Lambda Grammar

Ksenchik Nikita

May 28, 2023

1 Grammar

```
 < program > ::= let < variable > = < term > | < term > < < term > ::= < application > | < term' > < < term' > ::= < variable > | < abstraction > | "(" < term > ")" < < applique > ::= < term' > < term' > { < term' > } < < abstraction > ::= \ < variable > { < variable > }. < term > < < variable > ::= < keyword - exception > | < letter > { < letter > | < digit > } < < letter > ::= a | ... | z | A | ... | Z < digit > ::= 1 | ... | 9
```

2 Example 1

```
letK = \langle x \ y.x \implies
< program >
let < variable > = < term >
let < letter > \{ < letter > | < digit > \} = < term >
let K \{ \langle letter \rangle \mid \langle digit \rangle \} = \langle term \rangle
let K = < term >
let K = \langle term' \rangle
let K = \langle abstraction \rangle
let K = \langle variable \rangle \{\langle variable \rangle\}. \langle term \rangle
let \ K \ = \ \backslash < letter > \ \{ < variable > \}. < term >
let K = \langle x \{ \langle variable \rangle \}. \langle term \rangle
let K = \langle x < variable \rangle \{ \langle variable \rangle \}. \langle term \rangle
let K = \langle x < letter \rangle  { \langle letter \rangle  | \langle digit \rangle} {\langle variable \rangle}. \langle term \rangle
let K = \langle x | y | < letter > | < digit > \} \{ < variable > \}. < term > \}
let K = \langle x y \{ \langle variable \rangle \}. \langle term \rangle
let K = \langle x y. < term \rangle
let K = \langle x y. < term' \rangle
let K = \langle x y. \langle variable \rangle
let K = \langle x y. < letter \rangle \{ < letter \rangle | < digit \rangle \}
let K = \langle x y.x \{ \langle letter \rangle \mid \langle digit \rangle \}
let K = \langle x y.x \rangle
```

3 Example 2

```
let g < letter > \{ < letter > | < digit > \} = < term >
let gg \{ \langle letter \rangle \mid \langle digit \rangle \} = \langle term \rangle
let \ gg \ = < term >
let\ gg\ = < applique >
let gg = \langle term' \rangle \langle term' \rangle \{\langle term' \rangle\}
let \ gg \ = \ (< term >) \ < term' > \ \{< term' >\}
let gg = (\langle term' \rangle) \langle term' \rangle \{\langle term' \rangle\}
let gg = (\langle abstraction \rangle) \langle term' \rangle \{\langle term' \rangle\}
let \ qq = (\ < letter > \ | < digit > \} \{< variable > \}. < term >) < term' > \} 
term' > 
let gg = (\langle l = (letter > l = digit > ) = (letter > l = (letter > ) = 
let gg = (\langle l \{ \langle variable \rangle \}, \langle term \rangle) \langle term' \rangle \}
let gg = (\langle l < variable \rangle \{\langle variable \rangle\}, \langle term \rangle) \langle term' \rangle \{\langle term' \rangle\}
let \ gg = (\label{eq:continuous} \ \{ < letter > \ | \ < variable > \} \ \{ < variable > \}. < term >) \ < term' > \ \{ < variable > \}. < term >) \ < term' > \ \{ < variable > \}.
term' > 
let gg = (\lower black o \{ < letter > \ | < variable > \} \{ < variable > \}, < term >) < term' > \}
let gg = (\l o \{ \langle variable \rangle \}, \langle term \rangle) \langle term' \rangle \{ \langle term' \rangle \}
let gg = (\langle l o. \langle term \rangle) \langle term' \rangle \{\langle term' \rangle\}
let gg = (\langle l o. < term' >) < term' > \{ < term' > \}
let gg = (\langle lo. \langle variable \rangle) \langle term' \rangle \{\langle term' \rangle\}
let gg = (\langle lo. < letter > \{ < letter > | < digit > \}) < term' > \{ < term' > \}
let gg = (\langle lo.l \{\langle letter \rangle \mid \langle digit \rangle \}) \langle term' \rangle \{\langle term' \rangle \}
let gg = (\langle l o.l \rangle) < term' > \{ < term' > \}
let gg = (\label{eq:let_gg} (\label{eq:let_gg}) < variable > \{< term' > \}
let gg = (\lower let o.l) e \{ < letter > \ | \ < digit > \} \{ < term' > \}
let gg = (\langle l o.l \rangle) e \{\langle term' \rangle\}
let gg = (\langle l o.l \rangle) e < term' > \{ < term' > \}
let gg = (\lower let o.l) e < variable > \{ < term' > \}
let \ gg \ = \ (\label{eq:letter} \ \ \{ < letter \ | \ \ < digit > \} \ \{ < term' > \}
let gg = (\label{eq:gg} \ | \ \{ < letter \ | \ < digit > \} \ \{ < term' > \}
let gg = (\langle l o.l \rangle) e z \{\langle term' \rangle\}
let gg = (\label{eq:gg} let o.l) e z
```