Une syntaxe améliorée pour la description de circuits

Aliaume Lopez,
Encadré par le Professeur Dan R. Ghica,
University of Birmingham, UK

19 août 2016

Contexte général

L'étude des circuits imprimés se fait en général via des simulations (comme par exemple dans [4]). La définition de ce que fait le circuit (sa *sémantique*) étant alors dite *dénotationnelle*, ce qui contraste avec la majorité des langages de programmation dont la sémantique est donnée de manière *opérationnelle*, c'est à dire en opérant directement sur la *syntaxe*.

Le domaine étudié est celui des circuits digitaux, c'est à dire avec un sens précis pour les *entrées* et les *sorties* d'un circuit, ce qui n'est pas le cas pour un circuit quelconque.

La méthodologie utilisée a grandement été influencée par les récents développements sur le raisonnement par diagrammes pour une large variété de modèles de calculs tels que le calcul quantique [1], les signaux par flots [2] ou bien les circuits asynchrones [3].

Le problème étudié

Si l'étude syntaxique des circuits digitaux est intéressante, elle n'est pas propice à l'écriture de circuits réels pour plusieurs raisons. Une première est que les propriétés de distributivité des opérateurs syntaxiques permettent à plusieurs expressions très différentes de représenter le même circuit. Une seconde est qu'une très légère modification dans le circuit (ajout d'un cable) peut changer radicalement les expressions qui le représentent.

J'ai dans un premier temps été chargé de construire une syntaxe améliorée, qui permet de conserver la *composabilité* qui est la force de la syntaxe provenant de la théorie des catégories, ainsi que la vision *relationnelle* des composants, qui provient de la vision du circuit comme un graphe.

Cette construction est importante car elle permet l'écriture et la lecture de circuits complexes de manière beaucoup plus simple.

La contribution proposée

Construire une syntaxe est un travail difficile, et mon idée fût de m'appuyer le plus longtemps possible sur la syntaxe algébrique déjà présente. Dans un premier temps, j'ai simplement ajouté des sucres syntaxiques qui se transformaient en expressions classiques.

La première chose qu'il fallait ajouter était la notion de *lien* entre deux circuits. C'est en effet ce qui manque à la description algébrique des circuits pour être aussi agréable que la description en terme de graphe.

La solution retenue fût de ne pas construire explicitement des liens, puisque cela revenait à abandonner la notion de *composabilité*. En effet, je me suis inspiré de la syntaxe du lambda-

calcul, et j'ai introduit la notion de variable-circuit. Si cette construction était élégante et similaire à quelque chose de bien connu, elle ne respectait pas la symétrie inhérente aux dessins de circuits, c'est pourquoi un opérateur symétrique fût ajouté, complexifiant légèrement la syntaxe.

Exprimer un lien entre deux choses revenait alors simplement à connecter deux variables aux bons emplacements et tracer un fil entre les deux. Mais cette dernière opération restait laborieuse, et un dernier opérateur fût donc ajouté : *link*. Il s'est trouvé que ce dernier permettait exprimer les deux autres opérateurs, et donc cette seule construction a été retenue, simplifiant la syntaxe et la compréhension de celle-ci.

Après avoir décrit et implémenté cette nouvelle écriture, j'ai été chargé de l'intégrer au programme qui effectue la réduction en utilisant la sémantique opérationnelle. Plus tard enfin j'ai été amené à de le ré-écrire entièrement pour en garantir la correction et résoudre certains problèmes d'efficacité.

Les arguments en faveur de sa validité

La syntaxe proposée possède beaucoup de propriétés intéressantes. Par exemple, bien qu'ayant commencé comme un ajout à la syntaxe algébrique, elle permet d'exprimer une grande majorité de cette syntaxe simplement à partir de l'opérateur *link*.

On peut donc ignorer les axiomes des éléments de syntaxe exprimables et en déduire des axiomes qui portent uniquement sur ce constructeur. Les propriétés trouvées sont très naturelles (transitivité, symétrie …), et il est presque certain que ces axiomes suffisent pour démontrer que deux expressions représentant un même circuit sont équivalentes (via les axiomes).

La ré-écriture du programme s'est avérée fructueuse, car il est devenu plus simple à lire, vérifier, mais aussi correct et plus rapide sur des circuits de taille élevée.

Le bilan et les perspectives

Avec plus de temps il aurait été intéressant de travailler directement sur la syntaxe et d'établir précisément les axiomes minimaux pour représenter fidèlement des circuits.

Une autre direction de recherche serait de continuer à travailler sur le programme en lui-même, et d'ajouter des méthodes de réduction plus complexes, par exemple sur le traitement des circuits avec délais. Rendre ce programme utilisable plus facilement et ajouter une option de génération de code de description de circuit rendrait son utilisation pratique possible par d'autres personnes.

Références

[1] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. *eprint arXiv :quant-ph/0402130*, February 2004.

[2] Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. Full abstraction for signal flow graphs. *SIGPLAN Not.*, 50(1) :515–526, January 2015.

[3] Dan R. Ghica. *Diagrammatic Reasoning for Delay-Insensitive Asynchronous Circuits*, pages 52–68. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[4] R. P. Kurshan and K. L. McMillan. Analysis of digital circuits through symbolic reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11) :1356–1371, Nov 1991.