Под правилом после // записаны условия для атрибутов для данного правила. Для описания атрибутов используем синтаксис python3.

Функция update для обновления nonEmpty_table:

```
def update_nonEmpty(table):
  changed = True
  while changed:
    changed = False
    empty_nterms = [nterm for nterm, v in table.items() if not v[“nonEmpty”]]
    nonEmpty_nterms = [nterm for nterm, v in table.items() if v[“nonEmpty”]]
    for empty in empty_nterms:
      for nterm in table[empty][“nterms”]:
        if nterm in table and table[nterm][“nonEmpty”]:
          changed = True
          table[empty][“nonEmpty”] |= table[nterm][“nonEmpty”]
```

Функция update нужна для случая с такими грамматиками:
[S] → [A];
[A] → [B];
[B] → c;

Функция для вычисления reachable_table:

```
def create_reachable(left_nterms, right_nterms):
  table = {i: False for i in left_nterms}
  for nterm in right_nterms:
    table[nterm] = True
```

<start> ::= <grammar>	# nonEmpty
	grammar.nonEmpty_table = {}

<grammar> ::= <rule> <rule-sep> <grammar-tail>	# reachable
	grammar.left_nterms = grammar-tail.left_nterms | set([rule.left_nterm])
	grammar.right_nterms = grammar-tail.right_nterms | rule.right_nterms
	grammar.reachable_table = create_reachable(grammar.left_nterms, grammar.right_nterms)
	# nonEmpty
	rule.nonEmpty_table = grammar.nonEmpty_table
	grammar-tail.nonEmpty_table = grammar.nonEmpty_table

<grammar-tail> ::= <rule> <rule-sep> <grammar-tail>	# reachable
	grammar-tail$_0$.left_nterms = grammar-tail$_1$.left_nterms | set([rule.left_nterm])
	grammar-tail$_0$.right_nterms = grammar-tail$_1$.right_nterms | rule.right_nterms
	# nonEmpty
	rule.nonEmpty_table = grammar-tail$_0$.nonEmpty_table
	grammar-tail$_1$.nonEmpty_table = grammar-tail$_0$.nonEmpty_table

<grammar-tail> ::=	grammar-tail.left_nterms = set() grammar-tail.right_nterms = set() update_nonEmpty(grammar-tail.nonEmpty_table)

<rule> ::= <nonterm> <arrow> <rule-right>	# reachable
	rule.left_nterm = nonterm.value
	rule.right_nterms = rule-right.nterms
	# nonEmpty
	rule.nonEmpty_table[nonterm.value] = {“nterms”: rule-right.nterms,
	“nonEmpty”: rule-right.nonEmpty }

<rule-right> ::= <production> <rule-tail>	rule-right.nonEmpty = production.nonEmpty or rule-tail.nonEmpty rule-right.nterms = production.nterms | rule-tail.nterms
<rule-tail> ::= <production-sep> <production> <rule-tail>	$rule\text{-}tail_0$.nonEmpty = production.nonEmpty or $rule\text{-}tail_1$.nonEmpty $rule\text{-}tail_0$.nterms = production.nterms | $rule\text{-}tail_1$.nterms
<rule-tail> ::=	rule-tail.nonEmpty = False rule-tail.nterms = set()
<production> ::= <epsilon>	production.nonEmpty = False production.nterms = set()
<production> ::= <term> <production-tail>	production.nonEmpty = True production.nterms = production-tail.nterms
<production> ::= <nonterm> <production-tail>	production.nonEmpty = production-tail.nonEmpty production.nterms = production-tail.nterms | set([nonterm.value])
<production-tail> ::= <term> <production-tail>	$production\text{-}tail_0$.nonEmpty = True $production\text{-}tail_0$.nterms = $production\text{-}tail_1$.nterms
<production-tail> ::= <nonterm> <production-tail>	$production\text{-}tail_0$.nonEmpty = $production\text{-}tail_1$.nonEmpty $production\text{-}tail_0$.nterms = $production\text{-}tail_1$.nterms | set([nonterm.value])
<production-tail> ::=	production-tail.nonEmpty = False production-tail.nterms = set()
<term> ::= <term-regex>	
<nonterm> ::= <nterm-start> <nterm-regex> <nterm-end>	nonterm.value = str(nterm-regex)

// токены-параметры
<rule-sep> ::= ;
<arrow> ::= →
<production-sep> ::= |
<nterm-start> ::= [
<nterm-end> ::=]
<term-regex> ::= (a-zA-Z0-9)+
<nterm-regex> ::= (a-zA-Z0-9)+

Пример грамматики в этом синтаксисе:
[A] → b[A] | b[C]a;
[C] → c[C] | ε;