Rapport TAS

Par Mohamed Nizamuddin

&

Skander SERSI

<https://www.youtube.com/watch?v=LGqrXowiSC8>

<https://www.youtube.com/watch?v=QFE7kiFXw54>

Introduction

qu'est ce qu'ils ont écrit d'autres les auteurs ?

le contexte .

l'état initial de l'artcle

quel est le problème

qu'est ce qu'il propose comme solution

comprendre aumaximum ce qu'ils proposent

travail personnel de compréhension d'un article

pas de traduction d'article

ajouter quelque chose de personnel pour faire voir qu'on a compris (donner

dex exmeples qui ne sont pas dans l'article)

conclure:

en disant exactement ce qu'ona compris et ce qu'on a compris.

est ce qu'on pense qu'il est applicable en entreprise

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

l’apport des auteurs est la création d’un lambda calcul avec une syntaxe dirigé par les types et un operateur merge. corrige les problèmes du lambda calcul de Dunfield dans lequel il y avait de l'indéterminisme et le faite que les règles de reductions de λi ne preserve pas le type. L’idée que ont eu les auteurs et de faire une semantique dirigé par les tye en se servant d’annotations pour guider les reductions de façon explicite pour pas arriver a un terme mal typé. De la decoule des regles de reduction utilisant les informations d’annoations. Nous avons 2 regles step-annov et step-beta.

2.4)

Trouver les regles pour les fonctions eté dur pour les auteurs. Ils sont neanmoins arrive a le faire. D’abord en annotant les fonctions avec un type fonctionnel et la possibilité de rafiner les types des fonctions.

2.5) Pour eleminer le non determinisme utilisation de type disjoints lors de la fusion tel que les 2 termes de la fusion n’ont pas de type parents en commun sauf l’ancetre. De ceci nous otbenons la regle ETYPE-MERGE. Sauf si les deux termes de la fusion sont consistent c’est a dire syntaxiquement egale on a le droit de passer outre le fait que les types ne sont pas disjoints d’apres la regle ETYPE-MERGEV. Et 2 valeurs sont consistent si v1 ≈ v2

3.3 Typage

pour le typage

pour le generalisation du type ils se sont inspirés du calcul NeColus.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Transcript video 5 minutes a nettoyer pour garder ce qui est intéressant**

Notre travail porte sur la conception de système de type et les fonctionnalités du langage de modélisation dans un calcul de base.

La construction clé dont nous nous occupons ici est l'opérateur de fusion … qui fournit des mécanismes généraux qui englobent de nombreuses fonctionnalités POO.

L'opérateur de fusion combine deux termes. Ensemble, ils ont un type d'intersection.

Par exemple, 1,, True a le type Int & Bool. Et il peut être utilisé comme un entier ou un booléen,... sans éliminer explicitement comme projection pour les types de produits.

Comme vous pouvez le voir, les fusions sont très flexibles. Pour cette raison, la conception d'une sémantique opérationnelle directe pour un calcul avec l'opérateur de fusion est difficile.

Dans certains travaux existants, l'opérateur de fusion est biaisé pour éviter toute ambiguïté.

Jana Dunfield donne un ensemble de règles de réduction pour l'opérateur de fusion non biaisé. En voici deux.

Ces deux règles expriment qu'une fusion peut avancer vers sa partie gauche ou droite sans aucune contrainte. Donc, selon cette sémantique opérationnelle,

lors de l'évaluation d'une fusion bien typée, le résultat n'est pas unique ...

et peut être mal typé.

Notre sémantique opérationnelle est déterministe et de type sonore,

à l'aide d'annotations de type pour contrôler les fusions. Notre sémantique opérationnelle est déterministe et de type sonore,

à l'aide d'annotations de type pour contrôler les fusions.

Nous avons une relation de réduction typée,

qui «filtre» la valeur d'entrée par le type souhaité.

Lorsque les annotations de type déclenchent une réduction de type, le processus préserve les types.

En attendant, dans notre système de typage, nous nous assurons que ...

les parties des fusions peuvent être distinguées par leurs types,

la réduction typée est donc déterministe.

Premièrement, si deux termes ne se chevauchent pas dans les types,

les fusionner est autorisé.

Deuxièmement, si deux valeurs,

... ont toujours la même valeur pour leurs types qui se chevauchent, s'ils existent,

ils peuvent être fusionnés en toute sécurité.

Le concept de disjonction de type vient du calcul lambda-i.

Il n'autorise pas l'intersection de types non disjoints comme Int & Int.

Donc, il rejette les termes ambigus comme 1,, 2

et 1,, 1 également.

Nous soutenons les intersections illimitées

et généraliser la restriction de disjonction de type pour les valeurs fusionnées,

qui s'appelle la cohérence.

Prenons un avant-goût de notre sémantique opérationnelle dirigée par type

Ici j'utilise le langage Zord

qui est un projet en cours de notre groupe

Il utilise notre sémantique opérationnelle dirigée par type

a l'opérateur de fusion et les types d'intersection

Il prend également en charge certaines fonctionnalités POO plus sophistiquées

Vous pouvez essayer ces exemples ici

Et, à noter

la virgule unique désigne ici l'opérateur de fusion

Prenons l'exemple

que nous avons vu sur l'affiche

En supposant que x est une fusion d'un entier et d'un booléen

Et nous avons une fonction f

qui attend un entier

Nous avons également une fonction g

qui attend un booléen

Ici, nous allons utiliser x

comme entrée pour les deux fonctions

et les fusionner ensemble

Donc, enfin, nous avons un résultat qui est une fusion de 3 et de faux

Cliquez sur le 'StepTrace', vous pouvez voir les détails

comment la réduction se produit

Plus important encore, nous pouvons voir que

cette fusion avec l'annotation Int

lorsqu'il est ajouté à 1, seul 2 est sélectionné

De même, seul True est sélectionné à partir de la fusion

quand on l'utilise dans une fonction qui attend un booléen

Merci pour votre écoute

Si vous avez d'autres questions

vous êtes invités à nous envoyer un e-mail ou à venir à notre conférence

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

% λ-calcul avec un type intersection et un operateur fusion

% Mohamed Nizamuddin, Skander Sersi

% 17 Novembre 2020

\hfill

\begin{center}

\includegraphics{sorbonne-universite-logo.png}

\end{center}

\newpage

Cette article est un résumé de l'article A Type-Directed Operational Semantics For a Calculus with a Merge Operator[1] écrit par Xuejing Huang, Bruno C. d. S. Oliveira. Il propose une semantique operationnelle dirigé par les types pour un calcul avec le type intersection et l'operateur fusion.

# Introduction

L'operateur merge (,,) permet d'obtenir des valeurs ayant un type intersection. Voici un exemple de programme ocaml-like qui serait valide avec un tel operateur:

let v : String & Int = "Hello",, 2 in (x ^ " world !", x \* 3)

Dans notre exemple, la variable v peut etre soit un String avec "Hello", soit un Int avec 2. En evaluant cette expression nous obtenons ("Hello world!", 6).

Cette expression n'est pas valide dans certains languages avec le type intersection car les types de notre intersection sont disjoints.

Néanmoins, on voit bien que l'operateur merge donne de l'expressivité et pourrait meme remplacer d'autre features d'un langage orienté objet.

Des calculs avec l'operateur merge ont émergés mais ils sont complexes contrairement au lambda calcul simplement typé de base. Et aucun n'a reussi a prouvé un programme avec de la récursion. Le travail de Xuejing Huang fait suite à celui de Dunfield, qui présente une semantique operationnelle pour un calcul li avec l'operateur merge. Dunfield a utilisé une méthode complexe pour donné la sémantique. Premierement, son calcul a de l'indeterminisme, et deuxiemement les types ne sont pas preservés lors de l'evaluation des expressions. Les auteurs veulents corrigés les problèmes du calcul de Dunfield avec TDOS en proposant lianno

#dunfields semantics

# TDOS

# lambdiianno

# Applications

Le type intersection correspond a l'intersection de deux types. La variable x ayant un type intersection A & B peur etre soit de type A, soit de type B. Et l'operateur fusion permet de construire des valeurs ayant un type intersection qui donne beaucoup d'epressivité. Typescript est un language qui utilisant le type intersection. L'operateur fusion quant a lui, on le trouve dans des langages moins connus. Le duo forme donne un avantage non negligeable permettant d'obtenir de l'expressivité, de remplacer des d'autres features par exemple dans le language Sedel l'heritage est representé par l'operateur fusion avec l'intersection de type disjoints. Jusqu'a present il n'y a pas eu de language avec l'operateur fusion et une semantique simple. Des calculs ont émergés mais avait tous des inconvenients. Le travail de Xuejing Huang fait suite à celui de Dunfield, qui présente une semantique operationnelle pour un calcul li avec l'operteur fusion dans lequel il y a de l'indeterminisme, et les types ne sont pas preservé lors de reductions. Les ajouts

Etat de l'art :

probleme que les auteurs veulent traiter

et comment ils traite

a la fin de l'introduction : ce que les auteurs ont apporté de nouveaux

montrer que j'ai compris l'article

utilisez des exemples personnels pour montrez que l'on a compris

outils ???

synthetiser la'article:

- identifier les points importants de l'article

- quel a ete le probleme

- comment a ete resolu

- quels ont ete les difficultés et les challenges de la méthode de resolution

rapport : donner le contexte

qui l'a ecrit, qu'est ce qui ont ecrit d'autres

introduire l'article, de quoi il parle, qu'est qu'on va faire

section technique

conclusion : en disant exactement ce qu'on a compris, ce que l'on pense, le lien avec TAS et le monde du typage, et ce que cela sera utile.

#

---------------------------------------------------------------------------------------------------------------------------

Skander ( je suis entrain de retranscrire ce que je comprends de l’article ) Je ferai le tri + rédaction au propre quand je serai plus avancé

Qu’est ce qu’un type construit à partir d’une intersection ?

Est un type qui peut être alloué aux valeurs de type t1 et de type t2. Ce type est noté t1 inter t2. Généralement si le domaine des types t1 et t2, se chevauchent, Une valeur appartenant à l’intersection peut être donné à t1 inter t2. Cette valeur de type t1 inter t2 peut être passée aux fonctions qui attendent t1 et t2.

Qu’est ce qu’une opération de merge ?

Je sais pas encore

Page 1 :

Le calcul avec les types ayant une intersection disjointe et l’operateur merge fournissent généralement des mécanismes qui peuvent nous faire penser à d’autres fonctionnalités.

Ce genre de calcul peut aussi encoder des formes supérieurs de composition d’objets, qui capture des modèles de programmation communs dans des langages typés dynamiquement (comme javascript) dans une manière pleinement statique. Malheuresement comme d’autre fondement de calcul (comme System F, FeatherWeight Java), les calculs recents avec l’operateur merge n’ont pas d’operations sémantiques directes avec des propriétés standard et attendu comme le determinisme et la reduction de sujet (subject-reduction).

En outre, la théorie générale de ce genre de calculs comptent seulement dans des programmes qui terminent.

Ce papier propose des sémantiques opérationnelles dirigés vers les types pour λ:

Un calcul avec des intersections de type et un opérateur de merge, inspiré par d’autres calculs

comme celui de DunField(2014) et Oliveira(2016).

Le problème avec dunfield c’est que sa sémantique manque de derminisme et de reduction de sujet, mais propose une petit étape directe pour son calcul.

En utilisant le TDOS proposé par l’article on obtient une sémantique directe pour lambda, qui les deux propriétés (determinisme et subject-reduction).

Pour obtenir le determinisme, le lamba calcul proposé emploi une restruction disjointe proposée par Oliverira et Al. Le nouveau calcul lamba rajoute la résolution des recursions d’une manière directe, pas comme l’ancien calcul lambda où la recursion était problématique.

Pour relier la sémantique du nouveau calcul lambda au calcul de DUnfield et Oliveira, on montre deux resultats. Le premier, la sémantique du lambda respecte les sémantiques donnés par Dunfield. On montre aussi que le system de type du nouveau lambda est complet en respectant le systeme de type de l’ancien lambda.

Tous les résultats ont étés formalisés dans prouver de theoreme Coq.