Deuxième partie

TLA$^+$– les actions

Objectifs

Décrire des systèmes de transition

- en intention
- de manière abstraite

Le formalisme de description doit être

- aussi naturel que possible (= proche d'un langage de programmation)
- parfaitement rigoureux (pas d'ambiguïté ou de sémantique approximative)
- complet (tout système de transition est descriptible)
- minimaliste dans les concepts (pour axiomatiser)
- extensible (pour décrire des concepts dérivés)

$\rightarrow$ variables, ensembles et fonctions, actions de transition

TLA$^+$: Temporal Logic of Actions

TLA$^+$: Temporal Logic of Actions

- Un langage outillé pour modéliser les programmes et systèmes
- Particulièrement adapaté aux programmes et systèmes distribués / concurrents
- Basé sur les systèmes de transition
- Une toolbox embarquant un éditeur de texte, un outil de vérification de modèles (TLC) et pour visauliser les exécutions, un outil pour écrire et vérifier des preuves (TLAPS)
- `http://lamport.azurewebsites.net/tla/tla.html`

Plan

Structure d'une spécification

Un « programme » = une spécification de système de transition =

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

Exemple

MODULE *exemple*1

EXTENDS *Naturals*

VARIABLE x

États initiaux

$Init \triangleq x \in 0 \,..\, 2$ équivalent à $x \in Nat \wedge 0 \leq x \wedge x < 3$

Actions

$Plus \triangleq x' = x + 1$

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

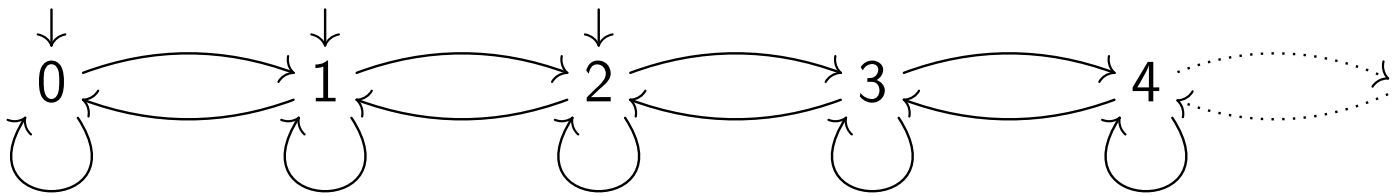
$Next \triangleq Plus \vee Moins$

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

Exemple

Correspond au système de transition :

$$
\begin{aligned}
V &\triangleq x \in \mathbb{N} \\
I &\triangleq 0 \leq x \leq 2 \\
R &\triangleq \quad x' = x + 1 \\
&\quad \lor x > 0 \land x' = x - 1 \\
&\quad \lor x' = x
\end{aligned}
$$



Constantes

- Constantes explicites : 0, 1, TRUE, FALSE, “toto”
- Constantes nommées : CONSTANT N
  généralement accompagnées de propriétés :
  ASSUME $N \in Nat \wedge N \geq 2$

Expressions autorisées

Tout ce qui est axiomatisable :

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

Opérateurs ensemblistes

$\{e_1, \ldots, e_n\}$	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 entiers impairs
$\{f(x) : x \in S\}$	l'ensemble des valeurs de l'operateur f en S
	$\{2 * n : n \in 1..5\} = \{2, 4, 6, 8, 10\}$
	$\{2 * n + 1 : n \in Nat\} =$ les entiers 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

Actions

Action

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

Une action n'est pas une affectation.

$$
\begin{aligned}
& x' = x + 1 \\
\equiv\ & x' - x = 1 \\
\equiv\ & x = x' - 1 \\
\equiv\ & (x > 1 \land x'/x = 1 \land x'\%x = 1) \lor (1 = x \land 2 = x') \\
& \quad \lor (x = 0 \land x' \in \{y \in \mathit{Nat} : y + 1 = 2 * y\})
\end{aligned}
$$

Autres exemples d'actions :

- $x' > x$ ou $x' \in \{x + 1, x + 2, x + 3\}$ (non déterministe)
- $x' \in \{y \in \mathbb{N} : \exists z \in \mathbb{N} : z * y = x \land z\%2 = 0\}$ (non évaluable)
- $x' = y \land y' = x$ (plusieurs variables)

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 $var' = \ldots$
   ou un prédicat de transition non déterministe $var' \in \ldots$

Se rapproche d'une instruction exécutable.

$$\begin{array}{rl} & x < 10 \wedge x' = x + 1 \\ \text{plutôt que} & x' = x + 1 \wedge x' < 11 \\ \text{ou} & x' - x = 1 \wedge x' < 11 \end{array}$$

Bégaiement

Bégaiement

$[\mathcal{A}]_f \triangleq \mathcal{A} \vee f' = f$, 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 bégaiement

$\langle \mathcal{A} \rangle_f \triangleq \mathcal{A} \wedge f' \neq f$

Variables non contraintes

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

UNCHANGED

UNCHANGED $e \triangleq e' = e$

MODULE *AlternatingBit*

EXTENDS *Naturals*

CONSTANT *Data*

VARIABLES *val*, *ready*, *ack*

$$
\begin{aligned}
\mathit{Init} \triangleq\ & \land\ \mathit{val} \in \mathit{Data} \\
& \land\ \mathit{ready} \in \{0, 1\} \\
& \land\ \mathit{ack} = \mathit{ready}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Send} \triangleq\ & \land\ \mathit{ready} = \mathit{ack} \\
& \land\ \mathit{val}' \in \mathit{Data} \\
& \land\ \mathit{ready}' = 1 - \mathit{ready} \\
& \land\ \text{UNCHANGED}\ \mathit{ack}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Receive} \triangleq\ & \land\ \mathit{ready} \neq \mathit{ack} \\
& \land\ \mathit{ack}' = 1 - \mathit{ack} \\
& \land\ \text{UNCHANGED}\ \langle \mathit{val}, \mathit{ready} \rangle
\end{aligned}
$$

$$\mathit{Next} \triangleq \mathit{Send} \lor \mathit{Receive}$$

$$\mathit{Spec} \triangleq \mathit{Init} \land \Box[\mathit{Next}]_{\langle \mathit{val}, \mathit{ready}, \mathit{ack} \rangle}$$

Mise en pratique : factorielle

Écrire la spécification d'un programme qui définit la factorielle d'un entier N, c'est-à-dire écrire une spécification telle qu'une variable contiendra, en un point déterminé d'une exécution, la valeur de $N!$ et ne changera plus ensuite.

- En une transition (!)
- En N transitions déterministes, par multiplications successives, par ordre croissant ou décroissant
- En $\lceil \frac{N}{2} \rceil$ à N transitions non déterministes, en pouvant faire deux multiplications en une transition
- En N transitions non déterministes, sans ordre particulier des multiplications
- En $1..N$ transitions non déterministes, en pouvant réaliser plusieurs multiplications en une transition

Plan

Fonctions

Fonction au sens mathématique : *mapping*, correspondance.

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

Une fonction est une valeur.
Une variable contenant une fonction peut changer de valeur
$\Rightarrow$ "la fonction change" par abus de langage.

Définition

Définition d'un symbole constant

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

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

Définition d'une valeur

$[x \in S \mapsto expr]$

Exemples : $[x \in 1..4 \mapsto 2 * x]$, $[x \in \{1, 2, 3, 5, 7, 11\} \mapsto 2 * x + 1]$

Tableaux

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

Domaine/Codomaine

Domain

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

Codomaine (range)

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

EXCEPT

Une variable contenant une fonction peut changer de valeur :

MODULE m

VARIABLE a

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

$$\begin{aligned} Act1 \triangleq\ & \land a[1] = 2 \\ & \land a' = [i \in 1 \,..\, 6 \mapsto i * 2] \end{aligned}$$

$$\begin{aligned} Act2 \triangleq\ & \land a[2] = 4 \\ & \land a' = [i \in 1 \,..\, 6 \mapsto \text{IF } i = 2 \text{ THEN } 8 \text{ ELSE } a[i]] \end{aligned}$$

EXCEPT

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

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

$(a' = [a \text{ EXCEPT } ![2] = 8]) \not\equiv (a[2]' = 8)$

Enregistrements

Enregistrement

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

Écriture simplifiée :

$$[\text{“}toto\text{”} \mapsto 1, \text{“}titi\text{”} \mapsto 2] = [toto \mapsto 1, titi \mapsto 2]$$
$$rec[\text{“}toto\text{”}] = rec.toto$$

Tuple

n-tuple

Notation : $\langle a, b, c \rangle$.
Un n-tuple est une fonction de domaine = {1,. . .,n} :
$\langle a, b, c \rangle[3] = c$

Pratique pour représenter des relations :
$\{\langle x, y \rangle \in X \times Y : R(x, y)\}$.
Exemple : $\{\langle a, b \rangle \in Nat \times Nat : a = 2 * b\}$.

Séquences

Séquences

$Seq(T) \triangleq \text{UNION } \{[1 \,..\, n \to T] : n \in Nat\}$
$\triangleq$ ensemble des séquences finies contenant des T.

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

Exemple de définition des opérateurs :

$$s \circ t \triangleq [i \in 1..(Len(s) + Len(t)) \mapsto \text{IF } i \leq Len(s) \text{ THEN } s[i] \text{ ELSE } t[i - Len(s)]]$$

$$Append(s, e) \triangleq s \circ \langle e \rangle$$

Fonction $\neq$ opérateur

$SuccF[x \in Nat] \triangleq x + 1$
$SuccO(x) \triangleq x + 1$

- *SuccF* est une définition de fonction au sens mathématique
  - Équivalent à $SuccF \triangleq [x \in Nat \mapsto x + 1]$
  - Son domaine est un ensemble : $\text{DOMAIN } SuccF = \mathbb{N}$
  - Son co-domaine est un ensemble : $\{SuccF[x] : x \in \text{DOMAIN } SuccF\} = \mathbb{N}^*$
  - $SuccF \in [X \to Y]$ a du sens
- *SuccO* est la définition d'un opérateur
  - Factorisation d'écriture : similaire à une macro dont on peut substituer le texte
  - N'a pas de domaine ou de co-domaine
  - $SuccO \in [X \to Y]$ n'a pas de sens

Définition récursive

Lors de la définition de symbole (fonction ou opérateur), il est possible de donner une définition récursive :

- Fonction :
  $fact[n \in Nat] \triangleq \text{IF } n = 0 \text{ THEN } 1 \text{ ELSE } n * fact[n-1]$
- Opérateur :
  $RECURSIVE\ fact(\_)$
  $fact(n) \triangleq \text{IF } n = 0 \text{ THEN } 1 \text{ ELSE } n * fact(n-1)$

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

Plan

Définition de symbole local

LET

Expression : $\text{LET } v \triangleq e \text{ IN } f$

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

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

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

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.

Maximum d'un ensemble

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

Somme des éléments d'un ensemble

$$
\begin{array}{l}
Somme[S \in \text{SUBSET } Nat] \triangleq \\
\quad \text{IF } S = \emptyset \text{ THEN } 0 \\
\quad \text{LET } e \triangleq \text{CHOOSE } x \in S : \text{TRUE} \\
\qquad \text{IN } \quad e + Somme[S \setminus \{e\}]
\end{array}
$$

Choix déterministe - 2

Choix déterministe

$\text{CHOOSE } x \in S : p = \text{CHOOSE } x \in S : p$ (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.

- La spécification
  $$(x = \text{CHOOSE } n : n \in Nat) \wedge \Box[x' = \text{CHOOSE } n : n \in Nat]_{\langle x \rangle}$$
  a une unique exécution : $x = c \rightarrow x = c \rightarrow \ldots$ où c est un nombre entier indéterminé (spécifié par le *choose*).
- La spécification
  $$(x \in Nat) \wedge \Box[x' \in Nat]_{\langle x \rangle}$$
  a une infinité d'exécutions, dont certaines où x est différent dans chaque état, d'autres où x finit par être constant...

Conclusion

Les actions TLA$^+$ sont un noyau minimal permettant d'exprimer toute spécification de système de transition (= tout programme) à partir de :

- la logique du premier ordre
- la théorie des ensembles
- les fonctions (*mapping*)
- les valeurs avant/après des variables