Programmation avec des réseaux d'interaction

Sylvain Lippi

7 février 1999

Les réseaux d'interaction introduits par Yves Lafont sont un modèle de calcul distribué inspiré de la logique linéaire de Jean-Yves Girard. Le but de l'exposé est de montrer qu'il s'agit aussi d'un nouveau paradigme de programmation que l'on peut aborder sans connaissance préalable de la logique. Après une introduction aux réseaux d'interaction, on montrera à travers quelques exemples classiques les spécificités de ce paradigme. Enfin, on présentera un interprète graphique permettant d'exécuter pas à pas un programme écrit avec des réseaux d'interaction. A ce niveau de l'exposé, il serait intéressant, si cela est possible, de faire une démonstration sur une machine.

Les réseaux d'interaction sont représentés par des graphes non orientés dont les sommets (les *cellules*) sont étiquetés par des symboles ayant une arité $n \geq 0$. Les cellules sont reliées entre-elles par l'intermédiaire de leurs *ports* ; une cellule d'arité n possède un *port principal* et n *port(s) auxiliaire(s)*. Certains ports ne sont reliés à aucun autre ; on dit qu'ils sont *libres*. La figure 1 donne un exemple de réseau avec un port libre construit avec les symboles $\mathbf{0}$ d'arité 0, $+$ d'arité 2 et $\mathbf{S}$ d'arité 1 (les cellules d'arité 0 sont représentées par des cercles et celles d'arité $n > 0$ par des triangles avec le port principal situé sur un des sommets).

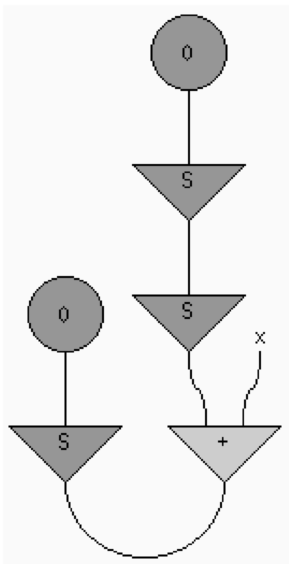

Fig 1 : un réseau d'interaction construit avec les symboles $\mathbf{0}$, $\mathbf{S}$ et $+$

On donne ensuite les *règles d'interaction* donnant les transitions du réseau. Le membre gauche d'une règle est formé de deux cellules reliées par leur port principal (une *coupure*) et le membre droit est un réseau ayant autant de ports libres que le membre gauche. Un programme ou *système d'interaction* est donc composé d'un ensemble fini de symboles accompagnés de leur arité, d'un ensemble fini de règles et d'un réseau.

Les principaux aspects du paradigme de l'interaction sont clairement mis en relief par l'écriture de quelques programmes très simples. On peut citer les suivants :

- Les opérations arithmétiques de base (addition, multiplication, puissance...) montrent que la gestion de la mémoire est explicite ; ce qui rend inutile l'utilisation d'un Glaneur de Cellules.
- La concaténation de listes peut être faite en un nombre constant d'étapes ; l'idée est d'utiliser des listes *différence* pour pouvoir directement connecter la fin de la première liste au début de la seconde.
- L'écriture d'un analyseur syntaxique pour une grammaire *LL(1)* montre comment associer mécaniquement une règle d'interaction à une règle de la grammaire.
- Les opérations de tri mettent en avant l'aspect distribué du calcul.

Un interprète pour les réseaux d'interaction a été réalisé en C avec les outils flex/bison pour l'analyse lexicale et syntaxique et GTK pour l'interface graphique. Dans un premier temps, le but de cet interprète est de se familiariser avec ce nouveau paradigme. Il s'agit aussi de tirer profit de la représentation graphique des programmes (les règles) et de leur exécution (les réseaux) en implantant une interface graphique où l'on peut suivre pas-à-pas l'évolution du réseau et la prochaine règle qui va être appliquée (voir figure 2). A chaque étape du calcul, on peut choisir la coupure qui doit être éliminée. On peut aussi demander d'éliminer toutes les coupures pour obtenir directement le résultat du calcul.

L'interprète n'en est encore qu'à un premier stade de développement et de nombreuses améliorations restent à faire : optimisation de l'affichage graphique, possibilité de faire des zooms, gestion des entrés/sorties, communication entre plusieurs systèmes d'interaction, interfaçage avec d'autres programmes. Il reste aussi à offrir un système de typage qui assure que l'on n'aura pas de coupures pour lesquelles aucune règle n'existe ainsi qu'un critère de correction garantissant l'absence de certains cycles empêchant une portion de réseau de communiquer avec l'extérieur.

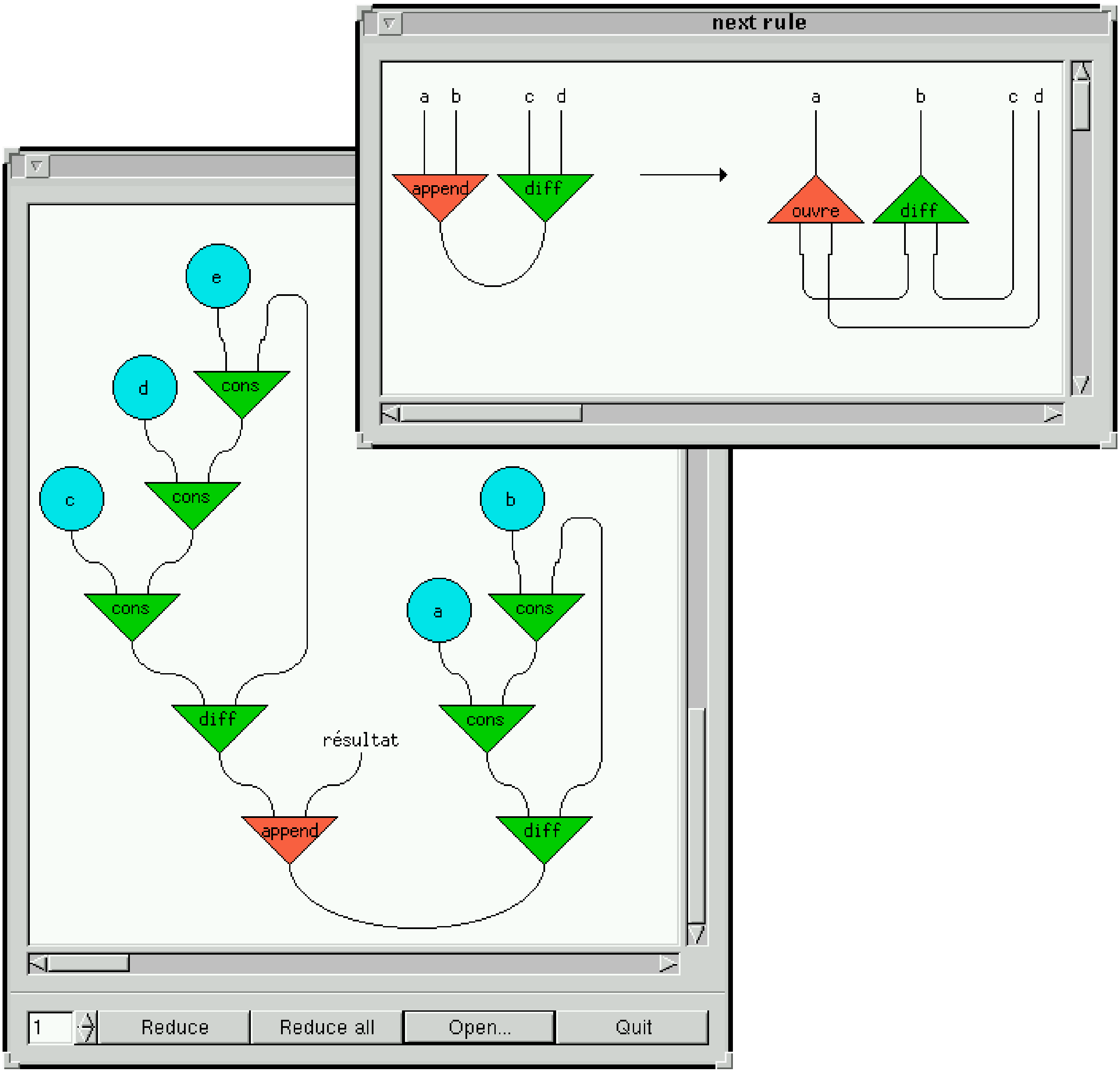

Fig 2 : Affichage graphique de l'exécution d'un programme

Références

– J.-Y. Girard, "Linear logic : its syntax and semantics", in "Advances in Linear Logic", London Mathematical Society Lecture Note Series 222, pp 1–42, Cambridge University Press, 1995 ;

– Y. Lafont, "From Proof-Nets to Interaction Nets", idem, pp 225–247 ;

– Y. Lafont, "Interaction Combinators", Information and Computation 137 (1), pp 69–101, Academic Press, 1997.