En finir avec la boxologie en revenant aux fondamentaux

Henri Habrias

IUT - Université de Nantes
Département informatique
3 rue Ml Joffre, 44041 Nantes Cedex 1
tel: 33 (0)2.51.12.58.23
henri.Habrias@univ-nantes.fr

Sections de rattachement : 27
Secteur : Tertiaire

RÉSUMÉ. *Il existe un très grand nombre de notations graphiques utilisées pour représenter ce qui est, en fait, un invariant d'état. Nous proposons un retour aux fondamentaux : la logique du premier ordre et le langage des ensembles pour spécifier des invariants d'état. Nous illustrons notre propos par des exemples.*

ABSTRACT. *There are a large number of graphical notations to specify what is in fact a state invariant. We propose to return to basic: the logic and the set language to specify a state invariant. We illustrate this point with examples.*

MOTS-CLÉS : *Spécifications, Invariant, Logique, Langage des ensembles,Méthodes B, Niam, Merise, schéma relationnel n-aire, UML*

KEYWORDS: *Specifications, Invariant, Logic, Set language, B method, Niam, Merise, n-ary relationnal schema, UML*

“ Une science qui accepte les images est, plus que toute autre, victime des métaphores. Aussi l’esprit scientifique doit-il sans cesse lutter contre les images, les analogies, les métaphores. “ G. Bachelard *in La formation du nouvel esprit scientifique, Paris, Vrin*

1. Introduction

Il y a au moins 40 ans (Habrias H. 1977) que sont lancées régulièrement de “nouvelles notations” graphiques pour spécifier des logiciels, citons SADT, MERISE, OMT parmi une cinquantaine. La dernière étant UML. Chacune de ces notations s’accompagne d’un vocabulaire cachant des concepts très simples et fondamentaux. Il est pour le moins étonnant que de vouloir spécifier en se servant de notations dont la sémantique est elle-même non spécifiée. La situation de l’informatique est tout à fait à part dans le domaine de l’ingénierie. Il ne s’agit pas pour nous de rejeter une notation graphique. Une telle notation peut être tout à fait formelle et des logiciens en ont proposé au cours de l’histoire. Pensons à Euler, à Venn, à Peano, à Lewis Caroll, à Pierce ! Mais elle est forcément limitée. Un langage linéaire comme le langage de la logique des prédicats ou celui des ensembles et des relations est plus puissant dans le sens où il permet de dire plus de choses et permet une vérification formelle. Dans cet article, nous illustrons la spécification de relations entre ensembles. Si le retour aux fondamentaux permettait d’éviter de nouvelles fausses nouvelles notations pour dire par exemple qu’une relation est une fonction injective, nous économiserions bien du papier de boxologie (Habrias H. 1987, Habrias H. 1997). Cet article ne présente aucun concept original. Il se limite à la spécification d’un invariant d’état. Il ne fait que plaider pour que la boxologie n’envahisse pas la recherche et l’enseignement. Et c’est une contribution au développement durable, thème fédérateur de ce colloque !

2. Quelques représentations d’une relation

Il existe plusieurs manières de spécifier et de représenter une relation quelconque de manière abstraite. L’informaticien pourra l’implanter de bien des manières. Un développement formel consistera à passer d’une représentation abstraite à une représentation plus concrète pour s’approcher par raffinages successifs à une implantation exécutable par une machine, et à prouver ces raffinages. Dans cet article, nous nous limitons à la spécification abstraite. Nous donnons un exemple de relation de l’ensemble $1..6$ vers l’ensemble $1..6$. Remarquons que notre spécification est insuffisante : nous ne précisons pas quel est le prédicat associé à cette relation. Que représentent ces entiers ? que représente un couple de ces entiers ? Nous en reparlerons en fin de cet article.
Les vertus des dessins sont souvent proclamées par les “modélisateurs” mais il n’existe que bien peu d’études (Lehman 89) pour confirmer ou infirmer ces vertus. Les logiciens avaient pourtant montré la voie en discutant des représentations graphiques (voir par exemple la différence entre diagrammes de Venn et diagrammes d’Euler (Blanché

1996). Il est aussi un domaine proche laissé pour compte, celui du nommage des variables (Habrias Lafaye 1999). T. Hoare (Hoare 1990)a reconnu son importance : *'One of the first discoveries of the research by the Z team was the necessity of separating small chunks of formal material by paragraphs of informal prose, explaining the relationship between the formal symbols and reality, and motivating the decisions that are captured by the formalisation. The drafting of the informal prose was even more difficult to teach, learn and practice than the mastery of mathematical notations and concepts'*. M. Jackson (Jackson 1995) est un des autres rares auteurs à avoir mis l'accent sur ce point en ce qui concerne l'étiquettage des diagrammes (il prend le cas des diagrammes états-transitions). Avec un humour tout britannique, il écrit : *" The difficulty is that the graphical notation itself is perfectly clear, but the meaning of boxes and diamonds and lines is very obscure and uncertain. "*

2.1. *Cinq représentations d'une relation en extension*

Voici illustrées par un exemple, cinq représentations que nous qualifierons de "scolaires".

– Notation linéaire ensembliste $\{(1 \mapsto 1), (1 \mapsto 5), (3 \mapsto 6), (4 \mapsto 2), (4 \mapsto 5), (5 \mapsto 1), (6 \mapsto 6)\}$

– Notation sagittale (Fig. 1)

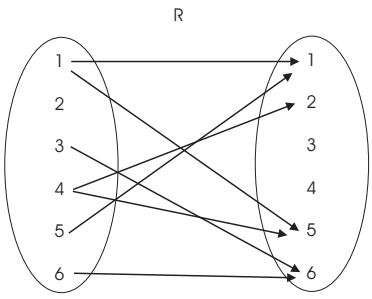

Figure 1. *Diagramme sagittal*

– Notation tabulaire (2). Ce que dans le domaine des bases de données, on appelle une *table relationnelle*. Une table relationnelle peut être considérée comme un ensemble de n-uplets, *i.e* une relation en extension. A ceci près que, le nom de chaque colonne étant porteur de sens, on peut changer la position des colonnes sans modifier la signification, ce qui n'est pas le cas pour un n-uplet, élément d'un produit cartésien. La troisième ligne de la table exemple est l'ensemble singleton $\{(de, 1), (vers, 5)\}$.

– Notation dictionnaire (2)

– Notation matricielle (2) En classe maternelle, à la place des 1 on utilise des $\times$ et à la place des 0, on laisse la case à blanc.

de	vers
1	1
1	5
3	6
4	2
4	5
5	1
6	6

de	vers
1	1, 5
3	6
4	2, 5
5	1
6	6
5	1

	1	2	3	4	5	6
1	×				×	
2						
3						×
4		×				
5	×					
6						×

Figure 2. *"Table relationnelle", Dictionnaire, Matrice*

2.2. *Six représentations de la spécification*

– Notation B : $R \in 1..6 \leftrightarrow 1..6$

– Diagrammes de Bachman (Fig. 3). Le sens inverse des flèches exprime une fonction totale.

– Schéma "conceptuel de données" Merise (Fig. 3).

– Schéma NIAM (Fig. 3)

– Diagramme de classes UML (Fig. 3).

– Schéma relationnel de Codd : $R1(\underline{c1, c2})$

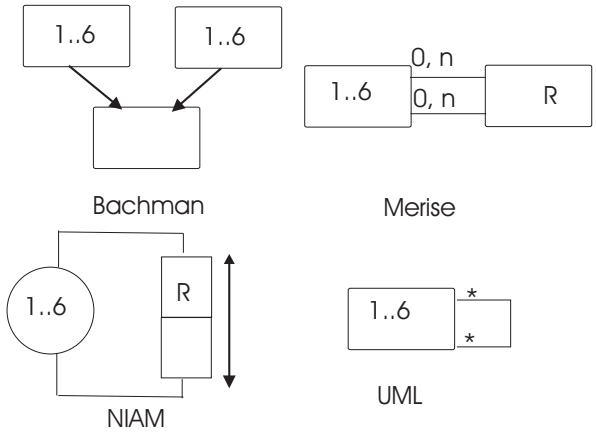

Figure 3. *Diagramme de Bachman, Merise, Niam et UML*

3. Présentation de cinq notations graphiques

Nous présentons quatre notations graphiques qui permettent de représenter les 16

cas de relations binaires. Il s'agit dans l'ordre, des notations NIAM, MERISE, UML, des relations n-aires *à la Codd*. Voici quelques mots sur ces notations.

– NIAM (Habrias 1989) n'utilise que deux symboles : $\forall$ pour la contrainte de totalité (on reconnaît le quantificateur universel) et $\leftrightarrow$ pour la contrainte d'unicité (*i.e.* la fonction). Rappelons que si on représente une fonction sous une forme tabulaire, la fonction allant de la colonne de gauche vers la colonne de droite, (les gens des bases de données, disent que la colonne de gauche est *clé*) on ne trouvera pas deux lignes de la table ayant la même partie gauche. Dit autrement, la colonne de gauche n'aura pas de valeur de ligne écrite deux fois. C'est la contrainte d'unicité. En NIAM, on peut représenter aussi d'autres contraintes ensemblistes. Mais nous ne les traiterons pas ici.

– Merise (Tardieu 1979), sous le nom de *cardinalités* utilise un couple (x, y) dont le premier élément, s'il est égal à 1 représente la contrainte de totalité (il prend la valeur 0 sinon) et dont le deuxième élément prend la valeur 1 si la relation est une fonction (il prend la valeur n sinon). Si par exemple, on a une fonction totale de S vers T, le couple (1, 1) est noté à côté du rectangle S.

– UML (Muller 1997) utilise l'étoile (*) quand Merise utilise (0, n), 1 quand Merise utilise (1, 1), 1..* quand Merise utilise (1, n), 0..1 quand Merise utilise (0, 1). Mais quand, par exemple, on a une fonction totale de S vers T, 1 est mis à côté de T et non à côté de S. Et ces valeurs au lieu d'être appelées des *cardinalités* comme dans Merise, ont pour nom, des *multiplicités*.

– Schéma relationnel n-aire *à la Codd* Cette notation (Codd 1990) est ce que les Merisiens appellent curieusement "le modèle logique de données", alors qu'ils appellent les schémas faits de rectangles, certangles et ficelles, un "modèle conceptuel de données". On ne voit pas en quoi l'un est plus conceptuel ou plus logique ! De quoi perturber l'étudiant à qui on demande de faire des ponts entre les différents modules enseignés en IUT. En NIAM, le schéma relationnel n-aire *à la Codd* s'appelle le "schéma regroupé". En effet, on regroupe les relations binaires ayant même clé, en relations n-aires. Le terme est technique et n'a pas d'autre connotation. On a préfixé par c, le nom des *champs* des schémas de relations,sauf pour les relations unaires (appelées *domaines*). On a souligné les clés. Les flèches représentent une contrainte de sous-ensemble (les gens des bases de données, disent "contrainte d'intégrité référentielle").

4. Les 16 cas de relations binaires

4.1. *Pourquoi 16 ?*

On a deux contraintes : totalité et unicité (ou fonction). Il suffit donc de dux symboles. Pour la relation de S vers T, on peut avoir totalité ou non et, pour chacun de ces cas, unicité ou non. Idem pour la relation inverse de T vers S. Ce qui fait $4 \times 4 = 16$. Remarquons que la notation OMT utilise des symboles quand il n'y a pas de contrainte, ce qui fait qu'avec OMT, lorsqu'il n'y a rien à écrire, c'est là où on écrit le plus.

4.2. *La notation B*

C'est notre *lingua franca*. La notation B (Abrial J.R. 1996, Habrias H. 2001) est facile à mémoriser. Voici son principe. Une fonction totale (ce que les professeurs de maths en France appellent une *application*, est notée (et vous conviendrez que B ne fait pas dans l'originalité, et c'est très bien !) par $\rightarrow$. Si l'on a ni une fonction dans un sens ni dans l'autre sens, alors on supprime toute orientation en utilisant le symbole $\leftrightarrow$. Si la fonction est partielle, on "coupe" l'hampe de la flèche ainsi $\nrightarrow$. Si la fonction "couvre" l'ensemble cible, alors on met une capote à la pointe de la flèche, ainsi $\twoheadrightarrow$. Si on a une fonction dans un sens et dans l'autre sens, on ne peut mettre une pointe d'un côté et de l'autre, on tomberait sur le symbole utilisé pour les relations.Aussi utilise-t-on : $\rightarrowtail$. En utilisant ces principes, on peut couvrir presque tous les cas, presque car pour certains cas, il faut utiliser en plus des contraintes sur le domaine (dom) ou le co-domaine (ran). Ensemble de départ (source), ensemble d'arrivée (cible), domaine, co-domaine (ou range) sont illustrés par la figure (Fig. 4). Figure qui correspond à l'invariant de $maPetiteMachine$ fourni ci-après.

MACHINE $maPetiteMachine$
SETS AA
VARIABLES$v1, v2$
INVARIANT
$v1 \subseteq AA \ \wedge v2 \in v1 \nrightarrow AA$
INITIALISATION
$v1 := \{a, b, c, d\} \parallel v2 := \{(a \mapsto b), (c \mapsto b), (d \mapsto d)\}$ END

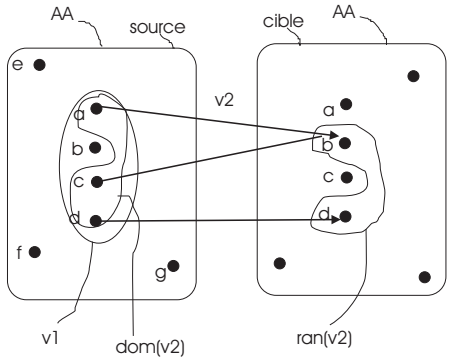

Figure 4. *Fonction partielle, Source, Cible, Domaine and Codomaine (range)*

Nous ne fournissons qu'un des 16 cas, figures (Fig. 5).

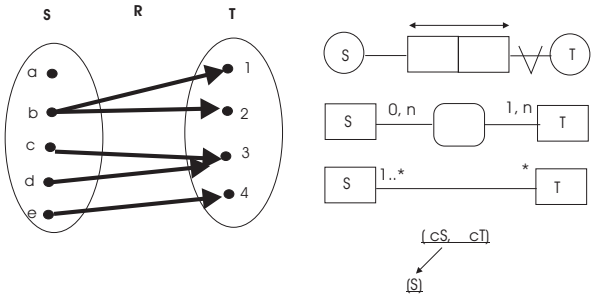

Figure 5. *Cas 1, $R \in S \leftrightarrow T \wedge ran(R) = T$, Relation quelconque de S vers T et de T vers R*

5. Illustration de l'utilité des mathématiques

Voici une définition extraite d'un livre français bien connu sur UML (Muller P-A 1997) : *"Qualification des associations*
La qualification des associations, aussi dénommée restriction d'une association, consiste à sélectionner un sous-ensemble d'objets parmi l'ensemble des objets qui participent à une association. La restriction est réalisée au moyen d'un tuple d'attributs particuliers, appelé qualificatif ou clé, qui est utilisé conjointement avec un objet de la classe source.
Le qualificatif est placé sur l'extrémité d'association au niveau de la classe source, dans un compartiment rectangulaire. Le qualificatif appartient pleinement à l'association et non aux classes associées.
L'instanciation d'une association qualifiée définit le nom des objets source et destination, et la valeur du qualificatif. Ainsi, chaque instance de la classe A, accompagnée de la valeur du qualificatif, identifie un sous-ensemble des instances de B qui participent à l'association. La qualification partitionne l'ensemble d'arrivée et réduit ainsi la multiplicité de l'association. La paire (instance de A, valeur du qualificatif) identifie un sous-ensemble des instances de B.
La restriction d'une association peut être opérée en combinant les valeurs des différents attributs qui forment le qualificatif."
On constate qu'on y trouve des termes mathématiques comme *ensemble, paire*. Pour ce dernier, manifestement l'auteur veut parler de *couple*. J'ai bien eu de peine à comprendre une telle définition. Grace à des exemples, j'ai fini par comprendre que l'auteur voulait dire, je prends un exemple, qu'étant données les deux relations $estDans$ et $aPourN^\circ$, telles que $estDans \in Salles \rightarrow Batiments \wedge aPourN^\circ \in Salles \rightarrow NAT$ il y a "association qualifiée" si $estDans \otimes aPourN^\circ \in Salles \rightarrowtail Batiments \times NAT$. En d'autres termes, le produit direct [1] des deux fonctions doit être

1. Le produit direct, noté par $\otimes$ de deux relations $r1 = \{(1, a), (1, b), (2, a), (4, c)\}$ et $r2 = \{(1, 5), (3, 5), (2, 1)\}$ est égal à $\{(1, (a, 5)), (1, (b, 5)), (2, (a, 1)\}$

une injection. Le $N^{\circ}DeSalle$ est le qualificatif. $N^{\circ}DeSalle = ran(ran(estDans \otimes aPourN))$.

6. Conclusion

*La simplicité ce n'est pas le commencement, c'est l'aboutissement.*faisait remarquer récemment un chef cuisinier romain. 40 ans c'est une longue gestation ! Si l'on s'en tient à l'alphabet unitilisé, la notation la plus simple est la notation NIAM : deux symboles graphiques (un pour la totalité, un pour l'unicité, ce qui fait que l'on peut coder en binaire les 16 cas, en allant du moins contraint (0000) au plus contraint (1111)) et un schéma "lisible" à condition bien sûr qu'on écrive les phrases pour une lecture de l'idée type dans les deux sens (voir figure 6). Mais toute contrainte

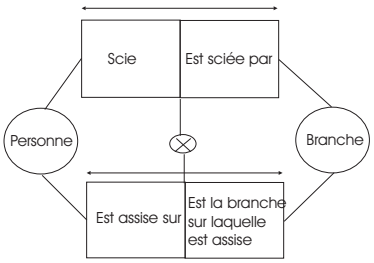

Figure 6. *On ne scie pas la branche sur laquelle on est assis*

ne peut bien sûr s'y exprimer graphiquement. Une notation formelle comme B (langage ensembliste, langage de la logique des prédicats du premier ordre) est alors nécessaire si on veut faire de la validation formelle. Et si l'on veut passer d'une telle notation graphique à la notation B, il faut ajouter quelques conventions (Habrias 2006). Par exemple, alors qu'on peut représenter, sans compliquer la notation graphique, le proverbe "On ne scie pas la branche sur laquelle on est assis", on ne peut le faire pour le précepte suivant "Qui que saille notre jument, le poulain est notre", ou, en d'autres termes, "Le propriétaire du poulain est le propriétaire de la mère du poulain". Ce qui se spécifie simplement comme suit. Etant donné : $aPourProprietaire \in chevaux \leftrightarrow personnes \wedge aPourMere \in chevaux \nrightarrow chevaux$, alors $aPourProprietaire \subseteq aPourProprietaire \fatsemi aPourMere$.$\fatsemi$ représente la "composition en avant", c'est l'inverse du "rond" habituellement utilisé dans les livres de mathématiques en France. On remarquera que du fait de la propriété de la relation de parenté biologique, on ne peut avoir une mère pour tout cheval (sinon, on aurait un ensemble infini de chevaux...ce qui pour un informaticien n'est pas implantable !). La lecture du MOF (Meta Object Facility) Specification, version 1.4 de l'OMG, montre que les mathématiques pour l'informatique commencent à intéresser les spécifieurs puisqu'on y lit *"The sets All-links is the Cartesian Product of the sets (...) This is a Link, which is a member of All-links..."*. On peut espérer que le rasoir d'Occam va contribuer au développement durable, thème fédérateur de cette confé-

rence et que le terme boxologie (“Un problème principal était de faire que notre domaine (le développement de système) soit accepté comme une recherche de première classe. A cette époque, on en parlait fréquemment comme étant de la ’boxologie”’, a écrit K. Nygaard, Prix Turing, créateur de la programmation objet) va quitter le génie logiciel.

7. Bibliographie

Abrial J.-R., *The B-Book, Assigning Programs to Meanings*, Cambridge University Press, 1996

Blanché R., Dubucs J., *La logique et son histoire*, Coll. U, Armand Colin, 1996

Broy M., Denert E. (Eds), *Software Pioneers, Contributions to Software Engineering*, Springer, 2002, ISBN :3-540-43081-4

Chen P.P.S., The Entity-Relationship Model, *ACM TODS*, March 1976, pp. 9-36, repris in Broy M., Denert E., p. 331 et s.

Codd E.F., *The Relational Model for Database Management, Version 2*, Addison-Wesley, 1990, ISBN : 020114192 2

Davenport, C., The role of graphical methods in the history of logic, *Methodos*, 1952,Milano, pages 145-164

Habrias H., Les graphiques de l’analyste, *O1 Informatique*, n˚ 11, juin-juillet 1977

Habrias H., Petit traité de boxologie, *L’Informatique Professionnelle*, n˚54, mai 1987

Habrias H., *Le modèle relationnel binaire, Méthode IA (NIAM)*, Eyrolles, 1989

Habrias H., *Dictionnaire encyclopédique du génie logiciel*, préface de J.P. Finance, Masson, 1997

Habrias H., *Spécification formelle avec B*, Hermes, Lavoisier, 2001

Habrias H., Du semi-formel au formel, un exemple, *Mosim’06*, Rabat, 3-5 avril 2006

Habrias H.,Teaching Specifications, hands on, *FORMED’08 Proceedings*, ETAPS, Budapest, March 2008

Habrias H., Lafaye J.Y., How to Name Variables of a State Based Formal Invariant, An experimental study, *FM’99 - Formal Methods, World Congress on Formal Methods, Toulouse, Sept. 99*, Proceedings, vol. 1, vol. 2, LNCS 1708, 1709, ISBN : 3-540-66587-0

Hoare C.A. R., Préface à *VDM’90, VDM and Z - Formal Methods in Software Developement*, LNCS, 1990, ISBN 3-540-52513-0 Jackson M., *Software Requirements & Specifications, a lexicon of practice, principles and prejudices*, Addison-Wesley, 1995, ISBN : 0-201-87712-0

Lehman J.A., An Empirical Comparison of Textual and Graphical Data Structure Documentation of Cobol Programs, *IEEE TOSE*, Vol. 15, N˚ 9, Sept. 1989, p. 1131-1135

Muller P-A., *Modélisation objet avec UML*, Eyrolles, 1997

Tardieu H et al., *Conception d’un système d’information*, Editions d’organisation, 1979