Implementacja języka funkcyjnego z rodziny ML z wykorzystaniem infrastruktury LLVM

Mateusz Lewko

6 września 2018

Spis treści

Obecnie

- Jest wiele języków z rodziny ML

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają
  - Polimorfizm parametryczny

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają
  - Polimorfizm parametryczny
  - Częściową aplikację

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają
  - Polimorfizm parametryczny
  - Częściową aplikację
  - Zagnieżdżone funkcje

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają
  - Polimorfizm parametryczny
  - Częściową aplikację
  - Zagnieżdżone funkcje
  - Funkcje wyższych rzędów

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają
  - Polimorfizm parametryczny
  - Częściową aplikację
  - Zagnieżdżone funkcje
  - Funkcje wyższych rzędów
  - System modułów (OCaml, SML) lub obiektowe klasy (F#)

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają
  - Polimorfizm parametryczny
  - Częściową aplikację
  - Zagnieżdżone funkcje
  - Funkcje wyższych rzędów
  - System modułów (OCaml, SML) lub obiektowe klasy (F#)
  - Trwałe rekordy, funkcje wzajemnie rekurencyjne, inferencja typów, algebraiczne typy danych, itp.

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają
  - **Polimorfizm parametryczny → Opakowywanie argumentów we wskaźnik**
  - Częściową aplikację
  - Zagnieżdżone funkcje
  - Funkcje wyższych rzędów
  - System modułów (OCaml, SML) lub obiektowe klasy (F#)
  - Trwałe rekordy, funkcje wzajemnie rekurencyjne, inferencja typów, algebraiczne typy danych, itp.

Obecnie

- Jest wiele języków z rodziny ML
- Zawierają
  - **Polimorfizm parametryczny $\Rightarrow$ Opakowywanie argumentów we wskaźnik**
  - Częściową aplikację
  - Zagnieżdżone funkcje
  - Funkcje wyższych rzędów
  - **System modułów (OCaml, SML)** lub obiektowe klasy (F#)
  - Trwałe rekordy, funkcje wzajemnie rekurencyjne, inferencja typów, inferencja typów, algebraiczne typy danych, itp.

Motywacja

- Wady opakowywania we wskaźnik

Motywacja

- Wady opakowywania we wskaźnik
  - Narzut pamięciowy — nawet 3x w przypadku typu int

Motywacja

- Wady opakowywania we wskaźnik
  - Narzut pamięciowy — nawet 3x w przypadku typu int
  - Narzut czasowy
    - Automatyczne zarządzenie pamięcią
    - Konieczność odczytywania pamięci ze sterty
    - Gorsze wykorzystanie pamięci cache

Motywacja

- Wady opakowywania we wskaźnik
  - Narzut pamięciowy — nawet 3x w przypadku typu int
  - Narzut czasowy
    - Automatyczne zarządzenie pamięcią
    - Konieczność odczytywania pamięci ze sterty
    - Gorsze wykorzystanie pamięci cache
- Wady systemu modułów

Motywacja

- Wady opakowywania we wskaźnik
  - Narzut pamięciowy — nawet 3x w przypadku typu int
  - Narzut czasowy
    - Automatyczne zarządzenie pamięcią
    - Konieczność odczytywania pamięci ze sterty
    - Gorsze wykorzystanie pamięci cache
- Wady systemu modułów
  - Brak możliwości przeładowaniu operatorów i funkcji (np. dla różnych typów numerycznych)

Motywacja

- Wady opakowywania we wskaźnik
  - Narzut pamięciowy — nawet 3x w przypadku typu int
  - Narzut czasowy
    - Automatyczne zarządzenie pamięcią
    - Konieczność odczytywania pamięci ze sterty
    - Gorsze wykorzystanie pamięci cache
- Wady systemu modułów
  - Brak możliwości przeładowaniu operatorów i funkcji (np. dla różnych typów numerycznych)
  - Nietrywialne w implementacji i skomplikowane w użyciu

Język MonoML

Język MonoML

- **Polimorfizm parametryczny $\longrightarrow$ Monomorfizacja**

Język MonoML

- **Polimorfizm parametryczny $\longrightarrow$ Monomorfizacja**
- **Częściową aplikację $\longrightarrow$ Bazowana na modelu push/enter**

Język MonoML

- **Polimorfizm parametryczny $\longrightarrow$ Monomorfizacja**
- **Częściową aplikację $\longrightarrow$ Bazowana na modelu push/enter**
- **Klasy typów (ad–hoc polimorfizm)**

Język MonoML

- **Polimorfizm parametryczny $\longrightarrow$ Monomorfizacja**
- **Częściową aplikację $\longrightarrow$ Bazowana na modelu push/enter**
- **Klasy typów (ad–hoc polimorfizm)**
- Zagnieżdżone funkcje
- Funkcje wyższych rzędów
- Trwałe rekordy, funkcje wzajemnie rekurencyjne, inferencja typów

Opis problemu

Kod do skompilowania

```
let twice f x = f (f x)
let _ =
  print_int   (twice identity 42  );
  print_float (twice identity 42.0)
```

Opis problemu

Kod do skompilowania

```
let twice f x = f (f x)
let _ =
  print_int   (twice identity 42  );
  print_float (twice identity 42.0)
```

Wygenerowany LLVM IR #1

```
define i32 @twice(i32 (i32)*, i32) {
    ...
}
```

Opis problemu

Kod do skompilowania

```
let twice f x = f (f x)
let _ =
  print_int   (twice identity 42  );
  print_float (twice identity 42.0)
```

Wygenerowany LLVM IR #2

```
define float @twice(float (float)*, float) {
    ...
}
```

Opis problemu

Kod do skompilowania

```
let twice f x = f (f x)
let _ =
  print_int   (twice identity 42  );
  print_float (twice identity 42.0)
```

Argumenty opakowane we wskaźnik

```
define i8* @twice(i8* (i8*)*, i8*) {
    ...
}
```

Moje podejście — Monomorfizacja

Po monomorfizacji

```
let twice_int (f : int -> int) (x : int) : int =
  f (f x)
let twice_float (f : float -> float) (x : float)
  : float = f (f x)

let _ =
  print_int   (twice_int identity_int 42  );
  print_float (twice_float identity_float 42.0)
```

Testy wydajnościowe
Cele

- Porównanie czasów wykonania funkcji polimorficznej i monomorficznej

Testy wydajnościowe
Cele

- Porównanie czasów wykonania funkcji polimorficznej i monomorficznej
  - w MonoMLu

Testy wydajnościowe
Cele

- Porównanie czasów wykonania funkcji polimorficznej i monomorficznej
  - w MonoMLu
  - w Haskellu, Javie i Standard MLu

Testy wydajnościowe
Cele

- Porównanie czasów wykonania funkcji polimorficznej i monomorficznej
  - w MonoMLu
  - w Haskellu, Javie i Standard MLu
- Narzut czasowy wywoływania funkcji w MonoMLu na tle innych języków

Testy wydajnościowe
Przygotowanie

Funkcja polimorficzna

```
let rec sum n (curr : 'a) (x : 'a) : 'a =
    if n = 0 then curr
    else sum (n - 1) (add curr x) x
```

Testy wydajnościowe
Przygotowanie

Funkcja polimorficzna

```
let rec sum n (curr : 'a) (x : 'a) : 'a =
    if n = 0 then curr
    else sum (n - 1) (add curr x) x
```

Monomorficzna

```
let rec sum n (curr : int) (x : int) : int =
    if n = 0 then curr
    else sum (n - 1) (add curr x) x
```

Testy wydajnościowe
Przygotowanie

Funkcja polimorficzna

```
sumPoly :: Num a => Int -> a ->  a -> a
sumPoly 0 curr _ = curr
sumPoly n curr x = sumPoly (n - 1) (curr + x) $! x
```

Monomorficzna

```
sumMono :: Int# -> Int# -> Int# -> Int#
sumMono 0# curr _ = curr
sumMono n curr x = sumMono (n -# 1#) (curr +# x) x
```

Testy wydajnościowe
Wyniki

Język	Wersja	Czas (ms)	σ	x
Haskell (GHC)	Mono	**39.1**	8.2	0.10
Haskell (GHC)	Poli	**696.8**	63.2	1.86

Testy wydajnościowe
Wyniki

Język	Wersja	Czas (ms)	σ	x
Haskell (GHC)	Mono	39.1	8.2	0.10
Haskell (GHC)	Poli	696.8	63.2	1.86
Java	Mono	**140.1**	65.7	0.37
Java	Poli	**564.9**	24.8	1.50

Testy wydajnościowe
Wyniki

Język	Wersja	Czas (ms)	σ	x
Haskell (GHC)	Mono	39.1	8.2	0.10
Haskell (GHC)	Poli	696.8	63.2	1.86
Java	Mono	140.1	65.7	0.37
Java	Poli	564.9	24.8	1.50
SML (MLton)	Mono	**151.0**	13.7	0.40
SML (SML/NJ)	Poli	**357.6**	14.4	0.95

Testy wydajnościowe
Wyniki

Język	Wersja	Czas (ms)	σ	x
Haskell (GHC)	Mono	39.1	8.2	0.10
Haskell (GHC)	Poli	696.8	63.2	1.86
Java	Mono	140.1	65.7	0.37
Java	Poli	564.9	24.8	1.50
SML (MLton)	Mono	151.0	13.7	0.40
SML (SML/NJ)	Poli	357.6	14.4	0.95
Mono ML	Mono	**327.0**	52.3	0.88
Mono ML	Poli	**375.4**	46.9	1.00

Blocks

Block Title

You can also highlight sections of your presentation in a block, with it's own title

Theorem

There are separate environments for theorems, examples, definitions and proofs.

Example

Here is an example of an example block.

Blocks

Block Title

You can also highlight sections of your presentation in a block, with it's own title

Theorem

There are separate environments for theorems, examples, definitions and proofs.

Example

Here is an example of an example block.

Summary

- The first main message of your talk in one or two lines.
- The second main message of your talk in one or two lines.
- Perhaps a third message, but not more than that.

- Outlook
  - Something you haven't solved.
  - Something else you haven't solved.

For Further Reading I


A. Author.
Handbook of Everything.
Some Press, 1990.


S. Someone.
On this and that.
Journal of This and That, 2(1):50–100, 2000.