une fonction associant à toute variable flexible x une valeur $s(x)$: l'état constitue l'instant d'observation.
- $\Sigma \triangleq Var \longrightarrow Valeurs$.

Actions

- Une observation de la variable x à différents instants conduit à mettre en évidence des actions ou des événements:

$$x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} x_2 \xrightarrow{\tau} x_3 \xrightarrow{\tau} \ldots \xrightarrow{\tau} x_i \xrightarrow{\tau} x_{i+1} \xrightarrow{\tau} \ldots$$

Actions

- Une observation de la variable x à différents instants conduit à mettre en évidence des actions ou des événements:

$$x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} x_2 \xrightarrow{\tau} x_3 \xrightarrow{\tau} \ldots \xrightarrow{\tau} x_i \xrightarrow{\tau} x_{i+1} \xrightarrow{\tau} \ldots$$

- τ cache des actions effectives:

$$x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} x_2 \xrightarrow{\alpha_3} x_3 \xrightarrow{\alpha_4} \ldots \xrightarrow{\alpha_i} x_i \xrightarrow{\alpha_{i+1}} x_{i+1} \xrightarrow{\alpha_{i+2}} \ldots$$

Actions

- Une observation de la variable x à différents instants conduit à mettre en évidence des actions ou des événements:

  $$x_0 \stackrel{\tau}{\rightarrow} x_1 \stackrel{\tau}{\rightarrow} x_2 \stackrel{\tau}{\rightarrow} x_3 \stackrel{\tau}{\rightarrow} \ldots \stackrel{\tau}{\rightarrow} x_i \stackrel{\tau}{\rightarrow} x_{i+1} \stackrel{\tau}{\rightarrow} \ldots$$

- τ cache des actions effectives:

  $$x_0 \stackrel{\alpha_1}{\rightarrow} x_1 \stackrel{\alpha_2}{\rightarrow} x_2 \stackrel{\alpha_3}{\rightarrow} x_3 \stackrel{\alpha_4}{\rightarrow} \ldots \stackrel{\alpha_i}{\rightarrow} x_i \stackrel{\alpha_{i+1}}{\rightarrow} x_{i+1} \stackrel{\alpha_{i+2}}{\rightarrow} \ldots$$

- Des occurences de τ peuvent être intercalées et conduisent à des bégaiements:

Actions

- Une observation de la variable x à différents instants conduit à mettre en évidence des actions ou des événements:

$$x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} x_2 \xrightarrow{\tau} x_3 \xrightarrow{\tau} \ldots \xrightarrow{\tau} x_i \xrightarrow{\tau} x_{i+1} \xrightarrow{\tau} \ldots$$

- τ cache des actions effectives:

$$x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} x_2 \xrightarrow{\alpha_3} x_3 \xrightarrow{\alpha_4} \ldots \xrightarrow{\alpha_i} x_i \xrightarrow{\alpha_{i+1}} x_{i+1} \xrightarrow{\alpha_{i+2}} \ldots$$

- Des occurences de τ peuvent être intercalées et conduisent à des bégaiements:

$$x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} x_2 \xrightarrow{\tau} x_2 \xrightarrow{\alpha_3} x_3 \xrightarrow{\alpha_4} \ldots \xrightarrow{\alpha_i} x_i \xrightarrow{\tau} x_i \xrightarrow{\alpha_{i+1}} x_{i+1} \xrightarrow{\alpha_{i+2}} \ldots$$

Actions comme Relations

- Une action α est une relation entre deux états s et s' :

$$s \xrightarrow{\alpha} s'$$

- Une action peut être modélisée sous la forme d'une relation entre l'état des variables **avant** et l'état des variables **après**:

$$\rho[\alpha)(x, x')$$

- x est la valeur de x en s et x' est la valeur de x en s'.

Primer une expression

- Une variable flexible x est primable et x' désigne la valeur suivante.
- Toute constante est non primable: sa valeur est constante.
- Primer une expression est réalisé par induction sur la syntaxe:

  $(x+6-y)'$ est traduit en $x'+6-y'$

  où x et y sont deux variables.

Propriété d'état d'un système

- Une propriété d'état d'un système P est interprétée sur l'ensemble des états Σ et l'expression $\mathsf{P}(\sigma)$ où $\sigma \in \Sigma$ signifie que P est vraie en σ.

- $\mathsf{P}(\mathsf{x}_1, \ldots, \mathsf{x}_N)(\sigma) \triangleq \mathbf{P}(\sigma(\mathsf{x}_1), \ldots, \sigma(\mathsf{x}_N))$ où x_I est une variable flexible du système.

- Un état d'un système σ est une fonction partielle de **Variables** dans **Valeurs**:

$$\Sigma \triangleq \textbf{Variables} \longrightarrow \textbf{Valeurs}$$

Propriétés de sûreté

- Exclusion mutuelle: soit une ressource R partagée par un ensemble de processus $\{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. R est utilisée par au plus un processus de $\{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$.

- Absence de blocage: soit les processus $\{\mathsf{P}_1, \ldots, \mathsf{P}_n\}$. Aucun des processus n'est bloqué c'est à dire que tout processus peut toujours exécuté une action sauf s'il est terminé.

- Correction Partielle: étant donné un processus de calcul caractérisé par un ensemble d'actions ou d'événements. Si les variables satisfont une précondition $\mathsf{PRE}(x)$, alors si le processus termine, les variables satisfont $\mathsf{POST}(x)$.

- Une propriété de sûreté exprime que rien de mauvais ne peut arriver!

Invariance et sûreté: sémantique

- Une propriété $P(x_1, \ldots, x_N)$ d'un système $\mathcal{S}$ sur l'ensemble de variables $\{x_1, \ldots,$ est inductivement invariante pour $\mathcal{S}$, si

$$\begin{cases} \text{INIT} \Rightarrow P \\ \forall \sigma, \sigma' \in \Sigma : P(\sigma) \wedge \sigma \longrightarrow \sigma' \Rightarrow P(\sigma') \end{cases}$$

- Une propriété $P(x_1, \ldots, x_N)$ d'un système $\mathcal{S}$ sur l'ensemble de variables $\{x_1, \ldots,$ est une propriété de sûreté pour $\mathcal{S}$, si

$$\forall \sigma, \sigma' \in \Sigma : \text{INIT}(\sigma) \wedge \sigma \stackrel{\star}{\longrightarrow} \sigma' \Rightarrow P(\sigma')$$

Vérification sémantique d'une propriété d'état d'un système

- Une propriété de sûreté pour $\mathcal{S}$, $\mathsf{P}(x_1, \ldots, x_N)$, est une propriété valide sémantiquement pour $\mathcal{S}$, si

  $\forall \sigma, \sigma' \in \Sigma : \textsc{Init}(\sigma) \wedge \sigma \xrightarrow{\star} \sigma' \Rightarrow \mathsf{P}(\sigma')$

- Si l'ensemble des états ou des configurations Σ de $\mathcal{S}$ est fini, alors le graphe constitué des états comme sommets et de la relation de transition comme arcs permet de vérifier exhaustivement tous les états accessibles à partir des états initiaux et de vérifier pour chaque état rencontré si la propriété courante est vraie ou fausse pour l'état visité.

- Si l'ensemble des états ou des configurations Σ de $\mathcal{S}$ est fini mais très grand (plusieurs millions d'états), alors ,

Vérification d'une propriété de sûreté

- $\forall \sigma, \sigma' \in \Sigma : \textsc{Init}(\sigma) \wedge \sigma \xrightarrow{\star} \sigma' \Rightarrow \mathsf{P}(\sigma')$
- Hypothèse 1 : Σ est un ensemble fini
- Méthode exhaustive de vérification :
  - Représenter Σ.
  - Vérifier que chaque état accessible du graphe satisfait P
- Σ. doit être de taille raisonnable.
- Vérification des modèles : Model Checking
- Si Σ. n'est pas fini, alors il faut trouver un modèle abstrait suffisant pour appliquer la méthode dans le cas finie.

Modélisation avec une relation

- Le système est modélisé par
  - une liste de variables x et une condition initiale notée $init(x)$
  - une relation de transition modélisant le passage des variabables flexibles de l'état courant à l'état suivant $next(x, x')$
  - éventuellement un invariant noté $I(x)$
  - éventuellement une liste de propriétés de sûreté
- La conception de ces éléments est réalisée à partir du cahier des charges selon une méthode que nous ne détaillons pas encore
- Un tel système est observé par des suites de valeurs des variables x à différents instants d'observation.

$$x_0 \overset{\alpha_1}{\rightarrow} x_1 \overset{\alpha_2}{\rightarrow} x_2 \overset{\tau}{\rightarrow} x_2 \overset{\alpha_3}{\rightarrow} x_3 \overset{\alpha_4}{\rightarrow} \ldots \overset{\alpha_i}{\rightarrow} x_i \overset{\tau}{\rightarrow} x_i \overset{\alpha_{i+1}}{\rightarrow} x_{i+1} \overset{\alpha_{i+2}}{\rightarrow} \ldots$$

Définir un protocole simple avec TLA$^+$

- Envoi et réception de messages

- Modélisation par des ensembles de messages envoyés ou reçus.

$$
\begin{array}{l}
sending(agent, message, bgent) \triangleq \\
\quad \wedge agent \in AGENTS \\
\quad \wedge bgent \in AGENTS \\
\quad \wedge message \in MESSAGES \\
\quad \wedge \neg(\langle agent, message, bgent\rangle \in sent) \\
\quad \wedge sent' \\
\quad sent \cup \{\langle agent, message, bgent\rangle\} \\
\\
\quad \wedge got' = got
\end{array}
\quad = \quad
\begin{array}{l}
receiving(agent, message, bgent) \triangleq \\
\quad \wedge agent \in AGENTS \\
\quad \wedge bgent \in AGENTS \\
\quad \wedge message \in MESSAGES \\
\quad \wedge \langle agent, message, bgent\rangle \in sent \\
\quad \wedge \langle agent, message, bgent\rangle \notin got \\
\quad \wedge got' \\
\quad got \cup \{\langle agent, message, bgent\rangle\} \\
\quad \wedge sent' = sent
\end{array}
$$

Expression complète

- Définir les données
- Définir les actions
- Définir le système
- Définir les propriétés

```
----------------MODULE simpleprotocol --------------------
CONSTANTS
      AGENTS,MESSAGES
VARIABLES
      sent,got
```

Expression complète

```
sending(agent,message,bgent) ==
   /\ agent \in AGENTS
   /\ bgent \in AGENTS
   /\ message \in MESSAGES
   /\ ~(<<agent,message,bgent>> \in  sent)
   /\ sent’ = sent \cup  {<<agent,message,bgent>>}
   /\ got’ = got

receiving(agent,message ,bgent) ==
   /\ agent \in AGENTS
   /\ bgent \in AGENTS
   /\ message \in MESSAGES
   /\ <<agent,message,bgent>> \in sent
   /\ <<agent,message,bgent>> \notin got
   /\ got’ = got \cup  {<<agent,message,bgent>>}
   /\ sent’ = sent

skip ==  got’ = got  /\ sent’ = sent
```

Expression complète

```
next == \E agent \in AGENTS :  \E bgent \in AGENTS :
          \E  message \in MESSAGES :
             \/ sending(agent,message,bgent)
             \/ receiving(agent,message,bgent)
             \/ skip

invariant ==  \A agent \in AGENTS :  \A bgent \in AGENTS :
                \A  message \in MESSAGES :
                   <<agent,message,bgent>> \in  got
                     =>  <<agent,message,bgent>> \in  sent

invariant == invariant1
init == sent = {} /\ got = {}
simpleprotocol == init /\ [next]_<<sent,got>>

THEOREM   simpleprotocol => []invariant
=============================================================
```

Validation par une configuration finie

- Définir un modèle de configuration:

```
CONSTANTS
   AGENTS = {"007","xx","xy"}
   MESSAGES = {"salut","allo","ah"}

INIT init
NEXT next

INVARIANTS
    invariant
```

- Engendrer tous les modèles finis associés à la configuration choisie et tester les propriétés comme les invariants et les propriétés de sûreté.

TLC: Outil d’analyse des modèles temporels

1. Ecrire un modèle fini du système dans un fichier préfixe .cfg.

2. Positionner les valeurs suivantes:

   (a) `setenv CLASSPATH <directory de TLC>`

   (b) `java tlatk.TLC -config simpleprotocol.cfg simpleprotocol.`

Modélisation de style impérative

- Le modèle obtenu est validé par l'outil TLC qui analyse les modèles finis: insuffisant pour garantir une absence complète de fautes.
- Une analyse fondée sur une approche symbolique couplée avec un prouveur est plus prometteuse.
- Une telle approche comporte des inconvénients dont l'un est l'emploi d'un prouveur.
- Objectif: diffuser la preuve par raffinement
- Reprise de notre exemple.

Une modélisation de style plus impérative sur les ensembles

SETS

$MESSAGES; AGENTS$

PROPERTIES

$MESSAGES \neq \emptyset \wedge$

$AGENTS \neq \emptyset$

VARIABLES

$sent, got, lost$

Une modélisation de style plus impérative sur les ensembles

INVARIANT

$$sent \subseteq AGENTS \times AGENTS \times MESSAGES \wedge$$

$$got \subseteq AGENTS \times AGENTS \times MESSAGES \wedge$$

$$lost \subseteq AGENTS \times AGENTS \times MESSAGES \wedge$$

$$(got \cup lost) \subseteq sent \wedge$$

$$lost = \emptyset$$

INITIALISATION

$$sent := \emptyset \parallel$$

$$got := \emptyset \parallel$$

$$lost := \emptyset$$

Une modélisation de style plus impérative sur les ensembles

SENDING $=$

any $agent1, agent2,$ *message* **where**

$agent1 \in$ *AGENTS* $\wedge$

$agent2 \in$ *AGENTS* $\wedge$

message $\in$ *MESSAGES* $\wedge$

$agent1 \mapsto agent2 \mapsto$ *message* $\notin$ *sent*

then

sent $:=$ *sent* $\cup \{agent1 \mapsto agent2 \mapsto$ *message*$\}$

end;

RECEIVING $=$ **any** $agent1, agent2,$ *message* **where**

$agent1 \in$ *AGENTS* $\wedge$

$agent2 \in$ *AGENTS* $\wedge$

message $\in$ *MESSAGES* $\wedge$

$agent1 \mapsto agent2 \mapsto$ *message* $\in$ *sent* $\wedge$

$agent1 \mapsto agent2 \mapsto$ *message* $\notin$ *got*

then

got $:=$ *got* $\cup \{agent1 \mapsto agent2 \mapsto$ *message*$\}$

end;

LOOSING $=$ **begin** SKIP **end**

Une modélisation de style plus impérative sur les ensembles

- La déclaration des ensembles n'exige pas de précision sur la taille de ces ensembles
- L'invariant est clairement exprimé mais doit être préservé par les opérations.
- Les conditiions de préservation sont fondées sur un principe d'induction.

Méthode de preuve de propriétés de sûreté

(I) $\forall \sigma, \sigma' \in \Sigma : \textsc{Initial}(\sigma) \wedge \sigma \xrightarrow{\star} \sigma' \Rightarrow \mathsf{P}(\sigma')$

(II) Il existe une propriété invariante pour $\mathcal{S}$, $\mathcal{I}$, telle que

$$\begin{cases} \textsc{Init} \Rightarrow \mathcal{I} \\ \forall \sigma, \sigma' \in \Sigma : \mathcal{I}(\sigma) \wedge \sigma \longrightarrow \sigma' \Rightarrow \mathcal{I}(\sigma') \\ \mathcal{I} \Rightarrow \mathsf{P} \end{cases}$$

- ◇ Correction de la méthode : (II) $\Rightarrow$ (I)
- ◇ Complétude de la méthode : (I) $\Rightarrow$ (II)
- ◇ Trouver $\mathcal{I}$ est une question critique et indécidable de cette méthode et I doit être écrit par le concepteur du système.

Méthode de preuve de propriétés de sûreté

- La propriété $\mathcal{I}$ n'est pas toujours simple à trouver et peut ne pas être exprimable dans le langage retenu.

- Soit P une propriété de sûreté pour $\mathcal{S}$ et un processus de calcul modélisé par une relation entre les valeurs avant et après la mise à jour $\mathcal{N}(\mathrm{x}, \mathrm{x'})$. On note DATA l'ensemble des données ou des valeurs du système.

  Il existe une propriété invariante pour $\mathcal{S}$, $\mathcal{I}$, telle que

$$\begin{cases} \textsc{Init} \Rightarrow \mathcal{I} \\ \forall \mathrm{x}, \mathrm{x'} \in \mathrm{DATA} : \mathcal{I}(\mathrm{x}) \wedge \mathcal{N}(\mathrm{x}, \mathrm{x'}) \Rightarrow \mathcal{I}(\mathrm{x'}) \\ \mathcal{I} \Rightarrow \mathrm{P} \end{cases}$$

Evénements

Un événement est de l'une des formes générales suivantes:

$nom \widehat{=}$
begin
$substitution\ gnralise$
end

$nom \widehat{=}$
select
$condition$
then
$substitution\ gnralise$
end

$nom \widehat{=}$
any $variables$ **where**
$condition$
then
$substitution\ gnralise$
end

Relation before-after notée $BA(x, x')$

.

Evénement	Predicat Before-After
begin $x : P(x_0, x)$ **end**	$P(x, x')$
select $G(x)$ **then** $x : Q(x_0, x)$ **end**	$G(x) \ \wedge \ Q(x, x')$
any t **where** $G(t, x)$ **then** $x : R(x_0, x, t)$ **end**	$\exists\, t \cdot \ (\ G(t, x) \ \wedge \ R(x, x', t) \ )$

Garde

.

Evénement: E	Garde: grd (E)
begin S **end**	true
select $G(x)$ **then** T **end**	$G(x)$
any t **where** $G(t,x)$ **then** U **end**	$\exists\, t \cdot G(t,x)$

Conditions de vérification

- Faisabilité des substitutions généralisées

$$I(x) \ \wedge \ \mathsf{grd}\,(E) \ \Rightarrow \ \exists x' \cdot P(x, x')$$

- Invariant d'un modèle

$$I(x) \ \wedge \ P(x, x') \Rightarrow I(x')$$

Conclusions

- Deux techniques de validation:
  - validation sur des modèles finis avec un outil d'analyse exhaustive TLC
  - validation par preuve avec un prouveur
- Nécessité d'une méthodologie de conception de modèles formels
- Concevoir graduellement un modèle formel de l'abstrait au concret