Implementation of static and dynamic semantics for a calculus with algebraic effects and handlers using PLT Redex

Maciej Buszka

Instytut Informatyki UWr

04.10.2019

Plan prezentacji

Formalizacja języków programowania

Wspiera:

- Ortogonalność
- Przewidywalność
- Spójność

Wynik: łatwiejsze utrzymanie i rozwój programów

Efekty obliczeniowe (*side effects*)

- Zjawisko powszechne i znane każdemu programiście `print "Hello World!"`
- Zrozumienie działania programu wymaga nielokalnego wnioskowania
- Wpływ programu na otoczenie
- Dwie ogólne kategorie:
  - Modyfikacja lub odczytanie stanu
  - Alternatywne ścieżki sterowania

Efekty obliczeniowe - podejścia konwencjonalne

- Program może mieć dowolne efekty, a system typów się nimi nie interesuje (C, C++, Java ..., OCaml, Scala)
  - Brak pewności co do zachowania funkcji
  - Niemożliwość wnioskowania równościowego
  - Niejawne zależności między programami
- Język jest czysty, a programy z efektami są modelowane za pomocą monad (Haskell, PureScript)
  - Osobna składnia dla programów z efektami
  - Łączenie różnych efektów jest skomplikowane (np transformatory monad)
  - Problem z dodaniem efektów *post-factum*

Efekty algebraiczne - możliwości

- Jednolita składnia
- Pomocny system typów:
  Program

```
let x = Get 0 in
let y = Get 1 in
5 + (Put (x + y))
```

  Ma typ `Num` i korzysta z efektów `Get` oraz `Put`
- Kompozycja efektów
- Modelowanie obu kategorii
- Definicja własnych efektów

Efekty algebraiczne - wyzwania

- Nowe, aktualnie rozwijane zagadnienie
- Dość skomplikowana semantyka, ważne niuanse
- Różne podejścia w literaturze:
  - Semantyka statyczna: row-types, capabilities, abstract effects
  - Semantyka dynamiczna: shallow/deep handlers, effect lifting
- Wiele rozbudowanych języków eksperymentalnych: Koka, Helium, Eff, Frank, Links

Cel pracy

- Stworzenie formalnego rachunku:
  - Efekty algebraiczne
  - Pomocny system typów
  - Relacja redukcji
- Implementacja modelu
  - Algorytmiczna inferencja typu i efektu
  - Ewaluator wyrażeń rachunku
    - Obrazowanie wykonania krok po kroku
    - Redukcja do wyniku końcowego

Rachunek

- Abstrakcyjne operacje
  - Nie muszą być zdefiniowane a priori
  - Przyjmują jeden argument, zwracają jedną wartość
- Wyrażenia obsługujące (*handler*)
  - Dostęp do wznowienia
  - Semantyka tzw. głębokiej obsługi
  - Obsługa wielu operacji naraz
  - Klauzula `return`
- Wyrażenia podnoszące (*lift*)
  - 'przeskoczenie' najbliższego wyrażenia obsługującego
- Wartości bazowe oraz operacje na nich

Model

- Składnia abstrakcyjna
  - Dodatkowo: przyjazny język
- System typów
  - Algorytmiczna inferencja typu i efektów wyrażenia
- Relacja redukcji
  - Wykonanie krok po kroku
- Maszyna abstrakcyjna
  - Jawne środowisko
- Testy

Podsumowanie

Wynik:

- Formalny rachunek z efektami algebraicznymi
- Wykonywalny model rachunku

Zastosowania:

- Edukacja
  - Inferencja typu i efektów
  - Wizualizacja przekształceń
- Eksploracja
  - Iteracyjna rozbudowa
  - Testy

Dalsza praca: rozszerzenie o polimorfizm

Dziękuję za uwagę

Implementation of static and dynamic semantics for a calculus with algebraic effects and handlers using PLT Redex

Maciej Buszka

Instytut Informatyki UWr

04.10.2019