

# Wymagający złamania wierszy tytuł pracy w języku polskim

(English title)

Mateusz Lewko

Praca licencjacka

**Promotor:** dr hab. Dariusz Biernacki

Uniwersytet Wrocławski  
Wydział Matematyki i Informatyki  
Instytut Informatyki

16 czerwca 2018



## Streszczenie

TODO polish abstract

---

TODO english abstract



# Spis treści

<b>1. Wprowadzenie</b>	<b>7</b>
1.1. Język ML . . . . .	7
1.1.1. Składnia . . . . .	7
1.2. Cechy języka . . . . .	7
1.3. Klasy typów . . . . .	7
1.4. Infrastruktura LLVM . . . . .	7
<b>2. Kompilator</b>	<b>9</b>
2.1. Etapy kompilacji . . . . .	9
2.2. Analiza leksykalna . . . . .	9
2.3. Parsowanie . . . . .	9
2.4. Inferencja typów . . . . .	9
<b>3. Generowanie kodu</b>	<b>11</b>
3.1. Częściowa aplikacja . . . . .	11
3.1.1. Opis działania . . . . .	11
3.1.2. Porównanie z innymi implementacjami . . . . .	11
3.2. Zagnieżdżone funkcje . . . . .	11
3.3. Rekordy . . . . .	11
3.4. Let polimorfizm . . . . .	12
3.5. Klasy typów . . . . .	12



# Rozdział 1.

## Wprowadzenie

### 1.1. Język ML

1. Dlaczego ML, jakie są inne języki ML
2. Bazowanie na  $F\#$

#### 1.1.1. Składnia

1. Opis, szczegóły składni, (przykłady: każda cecha języka i krótki przykład)

### 1.2. Cechy języka

1. Proste wyrażenia, rekurencja, let-polymorphism, rekordy, wzajemnie rekurencyjne funkcje na top levelu, klasy typów, proste moduły, wyrażanie na top levelu, efekty uboczne, inferencja typów, anotacje.

### 1.3. Klasy typów

1. Wprowadzenie czym są
2. Dlaczego? Jakie są alternatywy
3. Opis tego co zostało zaimplementowane, porównanie do innych języków, (Haskell, Rust, Scala)

### 1.4. Infrastruktura LLVM

1. Co to jest?

2. Dlaczego LLVM i jakie są inne opcje (C, assembler)?
3. Jak działa kompilowanie do LLVM?
4. Krótki opis high-llvm



## Rozdział 2.

# Kompilator

### 2.1. Etapy kompilacji

1. Jakie są etapy (lexer → parser → untyped ast → typed ast bez zagnieżdżonych funkcji → generowanie kodu (ast high-llvm) → wywoływanie funkcji z api llvma → llc → gcc i external)

2. Krótko o każdym etapie

### 2.2. Analiza leksykalna

1. Czego użyłem.

2. Analiza wcięć

### 2.3. Parsowanie

1. Czego użyłem, coś o Menhirze, dlaczego Menhir

2. Wyzwania (składnia bazująca na wcięciach)

3. Gramatyka

### 2.4. Inferencja typów

1. Po co? Jak działa u mnie



## Rozdział 3.

# Generowanie kodu

### 3.1. Częściowa aplikacja

#### 3.1.1. Opis działania

1. Dlaczego jest to nietrywialne
2. Jakie miałem cele
3. Jak to działa u mnie
4. Przykład (wygenerowanego pseudo-kodu)

#### 3.1.2. Porównanie z innymi implementacjami

1. Push/enter vs eval/apply

Porównanie z pracą "Making a fast curry: ..."

### 3.2. Zagnieżdżone funkcje

1. Co to są zagnieżdżone funkcje
2. Na czym polega trudność w ich implementacji
3. Jak zostały zaimplementowane: lambda lifting + closure conversion + wykorzystanie aplikacji częściowej

### 3.3. Rekordy

Implementacja, porównanie do rekordów w F#.

### 3.4. Let polimorfizm

1. Krótki opis, czym jest let-polimorfizm
2. Sposoby implementacji w różnych językach, zalety i wady
3. Sposób implementacji u mnie

### 3.5. Klasy typów

1. Czym są? Po co?
2. Sposoby implementowania, porównanie do pracy TODO
3. Jak zostały zaimplementowane, dlaczego tak