Грамматика для модельного языка Python

1 Введение

Сперва будет описана контекстно свободная грамматика, которая собственно задаёт синтаксис модельного языка Python, а затем к ней будет описан язык задания лексем языка с помощью конечного автомата с действиями.

2 Описание контекстно свободной грамматики

Грамматика состоит из следующих нетерминальных символов.

- <start> - стартовый символ грамматики
- <single_input> - на вход программы поступает файл
- <file_input> – программа запущена в интерактивном режиме
- <stmt> – оператор языка.
- <simple_stmt> – простой оператор, состоящий из малых операторов и перевода строки.
- <small_stmt> – состоит из оператора-выражения и других операторов.
- <expr_stmt> – оператор - выражение на языке.
- <pass_stmt> – оператор пропуска.
- <flow_stmt> – оператор управления потоком вычисления.
- <break_stmt> – оператор досрочного выхода из цикла.
- <continue_stmt> – оператор перехода на следующую итерацию цикла.
- <return_stmt> – оператор возврата из функции.
- <global_stmt> – оператор обьявляения переменной глобальной.

- <compound_stmt> – составной оператор.
- <funcdef> – описание функии.
- <parameters> – параметры функции.
- <argslist> – список аргументов.
- <if_stmt> – условный оператор.

- <while_stmt> – оператор цикла с предусловие.
- <for_stmt> – оператор цикла с итерированием.
- <suite> – блок кода, относяйщися к составным операторам.

- <testlist> – перечисление выражений.
- <exprlist> – перечисление арифметических выражений.
- <test> – единичное выражение .
- <or_test> – выражение с или.
- <and_test> – выражение с и.
- <not_test> – выражение с отрицанием.
- <comparison> – выражение сравнение.
- <comp_op> – перечисление лексем, участвующих в сравнении.
- <arith_expr> – арифметическое выражение.
- <sign> – знак.
- <term> – слагаемое.
- <mul_op> – операции с приоритетом у умножения.
- <factor> – множитель.
- <power> – степень.
- <atom> – атом, минимальная частица выражения, состоящая из кортежей, списков, чисел и так далее.
- <trailer> – постфикс после атома: (), [], . .
- <subscriptlist> – список возможных индексов.
- <subscript> – индекс или секция.
- <sliceop> – шаг секции.

Список список терминальных символов – лексем следующий.

- <SINGLE> – признак того, что программа запущена в интерактивном режиме.
- <FILE> – признак того, что программа запущена в режиме обработки файла
- <NEWLINE> – новая строка.
- <INDENT> – отступ.
- <DEDENT> – конец отступа.
- <NAME> – название переменной.

- <STRING> – строка в кавычках.
- <NUMBER> – число.
- <COM> – комментарий, начинающийся с #.
- <ENDMARKER> – признак конца (конец файла).
- Таблица TW – служебных слов языка.
- Таблица TD – таблица смиволов-разделителей.
- Таблица TID – таблица идентификаторов

Собственно сама контекстно свободная грамматика.

$$< start > \rightarrow < SINGLE > < single\_input > | < FILE > < file\_input >$$
$$< single\_input > \rightarrow < NEWLINE > | < simple\_stmt > | < compound\_stmt > < NEWLINE >$$
$$< file\_input > \rightarrow \{< NEWLINE > | < stmt >\} < ENDMARKER >$$

$$< stmt > \rightarrow < simple\_stmt > | < compound\_stmt >$$
$$< simple\_stmt > \rightarrow < small\_stmt > < NEWLINE >$$
$$< small\_stmt > \rightarrow < expr\_stmt > | < pass\_stmt > | < flow\_stmt > | < global\_stmt >$$
$$< expr\_stmt > \rightarrow < test > ['=' (< test >)]$$

$$< pass\_stmt > \rightarrow 'pass'$$
$$< flow\_stmt > \rightarrow < break\_stmt > | < continue\_stmt > | < return\_stmt >$$
$$< break\_stmt > \rightarrow 'break'$$
$$< continue\_stmt > \rightarrow 'continue'$$
$$< return\_stmt > \rightarrow 'return' [< testlist >]$$
$$< global\_stmt > \rightarrow 'global' < NAME > \{',' < NAME >\}$$
$$< compound\_stmt > \rightarrow < if\_stmt > | < while\_stmt > | < for\_stmt > | < funcdef >$$
$$< funcdef > \rightarrow 'def' < NAME > < parameters > ':' < suite >$$
$$< parameters > \rightarrow '(' [< argslist >] ')'$$
$$< argslist > \rightarrow < NAME > \{',' < NAME >\}$$
$$< if\_stmt > \rightarrow 'if' < test > ':' < suite > ['else' ':' < suite >]$$
$$< while\_stmt > \rightarrow 'while' < test > ':' < suite >$$
$$< for\_stmt > \rightarrow 'for' < exprlist > 'in' < testlist > ':' < suite >$$
$$< suite > \rightarrow < simple\_stmt > | < NEWLINE > < INDENT > < stmt > \{< stmt >\} < DEDENT >$$

$$< testlist > \rightarrow < test > \{',' < test >\}$$
$$< exprlist > \rightarrow < arith\_expr > \{',' < arith\_expr >\}$$
$$< test > \rightarrow < or\_test >$$
$$< or\_test > \rightarrow < and\_test > \{'or' < and\_test >\}$$
$$< and\_test > \rightarrow < not\_test > \{'and' < not\_test >\}$$
$$< not\_test > \rightarrow 'not' < not\_test > | < comparison >$$
$$< comparison > \rightarrow < arith\_expr > \{< comp\_op > < arith\_expr >\}$$
$$< comp\_op > \rightarrow '<' | '>' | '==' | '>=' | '<=' | '!=' | 'in' | 'not' 'in'$$
$$< arith\_expr > \rightarrow < term > \{< sign > < term >\}$$
$$< sign > \rightarrow '+' | '-'$$
$$< term > \rightarrow < factor > \{< mul\_op > < factor >\}$$
$$< mul\_op > \rightarrow '*' | '/' | '\%' | '//'$$
$$< factor > \rightarrow [< sign >] < factor > | < power >$$
$$< power > \rightarrow < atom > < trailer > ['**' < factor >]$$
$$< atom > \rightarrow '(' [< testlist >] ')' | '[' [< testlist >] ']' | < NAME > | < NUMBER > | < STRING > | 'None' | 'True' | 'False'$$

$$< trailer > \rightarrow '(' [< testlist >] ')' | '[' < subscriptlist > ']' | '.' < NAME >$$
$$< subscriptlist > \rightarrow < subscript > ',' < subscript >$$
$$< subscript > \rightarrow < test > | [< test >] ':' [< test >] [< sliceop >]$$
$$< sliceop > \rightarrow ':' [< test >]$$

3 Лексический разбор

Для лексического разбора опишем диаграмму состояний лексического анализатора модельного языка A также опишем ранее указанные таблицы TD и TW

Таблица 1: Таблица разделителей – TD

+	-	*	**	/	//	%
<	>	<=	>=	==	!=	(
)	[	]	,	:	.	=

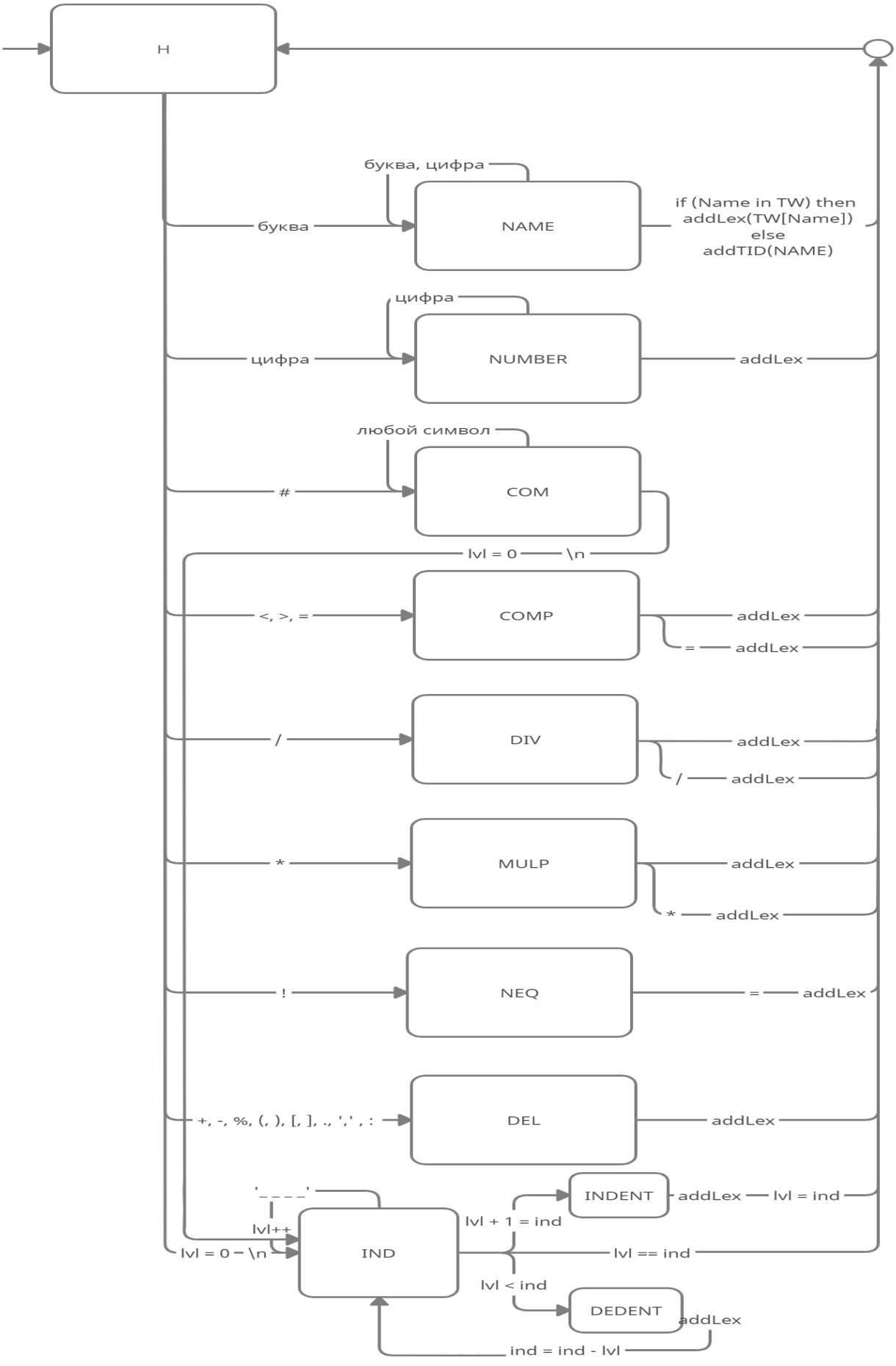
Таблица 2: Таблица служебных слов - TW

False	return	None	continue	for	not
True	def	while	and	global	in
if	or	else	pass	break	

Таблица TID является таблицей с идентфикаторами, которые встретились в программе, она заполняется во время работы анализатора.

На следующей странице собственно представлена таблица состояний. Обозначение '_' обозначает пробельный символ, '_ _ _ _' означает 4 пробела. Переход осуществляется по соответствующим символам дугам. Если по текущему символу нельзя перейти ни по одной дуге, то выводится ошибка. Переменная ind изначально инициализирована 0

Состояние <SINGLE> или <FILE> не указаны в автомате, одно из них выдается в начале лексического разбора по признаку того, как запщуена программа


H
буква, цифра
буква
NAME
if (Name in TW) then
addLex(TW[Name])
else
addTID(NAME)
цифра
цифра
NUMBER
addLex
любой символ
#
COM
lvl = 0
\n
<, >, =
COMP
addLex
=
addLex
/
DIV
addLex
/
addLex
*
MULP
addLex
*
addLex
!
NEQ
=
addLex
+, -, %, (,), [,], ., ',' , :
DEL
addLex
'_ _ _ _'
INDENT
addLex
lvl = ind
lvl++
lvl = 0
\n
IND
lvl + 1 = ind
lvl == ind
lvl < ind
DEDENT
addLex
ind = ind - lvl