Компиляция программ на OCaml в Lua

Цириков Семен Алексеевич, группа 18.Б11-мм

Научный руководитель: к.ф.-м.н. С.В. Григорьев

Консультант: программист ”IntelliJ Labs” Д.С. Косарев

СПбГУ

18 ноября 2020

Введение

- Ограничения встроенных систем
- Надёжность функционального программирования
- Сложность низкоуровневой разработки
- Эффективность компиляторных оптимизаций

Постановка задачи

Целью данной работы является создание инструмента компиляции из OCaml в Lua. Для этого сформулированы следующие задачи:

- Сравнить существующие решения:
  - BuckleScript
  - Js_of_ocaml
- Спроектировать трансляцию представления данных OCaml в Lua
- Реализовать прототип инструмента
- Протестировать созданное решение

Обзор аналогов

- Js_of_ocaml – трансляция из низкоуровневого байткода
- BuckleScript – трансляция из исходного кода

Трансляция

Проблемы стадии проектирования:

- Хэш-таблица – единственный составной тип в Lua
- Для многих конструкций отсутствует прямой аналог
- Некоторые техники требуют дополнительную кодогенерацию (пример: каррирование)
- Иной подход к организации модулей и областей видимости переменных

Архитектура

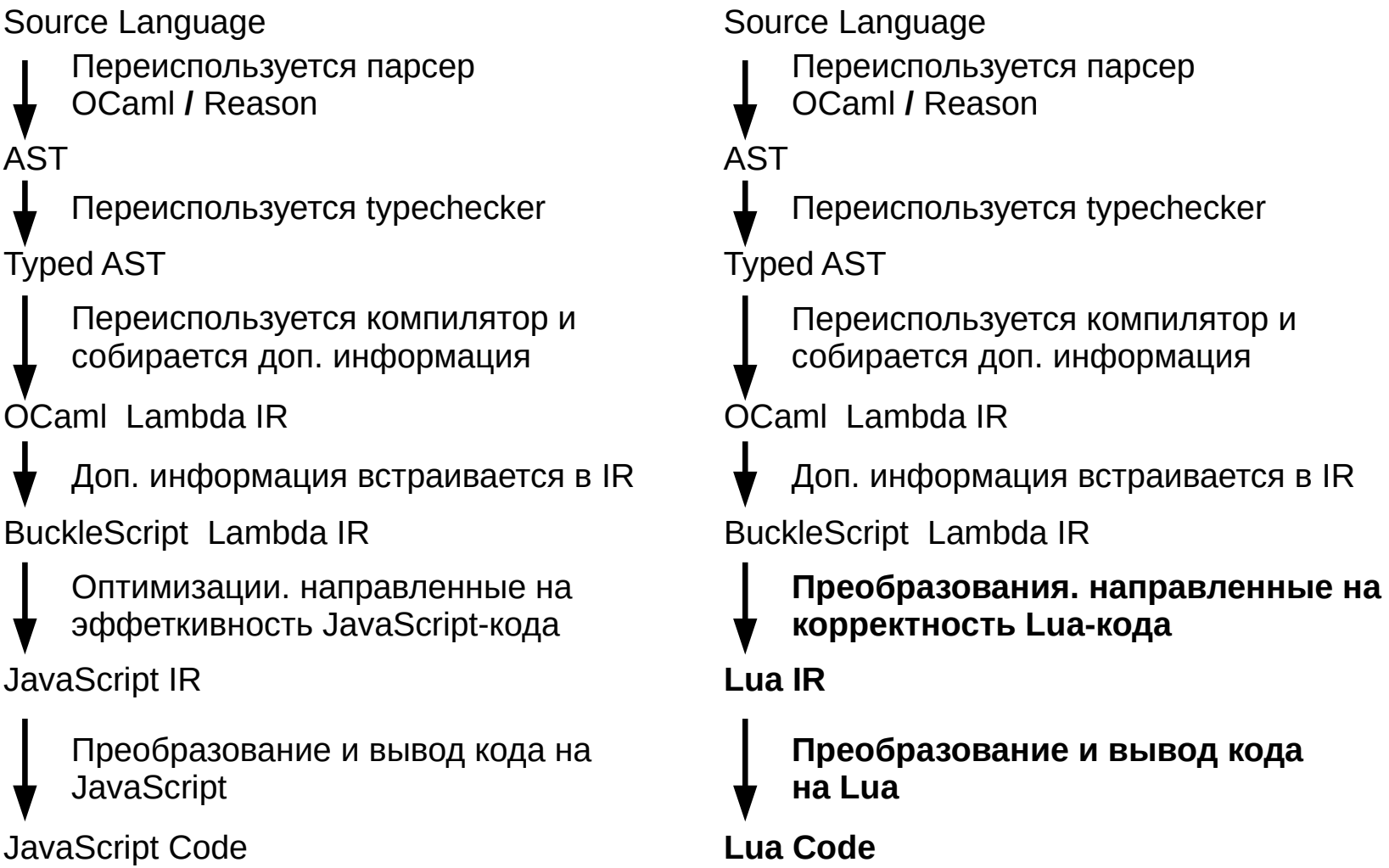

Рис. 1: Архитектура BuckleScript и её адаптация в проекте

Возможности и ограничения

Поддерживаются:

1. Базовые операции
2. Управляющие конструкции
3. Функции, рекурсия, каррирование
4. Обработка исключений
5. Импорт модулей

Не поддерживаются:

1. Массивы, списки
2. Функции ввода
3. Форматирование строк
4. Взаимодействие с ОС

Результаты

В ходе учебной практики достигнуты результаты:

1. Проведено исследование предметной области, в том числе изучены решения, принятые в аналогах проекта
2. Спроектирована трансляция представления данных из OCaml в Lua
3. Реализован прототип инструмента, доступный на GitHub репозитории
4. Полученное решение протестировано

Пример работы программы

Пример трансляции для пользовательского типа данных:

```
type expr =
    | Plus of expr * expr
    | Times of expr * expr
    | Value of string
let example = Times (
    Value ''n'',
    Plus (
        Value ''x'',
        Value ''y''
))
```

Listing 1: программа на OCaml

```
function Block.__(tag, block) do
  block.tag = tag
  return block
end end

example = ——[[Times]]Block.__(1, {
  ——[[Value]]Block.__(0, {''x''}),
  ——[[Value]]Block.__(0, {''y''}),
})
```

Listing 2: программа на Lua