Ибрагимов Саид M33371

Вариант 8. Описание лямбда функции в Python

Описание лямбда функции в Python. Описание начинается ключевым словом “lambda”, далее идет множество аргументов через запятую, двоеточие, выражение. Используйте логические и/или арифметические операции.

Используйте один терминал для всех имен переменных. Используйте один терминал для ключевых слов lambda и т. п. (не несколько ‘l’, ‘a’, ‘m’ и т. д.).

Пример: lambda n : n + 2

1 Разработка грамматики

INPUT -> “lambda” VARS0 : LE

VARS0 -> VARS
VARS0 -> eps

VARS -> VARS , var
VARS -> var

LE -> LE | LT
LE -> LT

LT -> LT & E
LT -> E

E -> E + T
E -> T

T -> T * F
T -> F

F -> (LE)
F -> var
F -> num

INPUT – входное выражение
VARS0 – объявление переменных, допускающее их отсутствие
VARS – объявление переменных (кол-во > 0)
LE – дизъюнкция
LT – конъюнкция
E – сложение
T – умножение
F – LE в скобках, либо переменная, либо число

В грамматике есть левая рекурсия. Устраним ее. Правого ветвления нет. Получим новую грамматику:

INPUT -> “lambda” VARS0 : LE

VARS0 -> VARS
VARS0 -> eps

VARS -> var VARS’
VARS’ -> , var VARS’
VARS’ -> eps

LE -> LT LE’
LE’ -> | LT LE’
LE’ -> eps

LT -> E LT’
LT’ -> & E LT’
LT’ -> eps

E -> T E’
E’ -> + T E’
E’ -> eps

T -> F T’
T’ -> * F T’
T’ -> eps

F -> (LE)
F -> var
F -> num

INPUT – входное выражение
VARS0 – объявление переменных, допускающее их отсутствие
VARS – объявление переменных (кол-во > 0)
VARS’ – продолжение объявления переменных
LE – дизъюнкция
LE’ – продолжение дизъюнкции
LT – конъюнкция
LT’ – продолжение конъюнкции
E – сложение
E’ – продолжение сложения
T – умножение
T’ – продолжение умножения
F – LE в скобках, либо переменная, либо число

2 Построение лексического анализатора

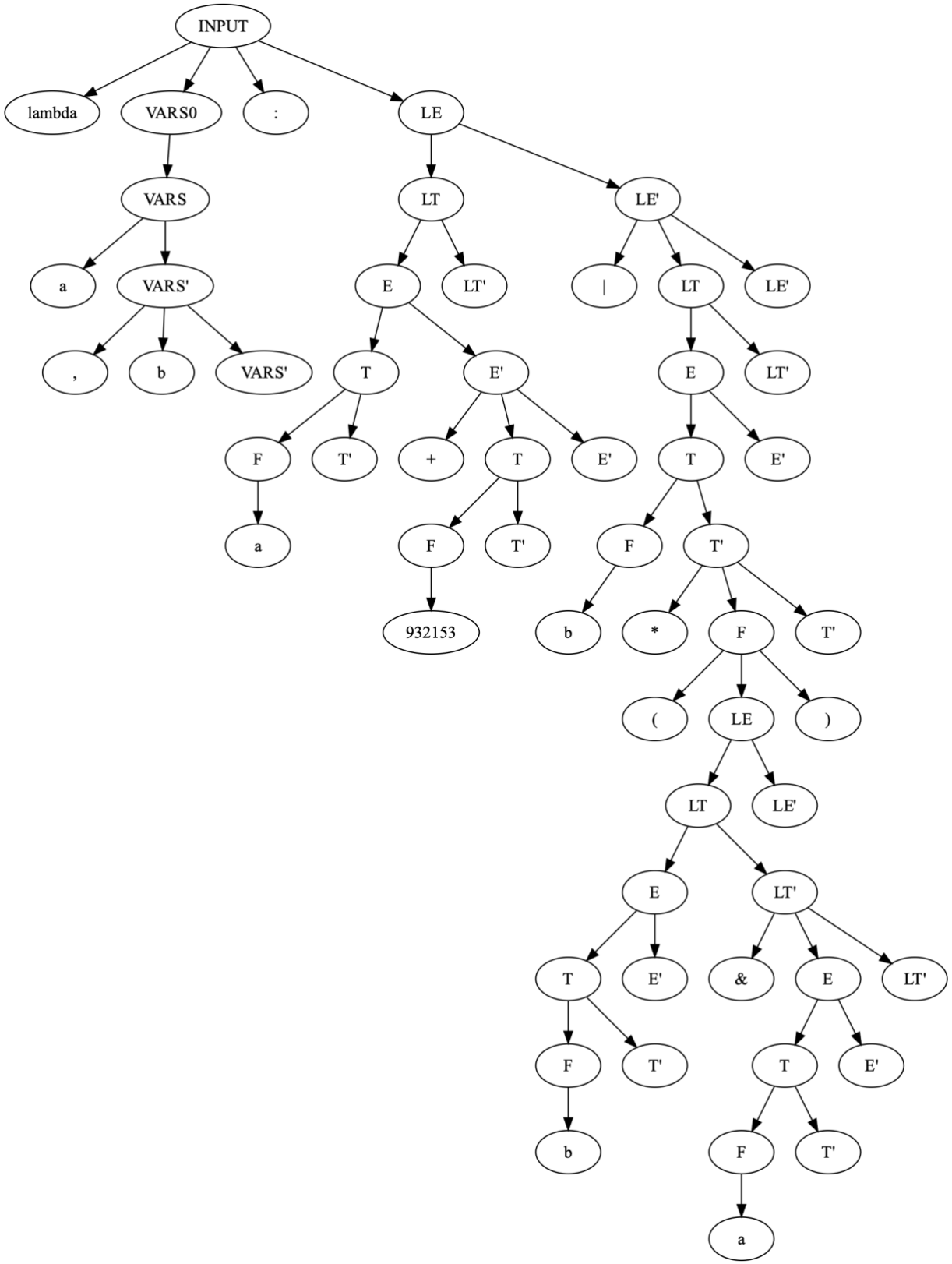
```
enum class Token(var value: String = "") {
    LAMBDA("lambda"),
    COLON(":"),
    COMMA(","),
    OR("|"),
    AND("&"),
    ADD("+"),
    MUL("*"),
    LPAREN("("),
    RPAREN(")"),
    VAR,
    NUM,

    END("$");

    override fun toString(): String = value
}
```

3 Построение синтаксического анализатора

НЕТЕРМИНАЛ	FIRST	FOLLOW
INPUT	“lambda”	$
VARS0	var	:
VARS	var eps	:
VARS’	, eps	:
LE	num var (	$)
LE’	| eps	$)
LT	num var (	$ |)
LT’	& eps	$ |)
E	num var (	$ | &)
E’	+ eps	$ | &)
T	num var (	$ | & +)
T’	* eps	$ | & +)
F	num var (	$ | & + *)

4 Визуализация дерева разбора

Lambda a, b : a + 932153 | b * (b & a)



5 Подготовка набора тестов

```
fun validPythonLambdas() = listOf(
    "lambda a : a" to "simple",
    "lambda : 22" to "without parameters",
    "lambda x, y, z : x + 1 + y + 2 + z" to "with multiple parameters",
    "lambda Z1ADc2f67sMb : 125125125" to "var naming",
    "lambda a, b : a | b" to "| operation",
    "lambda a, b : a & b" to "& operation",
    "lambda a, b : a * b" to "* operation",
    "lambda a, b : a + b" to "+ operation",
    "lambda a, b : (a + b) * 3" to "brackets",
    "lambda a, b : ((a + b) | 3) * 2" to "inner brackets",
    "lambda a, b : ((a + b)) * 3" to "repeated brackets",
    "lambda a1, B2, c1C2c : a1 + (B2 | 345) & c1C2c & 1100" to "final boss",
)
```

```
fun invalidPythonLambdas() = listOf(
    "lambda" to "\"lambda\" word",
    "lambda1 a : a" to "invalid lambda token",
    "lambda lambda : 1" to "var name cannot be \"lambda\"",
    "lambda a, b : a - b" to "invalid operator",
    "lambda a, b : a / b" to "invalid operator",
    "lambda a; b : a + b" to "invalid separator",
    "lambda a, b : # @ ! % № ^" to "# @ ! % № ^",
    "lambda a, b : \"hello\"" to "invalid return value",
    "lambda a :" to "without return",
    "lambda 1 : 1" to "number in parameters",
    "lambda 1a : 1" to "invalid var naming",
    "lambda a b : a" to "vars without comma",
    "lambda a, b ; a" to "without colon",
    "lambda a, b : a b" to "invalid body",
    "lambda a, b : + a b" to "invalid operators using (1)",
    "lambda a, b : a b |" to "invalid operators using (2)",
    "lambda a, b : a * & b" to "invalid operators using (3)",
    "lambda a, b : ()a + b" to "invalid brackets",
    "lambda a, b : )a + b(" to "invalid brackets",
    "lambda a, b : (a +) b" to "invalid brackets",
    "lambda a, b : a (+) b" to "invalid brackets",
    "lambda a, b : (a + b))" to "invalid brackets",
)
```