Au sujet des abus de langages en informatique

Dominique Duval

Résumé.

Les langages informatiques comportent de nombreux abus, c'est-à-dire des situations où les règles qui permettent d'écrire les programmes ne suffisent pas à décrire le sens de ces programmes : la syntaxe et la sémantique ne concordent pas. C'est le cas, par exemple, des "effets de bord", susceptibles de modifier l'état d'une machine, sans que cela apparaisse clairement dans l'écriture du programme. Dès qu'on souhaite appréhender l'informatique sous un angle théorique, ces abus de langages s'avèrent très gênants, et il est tentant de les supprimer, en se ramenant à une situation complètement explicite où syntaxe et sémantique concordent parfaitement. Cependant, ces abus de langages fournissent des informations intéressantes : typiquement, ce qui est caché est d'une autre nature que ce qui est montré, et en montrant tout on perd ce type d'informations.

Dans cet exposé, nous présentons un travail fait avec Christian LAIR (Université de Paris 7), qui montre que très souvent, en fait, ces abus de langage sont organisés selon une structure algébrique précise. On peut alors, pour les expliciter, utiliser une construction algébrique voisine d'un produit tensoriel. Mais surtout, il devient possible de prendre en compte directement les aspects implicites des langages informatiques, dans un formalisme algébrique adapté.

Spécifications.

L'approche de la *théorie des esquisses*, comme l'approche (plus classique en informatique) des *spécifications algébriques*, est essentiellement graphique. Plutôt que d'utiliser des formules logiques, cette approche fait appel à des graphes orientés, enrichis par de la composition de flèches, des équations entre flèches, et des *contraintes* permettant de traduire des propriétés non équationnelles (et qui, techniquement, sont aussi constituées de graphes). Un tel "graphe enrichi" est une *esquisse*, au sens de C. Ehresmann [Ehresmann 66], [Ehresmann 68], ou, plus généralement, une *trame*, au sens de C. Lair [Lair 87]. Une trame est une donnée formelle, ou *syntaxique*, dont le sens, *i.e.* la *sémantique*, est fourni par une (ou des) *réalisation(s)*. La plupart du temps, une telle réalisation est *ensembliste*, c'est-à- dire qu'elle interprète chaque point du graphe comme un ensemble, chaque flèche du graphe comme une application, et ainsi de suite, de façon cohérente : la notion de *foncteur* entre *catégories* est sous-jacente à cette notion. Ces questions sont traitées dans [guidel].

La notion d'état.

A titre d'exemple, considérons la notion *d'état* en informatique. Pour plus de précisions, et un exemple détaillé, on pourra se reporter à [state]. Il y a plusieurs façons de spécifier cette notion. En gros, ces méthodes peuvent être réparties en deux familles, selon que l'état y est *explicite* ou *implicite*.

Une *spécification avec état explicite* est une trame $\mathbf{S}^{ex}$ qui comporte un point distingué. Sa sémantique est fournie par une réalisation ensembliste de $\mathbf{S}^{ex}$, dans laquelle le point distingué est interprété comme l'ensemble des états de la machine considérée. Il s'agit donc d'une spécification tout-à-fait classique, mais souvent confuse. et dont il est difficile d'extraire une bonne notion de programme.

Une *spécification avec état implicite* est composée, entre autres, d'une trame $\mathbf{S}^{im}$. Mais celle-ci ne comporte aucun point pour représenter l'état, et ses réalisations ensemblistes n'ont aucun sens. Cependant, la trame $\mathbf{S}^{im}$ fournit une bonne notion de programme, et les états de la machine peuvent être vus comme des réalisations ensemblistes d'une partie de $\mathbf{S}^{im}$. D'autre part, c'est une réalisation *non ensembliste* de $\mathbf{S}^{im}$ qui en donne la sémantique. Pour définir précisément cette réalisation, il faut fournir des informations supplémentaires.

Une spécification avec état implicite est donc formée d'une trame $\mathbf{S}^{im}$ *et* d'autres composants. C'est une *mosaïque*, au sens de [guide2]. Les autres composants servent à préciser la nature de la réalisation qui fournit la sémantique de la trame $\mathbf{S}^{im}$. Pour cela, on utilise *encore* des trames $\mathbf{K}(i)$ (où i parcourt une certaine famille d'indices).

Le produit en ruban.

Considerons une mosaïque avec état implicite, comportant une trame $\mathbf{S}^{im}$ et des trames $\mathbf{K}(i)$. Un résultat fondamental est que :

> *En regroupant convenablement toutes les trames qui composent une mosaïque avec état implicite, il est possible de retrouver la trame avec état explicite* $\mathbf{S}^{ex}$

Cette construction s'appelle le *produit en ruban*, et ce résultat montre que le produit en ruban est, en fait, une forme de *produit tensoriel* [guide2].

Conclusion.

Les trames fournissent un outil de spécification bien adapté aux situations explicites, c'est-à-dire essentiellement à la programmation fonctionnelle. Pour traiter de la programmation impérative, ainsi que de divers autres aspects implicites des langages informatiques, nous proposons un second outil de specification : les mosaïques. Le lien entre les deux approches est fourni par le produit en ruban, qui permet d'expliciter les aspects implicites par une construction algébrique proche du produit tensoriel.

Références

[Ehresmann 66] Charles Ehresmann. Introduction to the theory of structured categories. Technical Report 10, University of Kansas at Lawrence, 1966.

[Ehresmann 68] Charles Ehresmann. Esquisses et types de structures algébriques. *Bulletin de l'Institut Polytechnique, Iasi*, 14, pp. 1-32, 1968.

[Lair 87] Christian Lair. Trames et sémantiques catégoriques des systèmes de trames. *Diagrammes*, 18 :CL1-CL47, 1987.

[ref] Christian Lair and Dominique Duval. Sketches and specifications : reference manual. *Rapport de recherche du LACO*, 2000.

http ://www.unilim.fr/laco/rapports.

[ref1] First part : Compositive graphs.

[ref2] Second part : Projective sketches.

[ref3] Third part : Models.

[guide] Domimque Duval and Christian Lair. Sketches and specifications : User's guide. *Rapport de recherche du LACO*, 2000.

http ://www.unilim.fr/laco/rapports.

[guide1] First part : Wefts for explicit specification.

[guide2] Second part : Mosaics for implicit specification.

[state] Dominique Duval and Christian Lair. Mosaics for specifications with implicit state. *Rapport de recherche du LACO*, 2000.

http ://www.unilim.fr/laco/rapports.

Dominique DUVAL
LACO, Université de Limoges
123 avenue Albert Thomas
F-87060 Limoges Cedex
dominique.duval@unilim.fr
http ://www.unilim.fr/laco/perso/dominique.duval/index.html