Allée de la Réflexion

Z, c'est quoi ce machin ?

Henri Habrias fait partie de l'équipe « Méthodes et Génie Logiciel » de l'IRIN (Institut de Recherche en Informatique de Nantes).

Introduction

Z [LIGHTFOOT 94] [RATCLIFF 94] [SPIVEY 94] [BOWEN 94] est une notation formelle pour spécifier un système d'information. Elle a pour origine les travaux de J.R. Abrial, travaux qui ont conduit à la méthode et aux outils B [ABRIAL 94].

Z utilise la théorie des ensembles typés et la logique des prédicats du premier ordre [GOCHET 94], des choses étudiées dans les cours de mathématiques pour l'informatique dispensés aux étudiants des IUT [JACQUEMIN 94].

Nous allons illustrer une petite partie de Z avec un exemple qui est un de nos sujets de Travaux Dirigés de première année à l'IUT de Nantes.

Le machin dans le machin

Le député Mr Santini interrogé, au sujet de la loi sur la défense de la langue française, par une journaliste de radio qui lui demandait comment il traduirait "Flirt", a répondu :

« Je donnerais une définition qu'on utilisait quand j'étais jeune :
Le machin dans la main
La main dans le machin
Mais jamais le machin dans le machin. »

Les grandes questions...

1) Si on veut stocker les flirts dans une base de données, quel sera le schéma relationnel de la base ?
2) Etant donnée votre réponse à la question 1, est-ce que la vérification du respect de la définition de Mr Santini pourra être faite par le système automatique ?
3) On décide de stocker les comportements amoureux que l'on observe à un instant donné. Lorsqu'Olga dit à Claudia qu'actuellement elle a deux flirts, Olga ne veut pas dire qu'à l'instant où elle parle elle est en train de flirter avec deux garçons. A l'instant où Olga parle, Olga ne flirte pas. Les flirts seront déduits de l'état du système. Faites la spécification en Z.
4) Faites le schéma Z de l'entrée d'un nouveau flirt.
5) Faites le schéma Z de la question « Est-ce un flirt ? ». On entre une relation amoureuse et on répond par oui ou par non.
6) Faites le schéma Z de la question « Quels sont les flirts ? »

... et leurs réponses

Réponse à la question 1

Si on veut stocker le minimum, on stockera la relation :

Flirt (Garçon, Fille)

Réponse à la question 2

Mais alors, la vérification du respect de la définition sera extérieure au système automatique.

Réponse à la question 3

[FILLE, GARCON]

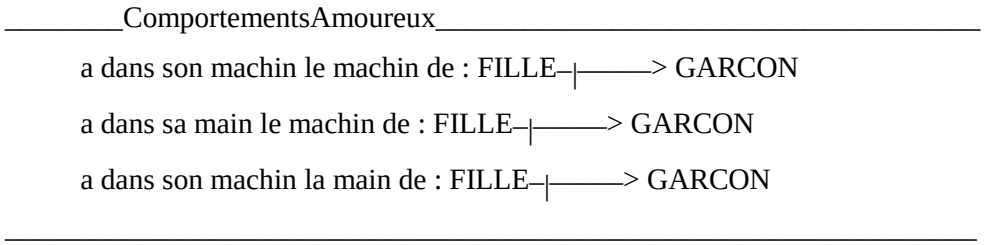
On a deux ensembles de base GARCON et FILLE. Il s'agit de l'ensemble des garçons et de l'ensemble des filles, passés, présents et futurs. On n'a pas le droit d'écrire, par exemple, qu'un ensemble d'éléments de type FILLE est un sous-ensemble d'un ensemble d'éléments de type GARCON.

En Z, les ensembles sont typés. Ainsi, si l'on considère deux types : Chevaux et Etudiants, *l'ensemble des étudiants idiots* et *l'ensemble des chevaux à dix têtes* ne sont pas des ensembles égaux bien qu'étant tous les deux vides.

On peut construire des types avec l'opérateur P qui donne l'ensemble des sous-ensembles d'un ensemble et avec le produit cartésien. (opérateur $\times$)

Au lieu d'écrire *a dans son machin le machin de* : FILLE–|———>GARCON comme ci-après, nous pouvions écrire :

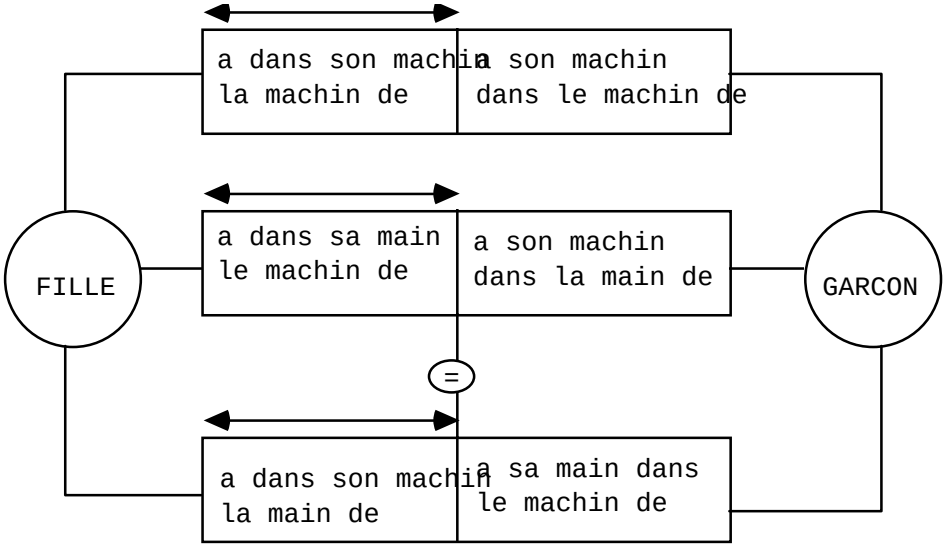
a dans son machin le machin de : P (FILLE $\times$ GARCON) et ajouter dans la partie prédicative du schéma, un prédicat indiquant que cette relation est une fonction.



On vient de définir ce qui s'appelle, en Z, un schéma. Il a pour nom *ComportementsAmoureux*. On y a déclaré trois variables : *a dans son machin le machin de, a dans sa main le machin de, a dans sa main le machin de*.

Ces variables ont un type indiqué après les deux points. Leur type est une fonction de FILLE vers GARCON.

C'est une relation qui est une fonction partielle (c'est ce qu'indique le trait vertical coupant la flèche) : c'est-à-dire que toutes les filles ne sont pas dans la fonction. Respectant les bonnes mœurs nous ne rentrerons pas plus dans le commentaire de l'interprétation de cette fonction.



Remarquons que nous n'avons pas décrit, sur notre schéma Z, — mais nous le pouvions ! , il y en a onze — tous les ensembles qui apparaissent sur le schéma NIAM [HABRIAS 88] ci-dessus. Nous n'avons décrit que les relations de FILLE vers GARCON.

_____SchémaFlirts_____

ComportementsAmoureux

_____

a dans son machin le machin de = a dans son machin la main de

a dans son machin le machin de = ∅

_____

On vient de définir un Schéma *SchémaFlirts*. On y a inclus le schéma *ComportementsAmoureux*. C'est comme si on avait recopié les déclarations de ce dernier[5]. Au-dessous du trait horizontal, on écrit des prédicats. Le saut de ligne exprime un *et* logique.

Il faut bien prendre conscience que nous avons écrit des prédicats. Le signe = est le signe d'égalité. Ce n'est pas le signe d'affectation !

Quand on écrit en langage PASCAL :

x := 4;

x := 8;

L'ordre de l'écriture est important. Le point virgule représente le séquencement. A la fin de l'exécution de ce petit bout de programme, on aura 8 dans x.

[5] *Dans notre exemple. Plus généralement, lors d'une inclusion de schéma dans un autre, on recopie dans ce dernier les déclaration et les prédicats du schéma inclus.*

Quand on écrit en Z :

x = 4

x = 8

C'est la même chose que x=4 /\ x=8. En l'occurrence, il s'agit d'une proposition qui est fausse.

Cela est souvent vite oublié par les débutants. Voici un petit exercice [Rann 94] pour bien se persuader de la différence entre l'écriture d'une séquence et celle d'un prédicat.

Soit la spécification Z suivante :

---

n' = m + 1

m' = n

Est-ce que le code suivant est une bonne implantation :

s_new.n := s.m + 1;

s_new.m := s.n;

Soit le schéma :

---

n' = m' + 1

m' = n

Est-ce que le code suivant est une bonne implantation ?

s_new.n := s_new.m + 1;

s_new.m := s.n;

A la première question, la réponse est oui, à la deuxième la réponse est non.

Si m=2 et n=4, on s'attend à avoir m'=4 et n'=5.

Or l'implantation donne m'=4 et n'=3.

Pour implanter directement, il aurait fallu remplacer

n' = m' + 1

m' = n

par

n' = n + 1

m' = n

en remplaçant m' de la première ligne par n (on peut le faire : la deuxième ligne indique que m'=n).

Quand on a écrit :

a dans son machin le machin de = a dans son machin la main de
a dans son machin le machin de = ∅

dans la partie prédicative du *SchémaFlirts*, on a écrit un prédicat :

a dans sa main le machin de = a dans son machin la main de /\ a dans son machin le machin de = Ø

qui est une contrainte que l'on impose. Ce prédicat doit être vrai.

En Z, nous n'avons pas la notion de séquence, laquelle est liée à l'implantation sur une machine de type Von Neuman.
Avec une spécification à la Z qui utilise les mathématiques et la logique, on peut faire des démonstrations.

_____EtatInitialComportementsAmoureux_____

_____

a dans sa main le machin de = ∅
a dans son machin la main de = ∅
a dans son machin le machin de = ∅

_____

On vient de définir l'état initial de notre système.

Réponse à la question 4

_____EntréeNouveauFlirt_____

Δ SchémaFlirts
g ? : GARCON
f ? : FILLE

_____

a dans sa main le machin de' = a dans sa main le machin de ⊕ {(f?,g?)}
a dans son machin la main de' = a dans sa machin la main de ⊕ {(f?,g?)}

_____

La notation Δ SchémaFlirts est la même chose que :

_____Δ SchémaFlirts_____

a dans son machin le machin de: FILLE–|——> GARCON

a dans son machin le machin de': FILLE–|——> GARCON

a dans sa main le machin de : FILLE—|—> GARCON

a dans sa main le machin de' : FILLE—|—> GARCON

a dans son machin la main de : FILLE—|——> GARCON

a dans son machin la main de' : FILLE—|——> GARCON

---

a dans sa main le machin de = a dans son machin la main de
a dans sa main le machin de' = a dans son machin la main de'
a dans son machin le machin de = ∅
a dans son machin le machin de' = ∅

Les variables décorées avec un prime représentent l'état après.

Les variables décorées avec un point d'interrogation sont des entrées.

Le signe ⊕ représente l'écrasement (*overriding*). Nous en donnons un exemple :

{ Julie |–> Dudule, Paulette |–> Aristide) ⊕ { Julie |–> Duchnock } =

{ Julie |–> Duchnock, Paulette |–> Aristide }

Au sujet de la notation, on aurait pu écrire :

Paulette |–> Aristide : (Paulette, Aristide), (f?, g?) : f? |–> g?

On remarquera que l'écrasement nous donne bien comme résultat une relation qui est une fonction.

Réponse à la question 5

REPONSE ::= oui|non

REPONSE est un type dit libre (*free type*). Nous sommes en effet libre d'énoncer ses valeurs, contrairement aux types de base pour lesquels nous ne pouvons énoncer les valeurs : les garçons ou les filles futurs nous ne les connaissons pas, ni d'ailleurs tous les garçons ou les filles passés.

_____Est-ce-un-flirt?0_____

ΞSchémaFlirt
g ? : GARCON
f ? : FILLE
r ! : REPONSE

((f?, g?) ∈ a dans sa main le machin de)
⇔ (r! = oui)

Le Ξ, qui ressemble à ≡, veut dire que l'état n'est pas modifié par l'opération que l'on est en train de définir. Il nous évite d'avoir à écrire toutes les variables de Flirt avec et sans prime dans la partie déclaration du schéma et d'avoir à écrire, dans la partie prédicative, (au-dessous du trait horizontal) des prédicats disant que l'ensemble avant l'opération est égal à l'ensemble après l'opération.

Peut-être, n'est-il pas mauvais de rappeler la signification de quelques connecteurs logiques [Jacquemin 94] :

p	**q**	**p ∧ q**	**p ∨ q**	**p => q**	**p ⇔ q**
1	1	1	1	1	1
0	1	0	1	1	0
1	0	0	1	0	0
0	0	0	0	1	1

Peut-être auriez-vous écrit le schéma :

_____Est-ce-un-flirt?1_____

ΞSchémaFlirt
g ? : GARCON
f ? : FILLE
r ! : REPONSE

_____

((f?, g?) ∈ a dans sa main le machin de)
⇒ (r! = oui)

_____

Mais dans ce cas, lorsque (f? g?) ∉ a dans sa main le machin de. (voir la table de vérité ci-dessus pour l'implication ⇒) la réponse pourra être oui ou non. L'implanteur pourra choisir la réponse.

C'est ce qu'on appelle une spécification lâche (*loose* en anglais) [Habrias 94].

Peut-être auriez-vous écrit le schéma :

_____Est-ce-un-flirt?2_____

ΞSchémaFlirt
g ? : GARCON
f ? : FILLE
r ! : REPONSE

_____

((f?, g?) ∈ a dans sa main le machin de)
/\ (r! = oui)

_____

Mais dans ce cas, *Est-ce-un-flirt?2* sera indéfini lorsque (f? g?) ∉ a dans sa main le machin de (voir la table de vérité du /\ ).

p	**q**	**p ∧ q**
1	1	1
0	1	0
1	0	0
0	0	0

Cela veut dire que toute implantation ne devra pas donner de réponse.

Et qu'en est-il de notre première spécification *Est-ce-un-flirt?0* ?

Si (f? g?) ∉ a dans sa main le machin de, ((f?, g?) ∈ a dans sa main le machin de) ⇔ (r! = oui) est vrai si r! = oui est faux (voir la table de vérité du ⇔).

p	**q**	**p ⇔ q**
1	1	1
0	1	0
1	0	0
0	0	1

Or comme r ! est du type libre REPONSE, on peut inférer que, de r! ≠ oui, r! = non.

Est-ce-un-flirt?0 est donc totalement défini **.**

On vient d'illustrer deux types de ce qu'on peut appeler des sous-spécifications : la spécification lâche (*loose*) (la spécification établit qu'une variable prend une unique valeur sous certaines circonstances, mais ne spécifie pas qu'elle est cette valeur), la spécification indéfinie (*undefined*) (il n'y a pas de valeur qui remplit la spécification.).

Il y a un troisième type, la spécification non déterministe (*non deterministic*) : la spécification autorise plusieurs valeurs, et ne contraint pas celle qui est choisie à être toujours la même. Au début d'un processus de spécification, on peut très bien être indéterministe. Voici un exemple donné par J.R. Abrial : à une certaine étape de la spécification d'un modèle de maison, on peut écrire : que la maison ait un toit en ardoise ou un toit en tuile (précondition), il n'y a pas d'eau dans la maison (invariant). On n'a pas encore déterminé le matériau.

En B, cela s'écrit ainsi :

[Toit en tuiles [] Toit en ardoises] Il n'y a pas d'eau dans la maison.

Le symbole [] est celui du choix non déterministe.

Selon un des axiomes des "substitutions généralisées" de B [Abrial 94], on a :

[Toit en tuiles] Il n'y a pas d'eau dans la maison /\ [Toit en ardoises] Il n'y a pas d'eau dans la maison.

Au premier abord, on est surpris de voir un choix s'exprimant avec un /\. Pourtant, c'est bien ce que l'on veut. On veut que si le toit est en tuiles, il n'y ait pas d'eau dans la maison et que si le toit est en ardoises, il n'y ait pas d'eau dans la maison. Si on avait mis un \/, cela aurait voulu dire, par exemple, qu'avec un toit en ardoises, il aurait pu y avoir de l'eau dans la maison

Réponse à la question 6

_____QuelsSontLesFlirts?_____

ΞComportementsAmoureux

Flirts!: FILLE ——|——>GARCON

_____

Flirts! = (a dans sa main le machin de ∩ a dans son machin la main de)/a dans son machin le machin de

_____

La variable Flirt décorée avec ! est une variable de sortie.

Ici elle est du type fonction partielle de FILLE vers GARCON.

Dans la partie prédicative, on a écrit, pour respecter la définition de Mr Santini, que l'ensemble des flirts est égal aux couples qui sont à la fois dans la relation *a dans sa main le machin de* et *a dans son machin la main de* mais qui ne sont pas dans la relation *a dans son machin le machin de*.

∩ représente l'intersection ensembliste et / la différence ensembliste.

Vous pouviez aussi utiliser le schéma *SchémaFlirt* ainsi :

```
_____QuelsSontLesFlirts?_____________________________________
    ΞSchémaFlirt
    Flirts!: FILLE ——|——>GARCON

_____________________________
    Flirts! = a dans sa main le machin de

_____________________________________________________________
```

Nous vous proposons maintenant de lire ce dialogue entre Huguette et Sophie :

Huguette : *Toujours autant de flirts Sophie ?*

Sophie : *Actuellement, j'ai Pierrot et Julot, et toi Huguette, flirtes-tu toujours avec Lulu et Zoé ?*

Huguette : *J'ai rompu avec Lulu et je n'ai pas fleurté avec Zoé depuis deux mois.*

Sophie : *Et, deux mois, on en convient tous c'est la limite à partir de laquelle on considère que le flirt est fini.*

Huguette : *Et, oui. Je n'ai pas de flirts actuellement.*

Sophie : *Pourtant, avec Dudule, il me semblait que…*

Huguette : *Oui, mais ça va plus loin que le flirt !*

et de refaire la spécification en conséquence.

***Nota bene** :*

Nous n'avons traité ici que d'une très petite partie de Z.

Nous n'avons pas traité du raffinage qui consiste à passer d'un niveau d'abstraction à un autre en se rapprochant petit à petit de l'implantation en changeant de représentation (par exemple, en implantant un ensemble par un array, qui est lui même analysé comme une fonction), en réduisant le non déterminisme (ce qui sera implanté devra être déterministe !), en introduisant la séquence (on a vu un petit exemple) et la boucle. Nous n'avons pas traité des preuves, preuve de la spécification, preuve du raffinage.

La méthode B [Abrial 94] est de la même famille que Z.

Mais elle utilise le concept de machine abstraite et traite complétement des preuves et du passage au code. Plusieurs applications ont été réalisées à ce jour en France, en particulier pour la RATP [Habrias 93]. Il existe un B User Group sur Internet que vous pouvez conctacter via mariano@inrets.fr. Il se réunit mensuellement à Paris.

Remerciements

Merci à Pascal Bernard pour ses remarques sur une première version de ce papier, aux étudiants de l'EPFL de Lausanne pour leurs questions, à Monsieur le député Santini pour nous avoir procuré un sujet sortant du classique carnet d'anniversaire.

Conclusion

Plus de 20 livres ont été publiés sur Z [HABRIAS 95].

Z est d'origine française.

B. Meyer dans son livre *Méthodes de programmation*, publié avec M. Baudouin en 1980, Delobel et Adiba dans leur livre *Bases de données et systèmes relationnels* publié en 1983, utilisaient une première ébauche de Z.

Curieusement, en France, ces livres n'ont guère influencé les enseignants[6] de ce qui s'est appelé sur les emplois du temps de nos départements informatiques "Analyse et conception de systèmes d'information".

Faudra-t-il que Z nous soit imposé par la normalisation internationale pour que nous nous y intéressions ? Peut-être qu'il faut profiter de notre retard pour passer directement à B ? [ABRIAL 95]

Les étudiant d'IUT doivent ingurgiter (mais qu'en font-ils ?) des mathématiques pour l'informatique, de la logique, des bases de données, etc. Et comme ce sont des étudiants d'Institut de Technologie, bien sûr, ils sont censés pouvoir utiliser, maîtriser cela. Leurs maîtres leur donnent-ils l'exemple de l'utilisation de tout ça ?

D'un côté, l'un parle de niveau logique, de dépendance fonctionnelle[7], de cardinalité, de contrainte d'intégrité référentielle, d'un autre côté, l'autre (parfois le même !) parle de logique, de relation, de fonction, d'inclusion d'ensembles.

Quand inscrirons-nous au fronton de nos établissements le principe du rasoir d'OCCAM[8] rappelé par J. Arsac dans plusieurs de ses livres :

" ENTIA, NON MULTIPLICANDA SUNT PRAETER NECESSITATEM "

" IL NE FAUT PAS MULTIPLIER LES ENTITES[9] SAUF NECESSITE " ▲

Henri Habrias

---

6 Nous ne nous adressons ici qu'à nos collègues enseignants. Un certain nombre de sociétés ont dès à présent compris l'enjeu… mais elles ne le crient pas sur les toits. Si les enseignants attendent que ça "sorte" dans les journaux "grand public" des informaticiens, pour commencer à l'enseigner, on aura pris un retard fort dommageable dans la situation de concurrence actuelle.

7 Et même de CIF, Contrainte d'intégrité fonctionnelle.

8 Guillaume d'Occam (XIV^e^). OCCAM est aussi le nom du langage de programmation des Transputer, langage dont, justement, une partie a été spécifiée en Z.

9 A ce sujet, les types de base dans une spécification Z sont en général très peu nombreux. Comme Z travaille avec des ensembles typés, la discussion sans fin et pseudo-philosophique que l'on a en Merise pour savoir si on a des entités date de commande, date de facture, date de naissance, date de décès ou si on a une entité date, et des propriétés (pourquoi pas des relations entre Date et Commande, Date et Facture, etc. ?) apparaît en Z comme un faux problème. Si on dit que Date de naissance et Date de décès sont deux ensembles de base (deux types de base), alors on ne peut exprimer que telle personne est née à une date où une autre est décédée !

Bibliographie

[ABRIAL 94] J.R. ABRIAL, *Introduction à la méthode B*, 6 cassettes vidéo d'une heure, Réalisation IUT de Nantes, Diffusion Teknea, 1994

[Abrial 95] J.R. Abrial, *Assigning programs to meaning*, Cambridge University Press, à paraître

[BOWEN 94] J.P. BOWEN, J.A. HALLS (Eds), *Z User Workshop*, Cambridge 1994, Springer-Verlag, 1991, ISBN : 3-540-19884-9

[CHAUVET 94] J.Y. CHAUVET., De Z à CAML, rapport Equipe Méthodes et Génie Logiciel, IRIN, IUT de Nantes, 1994

[GOCHET 94] P. GOCHET, P. GRIBOMONT, Logique, Méthodes formelles pour l'étude des programmes, Paris, Hermes, ISBN : 2-86601-395-6

[HABRIAS 88] H. HABRIAS, *Le modèle relationnel binaire, méthode I.A. (Niam)*, Paris, Eyrolles, 1988, ISBN : 2-12-08169

[HABRIAS 93] H. HABRIAS, *Introduction à la spécification*, Préface de H. GALLAIRE, Présentation de M. JACKSON, Paris, Masson, 1993, ISBN : 2-225-82768-0

[HABRIAS 94] H. HABRIAS, *Les spécifications formelles pour les systèmes d'information : quoi ? comment ? pourquoi ?* , in Actes du 10ième congrès Inforsid, 1994, ISBN : 2-906-855-10-3, pp. 1-31, une version corrigée et complétée est parue dans la revue ISI en 1995.

[HABRIAS 95] H. HABRIAS, *Bibliographie commentée d'ouvrages sur Z*, Rapport IRIN, 1995

[HABRIAS 95] H. HABRIAS, *Z Twenty Years on - What is its Future ?*, October 10-12, 1995, Proceedings, IUT de Nantes

[JACQUEMIN 94] C. JACQUEMIN, *Logique et mathématiques pour l'informatique et l'I.A.*, Paris, Masson, 1994, ISBN : 2-225-84642-1

[LIGHTFOOT 94] D. LIGHTFOOT, *La spécification formelle avec Z*, traduction française de *Formal Specification Uzing Z*, traduit par H. HABRIAS et P.M. DELPECH, Toulouse, Teknea (1994), ISBN : 2-87717-038-1

[RANN 94] RANN, TURNER, WHITWORTH, *Z: A Beguinner's Guide*, Chapman & Hall, ISBN : 0-412-55660-X

[RATCLIFF 94] B. RATCLIFF, *Introducing Specification Using Z, A Practical Case Study Approach*, London, McGraw-Hill, ISBN : 0-07-707965-5

[SPIVEY 94] J.M. SPIVEY, *La notation Z*, Traduction de M. LEMOINE, Paris, Masson, 1994, ISBN : 2-225-84367-8

☛ *Pour joindre Henri Habrias,e-mail : Habrias@irin.univ-nantes.fr*