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

Функция update для обновления generating_table:

```
def update_generating(table):
    changed = True
    while changed:
        changed = False
        nongenerating_nterms = [nterm for nterm, v in table.items() if not v[“is_generating”]]
        generating_nterms = [nterm for nterm, v in table.items() if v[“is_generating”]]
        for nongenerating in nongenerating_nterms:
            for nterm in table[nongenerating][“nterms”]:
                if nterm in table and table[nterm][“is_generating”]:
                    changed = True
                    table[nongenerating][“is_generating”] |= table[nterm][“is_generating”]
```

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>	# generating
	grammar.generating_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)
	# generating
	rule.generating_table = grammar.generating_table
	grammar-tail.generating_table = grammar.generating_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
	# generating
	rule.generating_table = grammar-tail$_0$.generating_table
	grammar-tail$_1$.generating_table = grammar-tail$_0$.generating_table

<grammar-tail> ::=	grammar-tail.left_nterms = set() grammar-tail.right_nterms = set() update_generating(grammar-tail.generating_table)

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

<rule-right> ::= <production> <rule-tail>	rule-right.is_generating = production.is_generating or rule-tail.is_generating rule-right.nterms = production.nterms | rule-tail.nterms
<rule-tail> ::= <production-sep> <production> <rule-tail>	rule-tail_0.is_generating = production.is_generating or rule-tail_1.is_generating rule-tail_0.nterms = production.nterms | rule-tail_1.nterms
<rule-tail> ::=	rule-tail.is_generating = False rule-tail.nterms = set()
<production> ::= <epsilon>	production.is_generating = False production.nterms = set()
<production> ::= <term> <production-tail>	production.is_generating = True production.nterms = production-tail.nterms
<production> ::= <nonterm> <production-tail>	production.is_generating = production-tail.is_generating production.nterms = production-tail.nterms | set([nonterm.value])
<production-tail> ::= <term> <production-tail>	production-tail_0.is_generating = True production-tail_0.nterms = production-tail_1.nterms
<production-tail> ::= <nonterm> <production-tail>	production-tail_0.is_generating = production-tail_1.is_generating production-tail_0.nterms = production-tail_1.nterms | set([nonterm.value])
<production-tail> ::=	production-tail.is_generating = 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;