Верификация структуры данных «зиппер»

Грахов Павел, ПМИ

Южный федеральный университет
Кафедра информатики и вычислительной техники

Научный руководитель — ст. преп. В.Н. Брагилевский

Ростов-на-Дону
2019

Постановка задачи

- Формальная верификация алгоритмов (Coq)
- Разработка надежного ПО
- Тестирование алгоритмов и ПО
- Алгоритмы $\rightarrow$ программы

- **Задача:**
  Разработать для некоторой структуры данных Coq-проект, содержащий:
  - Формализацию исходной структуры данных
  - Доказательство ее корректности
  - Тестирование
  - Генератор кода

  В качестве исходной структуры выбран «зиппер» для дерева

Формализация «зиппера»

- Пара курсор-контекст (Z_T, Z_C)
- Введенные операции
  - **MoveTop Z**
  - **MoveDown d Z**
  - **ZipperToTree Z**
  - **TreeToZipper T**
  - **Modify Z f**

Теоремы о «зиппере»

- Операции над зиппером сохраняют свойства исходного дерева
- Модификация курсора не меняет контекст
- Функции навигации не меняют исходное дерево

Coq-формализация

- Обработка ошибок усложняет доказательства
- Использование автоматизации

Обработка ошибок

- Проверяющий предикат

```
Definition CorrectMoveDownConditions D Z := ...
```

- Разные версии функции

```
Definition MoveDown D Z : ZipperTree := ...
Definition CheckAndMoveDown D Z :
    option ZipperTree := ...
Lemma SomethingAboutMoveDown:
    CorrectMoveDownConditions D Z -> ...
```

Автоматизация

- Использование специфических тактик (как `omega` либо `ring`)
- Механизм подсказок (`Hint`)
- Изменение конфигурации ядра (`Add Relation`, `Transparent`)

Тестирование

- Возможно, теорема неверна?
- Тестирование на большом количестве входных данных: QuickChick

```
Definition genInput : G (nat * (list nat)) :=
  genPair (choose (0, 100)) (listOf (choose (0, 100))).
Definition qc_test_insert_prop (nl: prod nat (list nat)) :=
  nth_insert_prop (fst nl) (snd nl).
QuickChick (forAll genInput qc_test_insert_prop).
```

Генерация кода

- Выходной язык: Haskell
- Прямая трансляция из Coq-кода будет работать **очень медленно**
- Необходимы оптимизации (возможно, в ущерб формальной корректности)

Оптимизация генерации кода

- Использование стандартных типов данных Haskell (`Integer`)
- `Inline` для простых определений

Полученные результаты

- Доказана корректность «зиппер» для древовидных структур
- Построен верифицированный генератор Haskell-кода для «зиппера»