Rust i napredni tipski sustavi

7. Rust - Jednostavan funkcijski jezik

Ivan Radiček

8. svibnja 2021.

Plan

- Prošli puta:
  - Rust: moduli i macroi
  - Dio sintakse kalkulatora
- Danas:
  - Dovršetak sintakse i semantike kalkulatora
  - Opis sintakse i semantike mini funkcijskog jezika
  - Rust: funkcije kao građani prvog reda

Podsjetnik

- Gradimo interpreter za jednostavan *kalkulator* jezik
  - Sintaksa: zadana gramatikom: $e := \overline{n} \mid e + e \mid e * e \mid (e)$
  - Semantika: standardna aritmetika/kalkulator
- Sintaksa/parsiranje u dva koraka (prisjetiti se definicije tokena i izraza):
  - `String -> Vec<Token>` (lexer, koji pretvara string u vektor tokena)
  - `Vec<Token> -> Expr` (parser, koji pretvara vektor tokena u izraz)
- Semantika: funkcija `Expr -> f64`

Ostala pravila

- Kako definirati pravila za + i $*$?

Ostala pravila

- Kako definirati pravila za + i ∗?

```
1    expr ::= expr(e1) Plus expr(e2) { Expr::Op(Op::Plus, Box::new(e1), Box::new(e2)) }
2    expr ::= expr(e1) Times expr(e2) { Expr::Op(Op::Times, Box::new(e1), Box::new(e2)) }
```

Ostala pravila

- Kako definirati pravila za + i ∗?

```
1    expr ::= expr(e1) Plus expr(e2) { Expr::Op(Op::Plus, Box::new(e1), Box::new(e2)) }
2    expr ::= expr(e1) Times expr(e2) { Expr::Op(Op::Times, Box::new(e1), Box::new(e2)) }
```

- Zašto ne funkcionira? Postoji konflikt . . .

Asocijativnost i prednost operatora

- Kako izgleda stablo za `3 + 4 + 5`?
- A kako za `3 * 4 + 5`?
- Postoji više opcija
- Mi želimo:
  - `*` i `+` su **lijevo**-asocijativni
  - `+` ima prednost pred `*`

Asocijativnost i prednost operatora

- Kako izgleda stablo za `3 + 4 + 5`?
- A kako za `3 * 4 + 5`?
- Postoji više opcija
- Mi želimo:
  - `*` i `+` su **lijevo**-asocijativni
  - `+` ima prednost pred `*`

```
1   %left Plus;
2   %left Times;
```

Asocijativnost i prednost operatora

- Kako izgleda stablo za `3 + 4 + 5`?
- A kako za `3 * 4 + 5`?
- Postoji više opcija
- Mi želimo:
  - `*` i `+` su **lijevo**-asocijativni
  - `+` ima prednost pred `*`

```
1   %left Plus;
2   %left Times;
```

- OK, kako pretvoriti string u vektor tokena?
- Prije toga ćemo još malo o funkcijama

Funkcije kao građani prvog reda

- Funkcije kao obične vrijednosti
- U funkcijskim jezicima funkcije stvarno jesu prvog reda (više o tome kasnije)
- U Rustu ovisi o tome kako koristi memoriju (naravno!)
- Tip funkcije (npr. u argumentu funkcije): `fn(T1, ..., Tn) -> T`
- Lambda/closure: `|x1, ..., xn| e` ili
  `|x1 : T1, ..., xn : Tn| -> T { e }`

Rezultat?

```
1   fn apply_twice<T>(x : T, f : fn(T) -> T) -> T {
2     f(f(x))
3   }
4
5   fn main() {
6     let f1 = |x| x + 1;
7     let f2 = |x : isize| -> isize { x + 2 };
8     let x = 3;
9     let y1 = apply_twice(x, f1);
10    let y2 = apply_twice(x, f2);
11    println!("f1^2(x)={}, f2^2(x)={}", y1, y2);
12  }
```

Pokazivači na funkcije vs. closures

- Closure "*zarobljava*" (captures) vrijednost koja dolazi **izvan funkcije**
- Za to treba dodati `move` ispred definicije funkcije
- No, tip više nije pointer na funkciju, već poseban closure tip (probaj promijeniti kod sa slajda gore) koji možemo koristiti sa `dyn Fn(T1, ..., Tn) -> T`

```
1   fn add_x(x : isize) -> Box<dyn Fn(isize) -> isize> {
2     Box::new(move |y| x + y)
3   }
4
5   fn apply_twice<T>(x: T, f: Box<dyn Fn(T) -> T>) -> T { f(f(x)) }
6
7   fn main() {
8     let f1 = add_x(1);
9     let f2 = add_x(2);
10    let x = 3;
11    let y1 = apply_twice(x, f1);
12    let y2 = apply_twice(x, f2);
13    println!("f1^2(x)={}, f2^2(x)={}", y1, y2);
14  }
```

- Knjiga: https://doc.rust-lang.org/book/ch13-01-closures.html i https://doc.rust-lang.org/book/ch19-05-advanced-functions-and-closures.html

Definicija lexera

- Lexer ćemo definirati pomoću cratea `regex-lexer`
- Svaki token se definira pomoću pripadajućeg regularnog izraza (regex, vidi: https://en.wikipedia.org/wiki/Regular_expression) i funkcije koja definira kako se string pretvara u vrijednost tokena
- Npr. `r"\("`, `"[0-9]+"`, ...

Generalna sintaksa `regex-lexera`

```
1  let lexer = regex_lexer::LexerBuilder::new()
2    .token(<regex_1>, <funkcija_1>)
3    ...
4    .token(<regex_n>, <funkcija_n>)
5    .build()
6    .unwrap();
7
8  let toks : Vec<Token> = lexer.tokens(s).collect();
```

Lexer kalkulatora

```
let lexer = regex_lexer::LexerBuilder::new()
    .token(r"\(", |_| Some(Token::LParen))
    .token(r"\)", |_| Some(Token::RParen))
    .token(r"\*", |_| Some(Token::Times))
    .token(r"\+", |_| Some(Token::Plus))
    .token(r"[0-9]+(\.[0-9]+)?",
           |s| Some(Token::Number(s.to_string())))
    .token(r"\s+", |_| None) // skip whitespace
    .build()
    .unwrap();
```

- Napomena: dodati
  `%token #[derive(Clone, Debug)] pub enum Token {};` na vrh pomelo definicije za definiranje tipa tokena i izvedbu potrebnih traitova

Semantika

- Jednostavna rekurzivna definicija funkcije koja prima `Expr` i vraća `f64`

Semantika

- Jednostavna rekurzivna definicija funkcije koja prima `Expr` i vraća `f64`

```
1  fn eval(e : &Expr) -> f64 {
2    match e {
3      Expr::Num(s) => s.parse().unwrap(),
4      Expr::BinOp(op, e1, e2) => {
5        let v1 = eval(e1);
6        let v2 = eval(e2);
7        match op {
8          Op::Plus => v1 + v2,
9          Op::Times => v1 * v2
10       }
11     }
12   }
13 }
```

Zadaci

- Dodati mogućnost da se izraz (string) učitava pomoću argumenata programa
- Proširiti jezik za funkcijama i konstantama
  - Npr. `"2.0 + cos(pi) + pow(2, 3)"`
  - Funkcije mogu biti "modularne":
    - Novi token - ime funkcije/konstante
    - Lista argumenata može se definirati sa definicijom: $args = e \mid e, args$ (da li je moguće dozvoliti 0 argumenata?)

Mini funkcijski jezik - brzi tečaj

- Koji je rezultat sljedećih izraza?

Običan `let` izraz

```
1   let x = 21 in x + x
```

Mini funkcijski jezik - brzi tečaj

- Koji je rezultat sljedećih izraza?

Običan `let` izraz

```
1  let x = 21 in x + x
```

Rezultat: 42

Mini funkcijski jezik - brzi tečaj

- Koji je rezultat sljedećih izraza?

Običan `let` izraz

```
1  let x = 21 in x + x
```

Rezultat: 42

Općenito - `let` izraz

```
1  let x = e1 in e2
```

Mini funkcijski jezik - brzi tečaj

- Koji je rezultat sljedećih izraza?

`let` definicija funkcije

```
1  let add x y = x + y in
2  let x = 22 in
3  let z = 20 in
4  add x z
```

(Rezultat: 42)

Mini funkcijski jezik - brzi tečaj

- Koji je rezultat sljedećih izraza?

`let` definicija funkcije

```
1  let add x y = x + y in
2  let x = 22 in
3  let z = 20 in
4  add x z
```

(Rezultat: 42)

Općenito - generalni `let` izraz

```
1  (* Ako ima argumenata definiramo funkciju,
2     inače običnu vrijednost *)
3  let f x1 ... xn = e1 in e2
```

Mini funkcijski jezik - brzi tečaj

- Koji je rezultat sljedećih izraza?

`let` definicija funkcije

```
1  let add x y = x + y in
2  let x = 22 in
3  let z = 20 in
4  add x z
```

(Rezultat: 42)

Općenito - generalni `let` izraz

```
1  (* Ako ima argumenata definiramo funkciju,
2     inače običnu vrijednost *)
3  let f x1 ... xn = e1 in e2
```

Općenito - poziv funkcije (aplikacija)

```
1  (* e je izraz koji definira funkciju, e1 ... en argumenti *)
2  e e1 ... en
```

Zašto **funkcijski** jezici?

- Funkcija je vrijednost kao bilo koja druga, građanka prvog reda!

```
1  let add n =
2    let f x = x + n in
3    f
4  in
5  let g1 = add 5 in
6  let g2 = add 10 in
7  (g1 8) + (g2 11)
```

Zašto **funkcijski** jezici?

- Funkcija je vrijednost kao bilo koja druga, građanka prvog reda!

```
1  let add n =
2    let f x = x + n in
3    f
4  in
5  let g1 = add 5 in
6  let g2 = add 10 in
7  (g1 8) + (g2 11)
```

Rezultat: 34

Zašto **funkcijski** jezici?

- Funkcija je vrijednost kao bilo koja druga, građanka prvog reda!

```
let add n =
  let f x = x + n in
  f
in
let g1 = add 5 in
let g2 = add 10 in
(g1 8) + (g2 11)
```

Rezultat: 34

```
let comp f g =
  let fc x = f (g x) in fc
in
let abs x = if x < 0 then (-x) else x in
let times2 x = 2 * x in
let f = comp abs times2 in
f (-22)
```

Zašto **funkcijski** jezici?

- Funkcija je vrijednost kao bilo koja druga, građanka prvog reda!

```
1  let add n =
2    let f x = x + n in
3    f
4  in
5  let g1 = add 5 in
6  let g2 = add 10 in
7  (g1 8) + (g2 11)
```

Rezultat: 34

```
1  let comp f g =
2    let fc x = f (g x) in fc
3  in
4  let abs x = if x < 0 then (-x) else x in
5  let times2 x = 2 * x in
6  let f = comp abs times2 in
7  f (-22)
```

Rezultat: 42

Osnovni sastojci - n-torke

```
1  let t = (5, 42, false) in t.1
```

Rezultat:

Osnovni sastojci - n-torke

```
1  let t = (5, 42, false) in t.1
```

Rezultat: 5 (po dogovoru elementi n-torke počinju sa indeksom jedan)

Osnovni sastojci - n-torke

```
1  let t = (5, 42, false) in t.1
```

Rezultat: 5 (po dogovoru elementi n-torke počinju sa indeksom jedan)

```
1  let swap t = (t.2, t.1) in
2  swap (0, 1)
```

Rezultat: `(1, 0)`

Osnovni sastojci - liste

Lista i match

```
1  let l0 = nil in
2  let l1 = 1::l0 in
3  let l2 = 2::l1 in
4  let len l =
5    match len with
6    | nil => 0
7    | h::t => (len t) + 1
8  in (len l2) * (len l1)
```

Rezultat:

Osnovni sastojci - liste

Lista i match

```
1  let l0 = nil in
2  let l1 = 1::l0 in
3  let l2 = 2::l1 in
4  let len l =
5    match len with
6    | nil => 0
7    | h::t => (len t) + 1
8  in (len l2) * (len l1)
```

Rezultat: 2

Još malo liste

```
1  let insert x l =
2    match l with
3    | nil => x::nil
4    | h::t => if x < h then x::h::t else h::(insert x t)
5  in
6  let isort l =
7    match l with
8    | nil => nil
9    | h::t => insert h (isort t)
10 in sort [1, 5, 4, 2, 3] (* skraćeno za 1::5::4::2::3::nil *)
```

Rezultat:

Još malo liste

```
1  let insert x l =
2    match l with
3    | nil => x::nil
4    | h::t => if x < h then x::h::t else h::(insert x t)
5  in
6  let isort l =
7    match l with
8    | nil => nil
9    | h::t => insert h (isort t)
10 in sort [1, 5, 4, 2, 3] (* skraćeno za 1::5::4::2::3::nil *)
```

Rezultat: `[1, 2, 3, 4, 5]`

Formalna sintaksa jezika

$$
\begin{array}{rcll}
e & := & x & \mid \texttt{let}\ x\ =\ e_1\ \texttt{in}\ e_2 \\
 & & e\ e_1 \cdots e_n & \mid \texttt{let}\ f\ x_1 \cdots x_n\ =\ e_1\ \texttt{in}\ e_2 \\
 & & e.i & \mid (e_1, \ldots, e_n) \\
 & & \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 & \mid \texttt{true} \mid \texttt{false} \\
 & & \texttt{match}\ e\ \texttt{with} & \\
 & & \texttt{nil} \Rightarrow e_1, h :: t \Rightarrow e_2 & \mid e_1 :: e_2 \mid \texttt{nil}
\end{array}
$$

Opis semantike jezika jezika

- v je vrijednost
- Γ je lista vrijednosti za varijable (zove se još i okolina)
  - npr. $x : 3, y : \texttt{true}, z : 4$
  - na početku izvođenja programma Γ je prazna ($\emptyset$)

Pod okolinom Γ, izraz e evaluira u vrijednost v

$$\Gamma \vdash e \to v$$

Vrijednosti

$$\begin{aligned} v \quad := \quad & n \mid \texttt{true} \mid \texttt{false} \\ \mid \quad & (v_1, \ldots, v_2) \\ \mid \quad & \texttt{nil} \mid v_1 :: v_2 \\ \mid \quad & \langle \Gamma, \lambda\ x_1, \ldots, x_n.\ e \rangle \end{aligned}$$

Prvo semantičko pravilo

- Svako pravilo može imati i jedan ili više preduvjet (piše se iznad crte)

Varijabla

$$\frac{(x : v) \in \Gamma}{\Gamma \vdash x \to v}$$

`let` izraz

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \Gamma, x : v_1 \vdash e_2 \to v_2}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \to v_2}$$

Primjer upotrebe pravila

$$\emptyset \vdash \begin{array}{l} \texttt{let } x = 2 \texttt{ in} \\ \texttt{let } y = x + 5 \texttt{ in } \rightarrow \\ x * y \end{array}$$

Primjer upotrebe pravila

$$\emptyset \vdash 2 \to 2$$

$$\emptyset \vdash \begin{array}{l} \texttt{let } x = 2 \texttt{ in} \\ \texttt{let } y = x + 5 \texttt{ in } \to \\ x * y \end{array}$$

Primjer upotrebe pravila

$$\dfrac{\emptyset \vdash 2 \to 2 \qquad \dfrac{}{x : 2 \vdash \begin{array}{l}\texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to}}{\emptyset \vdash \begin{array}{l}\texttt{let } x = 2 \texttt{ in} \\ \texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to}$$

Primjer upotrebe pravila

$$\dfrac{\emptyset \vdash 2 \to 2 \qquad \dfrac{x : 2 \vdash x + 5 \to 7}{x : 2 \vdash \begin{array}{l}\texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to}}{\emptyset \vdash \begin{array}{l}\texttt{let } x = 2 \texttt{ in} \\ \texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to}$$

Primjer upotrebe pravila

$$\dfrac{\emptyset \vdash 2 \to 2 \qquad \dfrac{x : 2 \vdash x + 5 \to 7 \qquad x : 2, y : 7 \vdash x * y \to 14}{x : 2 \vdash \begin{array}{l}\texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to}}{\emptyset \vdash \begin{array}{l}\texttt{let } x = 2 \texttt{ in} \\ \texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to}$$

Primjer upotrebe pravila

$$\dfrac{\emptyset \vdash 2 \to 2 \qquad \dfrac{x : 2 \vdash x + 5 \to 7 \qquad x : 2, y : 7 \vdash x * y \to 14}{x : 2 \vdash \begin{array}{l}\texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to 14}}{\emptyset \vdash \begin{array}{l}\texttt{let } x = 2 \texttt{ in} \\ \texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to}$$

Primjer upotrebe pravila

$$\dfrac{\emptyset \vdash 2 \to 2 \qquad \dfrac{x : 2 \vdash x + 5 \to 7 \qquad x : 2, y : 7 \vdash x * y \to 14}{x : 2 \vdash \begin{array}{l}\texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to 14}}{\emptyset \vdash \begin{array}{l}\texttt{let } x = 2 \texttt{ in} \\ \texttt{let } y = x + 5 \texttt{ in} \\ x * y\end{array} \to 14}$$

Semantika n-torki

Kreiranje n-torki

$$\frac{}{\Gamma \vdash (e_1, \ldots, e_n) \rightarrow}$$

Korištenje n-torki (projekcija)

$$\frac{}{\Gamma \vdash e.i \rightarrow}$$

Semantika n-torki

Kreiranje n-torki

$$\frac{\Gamma \vdash e_1 \to v_1}{\Gamma \vdash (e_1, \ldots, e_n) \to}$$

Korištenje n-torki (projekcija)

$$\frac{}{\Gamma \vdash e.i \to}$$

Semantika n-torki

Kreiranje n-torki

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \ldots \qquad \Gamma \vdash e_n \to v_n}{\Gamma \vdash (e_1, \ldots, e_n) \to}$$

Korištenje n-torki (projekcija)

$$\frac{}{\Gamma \vdash e.i \to}$$

Semantika n-torki

Kreiranje n-torki

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \ldots \qquad \Gamma \vdash e_n \to v_n}{\Gamma \vdash (e_1, \ldots, e_n) \to (v_1, \ldots, v_n)}$$

Korištenje n-torki (projekcija)

$$\frac{}{\Gamma \vdash e.i \to}$$

Semantika n-torki

Kreiranje n-torki

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \ldots \qquad \Gamma \vdash e_n \to v_n}{\Gamma \vdash (e_1, \ldots, e_n) \to (v_1, \ldots, v_n)}$$

Korištenje n-torki (projekcija)

$$\frac{\Gamma \vdash e \to (v_1, \ldots, v_n)}{\Gamma \vdash e.i \to}$$

Semantika n-torki

Kreiranje n-torki

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \ldots \qquad \Gamma \vdash e_n \to v_n}{\Gamma \vdash (e_1, \ldots, e_n) \to (v_1, \ldots, v_n)}$$

Korištenje n-torki (projekcija)

$$\frac{\Gamma \vdash e \to (v_1, \ldots, v_n) \qquad \text{tako da } 1 \leq i \leq n}{\Gamma \vdash e.i \to v_i}$$

Semantika Boolean tipa podataka

Kreiranje (uvjetno rečeno)

$$\Gamma \vdash \texttt{true} \to \texttt{true}$$
$$\Gamma \vdash \texttt{false} \to \texttt{false}$$

Korištenje Boolean tipa

$$\frac{}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \to}$$

Semantika Boolean tipa podataka

Kreiranje (uvjetno rečeno)

$$\Gamma \vdash \texttt{true} \to \texttt{true}$$
$$\Gamma \vdash \texttt{false} \to \texttt{false}$$

Korištenje Boolean tipa

$$\frac{\Gamma \vdash e \to \texttt{true}}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \to}$$

Semantika Boolean tipa podataka

Kreiranje (uvjetno rečeno)

$$\Gamma \vdash \texttt{true} \to \texttt{true}$$
$$\Gamma \vdash \texttt{false} \to \texttt{false}$$

Korištenje Boolean tipa

$$\frac{\Gamma \vdash e \to \texttt{true}}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \to} \qquad \frac{\Gamma \vdash e \to \texttt{false}}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \to}$$

Semantika Boolean tipa podataka

Kreiranje (uvjetno rečeno)

$$\Gamma \vdash \texttt{true} \to \texttt{true}$$
$$\Gamma \vdash \texttt{false} \to \texttt{false}$$

Korištenje Boolean tipa

$$\frac{\Gamma \vdash e \to \texttt{true} \qquad \Gamma \vdash e_1 \to v}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \to v} \qquad \frac{\Gamma \vdash e \to \texttt{false}}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \to}$$

Semantika Boolean tipa podataka

Kreiranje (uvjetno rečeno)

$$\Gamma \vdash \texttt{true} \to \texttt{true}$$
$$\Gamma \vdash \texttt{false} \to \texttt{false}$$

Korištenje Boolean tipa

$$\frac{\Gamma \vdash e \to \texttt{true} \qquad \Gamma \vdash e_1 \to v}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \to v} \qquad \frac{\Gamma \vdash e \to \texttt{false} \qquad \Gamma \vdash e_2 \to v}{\Gamma \vdash \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2 \to v}$$

Semantika liste

Kreiranje liste

$$\Gamma \vdash \texttt{nil} \rightarrow \qquad \frac{}{\Gamma \vdash e1 :: e2 \rightarrow}$$

Korištenje liste

$$\frac{}{\Gamma \vdash \texttt{match}\ e\ \texttt{with}\ \ \texttt{nil} \Rightarrow e_1,\ h :: t \Rightarrow e_2 \rightarrow}$$

Semantika liste

Kreiranje liste

$$\Gamma \vdash \texttt{nil} \rightarrow \texttt{nil} \qquad \frac{}{\Gamma \vdash e1 :: e2 \rightarrow}$$

Korištenje liste

$$\frac{}{\Gamma \vdash \texttt{match}\ e\ \texttt{with}\ \ \texttt{nil} \Rightarrow e_1,\ h :: t \Rightarrow e_2 \rightarrow}$$

Semantika liste

Kreiranje liste

$$\Gamma \vdash \texttt{nil} \to \texttt{nil}$$

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \Gamma \vdash e_2 \to v_2}{\Gamma \vdash e1 :: e2 \to v_1 :: v_2}$$

Korištenje liste

$$\frac{}{\Gamma \vdash \texttt{match}\ e\ \texttt{with}\ \ \texttt{nil} \Rightarrow e_1,\ h :: t \Rightarrow e_2 \to}$$

Semantika liste

Kreiranje liste

$$\Gamma \vdash \texttt{nil} \to \texttt{nil}$$

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \Gamma \vdash e_2 \to v_2}{\Gamma \vdash e1 :: e2 \to v_1 :: v_2}$$

Korištenje liste

$$\frac{\Gamma \vdash e \to \texttt{nil} \qquad \Gamma \vdash e_1 \to v}{\Gamma \vdash \texttt{match}\ e\ \texttt{with}\ \ \texttt{nil} \Rightarrow e_1,\ h :: t \Rightarrow e_2 \to v}$$

Semantika liste

Kreiranje liste

$$\Gamma \vdash \texttt{nil} \to \texttt{nil}$$

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \Gamma \vdash e_2 \to v_2}{\Gamma \vdash e1 :: e2 \to v_1 :: v_2}$$

Korištenje liste

$$\frac{\Gamma \vdash e \to \texttt{nil} \qquad \Gamma \vdash e_1 \to v}{\Gamma \vdash \texttt{match}\ e\ \texttt{with}\ \ \texttt{nil} \Rightarrow e_1,\ h :: t \Rightarrow e_2 \to v}$$

$$\frac{\Gamma \vdash e \to v_1 :: v_2}{\Gamma \vdash \texttt{match}\ e\ \texttt{with}\ \ \texttt{nil} \Rightarrow e_1,\ h :: t \Rightarrow e_2 \to}$$

Semantika liste

Kreiranje liste

$$\Gamma \vdash \texttt{nil} \to \texttt{nil}$$

$$\frac{\Gamma \vdash e_1 \to v_1 \qquad \Gamma \vdash e_2 \to v_2}{\Gamma \vdash e1 :: e2 \to v_1 :: v_2}$$

Korištenje liste

$$\frac{\Gamma \vdash e \to \texttt{nil} \qquad \Gamma \vdash e_1 \to v}{\Gamma \vdash \texttt{match}\ e\ \texttt{with}\ \ \texttt{nil} \Rightarrow e_1,\ \ h :: t \Rightarrow e_2 \to v}$$

$$\frac{\Gamma \vdash e \to v_1 :: v_2 \qquad \Gamma, h : v_1, t : v_2 \vdash e_2 \to v}{\Gamma \vdash \texttt{match}\ e\ \texttt{with}\ \ \texttt{nil} \Rightarrow e_1,\ \ h :: t \Rightarrow e_2 \to v}$$

Sematika funkcija

Kreiranje fukcije

$$\frac{}{\Gamma \vdash \texttt{let}\ f\ x_1\ \cdots\ x_n = e_1\ \texttt{in}\ e_2 \rightarrow}$$

Korištenje funkcije

$$\frac{}{\Gamma \vdash e\ e_1\ \cdots\ e_n \rightarrow}$$

Sematika funkcija

Kreiranje fukcije

$$\Gamma, f : \langle \Gamma\ , \lambda\ x_1, \ldots, x_n.\ e_1 \rangle \vdash e_2 \to v$$

$$\Gamma \vdash \texttt{let}\ f\ x_1\ \cdots\ x_n = e_1\ \texttt{in}\ e_2 \to v$$

Korištenje funkcije

$$\Gamma \vdash e\ e_1\ \cdots\ e_n \to$$

Sematika funkcija

Kreiranje fukcije

$$\frac{\Gamma, f : \langle \Gamma', \lambda\ x_1, \ldots, x_n.\ e_1 \rangle \vdash e_2 \to v \quad \text{t.d. } \Gamma' = \Gamma \cap \mathit{freevars}(\lambda\ x_1, \ldots, x_n.\ e_1)}{\Gamma \vdash \texttt{let}\ f\ x_1\ \cdots\ x_n = e_1\ \texttt{in}\ e_2 \to v}$$

Korištenje funkcije

$$\frac{}{\Gamma \vdash e\ e_1\ \cdots\ e_n \to}$$

Sematika funkcija

Kreiranje fukcije

$$\frac{\begin{array}{c}\Gamma, f : \langle \Gamma', \lambda\ x_1, \ldots, x_n.\ e_1 \rangle \vdash e_2 \to v \\ \text{t.d. } \Gamma' = \Gamma \cap \mathit{freevars}(\lambda\ x_1, \ldots, x_n.\ e_1)\end{array}}{\Gamma \vdash \texttt{let}\ f\ x_1\ \cdots\ x_n = e_1\ \texttt{in}\ e_2 \to v}$$

Korištenje funkcije

$$\frac{\Gamma \vdash e \to \langle (\Gamma', \lambda\ x_1, \ldots, x_n.\ e') \rangle}{\Gamma \vdash e\ e_1\ \cdots\ e_n \to}$$

Sematika funkcija

Kreiranje fukcije

$$\frac{\begin{array}{c}\Gamma, f : \langle \Gamma', \lambda\ x_1, \ldots, x_n.\ e_1 \rangle \vdash e_2 \to v \\ \text{t.d. } \Gamma' = \Gamma \cap \mathit{freevars}(\lambda\ x_1, \ldots, x_n.\ e_1)\end{array}}{\Gamma \vdash \texttt{let}\ f\ x_1\ \cdots\ x_n = e_1\ \texttt{in}\ e_2 \to v}$$

Korištenje funkcije

$$\frac{\Gamma \vdash e \to \langle (\Gamma', \lambda\ x_1, \ldots, x_n.\ e') \rangle \qquad \Gamma \vdash e_1 \to v_1 \qquad \ldots \qquad \Gamma \vdash e_n \to v_n}{\Gamma \vdash e\ e_1\ \cdots\ e_n \to}$$

Sematika funkcija

Kreiranje fukcije

$$\frac{\begin{array}{c}\Gamma, f : \langle \Gamma', \lambda\ x_1, \ldots, x_n.\ e_1 \rangle \vdash e_2 \to v \\ \text{t.d. } \Gamma' = \Gamma \cap \mathit{freevars}(\lambda\ x_1, \ldots, x_n.\ e_1)\end{array}}{\Gamma \vdash \texttt{let}\ f\ x_1\ \cdots\ x_n = e_1\ \texttt{in}\ e_2 \to v}$$

Korištenje funkcije

$$\frac{\begin{array}{ccc}\Gamma \vdash e \to \langle (\Gamma', \lambda\ x_1, \ldots, x_n.\ e') \rangle & \Gamma \vdash e_1 \to v_1 & \ldots \\ \Gamma \vdash e_n \to v_n \qquad \Gamma', x_1 : v_1, \ldots, x_n : v_n & & \vdash e' \to v\end{array}}{\Gamma \vdash e\ e_1\ \cdots\ e_n \to v}$$

Sematika funkcija

Kreiranje fukcije

$$\frac{\begin{array}{c}\Gamma, f : \langle \Gamma', \lambda\ x_1, \ldots, x_n.\ e_1 \rangle \vdash e_2 \to v \\ \text{t.d. } \Gamma' = \Gamma \cap \mathit{freevars}(\lambda\ x_1, \ldots, x_n.\ e_1)\end{array}}{\Gamma \vdash \texttt{let}\ f\ x_1\ \cdots\ x_n = e_1\ \texttt{in}\ e_2 \to v}$$

Korištenje funkcije

$$\frac{\begin{array}{c}\Gamma \vdash e \to \langle (\Gamma', \lambda\ x_1, \ldots, x_n.\ e') \rangle \qquad \Gamma \vdash e_1 \to v_1 \qquad \ldots \\ \Gamma \vdash e_n \to v_n \qquad \Gamma', x_1 : v_1, \ldots, x_n : v_n, f : \langle \Gamma', \lambda\ x_1, \ldots, x_n\ e' \rangle \vdash e' \to v\end{array}}{\Gamma \vdash e\ e_1\ \cdots\ e_n \to v}$$