Kompilator języka funkcyjnego na platformę JVM - szkic pracy dyplomowej

Jacek Duszenko

Planowane rozdziały:

1. Przedstawienie tematu pracy.
2. Wstęp do pracy.
3. Obecne rozwiązania w przemyśle i środowiskach akademickich.
4. Wirtualna maszyna Javy - opis i architektura.
5. Technologie użyte w implementacji kompilatora.
6. Opis implementowanego języka.
7. Przegląd zaimplementowanych funkcjonalności, struktur oraz elementów języka.
8. Przegląd napotkanych problemów oraz nieudanych implementacji.
9. Podsumowanie.
10. Bibliografia.

Szczegółowy opis rozdziałów:

1. Przedstawienie tematu pracy

W tym rozdziale omówiony zostanie temat pracy oraz motywacja oraz zakres implementacji.

2. Wstęp do pracy

We wstępie do pracy zostanie opisana geneza języków funkcyjnych, krótko opisane teoretyczne podwaliny języków funkcyjnych (rachunek lambda, teoria kategorii), przedstawienie zalet i wad programowania w paradygmacie funkcyjnym, przedstawienie cech języków funkcyjnych oraz powiązanie tych cech z pewnymi udogodnieniami oraz możliwościami, które one oferują.

3. Obecne rozwiązania w przemyśle i środowiskach akademickich

Opis kilku języków używanych w przemyśle i środowiskach akademickich z podziałem na języki czysto funkcyjne (Haskell) i nieczysto funkcyjne (Erlang, ML), języki implementowane na JVM (Scala, Clojure) oraz na innych platformach (F#). Porównanie różnic i podobieństw między tymi językami.

4. Wirtualna maszyna Javy - opis i architektura.

Opis architektury wirtualnej maszyny Javy, z podziałem na opisy poszczególnych części specyfikacji takich jak:

- Class loader
- Bytecode (opis niektórych instrukcji)
- Interpreter bytecode'u
- Kompilator JIT (Just in time)

5. Technologie użyte w implementacji kompilatora.

Opis technologii użytych w implementacji kompilatora. Motywacja wyboru języka implementacji kompilatora (Scala), opis technologii użytych do konstrukcji leksera, parsera (kombinatory parserów ze standardowej biblioteki Scali) oraz generatora kodu maszynowego (biblioteka asm do generowania kodu na JVM)

6. Opis implementowanego języka.

Gramatyka bezkontekstowa w formacie BNF implementowanego języka oraz jej opis. Opis s-wyrażeń w języku. Opis specyfikacji tego języka - system typów, natywne struktury sterowania, wywołania oraz aplikacje funkcji (w tym operatory arytmetyczne). Opis interakcji języka ze strumieniami wejścia-wyjścia.

7. Przegląd zaimplementowanych funkcjonalności, struktur oraz elementów języka.

Opis niektórych konstrukcji w języku wraz z ich teoretycznymi podwalinami i przełożeniem tego na kod kompilatora wraz z generowanym bytecodem. Pokazanie przykładowych programów wykorzystujących poszczególne konstrukcje języka wraz z kodem programu, wygenerowanym kodem pośrednim oraz efektami działania programu. Planowane konstrukcje:

- System typów.
- Wyrażenia let, letrec.
- Konstrukcje warunkowe (if, else, elif).
- Kompilowanie wyrażeń typu “pattern matching”. Opis przeistaczania wyrażenia do formy rozwiniętej funkcji (ang. curried function). Opis implementacji użytkowych podkonstrukcji, takich jak strażników dopasowania (ang. pattern guard), domyślnego dopasowania.
- Rekurencja.
- Optymalizacja wywołań ogonowych.

8. Przegląd napotkanych problemów oraz nieudanych implementacji.

Przegląd problemów oraz trudności napotkanych podczas implementacji kompilatora, opis tego, co można było zrobić lepiej oraz jak można było zapobiec napotkanym problemom zmieniając podejście implementacyjne. Przegląd nieudanych implementacji.

9. Podsumowanie

Wymienienie funkcjonalności, które udało się zaimplementować, zestawienie zaimplementowanych funkcjonalności z rozwiązaniami obecnymi w przemyśle. Decyzja czy udało się osiągnąć zaplanowany stan kompilatora. Perspektywy do dalszego rozwoju projektu - implementacja nowych cech języka, optymalizacja zaimplementowanych cech.

10. Bibliografia

Standardowy opis użytych źródeł w formacie bibtex.