CNAM - CRA Nancy
2003

Jacques Lonchamp

TROISIEME PARTIE

La spécification formelle en Z.

1. Introduction

La langage Z a été développé à l'Université d'Oxford à la suite des travaux de Jean René Abrial. C'est un langage formel qui utilise :

- les notions *ensemblistes*, le calcul des propositions (et, ou, non, implication, etc.) et des prédicats (quantificateurs existentiels – il existe - et universel – quel que soit -),
- les *relations* (partie du produit cartésien de plusieurs ensembles) et *fonctions* (relations avec au plus une image par valeur du domaine de définition),
- les *séquences* ou *suites* (fonctions des entiers naturels dans un autre ensemble pour imposer un ordre aux valeurs).

Nous commentons au paragraphe suivant un exemple de document de spécification en Z, utilisant uniquement les aspects ensemblistes.

2. Les ensembles

Le document de spécification contient tout d'abord une introduction informelle du problème traité.

> *Introduction*
> Cette spécification concerne l'enregistrement des passagers à bord d'un avion. Les places ne sont pas numérotées. Les passagers sont autorisés à embarquer selon la règle du 'premier arrivé, premier servi'.

Sont ensuite décrits tous les types et toutes les variables utiles dans la spécification. La spécification utilise des *'ensembles donnés'* dont on ne donne que le nom ('given sets'). Elle utilise aussi une boîte à outils mathématique avec des ensembles prédéfinis comme les entiers naturels ($\mathbb{N}$) ou les booléens.

Types

[PERSONNE]	ensemble des personnes identifiées de manière unique
capacité : $\mathbb{N}$	capacité de l'avion (c'est un entier naturel)

OUIOUNON ::= oui | non (la barre | indique un ou)
REPONSE ::= OK | déjàABord | plein | deuxErreurs

Puis vient la description de l'état du système. Il prend la forme d'un *schéma* avec un nom, des *déclarations précisant les types*, et des *propriétés (prédicats) précisant les valeurs*.

> *Etat du système*
> C'est l'ensemble des personnes à bord. Il ne peut excéder la capacité de l'avion.

Avion

àBord : $\mathbb{P}$ PERSONNE	($\mathbb{P}$ X est un élément de l'ensemble des sous ensembles de X, c'est à dire un ensemble d'éléments de X)
#àBord <= capacité	(#X indique le cardinal de l'ensemble X (nb éléments) ; cette propriété doit toujours être vraie ; c'est un invariant du système).

On décrit ensuite l'état initial du système comme un autre schéma.

Etat initial

Init

Avion	Référence au schéma Avion
àBord = ∅	L'avion est vide (ensemble vide)

On décrit les opérations qui peuvent faire évoluer l'état du système : embarquer, débarquer. Les schémas correspondants font référence au *schéma DAvion* qui indique *l'évolution du schéma Avion avant et après l'appel d'une opération* (les états après l'opération sont marqués par une apostrophe).

ΔAvion
àBord : ℙ PERSONNE
àBord' : ℙ PERSONNE

#àBord <= capacité
#àBord' <= capacité

Les opérations

Embarquer
ΔAvion
p? : PERSONNE — p? est une entrée de l'opération
---
p? ∉ àBord
#àBord < capacité
àBord' = àBord ∪ {p?} — ∪ est l'union des ensembles

Débarquer
ΔAvion
p? : PERSONNE — p? est une entrée de l'opération
---
p? ∈ àBord
àBord' = àBord \{p?} — \ est la différence ensembliste

Les interrogations laissent l'état du système inchangé. Ce *qu'indique le Xschéma* ('ksi schéma') suivant :

ΞAvion
àBord : ℙ PERSONNE
àBord' : ℙ PERSONNE

#àBord <= capacité
#àBord' <= capacité
àBord = àBord'

Les interrogations

Nombre
ΞAvion
n! : N — n! est une sortie de l'opération
---
n! = #àBord

Abord

ΞAvion
p? : PERSONNE
réponse! : OUIOUNON

(p? ∈ àBord ^ réponse ! = oui) v (p? ∉ àBord ^ réponse ! = non)

^ et v représentent le 'et' et le 'ou'

Enfin, le traitement des erreurs peut être spécifié dans un plusieurs schémas.

Traitement des erreurs

On donne en exemple le schéma lié aux erreurs d'embarquement.

ErreurEmbarquer

ΞAvion
p? : PERSONNE
rep! : REPONSE

(p? ∈ àBord ^ #àBord = capacité ^ rep! = deuxErreurs) v
(p? ∈ àBord ^ #àBord < capacité ^ rep! = déjàABord) v
(p? ∉ àBord ^ #àBord = capacité ^ rep! = plein)

3. Les fonctions

Les fonctions (ou plus généralement les relations) sont le moyen de *relier les uns aux autres les ensembles utilisés dans la spécification*. Une fonction f est définie dans un ensemble (son *domaine*, noté 'dom f') et prend ses valeurs dans un autre ensemble (son *image*, noté 'ran f' pour 'range'). Une fonction est partielle si toutes les valeurs de départ n'ont pas d'image par la fonction (notation f : X ⇸ Y). Sinon, elle est totale (notation f : X → Y).

(a) L'ajout d'un couple de valeurs à la fonction se note : f' = f ∪ {x ↦ y}

(b) La modification du couple (x, y) par le couple (x, y') se note : f' = f ⊕ {x ↦ y'}

(c) La suppression du couple (x, y) se note : f' = {x} ⩤ (soustraction au domaine).

(d) On donne ci-dessous un exemple de spécification de gestion d'un stock. Le niveau de stock est défini pour les produits stockés. Il peut exister des articles non stockés.

[ARTICLES] L'ensemble des articles (stockés ou non)

Magasin

stockés : ℙ ARTICLE
niveau : ARTICLE ⇸ ℕ (seuls les articles stockés ont un niveau)

dom niveau = stockés

Init

Magasin — Référence au schéma Magasin

---

stockés = ∅ — Initialement,
niveau = ∅ — il n'y a rien en stock

SortieStock

ΔMagasin
a? : ARTICLE
qté? : ℕ

---

a? ∈ stockés
niveau a? >= qté?
niveau' = niveau ⊕ {a? ↦ niveau a? – qté?}
stockés' = stockés

EntréeStock

ΔMagasin
a? : ARTICLE
qté? : ℕ

---

a? ∈ stockés
niveau' = niveau ⊕ {a? ↦ niveau a? + qté?}
stockés' = stockés

NouvelArticle

ΔMagasin
a? : ARTICLE

---

a? ∉ stockés
niveau' = niveau ∪ {a? ↦ 0 }
stockés' = stockés ∪ {a?}

AbandonArticle

ΔMagasin
a? : ARTICLE

---

a? ∈ stockés
niveau a? = 0 — (le niveau doit être nul)
stockés' = stockés \ {a?}
niveau' = {a ?} ⩤ niveau

Nous ne poursuivons pas plus avant la description du langage Z, qui possède bien d'autres concepts et notations, comme par exemple les suites (fonctions de ℕ dans X) et l'utilisation de la logique des prédicats (avec les quantificateurs existentiel ∃ et universel ∀) à la place de la logique des propositions.

Aujourd'hui, la *méthode B et son langage B*, également développés par J.R. Abrial, tend à remplacer la Z. B s'intéresse *au processus qui va de la spécification formelle au programme*. La méthode B offre :

- un *langage de spécification* à base de machines abstraites,
- une technique de *raffinage des spécifications* (des notions abstraites aux notions des langages de programmation),
- des obligations de *preuve* associées à chaque étape,
- *un outil* permettant de supporter ce processus (l'atelier B, Steria).

Elle a pu être appliquée dans quelques projets de taille réelle, comme le système Météor (ligne 14 du métro Parisien) avec 100 000 lignes de B compilées en 87 000 lignes de code ADA et 28 000 preuves.

EXERCICES

Exercice 3.1 : *ensembles*

On a enregistré des personnes comme utilisateurs d'un système informatique. A un moment donné, certains utilisateurs sont connectés à l'ordinateur (une seule fois) et d'autres non.

Spécifier en Z le système 'ordinateur' avec les opérations AjouterUtilsateur, SupprimerUti-lisateur, Connexion, Déconnexion, et les erreurs correspondantes.

Exercice 3.2 : *ensembles*

Soit l'extrait de spécification suivant, décrivant un terminal d'ordinateur.

Types

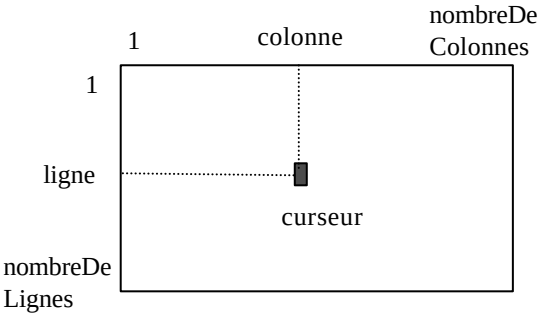
[TOUCHES] Caractères habituels plus touches spéciales suivantes :
gauche, droite, haut, bas (flèches de déplacement), début (home), entrée : TOUCHES

Etat du système

Curseur

ligne : ℕ (entiers naturels)
colonne : ℕ

ligne ∈ 1 .. nombreDeLignes
colonne ∈ 1 .. nombreDeColonnes



Etat initial

Init

Curseur

ligne = 1
colonne =1

Opérations

ToucheDébut

ΔCurseur
touche ? : TOUCHE

touche ? = début
ligne' = 1
colonne' =1

où les schéma ΔCurseur est le schéma conventionnel d'évolution du curseur défini par

ΔCurseur

ligne, ligne' : ℕ
colonne, colonne' : ℕ

ligne, ligne' ∈ 1 .. nombreDeLignes
colonne, colonne' ∈ 1 .. nombreDeColonnes

1. Définir sur le même modèle, le schéma de la touche droite (flèche à droite) ; attention aux divers cas liés à la fin de ligne (retour en début de ligne suivante) et à la fin de l'écran (retour en première ligne, première colonne).
2. Définir le schéma de l'opération LignesRestantes qui retourne le nombre de lignes restantes au dessous du curseur.

Exercice 3.3 : *fonctions*
Un système enregistre les réservations de chambres d'hôtel pour une nuit. Etant donné les types de base
[CHAMBRE] ensemble des chambres
[PERSONNE] ensemble de toutes les personnes possibles
on peut représenter l'état des réservations de l'hôtel par le schéma suivant :

Hôtel

$$\text{RéservéPour} : \text{CHAMBRE} \pfun \text{PERSONNE}$$

1. Pourquoi ReservéPour est-il une fonction, partielle ?
2. Donner l'état initial.
3. Définir les schémas de AccepterRéservation et de AnnulerRéservation.

Exercice 3.4 : *fonctions*
a. Expliquer la signification de la spécification Z suivante et de son schéma d'initialisation.

Types
[PERSONNE, BUREAU, TELEPHONE]

État

Annuaire

$$\begin{aligned}
&\text{Employés} : \mathbb{P}\ \text{PERSONNE} \\
&\text{Bureaux} : \mathbb{P}\ \text{BUREAU} \\
&\text{NumérosPossibles} : \mathbb{P}\ \text{TELEPHONE} \\
&\text{TéléphoneDuBureau} : \text{BUREAU} \pfun \text{TELEPHONE} \\
&\text{BureauDeEmployé} : \text{PERSONNE} \pfun \text{BUREAU} \\
&\text{PlacesLibres} : \text{BUREAU} \pfun \mathbb{N} \\
\hline
&\dom \text{TéléphoneDuBureau} = \text{Bureaux} \\
&\ran \text{TéléphoneDuBureau} \subseteq \text{NumérosPossibles} \\
&\dom \text{BureauDeEmployé} = \text{Employés} \\
&\ran \text{BureauDeEmployé} \subseteq \text{Bureaux} \\
&\dom \text{PlacesLibres} = \text{Bureaux}
\end{aligned}$$

InitAnnuaire

$$\begin{aligned}
&\text{Annuaire} \\
\hline
&\text{Employés} = \varnothing \\
&\text{Bureaux} = \varnothing \\
&\text{NumérosPossibles} = \varnothing \\
&\text{TéléphoneDeBureau} = \varnothing \\
&\text{BureauDeEmployé} = \varnothing \\
&\text{PlacesLibres} = \varnothing
\end{aligned}$$

b. Expliquer l'opération "EngagerEmployé"

EngagerEmployé

Δ Annuaire
p? : PERSONNE
b? : BUREAU

$p? \notin$ Employés
$b? \in$ Bureaux
PlacesLibres b? > 0
Employés' = Employés $\cup$ {p?}
Bureaux' = Bureaux
NumérosPossibles' = NumérosPossibles
TéléphoneDuBureau' = TéléphoneDuBureau
BureauDeEmployé' = BureauDeEmployé $\cup$ {(p? $\mapsto$ b?)}
PlacesLibres' = PlacesLibres $\oplus$ {(b? $\mapsto$ PlacesLibres b? - 1)}

c. En vous inspirant de l'opération précédente, écrire l'opération "LicencierEmployé".

d. Écrire l'opération "AjouterBureau".

ANNEXE

On trouvera ci-après un exemple "réaliste" de spécification formelle d'une bibliothèque.

Notations supplémentaires utilisées dans l'exemple :

$\mathbb{F}$ équivaut à $\mathbb{P}$ mais avec l'idée de partie finie. Un élément de $\mathbb{F}$ est un ensemble fini.
$f^{\sim}$ est la fonction inverse de f.
$f(|S|)$ est l'image de l'ensemble S par la fonction f.
$\widehat{=}$ indique la composition de schémas.

Remarques :

Book est un livre au catalogue.
Copy est un exemplaire de livre. Il peut être available (disponible) ou checkedout (sorti/emprunté).
Borrower est l'ensemble des emprunteurs. Staff est l'ensemble des membres du personnel de la bibliothèque.
MaxCopiesAllowed est le nombre maximum d'emprunts simultanés autorisés.