Típuselmélet

Kaposi Ambrus

Programozási Nyelvek és Fordítóprogramok Tanszék

ELTE IK PhD nyílt nap

2019. április 10.

Mi az, hogy program?

Mi az, hogy program?

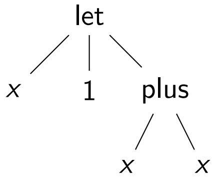
1. karakterlánc `let x := 1 in x + x`

Mi az, hogy program?

1. karakterlánc `let x := 1 in x + x`
2. lexikális elemek sorozata $[\mathsf{let}, x, :=, 1, \mathsf{in}, x, \mathsf{plus}, x]$

Mi az, hogy program?

1. karakterlánc — `let x := 1 in x + x`
2. lexikális elemek sorozata — $[\text{let}, x, :=, 1, \text{in}, x, \text{plus}, x]$
3. szintaxisfa

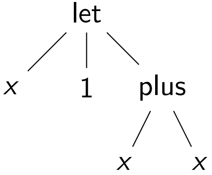


Mi az, hogy program?

1. karakterlánc `let x := 1 in x + x`

print

2. lexikális elemek sorozata $[\mathrm{let}, x, :=, 1, \mathrm{in}, x, \mathrm{plus}, x]$
3. szintaxisfa



Mi az, hogy program?

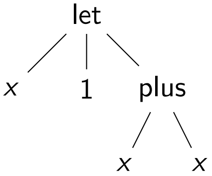
lexing

1. karakterlánc
2. lexikális elemek sorozata
3. szintaxisfa

print

```
let x := 1 in x + x
```

$[\text{let}, x, :=, 1, \text{in}, x, \text{plus}, x]$



Mi az, hogy program?

lexing

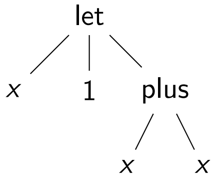
parsing

1. karakterlánc
2. lexikális elemek sorozata
3. szintaxisfa

print

```
let x := 1 in x + x
```

$[\mathrm{let}, x, :=, 1, \mathrm{in}, x, \mathrm{plus}, x]$

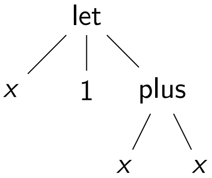


Mi az, hogy program?

1. karakterlánc
   - lexing ↓ / print ↑
2. lexikális elemek sorozata
   - parsing ↓ / print ↑
3. szintaxisfa

```
let x := 1 in (x + (x))
let x := 1 in x + x
```

$[\mathrm{let}, x, :=, 1, \mathrm{in}, x, \mathrm{plus}, x]$



Mi az, hogy program?

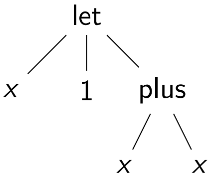
`let x := 1 in (x + (x))`

lexing

parsing

print

1. karakterlánc — `let x := 1 in x + x`
2. lexikális elemek sorozata — $[\mathsf{let}, x, :=, 1, \mathsf{in}, x, \mathsf{plus}, x]$
3. szintaxisfa



4. jól kötésezett szintaxisfa

$(\texttt{let x := 1 in x + x}) = (\texttt{let y := 1 in y + y})$

Mi az, hogy program?

`let x := 1 in (x + (x))`

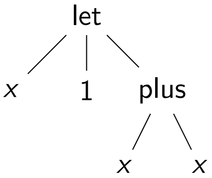
1. karakterlánc   `let x := 1 in x + x`

lexing

2. lexikális elemek sorozata $[\mathsf{let}, x, :=, 1, \mathsf{in}, x, \mathsf{plus}, x]$

parsing

3. szintaxisfa



scope checking

4. jól kötésezett szintaxisfa

print

$$(\texttt{let x := 1 in x + x}) = (\texttt{let y := 1 in y + y})$$

Mi az, hogy program?

`let x := 1 in (x + (x))`

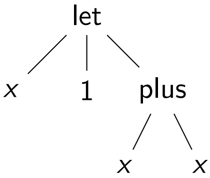
1. karakterlánc — `let x := 1 in x + x`

lexing

2. lexikális elemek sorozata — $[\text{let}, x, :=, 1, \text{in}, x, \text{plus}, x]$

parsing

3. szintaxisfa



scope checking

4. jól kötésezett szintaxisfa

print

típusellenőrzés

(`let x := 1 in x + x`) = (`let y := 1 in y + y`)

5. jól típusozott szintaxisfa

nincs olyan, hogy (`let x := 1 in x + true`)

Mi az, hogy program?

`let x := 1 in (x + (x))`

1. karakterlánc — `let x := 1 in x + x`

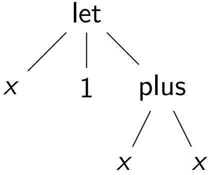
lexing

print

2. lexikális elemek sorozata — $[\text{let}, x, :=, 1, \text{in}, x, \text{plus}, x]$

parsing

3. szintaxisfa



scope checking

4. jól kötésezett szintaxisfa

típusellenőrzés

(`let x := 1 in x + x`) = (`let y := 1 in y + y`)

5. jól típusozott szintaxisfa

nincs olyan, hogy (`let x := 1 in x + true`)

6. algebrai szintaxis — (`let x := 1 in x + x`) = 2

Értelmetlen tételek

1. karakterlánc
2. lexikális elemek sorozata

   szóközelhagyás tétele

3. szintaxisfa

   a zárójelek és a szóközök elhagyásának sorrendje nem számít

4. jó kötésezett szintaxisfa

   a változóátnevezés megőrzi a zárójel-párokat

5. jól típusozott szintaxisfa

   a változóátnevezés megőrzi a típusokat

6. algebrai szintaxis

   tárgyredukció tétele

Egy algebrai struktúra

$$
\begin{aligned}
\mathsf{C} &: \mathsf{Set} \\
- \otimes - &: \mathsf{C} \to \mathsf{C} \to \mathsf{C} \\
\mathsf{u} &: \mathsf{C} \\
-^{-1} &: \mathsf{C} \to \mathsf{C} \\
\mathsf{ass} &: (a \otimes b) \otimes c = a \otimes (b \otimes c) \\
\mathsf{idl} &: \mathsf{u} \otimes a = a \\
\mathsf{idr} &: a \otimes \mathsf{u} = a \\
\mathsf{invl} &: a^{-1} \otimes a = \mathsf{u} \\
\mathsf{invl} &: a \otimes a^{-1} = \mathsf{u}
\end{aligned}
$$

Egy másik algebrai struktúra

$$
\begin{aligned}
\mathsf{Ty} &: \mathsf{Set} \\
\mathsf{Tm} &: \mathsf{Ty} \to \mathsf{Set} \\
\mathsf{Bool} &: \mathsf{Ty} \\
\mathsf{Nat} &: \mathsf{Ty} \\
\mathsf{true} &: \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{false} &: \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{if}{-}\mathsf{then}{-}\mathsf{else}{-} &: \mathsf{Tm}\,\mathsf{Bool} \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \to \mathsf{Tm}\,A \\
\mathsf{num} &: \mathbb{N} \to \mathsf{Tm}\,\mathsf{Nat} \\
\mathsf{isZero} &: \mathsf{Tm}\,\mathsf{Nat} \to \mathsf{Tm}\,\mathsf{Bool} \\
\mathsf{if}\beta_1 &: \mathsf{if}\,\mathsf{true}\,\mathsf{then}\,t\,\mathsf{else}\,t' = t \\
\mathsf{if}\beta_2 &: \mathsf{if}\,\mathsf{false}\,\mathsf{then}\,t\,\mathsf{else}\,t' = t' \\
\mathsf{isZero}\beta_1 &: \mathsf{isZero}\,(\mathsf{num}\,0) = \mathsf{true} \\
\mathsf{isZero}\beta_2 &: \mathsf{isZero}\,(\mathsf{num}\,(1+n)) = \mathsf{false}
\end{aligned}
$$

Típuselmélet

- Egyszerre programozási nyelv és számítógépes bizonyításellenőrző rendszer
- Leírható vele az algebrai szintaxis
- Feladatok: algebrai szintaxissal...
  - ... megadni egy interpretert
  - ... leírni egy fordítóprogramot: a helyessége triviális!
  - ... leírni az összes lehetséges algebrai szintaxist
  - ... leírni a típuselméletet
- Szeminárium minden hétfő este 6-kor
  - `http://bitbucket.org/akaposi/tipuselmelet`