Formalismes de Description de Systèmes

Alain Griffault
Université Bordeaux I

Frédéric Herbreteau
Enseirb

21 octobre 2003

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Le besoin de formalismes

Montrer le respect du cahier des charges.
Dialoguer sans ambiguité.
Comparer des solutions.
Abstraire un système.

1.2 À quoi sert la formalisation des systèmes ?

systèmes critiques et ou complexes.
historiques (industriels et recherche).

1.3 Comment comparer les formalismes

theorique : expressivite,
pratique : efficacité, adéquation avec les besoins utilisateurs, outillage.

1.4 Le contenu de ce cours

Approche intuitive des besoins pour la représentation des systèmes.
Les systèmes à contrôle fini.
Les systèmes à contrôle infini.
Les langages de description.
À chaque fois, définition, exemples, proprietés, utilisation.

Chapitre 2

Modélisation de systèmes

2.1 Étude de programmes parallèles

2.1.1 Communication par variables partagées

Soient les deux processus suivant et la variable globale `compte` initialisée à 10.

```
Processus A
var aux : integer

begin
aux := compte ;
aux := aux + 10 ;
compte := aux ;
end.
```

```
Processus B
var aux : integer

begin
aux := compte ;
aux := aux - 5 ;
compte := aux ;
end.
```

Question 2.1 *Quelles sont les valeurs possibles de la variable* `compte` *après les exécutions des deux processus A et B ?*

Question 2.2 *Cherchez une méthode simple qui permette aisément de montrer que votre réponse à la question précédente est correcte ?*

2.1.2 Communication par messages

Soient les deux processus suivant :

```
Processus P(i) /* i de 0 a 1 */
begin
while true do
        begin
        F(i) ! prendre ;
        F(1-i) ! prendre ;
        F(1-i) ! poser ;
        F(i) ! poser ;
        end ;
end.
```

```
Processus F(i) /* i de 0 a 1 */
begin
while true do
        begin
        P(0) ? prendre → P(0) ? poser ;
        or
        P(1) ? prendre → P(1) ? poser ;
        end ;
end.
```

Question 2.3 *Appliquez la méthode précédente pour mettre en évidence le problème principal de ce programme.*

2.2 Analyse de la méthode

Essayez de répondre aux questions suivantes :

Question 2.4 *A quoi correspondent les sommets d'un tel graphe ?*

Question 2.5 *A quoi correspondent les arrêtes d'un tel graphe ?*

Question 2.6 *A quoi correspond un sommet duquel part aucune transition ?*

Première partie

Les systèmes à contrôle fini

Chapitre 3

Les systèmes réactifs discrets

3.1 Quelques définitions

Definition 3.1 *Un système de transition est un triplet* $S = < Q, I, T >$ *où :*
- Q *est un ensemble d'états.*
- I *est le sous-ensemble des états* initiaux.
- $T \subseteq Q \times Q$ *est un ensemble de transitions.*

Parmi les variantes des systèmes de transition on distingue :

Definition 3.2 *Un système de transition* **étiqueté** *est un quadruplet* $S = < Q, I, A, T >$ *où :*
- Q *est un ensemble d'états.*
- I *est le sous-ensemble des états* initiaux.
- A **est un ensemble d'actions ou alphabet.**
- $T \subseteq Q \times A \times Q$ *est un ensemble de transitions.*

Definition 3.3 *Une machine de Mealy est un quintuplet* $S = < Q, I, In, Out, T >$ *où :*
- Q *est un ensemble d'états.*
- I *est le sous-ensemble des états* initiaux.
- In **est un alphabet représentant des actions.**
- Out **est un alphabet représentant des réactions.**
- $T \subseteq Q \times In \times Out \times Q$ *est un ensemble de transitions.*

On distingue également les systèmes de transition finis des systèmes de transition infinis.

Definition 3.4 *Un système de transition* **étiqueté fini** *est un quadruplet* $S = < Q, I, A, T >$ *où :*
- Q *est un ensemble d'états.*
- I *est le sous-ensemble des états* initiaux.
- A *est un ensemble d'actions ou alphabet.*
- $T \subseteq Q \times A \times Q$ *est un ensemble* **fini** *de transitions.*

Remarque : si le nombre de transitions est fini, le nombre d'états *interressant* et le nombre d'actions *interressantes* de l'alphabet sont finis.

3.2 Les systèmes réactifs discrets finis

Ils sont modélisables par des systèmes de transition finis. L'avantage principal est que tous les problèmes sur une telle structure sont décidables.

3.2.1 Modélisation par systèmes de transition finis

Modélisation des données

Dessiner les systèmes de transition des objets suivants :

Question 3.1 *Une variable entière comprise entre 0 et 3*
- *les états ;*
- *l'opération d'affectation ;*
- *l'opération de lecture*

Question 3.2 *Un buffer de taille 1.*

Question 3.3 *Une file d'attente de taille 2.*

Question 3.4 *Une pile d'attente de taille 2.*

Modélisation de comportements simples

Dessiner les systèmes de transition des objets suivants :

Question 3.5 *Un contrôleur d'une salle limitée à dix places.*

Question 3.6 *Un contrôleur de bibliothèque dans laquelle un écrivain est nécessairement seul, et ou les lecteurs peuvent se côtoyer. La première personne à accéder à la bibliothèque est un écrivain.*

Question 3.7 *Une porte logique* **ET**.

Question 3.8 *Un processus de l'algorithme d'exclusion mutuelle de Peterson dans le cas de deux processus.*

```
while true do begin
              F[i] := true;
              Tour := i;
              wait (F[j]=false or Tour = j);
              F[i] := false;
              end;
```

Question 3.9 *L'émetteur dans le protocole du bit alterné.*

```
parite := true
while true do begin
              recevoir(message); /* du producteur */
              emettre(message,parite);
              recevoir(ack,pariteAck);
              while parite <> pariteAck do begin
                                           emettre(message,parite);
                                           recevoir(ack,pariteAck);
                                           end;
              parite := not parite;
              end;
```

Question 3.10 *le récepteur dans le protocole du bit alterné.*

```
pariteAck := true
while true do begin
              recevoir(message,parite);
              emettre(ack,parite);
              while parite <> pariteAck do begin
```

```
                                    recevoir(message,parite);
                                    emettre(ack,parite);
                                    end;
emettre(message); /* vers le consommateur */
pariteAck := not pariteAck;
end;
```

Modélisation de comportements complexes

Question 3.11 *Essayer de dessiner le système de transition de l'algorithme d'exclusion mutuelle de Peterson dans le cas de deux processus.*

3.2.2 Le produit de synchronisation

Le comportement global d'un ensemble de processus concurrents est calculé à partir : d'une part des descriptions des comportements de chacun des processus ; d'autre part d'une description de leurs interactions. Les comportements des composants sont décrits par des systèmes de transition, et les interactions par un système de synchronisation. Le produit de synchronisation calcule le système de transition comme la restriction du produit libre des systèmes de transition aux seules transitions qui correspondent aux interactions permises. Plus formellement :

Un système de synchronisation pour un ensemble $\{S_1, \ldots, S_n\}$ de systèmes de transition est un ensemble de vecteurs de synchronisation.
Un vecteur de synchronisation est un élément de $\prod_{i=1}^{i=n} A_i$, donc un n-uplet $(a_1, \ldots, a_n)$ et intuitivement représente une évolution globale permise du système à un instant donné et indépendamment du contexte (i.e. de l'état du système).

Le produit de synchronisation d'un ensemble $\{S_1, \ldots, S_n\}$ de systèmes de transition, et d'un système de synchronisation V est le système de transition $S =< Q, I, A, T >$ défini par :
- $Q = \prod_{i=1}^{i=n} Q_i$.
- $I = \prod_{i=1}^{i=n} I_i$.
- $A = \prod_{i=1}^{i=n} A_i$.
- $T = T_l \cap Q \times V \times Q$ avec
  $T_l = \{((q_1, \ldots, q_n)(a_1, \ldots, a_n)(q'_1, \ldots, q'_n)) \ / \ (q_i, a_i, q'_i) \in T_i\}$

Intuitivement, T est égal au produit libre des systèmes de transition T_l diminué de toutes les transitions dont l'étiquette $(a_1, \ldots, a_n)$ n'est pas dans le système de synchronisation V.

Un exemple

La figure 3.1 représente le comportement d'un compteur à un chiffre (2 valeurs possibles) avec une possibilité d'un comportement erroné lorsqu'il se trouve dans l'état `1`.
Il faut noter, sur chacun des états, la présence d'une boucle étiquetée par `e`. La sémantique intuitive de cette transition est : *je ne fais rien, je ne participe pas.* Cette transition lorsqu'elle est présente sur tous les états de tous les composants permet de modéliser tous les phénomènes asynchrones dans ce modèle fondamentalement synchrone.

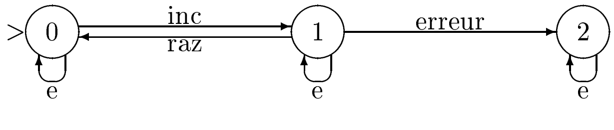

FIG. 3.1 – Une roue de compteur kilométrique

La définition formelle de ce système de transition est :
- $Q = \{0, 1, 2\}$ l'ensemble des états.

- $I = \{0\}$ l'état initial.
- $A = \{e, erreur, inc, raz\}$ l'ensemble d'actions.
- $T = \{(0, e, 0), (1, e, 1), (2, e, 2), (0, inc, 1), (1, raz, 0), (1, erreur, 2)\}$ l'ensemble des transitions.

Considérons un système global composé de deux instances de l'objet représenté par la figure 3.1. La table 3.1 représente les actions globales autorisées à un instant donné. Dans cet exemple, deux variables à un chiffre permettent de construire un compteur kilométrique à deux chiffres. Le phénomène d'entraînement des roues dentées est modélisé par une synchronisation entre : d'une part une incrémentation `inc` sur la première composante du système ; d'autre part une remise à zéro `raz` sur la deuxième composante du système.

ST variable	ST variable
e	inc
inc	raz
erreur	e
e	erreur
raz	raz

TAB. 3.1 – Un système de synchronisation

Le système de synchronisation correspondant a ce tableau est :
- $V = \{(e, inc), (inc, raz), (erreur, e), (e, erreur), (raz, raz)\}$.

La figure 3.2 représente le comportement global des deux variables en tenant compte des contraintes définies dans la table 3.1. Elle est obtenue en effectuant le produit de synchronisation de deux instances du système de transition et du système de synchronisation définis précédemment.

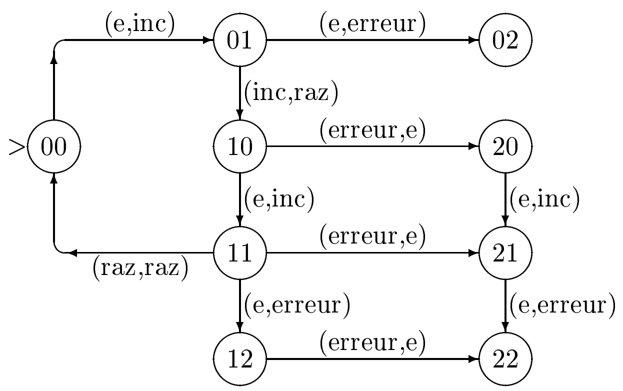

FIG. 3.2 – Un compteur kilométrique obtenu par produit synchronisé

3.2.3 Calcul de comportements complexes

Après avoir modélisé tous les composants par des systèmes de transitions et les interactions entre ces composants par un système de synchronisation :

Question 3.12 *Calculer le système de transition du protocole du bit alterné.*

Question 3.13 *Calculer le système de transition de l'algorithme d'exclusion mutuelle de Peterson dans le cas de deux processus.*

Chapitre 4

Travaux Pratiques

4.1 AltaRica et acheck

Présentation rapide et restreinte du langage AltaRica et de l'outil acheck à travers l'exemple du compteur kilométrique.

4.2 Exercice

Modéliser l'algorithme d'exclusion mutuelle de Peterson en utilisant AltaRica comme Mec.

Chapitre 5

Les automates à contraintes

5.1 Bilan TP

- ST pour algo : correct
- ST pour données : pénible
- Produit de synchro lorsque le nombre de sous-systèmes est grand peut être pénible.

5.2 Idée des automates à contraintes

1. Dans un ST, les états sont nommés.
2. Une variable `nomEtat` et un domaine énuméré de chaines de caractères.
3. Une variable sur un domaine quelconque.
4. Un vecteur de variables et des contraintes sur les domaines.

5.3 Quelques définitions

Definition 5.1 *Un système de transition à contraintes est un triplet :*
- *un ensemble de variables $\vec{s}$,*
- *un ensemble d'événements E,*
- *un ensemble de transition :*

$$G(\vec{s}) \quad \xrightarrow{e} \quad \vec{s} := \sigma(\vec{s})$$

$G(\vec{s})$ est un(e) garde (c-à-d une expression booléenne) et $e \in E$,

5.3.1 Un exemple

Un interrupteur est ouvert ou fermé.

```
variable ouvert : boolean;
event    push;
transitions
  ouvert = true |- push -> ouvert := false;
  ouvert = false |- push -> ouvert := true;
```

Une version plus élégante du même automate :

```
variable ouvert : boolean;
event    push;
transitions
  true |- push -> ouvert := not ouvert;
```

5.4 Quelques propriétés des automates à contraintes

Definition 5.2 *Un système de transition à contraintes avec assertion est un quatruplet :*
- *un ensemble de variables $\vec{s}$,*
- *un ensemble d'événements E,*
- *un ensemble de transition :*

$$G(\vec{s}) \quad \xrightarrow{e} \quad \vec{s} := \sigma(\vec{s})$$

$G(\vec{s})$ est un(e) garde (c-à-d une expression booléenne) et $e \in E$,
- *une contrainte globale (ou assertion) $A(\vec{s})$,*

Attention, introduction de post-condition.

5.4.1 Un exemple

Un interrupteur est ouvert ou fermé. Il possède deux vitesses, mais la fermeture et l'ouverture ne sont possibles qu'en vitesse lente.

```
variable ouvert, rapide : boolean;
event    push, plus, moins;
transitions
  true |- push -> ouvert := not ouvert;
  not rapide |- plus -> rapide := true;
  rapide |- moins -> rapide := false;
assertion
  rapide => not ouvert; // not (rapide & ouvert)
```

Question 5.1 *Montrez que les systèmes de transition à contraintes avec assertion définis en 5.2 ont la même expressivite que ceux de la définition 5.7.*

5.4.2 Un exemple

La transformation donne sur l'exemple précédent :

```
variable ouvert, rapide : boolean;
event    push, plus, moins;
transitions
  not(rapide & ~ouvert) &
  not(rapide & ouvert) &
  true                        |- push -> ouvert := not ouvert;
  not(true & ouvert) &
  not(false & ouvert) &
  not rapide                  |- plus -> rapide := true;
  not(false & ouvert) &
  not(true & ouvert) &
  rapide                      |- moins -> rapide := false;
```

qui se simplifie en :

```
variable ouvert, rapide : boolean;
event    push, plus, moins;
transitions
  not rapide                  |- push -> ouvert := not ouvert;
  not ouvert & not rapide     |- plus -> rapide := true;
  not ouvert & rapide         |- moins -> rapide := false;
```

Definition 5.3 *Un système de transition à contraintes à priorité est un quatruplet :*
- *un ensemble de variables $\vec{s}$,*
- *un ensemble d'événements E,*
- *un ensemble de transitions :*

$$G(\vec{s}) \quad \xrightarrow{e} \quad \vec{s} := \sigma(\vec{s})$$

$G(\vec{s})$ est un(e) garde (c-à-d une expression booléenne) et $e \in E$,
- *des priorités (un ordre partiel) sur E noté $\prec_E$.*

5.4.3 Un exemple

```
variable ouvert, rapide : boolean;
event    {push, plus} > moins;
transitions
  not rapide                |- push -> ouvert := not ouvert;
  not ouvert & not rapide   |- plus -> rapide := true;
  not ouvert                |- moins -> rapide := false;
```

est équivalent à :

```
variable ouvert, rapide : boolean;
event    push, plus, moins;
transitions
  not rapide                |- push -> ouvert := not ouvert;
  not ouvert & not rapide   |- plus -> rapide := true;
  not(not rapide) &
  not(not ouvert & not rapide) &
  not ouvert                |- moins -> rapide := false;
```

qui se réduit en :

```
variable ouvert, rapide : boolean;
event    push, plus, moins;
transitions
  not rapide                |- push -> ouvert := not ouvert;
  not ouvert & not rapide   |- plus -> rapide := true;
  rapide & not ouvert       |- moins -> rapide := false;
```

Question 5.2 *Montrez que les systèmes de transition à contraintes à priorité définis en 5.3 ont la même expressivite que ceux de la définition 5.7.*

Question 5.3 *Montrez que les systèmes de transition à contraintes avec assertion et priorité sont ambigüs.*

Question 5.4 *Proposez une extension du produit de synchronisation de systèmes de transition pour les systèmes de transition à contraintes.*

Question 5.5 *Le produit synchronisé de deux systèmes de transition à contraintes avec assertion est-il ambigü ?*

Question 5.6 *Le produit synchronisé de deux systèmes de transition à contraintes avec priorité est-il ambigü ?*

Question 5.7 *Donnez un argument convaincant pour la première sémantique.*

Question 5.8 *Proposez une sémantique pour la synchronisation hiérarchique de systèmes de transition à contraintes avec assertions et priorites.*

5.5 Modélisation avec des automates à contraintes

Question 5.9 *Modélisez avec des systèmes de transition à contraintes l'algorithme d'exclusion mutuelle de Peterson*

Question 5.10 *Modélisez avec des systèmes de transition à contraintes le protocole du bit alterné.*

5.6 Expressivité des systèmes de transition à contraintes

5.6.1 Les problèmes de décision pour la vérification

Les STE sont classés en fonction de la décidabilité de certaines questions.

L'accessibilité : l'ensembles des triplets (STE, s, s'), où $s \in STE, s' \in STE$, et $s \xrightarrow{*}_{STE} s'$.

Borné : l'ensembles des couples (STE, s_o), où $s_o \in STE$, et $\{s' \in STE, s \xrightarrow{*}_{STE} s'\}$ est fini.

Blocage : l'ensembles des couples (STE, s_o), où $s_o \in STE$, et $\exists s \in STE$ tel que $\{s' \in STE, s \xrightarrow{*}_{STE} s'\} = \emptyset$.

Model-Checking pour une classe Φ de formules : l'ensembles des triplets (STE, s_o, ϕ), où $s_o \in STE, \phi \in \Phi$, et $(STE, s_o) \models \phi$.

5.6.2 Quelques résultats

1. Le problème du model-checking est décidable pour les STE finis.
2. Le problème du model-checking est indécidable pour les systèmes de transition à contraintes.
3. Le problème de l'accessibilité est indécidable pour les systèmes de transition à contraintes.

Les recherches ont abouti à la définition de classes pour lesquelles certains problèmes sont décidables et d'autres indécidables.

5.7 Quelques systèmes de transition à contraintes particuliers

5.7.1 Les systèmes finis

Une restiction consiste à prendre toutes les variables du système à valeurs dans des domaines finis.

Definition 5.4 *Un système de transition à contraintes fini est un triplet :*
- *un ensemble de variables* **entières bornées $\vec{s}$ fini**
- *un ensemble d'événements E,*
- *un ensemble de transition* **fini** *:*

$$G(\vec{s}) \quad \xrightarrow{e} \quad \vec{s} := \sigma(\vec{s})$$

$G(\vec{s})$ est un(e) garde (c-à-d une expression booléenne) et $e \in E$, et toutes les opérations sont permises dans σ.

Cette restriction est importante. De nombreux travaux ont regardé les systèmes dans lesquels les variables représentent les objets les plus utilisés en contrôle de systèmes : les compteurs et les buffers.

5.7.2 Les systèmes à compteurs

Definition 5.5 *Un système de transition à contraintes à compteurs est un quadruplet :*
- *un ensemble de variables de contrôle* **entières bornées $\vec{s}$ fini**
- *un ensemble de compteurs* **entiers $\vec{c}$ fini**
- *un ensemble d'événements E,*

– *un ensemble de transition* **fini** *:*

$$G(s,\vec{c}) \quad \xrightarrow{e} \quad s,\vec{c} := \sigma(s,\vec{c})$$

$G(s,\vec{c})$ est un(e) garde (c-à-d une expression booléenne) et $e \in E$, et les opérations permises dans σ sont $\{assign(s, cste),\ add(c, v),\ test(c, 0),\ reset(c),\ transfert(c, c')\}$

Quelques résultats pour les systèmes de transition à contraintes à compteurs :

1. si $\sigma \in \{add(c, v)\}$, l'accessibilite est décidable et la logique LTL est décidable.
2. si $\sigma \in \{add(c, v)\}$, et que le nombre de compteurs est au maximum 2, l'accessibilite est calculable.
3. si $\sigma \in \{add(c, v),\ test(c, 0)\}$, la logique LTL n'est pas décidable.
4. ...

5.7.3 Les systèmes à files

Definition 5.6 *Un système de transition à contraintes à files est un quintuplet :*
- *un ensemble de variables de contrôle* **entières bornées** $\vec{s}$ **fini**
- *un ensemble de canaux* $\vec{c}$ **fini**
- *un ensemble d'événements E,*
- **un ensemble fini de messages** M.
- *un ensemble de transition* **fini** *:*

$$G(\vec{s}) \quad \xrightarrow{e} \quad s,\vec{c} := \sigma(s,\vec{c})$$

$G(\vec{s})$ est un(e) garde (c-à-d une expression booléenne) et $e \in E$, et les opérations permises dans σ sont $\{assign(s, cste),\ c?m,\ c!m\}$, où $m \in M$ et c est une variable représentant un canal.

Quelques résultats pour les système de transition à contraintes à files :

1. l'accessibilité est indécidable.
2. l'accessibilité avec un seul canal est indécidable.
3. ...

5.7.4 Les systèmes à contrainte AltaRica

Definition 5.7 *Un système de transition à contraintes à files est un quintuplet :*
- *un ensemble de variables de contrôle* **entières bornées** $\vec{s}$ **fini**
- *un ensemble de variables de flux* $\vec{f}$ **fini**
- *un ensemble d'événements E,*
- *un ensemble de transition* **fini** *:*

$$G(s,\vec{f}) \quad \xrightarrow{e} \quad \vec{s} := \sigma(\vec{s})$$

$G(\vec{s})$ est un(e) garde (c-à-d une expression booléenne) et $e \in E$, et les opérations permises dans σ sont $\{assign(s, cste)\}$.
- *une assertion $A(s,\vec{f})$*
- *un ordre partiel $\prec_E$ pour des priorités.*