

# Parser

## Goian Tudor – 933/2

<https://github.com/VaruTudor/Formal-Languages-and-Compiler-Design>

The Grammar has 5 fields:

N – non-terminals

E – terminals

P – productions

S – start symbol

isCFG – flag which checks if the grammar is context-free

For the production we use a dictionary, where for each key (symbol) there is a list of tuples (symbol, destination). Most methods are for file parsing except checkCFG(rules, N) which receives the rules and the set of non-terminals and getProductionsFor(nonTerminal) which receives a non-terminal symbol.

Below there are some test examples.

```
N = { S, A, B, C, D }
E = { (, ), +, *, int }
S = S
P = {
    S -> A B C,
    A -> ( S ),
    B -> * S | E,
    C -> + A | C
}
```

```
N = { S, A, B, C, D }
E = { (, ), +, *, int }
productions
[[('A B C', 1)], [(' ( S )', 2)], [(' * S', 3), (' E', 4)], [(' + A', 5), (' C', 6)], None]
productions for A
[('( S )', 2)]
CFG
```

```

closure(itemList):
    For each item in the item list, if after dot there is a non-terminal, add
    it to the list of items and repeat the process.
    pseudocode:
        repeat
            for any  $[A \rightarrow \alpha.B\beta]$  in  $C$  do
                for any  $B \rightarrow \gamma$  in  $P$  do
                    if  $[B \rightarrow \gamma] \notin C$  then
                         $C = C \cup [B \rightarrow \gamma]$ 
                    end if
                end for
            end for
        until  $C$  stops changing
    input:
        itemList: a list of LR(0) items
    :return:
        the resulting state

goto(state, symbol):
    For each LR0 item in the state move the dot if the symbol follows it.
    Perform closure on the modified item.
    pseudocode:
        goto( $s, X$ ) = closure( $\{[A \rightarrow \alpha X \beta] \mid [A \rightarrow \alpha X \beta] \in s\}$ )
    input:
        state: the state to be checked
        symbol: the symbol to be checked
    output:
        closure of the updated items

computeCanonicalCollection():
    For each state  $s$  in the canonical collection, for each symbol  $X$  (in both
    terminals and non-terminals), check if goto( $s, X$ ) result is not an empty list
    nor exists already in the canonical collection and if so add it to the
    canonical collection
    pseudocode:
        repeat
            for any  $s$  in  $C$  do
                for any  $X$  in  $N \cup \Sigma$  do
                    if goto( $s, X$ )  $\neq \emptyset$  and goto( $s, X$ ) not in  $C$  then
                         $C = C \cup \text{goto}(s, X)$ 
                    end if
                end for
            end for
        until  $C$  stops changing

computeTableActions():
    For each state in the canonical collection add in the LR0 Table it's
    appropriate action

```

Bellow there is a test example computing the Canonical Collection for a grammar.

1)  $S' \rightarrow S$   
 $S \rightarrow aA$   
 $A \rightarrow bA \mid c$

Canonical collection:

$$s_0 = \text{closure}(\{[S' \rightarrow \cdot S]\}) = \{[S' \rightarrow \cdot S], [S \rightarrow \cdot aA]\}$$

$$s_1 = \text{goto}(s_0, S) = \text{closure}(\{[S' \rightarrow S \cdot]\}) = \{[S' \rightarrow S \cdot]\}$$

$$s_2 = \text{goto}(s_0, a) = \text{closure}(\{[S \rightarrow a \cdot A]\}) = \{[S \rightarrow a \cdot A], [A \rightarrow \cdot bA], [A \rightarrow \cdot c]\}$$

$$\text{goto}(s_1, X) = \emptyset, \quad \forall X \in N \cup \Sigma$$

$$s_3 = \text{goto}(s_2, A) = \text{closure}(\{[S \rightarrow aA \cdot]\}) = \{[S \rightarrow aA \cdot]\}$$

$$s_4 = \text{goto}(s_2, b) = \text{closure}(\{[A \rightarrow b \cdot A]\}) = \{[A \rightarrow b \cdot A], [A \rightarrow \cdot bA], [A \rightarrow \cdot c]\}$$

$$\text{goto}(s_3, X) = \emptyset, \quad \forall X \in N \cup \Sigma$$

$$s_5 = \text{goto}(s_4, A) = \text{closure}(\{[A \rightarrow bA \cdot]\}) = \{[A \rightarrow bA \cdot]\}$$

$$s_6 = \text{goto}(s_4, c) = \text{closure}(\{[A \rightarrow c \cdot]\}) = \{[A \rightarrow c \cdot]\}$$

$$\text{goto}(s_5, c) = \emptyset$$

Non-terminals and Terminals:

$N = \{ S, A \}$

$E = \{ a, b, c \}$

Productions:

$\{ 'S': [( 'aA', 1)], 'A': [( 'bA', 2), ( 'c', 3)] \}$

Productions for A:

$[( 'bA', 2), ( 'c', 3)]$

The grammar is CFG

Canonical Collection:

$S_0( [S' \rightarrow \cdot S, S \rightarrow \cdot aA] )$

$S_1( [S' \rightarrow S \cdot] )$

$S_2( [S \rightarrow a \cdot A, A \rightarrow \cdot bA, A \rightarrow \cdot c] )$

$S_3( [S \rightarrow aA \cdot] )$

$S_4( [A \rightarrow b \cdot A, A \rightarrow \cdot bA, A \rightarrow \cdot c] )$

$S_5( [A \rightarrow c \cdot] )$

$S_6( [A \rightarrow bA \cdot] )$