

Deuxième partie

TLA⁺ – les actions



Plan

- 1 Programme
 - Structure
 - Constantes
 - Expressions
- 2 Actions
- 3 Fonctions
 - Fonctions de X dans Y
 - Les enregistrements (records)
 - Définition récursive
 - Tuples
 - Séquences
- 4 Divers



Structure d'un programme

Un « programme » = une spécification de ST =

- des constantes
- des variables
- des actions = prédicat de transition reliant deux états :
 - l'état courant, variables non primées
 - l'état d'arrivée, variables primées
- un ensemble d'états initiaux défini par un prédicat d'état
- un prédicat de transition construit par disjonction des actions
(\approx actions répétées infiniment)



Exemple

MODULE *exemple1*

EXTENDS *Naturals*

VARIABLE *x*

États initiaux

Init $\triangleq 0 \leq x \wedge x < 3$

Actions

Plus $\triangleq x' = x + 1$

Moins $\triangleq x > 0 \wedge x' = x - 1$

Next $\triangleq Plus \vee Moins$

Spec $\triangleq Init \wedge \square[Next]_{\langle x \rangle}$

27

Exemple

Correspond au système de transitions :

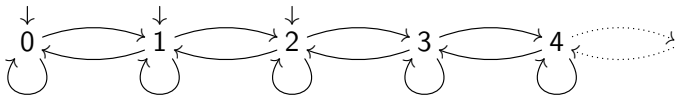
$$V \triangleq x \in \mathbb{N}$$

$$I \triangleq 0 \leq x < 3$$

$$R \triangleq x' = x + 1$$

$$\vee x > 0 \wedge x' = x - 1$$

$$\vee x' = x$$



Constantes

- Constantes explicites : 0, 1, TRUE, FALSE, “toto”
- Constantes nommées : `CONSTANT N`
généralement accompagnées de propriétés :
`ASSUME N ∈ Nat`



Expressions autorisées

Tout ce qui est axiomatisable :

- expressions logiques : $\neg, \wedge, \vee, \forall x \in S : p(x), \exists x \in S : p(x) \dots$
- expressions arithmétiques : $+, -, > \dots$
- expressions ensemblistes : $\in, \cup, \cap, \subset, \{e_1, e_2, \dots, e_n\}, n..m, \{x \in S : p(x)\}, \{f(x) : x \in S\}, \text{UNION } S, \text{SUBSET } S$
- IF *pred* THEN e_1 ELSE e_2
- fonctions de X dans Y
- tuples, séquences, ...



Opérateurs ensemblistes

$\{e_1, \dots, e_n\}$	ensemble en extension
$n..m$	$\{i \in Nat : n \leq i \leq m\}$
$\{x \in S : p(x)\}$	l'ensemble des éléments de S vérifiant la propriété p $\{n \in 1..10 : n \% 2 = 0\} = \{2, 4, 6, 8, 10\}$ $\{n \in Nat : n \% 2 = 1\} =$ les nombres impairs
$\{f(x) : x \in S\}$	l'ensemble des valeurs de l'opérateur f en S $\{2 * n : n \in 1..5\} = \{2, 4, 6, 8, 10\}$ $\{2 * n + 1 : n \in Nat\} =$ les nombres impairs
UNION S	l'union des éléments de S UNION $\{\{1, 2\}, \{2, 3\}, \{3, 4\}\} = \{1, 2, 3, 4\}$
SUBSET S	l'ensemble des sous-ensembles de S SUBSET $\{1, 2\} = \{\{\}, \{1\}, \{2\}, \{1, 2\}\}$



Plan

- 1 Programme
 - Structure
 - Constantes
 - Expressions
- 2 **Actions**
- 3 Fonctions
 - Fonctions de X dans Y
 - Les enregistrements (records)
 - Définition récursive
 - Tuples
 - Séquences
- 4 Divers



Actions

Action

Action = prédicat de transition = expression booléenne contenant des constantes, des variables et des variables primées.

Une action n'est pas une affectation.

$$x' = x + 1$$

$$\equiv x' - x = 1$$

$$\equiv x = x' - 1$$

$$\equiv (x > 1 \wedge x'/x = 1 \wedge x' \% x = 1) \vee (1 = x \wedge 2 = x')$$

$$\vee (x = 0 \wedge x' \in \{y \in \text{Nat} : y + 1 = 2 * y\})$$

Action non déterministe : $x' > x$ ou $x' \in \{x + 1, x + 2, x + 3\}$

Action non évaluable : $x' \in \{y : \exists z : z * y = x \wedge z \% 2 = 0\}$

Action multiple : $x' = y \wedge y' = x$

Action gardée

Action gardée

Action constituée d'une conjonction :

- 1 un prédicat d'état portant uniquement sur l'état de départ
- 2 un prédicat de transition déterministe $var' = \dots$
ou un prédicat de transition non déterministe $var' \in \dots$

Se rapproche d'une instruction exécutable.

$$x < 10 \wedge x' = x + 1$$

plutôt que $x' = x + 1 \wedge x' < 11$

ou $x' - x = 1 \wedge x' < 11$



Bégaïement

Bégaïement

$[A]_f \triangleq A \vee f' = f$, où f est un tuple de variables.

exemple : $[x' = x + 1]_{\langle x, y \rangle} = (x' = x + 1 \vee (\langle x, y \rangle' = \langle x, y \rangle))$
 $= (x' = x + 1 \vee (x' = x \wedge y' = y))$

Non bégaïement

$\langle A \rangle_f \triangleq A \wedge f' \neq f$

Variables non contraintes

$(x' = x + 1) = (x' = x + 1 \wedge y' = \text{n'importe quoi})$
 $\neq (x' = x + 1 \wedge y' = y)$

UNCHANGED

UNCHANGED $e \triangleq e' = e$

EXTENDS *Naturals*CONSTANT *Data*VARIABLES *val, ready, ack*

$$\begin{aligned} \textit{Init} &\stackrel{\Delta}{=} \wedge \textit{val} \in \textit{Data} \\ &\wedge \textit{ready} \in \{0, 1\} \\ &\wedge \textit{ack} = \textit{ready} \end{aligned}$$

$$\begin{aligned} \textit{Send} &\stackrel{\Delta}{=} \wedge \textit{ready} = \textit{ack} \\ &\wedge \textit{val}' \in \textit{Data} \\ &\wedge \textit{ready}' = 1 - \textit{ready} \\ &\wedge \text{UNCHANGED } \textit{ack} \end{aligned}$$

$$\begin{aligned} \textit{Reveive} &\stackrel{\Delta}{=} \wedge \textit{ready} \neq \textit{ack} \\ &\wedge \textit{ack}' = 1 - \textit{ack} \\ &\wedge \text{UNCHANGED } \langle \textit{val}, \textit{ready} \rangle \end{aligned}$$

$$\textit{Next} \stackrel{\Delta}{=} \textit{Send} \vee \textit{Reveive}$$

$$\textit{Spec} \stackrel{\Delta}{=} \textit{Init} \wedge \square[\textit{Next}]_{\langle \textit{val}, \textit{ready}, \textit{ack} \rangle}$$

Plan

- 1 Programme
 - Structure
 - Constantes
 - Expressions
- 2 Actions
- 3 **Fonctions**
 - Fonctions de X dans Y
 - Les enregistrements (records)
 - Définition récursive
 - Tuples
 - Séquences
- 4 Divers



Fonctions

Fonction au sens “mapping”, correspondance.

- $[X \rightarrow Y]$ = ensemble des fonctions de X dans Y .
- f fonction de X dans Y : $f \in [X \rightarrow Y]$
- $f[x] \triangleq$ la valeur de f en x .

Une fonction est une **valeur**.

Une variable contenant une fonction peut changer de valeur \Rightarrow la “fonction change”.



Définition

Définition d'un symbole

$f[x \in Nat] \triangleq$ expression utilisant x

Exemple : $Inc[x \in Nat] \triangleq x + 1$

Définition d'une valeur

$[x \in S \mapsto expr]$

Exemple : $[x \in 1..4 \mapsto 2 * x]$

Tableaux

Tableau : fonction $t \in [X \rightarrow Y]$ où X est un intervalle d'entiers.

Domaine/Codomaine

Domain

$\text{DOMAIN } f = \text{domaine de définition de } f$

Codomaine (range)

$\text{Codomain}(f) \triangleq \{f[x] : x \in \text{DOMAIN } f\}$



EXCEPT

Une variable contenant une fonction peut changer de valeur :

MODULE *m*

VARIABLE *a*

Init $\triangleq a = [i \in 1..3 \mapsto i + 1]$

Act1 $\triangleq \wedge a[1] = 2$
 $\wedge a' = [i \in 1..6 \mapsto i * 2]$

Act2 $\triangleq \wedge a[2] = 4$
 $\wedge a' = [i \in 1..6 \mapsto \text{IF } i = 2 \text{ THEN } 8 \text{ ELSE } a[i]]$

EXCEPT

$[a \text{ EXCEPT } ![i] = v]$ équivalent à

$[j \in \text{DOMAIN } a \mapsto \text{IF } j = i \text{ THEN } v \text{ ELSE } a[j]]$

$(a' = [a \text{ EXCEPT } ![2] = 8]) \neq (a[2]' = 8)$

Enregistrements

Enregistrement

Un enregistrement (record) est une fonction de $[X \rightarrow Y]$ où X est un ensemble de chaînes.

Écriture simplifiée :

$$\begin{aligned} [\text{"toto"} \mapsto 1, \text{"titi"} \mapsto 2] &= [\text{toto} \mapsto 1, \text{titi} \mapsto 2] \\ \text{rec}[\text{"toto"}] &= \text{rec.toto} \end{aligned}$$



Définition récursive

Lors de la définition de symbole, il est possible de donner une définition récursive :

$$fact[n \in Nat] \stackrel{\Delta}{=} \text{IF } n = 0 \text{ THEN } 1 \text{ ELSE } n * fact[n - 1]$$

En théorie, il faudrait démontrer la validité de cette définition (terminaison dans tous les cas).



Tuple

n-tuple

Notation : $\langle a, b, c \rangle$.

Un n-tuple est une fonction de domaine $= \{1, \dots, n\}$:

$$\langle a, b, c \rangle[3] = c$$

Pratique pour représenter des relations :

$$\{\langle x, y \rangle \in X \times Y : R(x, y)\}.$$

$$\text{Exemple : } \{\langle a, b \rangle \in \text{Nat} \times \text{Nat} : a = 2 * b\}.$$



Séquences

Séquences

$Seq(T) \triangleq \text{UNION } \{[1 .. n \rightarrow T] : n \in \text{Nat}\}$

\triangleq ensemble des séquences finies contenant des T .

Opérations $Len(s)$, $s \circ t$ (concaténation), $Append(s, e)$, $Head(s)$,
 $Tail(s)$.



Plan

- 1 Programme
 - Structure
 - Constantes
 - Expressions
- 2 Actions
- 3 Fonctions
 - Fonctions de X dans Y
 - Les enregistrements (records)
 - Définition récursive
 - Tuples
 - Séquences
- 4 Divers



Définition de symbole local

LET

Expression : $\text{LET } v \stackrel{\Delta}{=} e \text{ IN } f$

Équivalent à l'expression f où toutes les occurrences du symbole v sont remplacées par e .

Exemple : $\text{LET } i \stackrel{\Delta}{=} g(x) \text{ IN } f(i)$
 $\equiv f(g(x))$

$\text{pythagore}(x, y, z) \stackrel{\Delta}{=} \text{LET } \text{carre}(n) \stackrel{\Delta}{=} n * n \text{ IN}$
 $\text{carre}(x) + \text{carre}(y) = \text{carre}(z)$

Choix déterministe

Opérateur de choix

$\text{CHOOSE } x \in S : p \triangleq$ choix arbitraire *déterministe* d'un élément dans l'ensemble S et qui vérifie le prédicat p .

$\text{max}[S \in \text{SUBSET } \text{Nat}] \triangleq \text{CHOOSE } m \in S : (\forall p \in S : m \geq p)$

Choix déterministe

$\text{CHOOSE } x : p = \text{CHOOSE } x : p$ (aïe)

Pour un ensemble S et une propriété p , l'élément choisi est toujours le même, dans toutes les exécutions et tout au long de celles-ci. Ce n'est pas un sélecteur aléatoire qui donne un élément distinct à chaque appel !

Choix déterministe - 2

- Le programme

$$(x = \text{CHOOSE } n : n \in \text{Nat}) \wedge \square [x' = \text{CHOOSE } n : n \in \text{Nat}]_x$$

a une **unique** exécution : $x = c \rightarrow x = c \rightarrow \dots$ où c est un nombre entier inderterminé (spécifié par le choose).

- Le programme

$$(x \in \text{Nat}) \wedge \square [x' \in \text{Nat}]_x$$

a une infinité d'exécutions, dont certaines où x est différent dans chaque état, d'autres où x est constant, d'autres où x cycle. . .

