Cours Modélisation et Vérification des Systèmes Informatiques (MVSI)

Modélisation, spécification et vérification CM2

Dominique Méry
Telecom Nancy, Université de Lorraine

Année universitaire 2024-2025

Plan

Sommaire

Current Summary

Current Subsection Summary

Calcul d'un pgcd

MODULE *pgcd*

EXTENDS *Naturals*, *TLC*

CONSTANNOTATNTS a, b

VARIABLES x, y

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

$$a1 \triangleq x > y \land x' = x - y \land y' = y$$
$$a2 \triangleq x < y \land y' = y - x \land x' = x$$
$$over \triangleq x = y \land x' = x \land y' = y$$

$$Next \triangleq a_1 \lor a_2 \lor over$$

$$test \triangleq x \neq y$$

Calcul du pgcd

```
-------------------- 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
=============================================
```

Current Subsection Summary

Exemple de modélisation TLA$^+$

test

MODULE $ex1$

modules de base importables

EXTENDS *Naturals*, *TLC*

un système contrôle l'accès à une salle dont la capacité est de 19 personnes ; écrire un modèle de ce système en vérifiant la propriété de sûreté

VARIABLES *np*

Exemple de modélisation TLA$^+$

Première tentative

$$\mathit{entrer} \triangleq np' = np + 1$$
$$\mathit{sortir} \triangleq np' = np - 1$$
$$\mathit{next} \triangleq \mathit{entrer} \lor \mathit{sortir}$$
$$\mathit{init} \triangleq np = 0$$

Seconde tentative

$$\mathit{entrer}_2 \triangleq \mathit{np} < 19 \land \mathit{np}' = \mathit{np}+1$$
$$\mathit{next}_2 \triangleq \mathit{entrer}_2 \lor \mathit{sortir}$$

Exemple de modélisation TLA$^+$

Troisième tentative

$$sortir_2 \triangleq np > 0 \land np' = np - 1$$
$$next_3 \triangleq entrer_2 \lor sortir2$$

$$safety_1 \triangleq np \leq 19$$
$$question_1 \triangleq np \neq 6$$

```
-------------------------------- 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
==============================================================================
```

Traduction de la définition

Soit $(Th(s,c), x, \textsc{Vals}, \mathsf{Init}(x), \{r_0, \ldots, r_n\})$ un modèle relationnel M d'un système $\mathcal{S}$. Une propriété A est une propriété de sûreté pour le système $\mathcal{S}$, si
$\forall x_0, x \in \textsc{Vals}.Init(x_0) \wedge \mathsf{Next}^\star(x_0, x) \Rightarrow A(x)$.

Traduction de la définition

Soit $(Th(s,c), x, \textsc{Vals}, \mathsf{Init}(x), \{r_0, \ldots, r_n\})$ un modèle relationnel M d'un système $\mathcal{S}$. Une propriété A est une propriété de sûreté pour le système $\mathcal{S}$, si
$\forall x_0, x \in \textsc{Vals}.Init(x_0) \wedge \mathsf{Next}^\star(x_0, x) \Rightarrow A(x)$.

- x est une variable ou une liste de variable : `VARIABLES x`

Soit $(Th(s,c), x, \textsc{Vals}, \mathsf{Init}(x), \{r_0, \ldots, r_n\})$ un modèle relationnel M d'un système $\mathcal{S}$. Une propriété A est une propriété de sûreté pour le système $\mathcal{S}$, si
$\forall x_0, x \in \textsc{Vals}.Init(x_0) \wedge \mathsf{Next}^\star(x_0, x) \Rightarrow A(x)$.

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

Traduction de la définition

Soit $(Th(s,c), x, \textsc{Vals}, \mathsf{Init}(x), \{r_0, \ldots, r_n\})$ un modèle relationnel M d'un système $\mathcal{S}$. Une propriété A est une propriété de sûreté pour le système $\mathcal{S}$, si
$\forall x_0, x \in \textsc{Vals}.Init(x_0) \wedge \mathsf{Next}^\star(x_0, x) \Rightarrow A(x)$.

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

Traduction de la définition

Soit $(Th(s,c), x, \textsc{Vals}, \mathsf{Init}(x), \{r_0, \ldots, r_n\})$ un modèle relationnel M d'un système $\mathcal{S}$. Une propriété A est une propriété de sûreté pour le système $\mathcal{S}$, si
$\forall x_0, x \in \textsc{Vals}.Init(x_0) \wedge \mathsf{Next}^\star(x_0, x) \Rightarrow A(x)$.

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

Current Subsection Summary

- 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 :
  - `<def> == <expression>` : une définition `<def>` est la donnée d'une expression `<expreession>` 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 EXCEPT! [i]=e]` est la fonction f où seule lavaleur en i a changé et vaut .
- Une configuration doit être définie pour évaluer une spécification

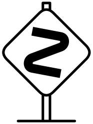
- Limitation des actions :

$$
\begin{array}{l}
\texttt{nom} \triangleq \\
\quad \land\ cond(v,w) \\
\quad \land\ v' = e(v,w) \\
\quad \land\ w' = w
\end{array}
$$

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

Commentaires

- 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