ÉLÉMENT DE PORTFOLIO 06

Publication

1 DÉFINITION DE CET ÉLÉMENT

Titre de l'élément : Causal Computational Complexity of Distributed Processes

URL de l'élément : `https://hal.science/hal-02074534v1`

2 MOTIVATIONS DU CHOIX DE CET ÉLÉMENT

Cet élément a été choisi parce qu'il représente :

1. un lien entre deux mondes des méthodes formelles (la théorie de la concurrence et la complexité implicite) ;
2. une collaboration entre l'équipe APR et *Imperial College* (puis *Oxford*).

L'équipe APR s'efforce de bâtir des ponts entre des domaines distincts de l'algorithmique et de la programmation. À ce titre, cet article représente un exemple d'un rapprochement entre deux thématiques non-connexes :

- Le *monde de la complexité implicite* d'une part : il s'agit d'un domaine, actif depuis une vingtaine d'années, qui vise à produire des résultats de complexité (en temps, en espace) *a priori* pour des algorithmes à partir de contraintes syntaxiques (restriction du langage) ou d'analyses statiques (systèmes de types). L'idée principale est de construire un cadre formel dans lequel tous les programmes *définissables* (soit syntaxiquement, soit après une étape de vérification de types) satisfont une borne de complexité.
- Le monde des *algèbre de processus* d'autre part : un domaine dans lequel des abstractions mathématiques de programmes (ou protocoles) concurrents et distribués sont définies et étudiées.

Cet élément a été publié dans une conférence internationale majeur du domaine (LICS).

Le caractère pertinent de cet élément s'exprime dans :

- la définition d'une notion de complexité nouvelle (complexité causale des messages générés dans un réseau de services) ;
- l'adaptation d'une technique classique d'un domaine (la complexité implicite) à un autre domaine (les algèbres de processus) ;
- l'introduction d'analyses spécifiques supplémentaires qui prennent en compte le caractère particulier du cadre concurrent.

3 PRÉSENTATION DE CET ÉLÉMENT

Cet article décrit une méthode d'analyse pour les algèbres de processsus (des modèles de programmes et protocoles distribués) qui décrit que le nombre global de messages générés par un appel au service est borné par un polynôme sur la taille de cet appel.

Le formalisme utilisé dans l'article est celui du pi-calcul, un langage mathématique (une algèbre de processus) décrivant les actions observables des différentes composantes d'un système. Ce formalisme permet de décrire facilement des services, acceptant des requêtes, générant des messages (par exemple, des appels à d'autres services), et renvoyant finalement une réponse à la requête initiale. Des services arithmétiques (calculant des fonctions simples des entiers dans les entiers) sont utilisés comme exemples, bien que l'analyse s'applique de la même façon à des services manipulant des données complexes. Les services peuvent être construits de manière récursive (un service peut envoyer une requête à lui-même pour obtenir une réponse intermédiaire). Certaines définitions de services récursifs génèrent un nombre exponentiel de messages (par exemple, un service récursif pour le calcul de la factorielle, qui appelle un service récursif pour le calcul de la multiplication, qui lui même appelle un service récursif pour le calcul de l'addition). L'objectif de l'analyse présentée dans le portfolio est de détecter et de rejeter de tels services.

3.1 Complexité Causale

L'article s'attelle d'abord à définir une notion de complexité *en messages* pour les réseaux de services. L'idée est, pour la première fois dans ce domaine, de lier la complexité d'un processus au nombre de communications générées par un appel externe. Cela nécessite l'introduction d'une notion de complexité causale, qui permet de lier à chaque communication, la requête initiale qui l'a causée, et ainsi d'obtenir une définition dynamique de la complexité : plusieurs requêtes différentes peuvent être traitées simultanément. Ce que garantit l'analyse, c'est que les messages causalement produits par une requête sont bornés par une fonction en la taille de la requête.

3.2 Adaptation de [1]

L'article [1] est une article important du domaine de la complexité implicite, qui caractérise les fonctions polynomiales à l'aide de règles syntaxiques simples : toutes les fonctions séparent leurs paramètres en deux ensembles : ceux qui sont $\ll$ sûrs $\gg$ et les autres. L'analyse empêche les produits de récursion (les résultats obtenus par un appel récursif) à être utilisés comme argument non-sûr d'une fonction. Les auteurs montrent que cette discipline de séparation suffit à caractériser les fonctions récursives ayant une complexité en temps.

Dans cet article, on adapte dans un premier temps l'analyse de [1] à un cadre concurrent, à l'aide d'un système de types. Les règles de typage garantissent une division sûr/non-sûr des contenus des messages, et donc des requêtes aux différents services.

3.3 Perturbations concurrentes et solution

Pourtant, cette analyse n'est pas correcte. En effet, le caractère concurrent du cadre peut tromper l'analyse, en perturbant le flot de calcul d'un service à l'aide de messages externes portant des contenus non-bornés et susceptibles de produire des calculs arbitrairement grand (et donc des messages arbitrairement nombreux). Pour que les contraintes du système de types soient respectées, il est nécessaire d'ajouter une étude du flot de calul (de la provenance et du devenir des contenus des messages échangés au sein du système).

La combinaison des deux analyses (systèmes de types et analyse d'origines) permet d'obtenir la borne polynomiale souhaitée.

4 RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Stephen J. Bellantoni and Stephen A. Cook. A new recursion-theoretic characterization of the polytime functions. *Comput. Complex.*, 2 :97–110, 1992.