Analyse d'Article :
The Generation of Optimal Code for Arithmetic Expressions
UE Introduction à la Recherche Scientifique

François Flandin

Decembre 2024

1 Contexte

L'article, *The Generation of Optimal Code for Arithmetic Expressions RAVI SETHI AND J. D. ULLMAN*, a été publié en 1970, et se situe dans le domaine de l'informatique, plus précisément dans la compilation d'expressions arithmétiques.
A cette époque, les techniques de minimisation de code d'expression arithmétiques ne sont pas très poussées et n'exploitent pas les différentes propriétés des opérations.
L'article prend beaucoup des travaux de Nakata et de Meyers, qui ont réalisé des algorithmes qui minimisent le nombre de registres requis pour évaluer une expression arithmétique, et prend aussi des travaux d'autres articles qui commencent à exploiter la commutativité des opérations.
Les auteurs sont, quand à eux, des personnes très importantes dans le monde de la compilation et de l'informatique théorique avec notamment le *Dragon Book*, référence dans le domaine de la compilation, ainsi que les algorithmes de parsing d'expressions *LL(k)/LR(k)*.

2 Résumé

Au cours de cet article, 3 algorithmes sont présentés, le but final est de minimiser le nombre d'instructions du code généré, ainsi que minimiser le nombre de références mémoires, car la mémoire est bien plus lente que les registres du processeur.

Les 3 algorithmes fonctionnent de la même manière, ils se servent d'un arbre binaire qui représente l'expression arithmétique que l'on souhaite compiler, les nœuds représentent les opérations et les feuilles les variables définies avant l'expression, de plus, les nœuds et les feuilles de l'arbre sont étiquetés en fonction du nombre de registres nécessaires pour les enregistrer temporairement. Ce qui différencie ces algorithmes, c'est que cet arbre pourra être modifié en fonction des propriétés que l'on veut exploiter.

Le 1er algorithme reprend l'idée des algorithmes de Nakata et Meyers, mais rajoute le problème de minimiser le nombre de références mémoires en plus de celui de minimiser le nombre d'instructions, il ne touche pas a l'arbre, et reste plutôt basique même si il reste très efficace.

Le deuxième algorithme est légèrement plus complexe et pousse plus loin la réduction du nombre

d'étapes du code généré, pour cela, il exploite les expressions commutatives en modifiant l'arbre de manière a échanger les fils des opérations commutatives si le fils gauche est une feuille. C'est une particularité qui change de ce qui était fait auparavant.

Le troisième algorithme est encore un peu plus complexe et exploite cette fois-ci l'associativité des opérations, en générant un "arbre associatif", c'est a dire que l'on regroupe les fils des nœuds d'opérateurs associatifs ensemble, par exemple a + b + c donnerait (+) en racine et [a,b,c] en fils.

3 Analyse

La structure de l'article respecte bien son objectif, qui est de minimiser le nombre d'instructions du code généré après compilation d'une expression arithmétique. En effet, plus on avance dans l'article, plus on progresse dans la "complexité" de l'algorithme, on commence par nous présenter un algorithme simple qui a déjà été présenté auparavant, puis on nous introduit de plus en plus de concepts qui permettent d'améliorer la réduction du code généré. On nous présente même plusieurs exemples et définit correctement ses termes, ce qui permet une bonne compréhension de l'article.

De plus, l'article présente plusieurs lemmes et théorèmes pour prouver que l'algorithme rempli bien sa tâche, et sont démontrés à l'aide de preuves par induction ou par d'autres méthodes mathématiques, ce qui est adéquat dans le domaine algorithmique.

L'article n'extrapole pasFeuille de calcul sans titre ses résultats et ces résultats sont même plutôt importants car ils restent un des fondamentaux de la compilation moderne.

Le seul point "négatif" serait que l'article affirme que la complexité temporelle de l'algorithme est de O(n) sans fournir de preuve. Cependant, on peut facilement le vérifier, en effet, l'algorithme s'applique une fois a chaque nœud de l'arbre.