## Lambda Grammar

#### Ksenchik Nikita

May 27, 2023

### 1 Grammar

```
 < program > ::= (let < variable > = < term >) \mid < term > < < term > ::= < variable > \mid < applique > \mid < abstraction > \mid (< term >) < abstraction > ::= \ < variable > \ < variable > \ < term > < < applique > ::= < term > < term > \ < term > \ < variable > ::= < letter > \ < letter > \ < digit > \ < digit > ::= < letter > \ < digit > ::= \ < digit \
```

### 2 Example 1

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

# 3 Example 2

```
 \begin{array}{lll} let \ gg \ = \ (\  \  \, lot) \ e \ z \implies \\ & < program > let \  \  \, < letter > \  \  \, \{ < letter > \  \  \, | \  \  \, < digit > \} \  \  \, = < term > \\ let \ g \  \  \, \{ < letter > \  \  \, | \  \  \, < digit > \} \  \  \, = < term > \\ let \ gg \  \  \, \{ < letter > \  \  \, | \  \  \, < digit > \} \  \  \, = < term > \\ let \ gg \  \  \, \{ < letter > \  \  \, | \  \  \, < digit > \} \  \  \, = < term > \\ let \ gg \  \  \  \, = \  \  \, < term > \\ let \ gg \  \  \  \  \, = \  \  \, < term > \} \\ \end{array}
```

```
let gg = (\langle term \rangle) \langle term \rangle \{\langle term \rangle\}
let gg = (\langle abstraction \rangle) \langle term \rangle \{\langle term \rangle\}
let gg = (\ < letter > \ | < digit >) \ < variable >). < term >) < term > \ < term >)
let gg = (\langle l \{\langle letter \rangle \mid \langle digit \rangle \} \{\langle variable \rangle \}, \langle term \rangle) \langle term \rangle \}
let gg = (\label{eq:gg} \{ \langle variable \rangle \}, \langle term \rangle) \langle term \rangle \}
let gg = (\label{eq:continuous} \{ < variable > \}, < term > \} < term > \}
term > 
let gg = (loo(setter > | set solution | set solut
let gg = (\label{eq:gg} \{ \langle variable \rangle \}, \langle term \rangle) \langle term \rangle \}
let gg = (\langle l o. \langle term \rangle) \langle term \rangle \{\langle term \rangle\}
let \ gg \ = \ (\label{eq:letter} \ \ \{ < letter > \ \ | \ \ < digit > \}) \ \ \ < term > \ \ \{ < term > \}
let gg = (\label{eq:gg} let gg = (\label{eq:gg} let o.l \{ < letter > | < digit > \}) < term > \{ < term > \}
let gg = (\langle l o.l \rangle) < term > \{\langle term \rangle\}
let gg = (\label{eq:let_gg} (\label{eq:let_gg}) < variable > \{ < term > \}
let gg = (\lower let o.l) < letter > \{ < letter > \lower | < digit > \} \{ < term > \}
let gg = (\label{eq:gg} let gg = (\label{eq:gg} let o.l) e \{ < letter > | < digit > \} \{ < term > \}
let gg = (\langle l o.l \rangle) e \{\langle term \rangle\}
let gg = (\label{eq:let} let gg = (\label{eq:let} let gg = (\label{eq:let} elet gg = (\label{eq:let} let gg = (\label{eq:let} elet gg = (\label{eq
let gg = (\label{eq:gg} (\label{eq:gg} let gg = (\label{eq:gg} (\label{eq:gg} let o.l) e < variable > \{< term >\}
let~gg~=~(\backslash l~o.l)~e~< letter >~ \{< letter ~|~ < digit >\}~ \{< term >\}
let gg = (\label{eq:letter} | et gg = (\label{eq:letter} | et git >) \{ < term > \}
let gg = (\label{eq:gg} let gg = (\label{eq:gg} let o.l) e z \{< term > \}
let gg = (\langle l o.l \rangle) e z
```