
Cours Modélisation et Vérification des Systèmes Informatiques (MVSI)

Modélisation, spécification et vérification CM2

Dominique Méry
Telecom Nancy, Université de Lorraine

Année universitaire 2024-2025

- ## 1 Cours 2
- ### 2 Modélisation relationnelle en action

Exemple du PGCD

Exemple de la modélisation d'un dispositif de comptage

Logique TLA et langage TLA⁺

① Cours 2

② Modélisation relationnelle en action

Exemple du PGCD

Exemple de la modélisation d'un dispositif de comptage

Logique TLA et langage TLA⁺

① Cours 2

② Modélisation relationnelle en action

Exemple du PGCD

Exemple de la modélisation d'un dispositif de comptage

Logique TLA et langage TLA⁺

① Cours 2

② Modélisation relationnelle en action

Exemple du PGCD

Exemple de la modélisation d'un dispositif de comptage

Logique TLA et langage TLA⁺

① Cours 2

② Modélisation relationnelle en action

Exemple du PGCD

Exemple de la modélisation d'un dispositif de comptage

Logique TLA et langage TLA⁺

MODULE *pgcd*

EXTENDS *Naturals, TLC*

CONSTANNOTATNTS a, b

VARIABLES x, y

$Init \triangleq x = a \wedge y = b$

$a1 \triangleq x > y \wedge x' = x - y \wedge y' = y$

$a2 \triangleq x < y \wedge y' = y - x \wedge x' = x$

$over \triangleq x = y \wedge x' = x \wedge y' = y$

$Next \triangleq a1 \vee a2 \vee over$

$test \triangleq x \neq y$

```
----- MODULE pgcd -----  
EXTENDS Naturals,TLC  
CONSTANTS a,b  
VARIABLES  x,y  
-----  
Init == x=a /\ y=b  
-----  
a1 == x > y /\ x'=x-y /\ y'=y  
a2 == x < y /\ y'=y-x /\ x'=x  
over == x=y /\ x'=x /\ y'=y  
-----  
Next == a1 \/ a2 \/ over  
-----  
test == x # y  
=====
```


① Cours 2

② Modélisation relationnelle en action

Exemple du PGCD

Exemple de la modélisation d'un dispositif de comptage

Logique TLA et langage TLA⁺

VARIABLES np

Première tentative

$\text{entrer} \triangleq np' = np + 1$

$\text{sortir} \triangleq np' = np - 1$

$\text{next} \triangleq \text{entrer} \vee \text{sortir}$

$\text{init} \triangleq np = 0$

Seconde tentative

$$\text{entrer}_2 \triangleq np < 19 \wedge np' = np + 1$$
$$\text{next}_2 \triangleq \text{entrer}_2 \vee \text{sortir}$$

Troisième tentative

$$\text{sortir}_2 \triangleq np > 0 \wedge np' = np - 1$$

$$\text{next}_3 \triangleq \text{entrer}_2 \vee \text{sortir}_2$$

$$\text{safety}_1 \triangleq np \leq 19$$

$$\text{question}_1 \triangleq np \neq 6$$

```

----- MODULE ex1-----
(* modules de base importables *)
EXTENDS Naturals,TLC

-----
(* un syst\`eme contr\`ole l'acc\`es \`a une salle dont la capacit\`e est de 19 personnes
VARIABLES np
-----

(* Premi\`ere tentative *)
entrer == np '=np +1
sortir == np'=np-1
next == entrer \/ sortir
init == np=0\fora
-----

(* Seconde tentative *)
entrer2 == np<19 /\ np'=np+1
next2 == entrer2 \/ sortir
-----

(* Troisi\`eme tentative *)
sortir2 == np>0 /\ np'=np-1
next3 == entrer2 \/ sortir2
-----

safety1 == np \leq 19
question1 == np # 6
=====

```

Soit $(Th(s, c), x, VALS, Init(x), \{r_0, \dots, r_n\})$ un modèle relationnel M d'un système \mathcal{S} . Une propriété A est une propriété de sûreté pour le système \mathcal{S} , si

$$\forall x_0, x \in VALS. Init(x_0) \wedge Next^*(x_0, x) \Rightarrow A(x).$$

Soit $(Th(s, c), x, VALS, Init(x), \{r_0, \dots, r_n\})$ un modèle relationnel M d'un système \mathcal{S} . Une propriété A est une propriété de sûreté pour le système \mathcal{S} , si

$\forall x_0, x \in VALS. Init(x_0) \wedge Next^*(x_0, x) \Rightarrow A(x).$

► x est une variable ou une liste de variable : VARIABLES x

Soit $(Th(s, c), x, VALS, Init(x), \{r_0, \dots, r_n\})$ un modèle relationnel M d'un système \mathcal{S} . Une propriété A est une propriété de sûreté pour le système \mathcal{S} , si

$\forall x_0, x \in VALS. Init(x_0) \wedge Next^*(x_0, x) \Rightarrow A(x).$

- ▶ x est une variable ou une liste de variable : `VARIABLES x`
- ▶ $Init(x)$ est une variable ou une liste de variable : `init == Init(x)`

Soit $(Th(s, c), x, VALS, Init(x), \{r_0, \dots, r_n\})$ un modèle relationnel M d'un système \mathcal{S} . Une propriété A est une propriété de sûreté pour le système \mathcal{S} , si

$$\forall x_0, x \in VALS. Init(x_0) \wedge Next^*(x_0, x) \Rightarrow A(x).$$

- ▶ x est une variable ou une liste de variable : `VARIABLES x`
- ▶ $Init(x)$ est une variable ou une liste de variable : `init == Init(x)`
- ▶ $Next^*(x_0, x)$ est la définition de la relation définissant ce que fait le système : `Next == a1 \ / \ a2 \ / \ \ / \ an`

Soit $(Th(s, c), x, VALS, Init(x), \{r_0, \dots, r_n\})$ un modèle relationnel M d'un système \mathcal{S} . Une propriété A est une propriété de sûreté pour le système \mathcal{S} , si

$$\forall x_0, x \in VALS. Init(x_0) \wedge Next^*(x_0, x) \Rightarrow A(x).$$

- ▶ x est une variable ou une liste de variable : `VARIABLES x`
- ▶ $Init(x)$ est une variable ou une liste de variable : `init == Init(x)`
- ▶ $Next^*(x_0, x)$ est la définition de la relation définissant ce que fait le système : `Next == a1 \ / a2 \ / \ / an`
- ▶ $A(x)$ est une expression logique définissant une propriété de sûreté à vérifier sur toutes les configurations du modèle : `Safety == A(x)`

1 Cours 2

2 Modélisation relationnelle en action

Exemple du PGCD

Exemple de la modélisation d'un dispositif de comptage

Logique TLA et langage TLA⁺

- ▶ TLA (Temporal Logic of Actions) sert à exprimer des formules en logique temporelle : $\Box P$ ou *toujours P*
- ▶ TLA⁺ est un langage permettant de déclarer des constantes, des variables et des définitions :
 - $\langle \text{def} \rangle == \langle \text{expression} \rangle$: une définition $\langle \text{def} \rangle$ est la donnée d'une expression $\langle \text{expression} \rangle$ qui utilise des éléments définis avant ou dans des modules qui *étendent* ce module.
 - Une variable x est soit sous la forme x soit sous la forme x' : x' est la valeur de x après.
 - Un module a un nom et rassemble des définitions et il peut être une extension d'autres modules.
 - $[f \text{ EXCEPT! } [i]=e]$ est la fonction f où seule la valeur en i a changé et vaut .
- ▶ Une configuration doit être définie pour évaluer une spécification

- ▶ Limitation des actions :

$$\begin{aligned} \text{nom} &\triangleq \\ &\wedge \text{cond}(v, w) \\ &\wedge v' = e(v, w) \\ &\wedge w' = w \end{aligned}$$

- ▶ $e(v, w)$ doit être codable en Java.
- ▶ Modules standards : Naturals, Integers, TLC ...

- ▶ Téléchargez l'application le site de Microsoft pour votre ordinateur.
- ▶ Ecrivez des modèles et testez les !
- ▶ Limitations par les domaines des variables.



Permettre un raisonnement symbolique quel que soit l'ensemble des états