Rapport: Implementing a proof assistant
Charles Meyer

1. Introduction

Ce rapport documente la mise en œuvre d'un assistant de preuve pour la logique propositionnelle et les types dépendants. Ce projet vise à créer un outil capable de vérifier et d'élaborer des preuves de manière interactive, facilitant ainsi l'apprentissage et l'application des concepts liés au lambda-calcul typé.

2. Ce qui a été implémenté

Simple Types

Pour la partie concernant les nombres naturels, j'ai travaillé sur différentes étapes sans parvenir à faire fonctionner cette section ni à prouver les fonctions *predecessor* et *addition*.

1 Type inference for a simply typed calculus
- ☑ ~~1.1 Simple types~~
- ☑ ~~1.2 λ-terms~~
- ☑ ~~1.3 String representation~~
- ☑ ~~1.4 Type inference~~
- ☑ ~~1.5 Type checking~~
- ☑ ~~1.6 Type inference and checking together~~
- ☑ ~~1.7 Other connectives~~
- ☑ ~~1.8 Conjunction~~
- ☑ ~~1.9 Truth~~
- ☑ ~~1.10 Disjunction~~
- ☑ ~~1.11 Falsity~~
- ☑ ~~1.12 Parsing strings~~

2 Interactive prover
- ☑ ~~2.1 String representation of contexts~~
- ☑ ~~2.2 Sequents~~
- ☑ ~~2.3 An interactive prover~~
- ☑ ~~2.4 Elimination of arrows~~
- ☑ ~~2.5 Proofs in files~~
- ☑ ~~2.6 Full arrow elimination~~
- ☑ ~~2.7 Conjunctions~~
- ☑ ~~2.8 Truth~~
- ☑ ~~2.9 Disjunction~~
- ☑ ~~2.10 Falsity~~

3 Natural numbers
- ☐ 3.1 A type for natural numbers
- ☐ 3.2 Introduction rules

- [ ] 3.3 Elimination rule

4 Optional: small extensions

- [ ] 4.1 Reduction
- [ ] 4.2 Declarations
- [ ] 4.3 Other inductive types

Dependent Types

Je n'ai pas implémenté l'étape 5.12 et les suivantes par manque de temps.

5 Dependent types

- [x] ~~5.1 Expressions~~
- [x] ~~5.2 String representation~~
- [x] ~~5.3 Fresh variable names~~
- [x] ~~5.4 Substitution~~
- [x] ~~5.5 Contexts~~
- [x] ~~5.6 Normalization~~
- [x] ~~5.7 α-conversion~~
- [x] ~~5.8 Conversion~~
- [x] ~~5.9 Type inference~~
- [x] ~~5.10 Type checking~~
- [x] ~~5.11 Interaction loop~~
- [ ] 5.12 Natural numbers
- [ ] 5.13 Equality
- [ ] 5.14 Using the prover
- [ ] 5.15 Optional: inductive types
- [ ] 5.16 Optional: interactive prover
- [ ] 5.17 Optional: universes
- [ ] 5.18 Optional: better handling of indices

3. Difficultés rencontrées

Je vais détailler dans les sections suivantes les étapes de l'implémentation de l'assistant de preuve qui m'ont coûté le plus de temps et expliquer pourquoi.

1. Disjunction (1.10)

La première grande difficulté rencontrée a été l'ajout des constructeurs du type et du terme pour la disjonction. Ce qui a rendu cette partie complexe est une confusion de ma part entre l'analyse de cas (case) et la disjonction. Cette difficulté n'était pas liée à l'implémentation, mais plutôt à la compréhension des règles d'introduction et d'élimination associées à *case*.

2. Disjunction (2.9)

Le deuxième point qui m'a posé problème est la preuve de la distributivité de la conjonction :
dist.proof: (A /\ (B \/ C)) => (A /\ B) \/ (A /\ C)

3. Falsity (2.10)

Une autre preuve m'a pris beaucoup de temps :
impdm.proof: ((not A) \/ B) => A => B

4. Natural numbers (3)

L'avant dernière difficulté que j'ai rencontrée est liée à l'implémentation des entiers naturels, notamment l'ajout des types et des termes dans le lexer et le parser. J'ai réalisé une première version, mais lorsque j'essayais d'exécuter l'assistant de preuve et de déclarer un entier naturel, je tombais sur une erreur. Je n'ai pas réussi à résoudre cette erreur, qui provient probablement de ma compréhension approximative du système de générateurs de *lexer* et de *parser*.
Une seconde difficulté liée aux entiers naturels concernait l'ajout des deux règles d'introduction et la capacité de la fonction *prove* à déterminer quand utiliser quelle règle. En effet, du point de vue de la fonction, son paramètre est le même pour les deux règles d'introduction.

4. Natural numbers (5.12)

Enfin, la dernière section du devoir sur laquelle je me suis heurté, et que je n'ai pas réussi à terminer, concerne les entiers naturels dans le cadre des types dépendants. La difficulté provenait du peu d'informations fournies dans la consigne, mais aussi de l'apparition du principe d'induction qui, bien que familier, m'a posé problème dans le contexte de ce projet. Je n'ai pas pu achever cette section par manque de temps.

4. Choix d'implémentation

Pour les choix d'implémentation, je me suis majoritairement référé au cours, notamment aux règles de déduction naturelle, qui m'ont aidé pour les fonctions d'inférence de types. Je me suis également inspiré des programmes réalisés précédemment, en particulier le programme de lambda-calcul, pour certaines fonctions similaires.

5. Extensions possibles

Pour les extensions possibles, je pense que dans un premier temps, terminer toutes les questions et implémenter les parties optionnelles aurait été une bonne première étape.

6. Conclusion

L'implémentation d'un assistant de preuve et les difficultés rencontrées m'ont permis d'approfondir ma compréhension des différents concepts du cours, mais aussi de faire le lien entre les différents travaux dirigés en OCaml, en Agda, et le contenu du cours, qui était parfois confus dans mon esprit. Enfin, ce projet m'a aussi permis de confirmer et de développer ma compréhension du langage OCaml, que j'ai découvert avec *Computational Logic*.