Rapport Devoir 3 - IFT 2035
Groupe 68

On est commencé par trouver la similarité entre ce merveilleux langage "Girald" et la système F.

Tout d'abord, nous nous sommes familirisé avec la lambda calcul simplement typé pour comprendre le type de Var et Arrow, puis on continue de passer à polymorphisme du Système F pour le type forall (pour tout).

Pour type_freevars, env_type, type_subst et type_equiv:

- Le Type de chaque predicat peut prendre l'un de ces 3 types: soit Var(X), Arrow (A, B), ou forall(A, Body). Chaque type nécessite un mode de résolution différent.
- La difficulté de l'implémentation de forall est dans la gestion de la scope entre la bounded variable et le corps.

Pout type_verifications:

- On traduit chaque règles d'inférence à une clause Prolog en nous assurant de gérer correctement l'évolution des deux environnements : celui des types (Δ) et celui des termes (Γ).

Pour l'évaluation (la partie plus dificile pour nous - la LLM nous avons aidé à saisir le concept et implémenter lambda et poly):

- expr_freevars: on collecte des variables recoursivement, puis le retire de la liste des variables libres uniquement lors on voit un lambda (qui lie des termes), en ignorant les polymorphes (qui ne lient que des types).
- expr_subst: implémente la substitution évitant la capture (capture-avoiding substitution), spécifiquement pour les nœuds lambda (on renomme des variables conflictuelles si nécessaire)

Nous avons testé le code de manière approfondie avec les cas de test fournis dans le devoir.

Les LLMs sont utilisés pour aider à comprendre la portée des variables liées, le concept d'évaluation (la partie la plus ardue) ainsi que pour le débogage.