Rapport

Evaluateur-Typeur de Lambda-Calcul

Romain Demangeon

TAS - Typage et Analyse Statique

Réalisée par :

Dilyara Babanazarova, 28709428

Année universitaire : 2024/2025

Résumé des Parties Traitées

Ce projet a pour objectif de développer un évaluateur et un typeur pour un λ-calcul étendu, incluant des fonctionnalités comme les entiers de Church, les combinateurs, les opérateurs arithmétiques (`Add`, `Sub`), les listes, ainsi que des constructions comme `Let`, `IfZero`, `IfEmpty`, `Fix`, et les opérateurs de référence (`Ref`, `Deref`, `Assign`). Les types définis dans l'AST incluent `Var`, `Arr`, `Nat`, `N`, `List`, `Forall`, `Ref`, `Unit`, et `Weak`, couvrant une large gamme de constructions et de types nécessaires pour représenter des expressions complexes. Les principales parties du projet incluent la création d'un interpréteur capable de réduire des termes lambda en forme normale avec la règle "Call By Value" (CBV), ainsi qu'un système de typage pour vérifier et inférer les types des expressions.
L'évaluateur implémente la définition des termes (abstractions, applications, combinateurs), les règles de réduction, et le support des évaluations conditionnelles (`IfZero` et `IfEmpty`). Il prend également en charge des constructions comme `Let`, `Fix`, les opérateurs sur les listes (`Cons`, `Tete`, `Queue`), et la manipulation des références (`Ref`, `Deref`, `Assign`). Le typeur génère et résout des équations de types en utilisant des techniques comme la généralisation et l'unification, et gère également les types avancés tels que les quantifications universelles (`Forall`) et les types faibles (`Weak`).
Les tests mis en place couvrent des cas simples (comme l'identité et les combinateurs de base) ainsi que des cas complexes (combinaison de extttLet, quantifications universelles, applications imbriquées) pour vérifier la justesse des réductions et des inférences de types. Ces tests ont permis d'assurer la robustesse et la cohérence du système face à des scénarios variés.

Points Forts du Projet

- **Précision des Réductions** : L'évaluateur gère efficacement la réduction des termes en CBV et parvient à réduire des expressions complexes.
- **Typage Avancé** : Le typeur offre des capacités avancées pour généraliser les types et gérer les quantifications universelles, permettant une inférence correcte des types, même pour des expressions complexes. Il prend également en charge la propagation des types faibles (`Weak`) et les références.
- **Gestion des Équations de Types** : La fonction `genere_equa` permet de générer des équations de types pour des expressions complexes, en prenant en compte les différentes constructions de l'AST, ce qui permet de vérifier la cohérence des types, y compris pour des constructions imbriquées.

Points Faibles du Projet

- **Améliorations Possibles** : Le projet aurait pu être amélioré en implémentant d'autres stratégies d'évaluation en ajoutant un lexer/parser pour le λ-calcul, ou en incluant des types produits et des types sommes.
- **Tests Limités** : Bien que de nombreux tests aient été effectués, il aurait été bénéfique d'inclure des tests encore plus détaillés, notamment sur des scénarios complexes impliquant des combinaisons de plusieurs constructions (par exemple, des termes imbriqués avec `Let`, `IfZero`, et des références) pour vérifier la gestion de tous les cas limites possibles.

Difficultés Rencontrées

- **Gestion des Types Faibles** : La manipulation des types faibles (via `Weak`) et la mise à jour des environnements se sont révélées complexes, entraînant des difficultés dans la propagation des substitutions et l'éviction correcte des variables liées. Cela a nécessité une révision approfondie des algorithmes de substitution et de généralisation pour garantir leur bonne gestion.
- **Unification et Substitution** : La partie du typeur concernant l'unification et la substitution a posé des problèmes, notamment lorsqu'il s'agissait de gérer des substitutions multiples et des cas où

plusieurs types faibles devaient être résolus. La gestion correcte de la propagation des substitutions, surtout dans des contextes imbriqués, a représenté une difficulté de comprehension.

Conclusion

En conclusion, malgré ces défis ce projet a abouti à un système fonctionnel, offrant une bonne précision des réductions et un typage avancé pour des expressions lambda variées. Bien que certaines améliorations soient encore possibles, le travail réalisé a permis de poser des bases solides pour la gestion et l'évaluation de langages basés sur le λ-calcul, offrant ainsi une plateforme d'expérimentation pour des extensions futures, telles que la gestion des types produits, des types sommes, ou encore l'implémentation de nouvelles stratégies d'évaluation