Construction d'une sémantique concurrente par instrumentation d'une sémantique opérationelle structurelle

Matthieu Perrin, Claude Jard, Achour Mostefaoui

Construction d'une sémantique concurrente par instrumentation d'une sémantique opérationelle structurelle

Matthieu Perrin [1], **Claude Jard** [2], **Achour Mostefaoui** [3]

1. *LINA, Université de Nantes*
   Faculté des Sciences et Techniques
   2, rue de la Houssinière, 44322 Nantes Cedex 3
   France
   matthieu.perrin@univ-nantes.fr
2. *LINA, Université de Nantes*
   claude.jard@univ-nantes.fr
3. *LINA, Université de Nantes*
   achour.mostefaoui@univ-nantes.fr

RÉSUMÉ. *De par la séquentialité qu'elles induisent, les sémantiques opérationnelles structurelles ne sont pas parfaitement adaptées pour décrire les langages répartis modernes comme Orc, un langage d'orchestration des sites web devenu un véritable modèle de programmation concurrente. Nous proposons deux nouvelles sémantiques pour le langage Orc, un instrumentation légère de la sémantique originale capturant la concurrence au sein d'une exécution et une sémantique concurrente complète décrivant toutes les exécutions possibles pour une expression.*

ABSTRACT. *Because they are intrinsically sequential, structural operational semantics are not completely suitable to describe modern distributed languages such as Orc, a language for web site orchestration that became a true model for concurrent programming. We propose two new semantics for the Orc language, a lightweight instrumentation of the original semantics capturing concurrency in one execution and a whole concurrent semantics able to describe all the possible executions for an expression.*

MOTS-CLÉS : *Orc, sémantique, causalité, préemption, conflit, concurrence, structure d'événements asymétrique*

KEYWORDS: *Orc, semantics, causality, preemption, conflict, concurrence, asymmetric event structure*

1. Introduction

Orc (Kitchin *et al.*, 2009) est un langage de programmation initialement conçu pour l'orchestration des sites web. Sa philosophie sous-jacente est que la modularité est essentielle à la conception de grands programmes modernes, assemblages de composants qui doivent être coordonnés par un chef d'orchestre pour travailler ensemble. En Orc, ces composants, appelés *sites*, peuvent être des services définis de manière externe dans n'importe quel langage, ainsi que des encapsulations d'expressions Orc. À la différence des fonctions dans un langage fonctionnel, les sites de Orc ne retournent pas de résultat, mais publient une ou plusieurs valeurs, qui peuvent alors être combinées via les opérateurs du langage, permettant d'exprimer les différentes facettes de la programmation répartie : le parallélisme ($|$), la séquentialité ($> x >$ et ;) et la préemption ($<x<$).

De par sa richesse d'expression, le langage Orc est un candidat de choix pour étudier les systèmes répartis. Cependant, sa sémantique est formellement définie par une sémantique opérationnelle structurée (Plotkin, 1981) qui décrit un système de transitions. Aussi la sémantique d'une expression est-elle un ensemble de séquences d'exécution possibles. Le problème de cette représentation est qu'elle confond non-déterminisme et concurrence. Par exemple, le sens de l'expression $1|2$ est la publication des deux valeurs 1 et 2, sans favoriser l'une par rapport à l'autre temporellement. Pourtant, sa sémantique ordonne les publications ce qui définit artificiellement deux comportements disjoints et incompatibles pour ce programme.

2. Contributions

2.1. La sémantique instrumentée

Pour pallier ce problème, nous proposons une instrumentation de la sémantique qui calcule la concurrence au fur et à mesure de l'exécution. À chaque pas de calcul, en plus de l'étiquette de la sémantique standard, on produit deux nouvelles informations permettant de définir une structure d'événements asymétrique, c'est à dire un triplet $(E, \leq, \nearrow)$ dans lequel E est l'ensemble des événements observés pendant l'exécution et $\leq$ et $\nearrow$ sont respectivement les relations de causalité et de causalité faible entre les événements. Intuitivement, un événement e_1 est une *cause* d'un événement e_2 s'il se produit *toujours* avant ce dernier, et $e_1 \nearrow e_2$ si e_1 ne se produit *jamais* après e_2. En plus de la causalité, la causalité faible permet de capturer la *préemption* selon laquelle un événement peut empêcher un autre de se produire. Dans Orc, l'opérateur $<x<$, qui termine une exécution lors de la publication d'une valeur, permet d'exprimer simplement la préemption, ce qui rend l'utilisation de structure d'événements *asymétriques* nécessaire. De plus, la *concurrence* est définie comme l'absence de lien de causalité faible entre deux événements.

Une linéarisation d'une structure d'événements asymétrique est un sous-ensemble d'événements totalement ordonnés clos à gauche pour la relation de causalité et dont l'ordre ne rentre pas en conflit avec la causalité faible. On prouve que notre instru-

mentation est correcte, dans le sens où toute linéarisation correspond à une exécution séquentielle réellement présente dans la sémantique originale. On a donc une inclusion de l'ensemble des comportements capturés dans une exécution de la sémantique instrumentée, dans l'ensemble des comportements autorisés pour un programme.

2.2. La sémantique concurrente

On souhaite aller plus loin et obtenir l'égalité entre ces deux ensembles, c'est à dire qu'une unique exécution sous une sémantique concurrente capture à la fois toutes les exécutions de la sémantique officielle. Pour cela, il est nécessaire de nous pencher sur une nouvelle relation entre les événements, le *conflit*. Informellement, plusieurs événements sont en conflit s'ils ne peuvent pas être tous présents dans une même exécution. Plus formellement, il est défini comme un cycle dans la relation de préemption, et c'est donc un cas particulier de la causalité faible. Autrement dit, les structures d'événements asymétriques restent l'objet mathématique adapté pour étudier le conflit.

La définition d'une telle sémantique concurrente est notre deuxième contribution. Pour pouvoir capturer le conflit, il est nécessaire de générer plus d'événements que ce que l'on peut observer dans une seule exécution séquentielle. La sémantique concurrente génère donc tous les événements possibles lors de l'évaluation d'une expression Orc, et instrumente les transitions pour calculer une structure d'événements asymétrique. Notre sémantique concurrente étant une extension des sémantiques précédentes, il existe toujours plusieurs exécutions possibles. Cependant, toutes définissent la même structure d'événements qui contient l'information nécessaire pour reconstruire toutes les exécutions séquentielles d'une expression Orc. C'est la raison pour laquelle on peut bien parler de *la* sémantique concurrente de Orc.

3. Conclusion

Dans nos travaux, nous définissons deux nouvelles sémantiques de Orc : la sémantique instrumentée, qui extrait autant d'information que possible d'une unique exécution et une sémantique concurrente qui décrit tous les comportements possibles du programme. La prochaine étape de notre travail sera l'utilisation de ces sémantiques dans la conception d'un débogueur pour le langage Orc qui permette de rejouer des scénarios d'exécution en fournissant des outils d'analyse des causes premières et des situations de compétition.

Bibliographie

Kitchin D., Quark A., Cook W., Misra J. (2009). The orc programming language In *Formal techniques for distributed systems; proceedings of fmoods/forte*, vol. 5522, p. 1-25. Lisbon, Portugal, Springer.

Plotkin G. D. (1981). *A structural approach to operational semantics* Rapport technique n° DAIMI FN-19. Aarhus, Denmark, Aarhus University. (réimprimé avec corrections dans J. Log. Algebr. Program. 60-61: 17-139 (2004))