Communicating Sequential Processes

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 mai 2010

https://www.bortzmeyer.org/communicating-sequential-processes.html

Auteur(s) : C. A. R. Hoare
ISBN n°0-13-153271-5
Éditeur : Prentice-Hall
Publié en 1985

Pendant toutes les années 1970, un bouillonnement de recherche sur la programmation parallèle a apporté dans la boîte à outils du programmeur tout un tas de concepts nouveaux, qui ont permis d'écrire des programmes non-séquentiels, sinon facilement, du moins sans se prendre les pieds dans le tapis à chaque fois. Le célèbre livre de Hoare est un reflet de cette époque, dont son auteur a été un des grands contributeurs, avec sa théorie des processus séquentiels communiquants.

Toute une génération de programmeurs a appris la programmation parallèle dans ce livre et se souvient des exemples avec une machine à café, comme le client fou (chapitre 2.2) qui insère au hasard une pièce de un ou de deux "*pence*" et se bloque si la machine ne se contente pas d'une seule pièce, si c'est celle de un "*penny*". Ou de la machine à café bruyante (chapitre 2.3) qui sert à illustrer l'indépendance d'événements non synchronisés (entre le « cling » de la machine et le juron du client qui n'a pas eu ce qu'il voulait).

Le livre est solidement mathématique. Il commence doucement, et même lentement, mais est vraiment difficile à suivre vers la fin. À noter que la communication explicite entre processus ne commence qu'au chapitre 4, ce qui donne une idée du souci de l'auteur d'établir des bases théoriques sérieuses avant de commencer à rentrer dans les détails.

L'auteur n'oublie pas pour autant la mise en œuvre et fournit des pistes pour programmer ses processus séquentiels communiquants, dans un dialecte de Lisp. Attention, le langage utilisé utilise les concepts de Lisp mais Hoare lui a donné une syntaxe différente, ressemblant à Pascal, et sans les célèbres parenthèses. Il ne faut pas compter écrire directement un programme avec ces exemples, il faut d'abord trouver un langage exécutable !

Bien des langages ont d'ailleurs repris les concepts de ce livre, le plus connu était à l'époque Occam, qui était censé permettre la programmation facile d'une puce révolutionnaire, le Transputer, et qui n'a pas été un grand succès. Ada a ensuite beaucoup suivi ces idées. Le dernier langage qui les ai reprises explicitement est Go, dont le mécanisme de communication a repris le terme hoarien de "*channel*".

La variété des solutions au problème de la programmation paralléle ne s'est d'ailleurs pas arrêté là et le chapitre 7 discute des solutions alternatives, que ce soit pour la structuration des programmes comme les coroutines ou les moniteurs, ou pour la communication comme les tubes.