GENERAL PARSING: A NEW APPROACH

Roman LUKÁŠ, Master Degree Programme (5)
Dept. of Information Systems, FIT, BUT
E-mail: xlukas03@stud.fit.vutbr.cz

Supervised by Dr. Alexander Meduna

ABSTRACT

This paper presents a new approach to general parsing of context-free languages. This approach represents a significant simplification compared to most other parsing methods because it is based on regular expressions rather than pushdown automata. In addition, it works in a parallel and deterministic way.

1 DEFINITION OF REGULAR EXPRESSIONS

Let M be a finite set of symbols. We define a regular expression over M recursively in the following manner:

- $\varnothing$ is a regular expression over M and denoting regular language $L = \{\}$
- ε is a regular expression over M and denoting regular language $L = \{\varepsilon\}$
- a, where $a \in M$, is a regular expression over M and denoting regular language $L = \{a\}$
- If p, q are regular expressions over M and denoting regular languages L_p, L_q, then
  - $[p \mid q]$ is a regular expression over M and denoting regular language $L = L_p \cup L_q$,
  - pq is a regular expression over M and denoting regular language $L = L_p.L_q$,
  - $\{p\}$ is a regular expression over M and denoting regular language $L = L_p^*$

Note: For a set M, $R(M)$ denotes the set of all regular expression over M.

2 RESULTS

2.1 MAPPING α

Let $G = (N, T, P, S)$ be a context-free grammar without any ε-rules, $C = \{<i>: i \in 1..\mathrm{card}(P)\}$. For every grammar G we define the mapping α from $T \times N$ to $R(N \cup T \cup C)$, which is constructed using algorithm 1.

2.2 ALGORITHM 1

Input: $G = (N, T, P, S)$ without ε-rules
Output: mapping α for grammar G

Note: β is any mapping from $(N \cup T) \times N$ to $R(N \cup T \cup C)$.

Method:
1) for $\forall V \in N \cup T$, **for** $\forall A \in N$ **do**
    $\beta(V, A) := \varnothing$
2) for $\forall (A \to V\chi) \in P$, where $A \in N$; $V \in N \cup T$; $\chi \in (N \cup T)^*$ **do**
    if $A \to V\chi$ is i-th rule in G **then** $\beta(V, A) := [\beta(V, A) \mid \chi<i>]$
3) for $\forall A \in N$ **do begin**
    for $\forall V \in (N \cup T) \setminus \{A\}$ **do**
        $\beta(V, A) := \beta(V, A)\{\beta(A, A)\}$
    $\beta(A, A) := \varnothing$
    for $\forall B \in N \setminus \{A\}$ **do begin**
        for $\forall V \in (N \cup T) \setminus \{A\}$ **do**
            $\beta(V, B) := [\beta(V, B) \mid \beta(V, A)\beta(A, B)]$
        $\beta(A, B) := \varnothing$
    end
end

4) for $\forall A \in N$, **for** $\forall a \in T$ **do**

    $\alpha(a, A) := \beta(a, A)$

2.3 SYNTACTIC ANALYSIS

Let $G = (N, T, P, A_1)$ be a context-free grammar without any ε-rules, where $N = \{A_1, \dots, A_n\}$, $C = \{<i>: i \in 1..\text{card}(P)\}$, $Q = N \cup T \cup C \cup \{\#\}$.

Let Θ denote the set of k-tuple of the form $(v\#, R_{A_1}, R_{A_2}, \dots, R_{A_n}, R_t)$, where $v \in T^*$; $R_{A_1}, R_{A_2}, \dots, R_{A_n}, R_t \in R(Q)$. Each member of Θ is called a configuration.

Let ${}_w\Sigma$ be a configuration of the form $(w\#, \#, \varnothing, \dots, \varnothing, \varnothing)$, where $w \in T^*$ is an input string. Then, ${}_w\Sigma$ is called a starting configuration for w.

Let ${}_w\Phi_1 \subseteq \Theta$ and ${}_w\Phi_1 = \{(\#, \varnothing, \varnothing, \dots, \varnothing, R_{parse}): R_{parse} \in R(C), R_{parse} \neq \varnothing\}$. Then, each member of ${}_w\Phi_1$ is called a final accepting configuration for w and for every $x \in L(R_{parse})$ holds: reverse(x) is the right parse of string w.

Let ${}_w\Phi_0 \subseteq \Theta$ and ${}_w\Phi_0 = \{(x\#, \varnothing, \varnothing, \dots, \varnothing, \varnothing): x \in \mathit{suffix}(w)\}$. Then, each member of ${}_w\Phi_0$ is called a final unaccepting configuration for w.

Let $\vdash$ be a mapping from Θ to Θ defined as:

Let $\chi, \delta \in \Theta$, $\chi = (abv, R_{A_1}, R_{A_2}, \dots, R_{A_n}, R_t)$, $\delta = (bv, R'_{A_1}, R'_{A_2}, \dots, R'_{A_n}, R'_t)$, where $a \in T$; $b \in T \cup \{\#\}$; $v \in T^*\# \cup \{\varepsilon\}$; $R_{A_1}, R_{A_2}, \dots, R_{A_n}, R_t, R'_{A_1}, R'_{A_2}, \dots, R'_{A_n}, R'_t \in R(Q)$, and regular expressions $R'_{A_1}, R'_{A_2}, \dots, R'_{A_n}, R'_t$ are computed as:

1) **Perform in parallel:** $R'_t := \varnothing$, **for** $\forall i = 1..n$: $R'_{Ai} := \varnothing$

2) **Perform in parallel:** Parse(R_t), **for** $\forall i = 1..n$: Parse($\alpha(a, A_i)\, R_{A_i}$)

procedure Parse($R : R(Q)$)
begin
  case R is in form:
  $\varnothing$: **do** nothing
  a_xS; $a_x \in T \cup \{\#\}$; $S \in R(Q)$: **if** $a_x = b$ **then** $R'_t := [\, R'_t \mid S]$
  A_xS; $A_x \in N$; $S \in R(Q)$: **if** $b \neq \#$ **then if** $\alpha(b, A_x) \neq \varnothing$ **then** $R'_{A_x} := [\, R'_{A_x} \mid S]$
  cS; $c \in C$; $S \in R(Q)$: Parse(Sc)
  $[S_1 \mid S_2]S$; $S_1, S_2, S \in R(Q)$: **Perform in parallel:** Parse(S_1S), Parse(S_2S)
  $\{S_1\}S$; $S_1, S \in R(Q)$: **Perform in parallel:** Parse(S), Parse($S_1\{S_1\}S$)
  end case
end

Then, we write: $\chi \vdash \delta$. $\vdash^*$ denotes transitive and reflexive closure of $\vdash$.

Claim: Let $G = (N, T, P, A_1)$ be a context-free grammar without any ε-rules. For every $w \in T^*$ holds: $w \in L(G) \Leftrightarrow ({}_w\Sigma \vdash^* F \wedge F \in {}_w\Phi_1)$ and $w \notin L(G) \Leftrightarrow ({}_w\Sigma \vdash^* F \wedge F \in {}_w\Phi_0)$.

3 EXAMPLE

Consider the context-free grammar $G_v = (N, T, P, E)$, where: $N = \{E, T, F\}$, $T = \{i, +, *, (,)\}$, $P = \{(1)\ E \to E+T,\ (2)\ E \to T,\ (3)\ T \to T{*}F,\ (4)\ T \to F,\ (5)\ F \to (E),\ (6)\ F \to i\}$.

Task: $i{+}i \in L(G_v)$?

3.1 MAPPING α FOR A GRAMMAR G_v

The following tables present a computation of final mapping α for G_v:

	E	**T**	**F**	**i**	**(**	**)**	+	*
E	+T<1>	<2>	∅	∅	∅	∅	∅	∅
T	∅	*F<3>	<4>	∅	∅	∅	∅	∅
F	∅	∅	∅	<6>	E)<5>	∅	∅	∅

Tab. 1: *Mapping β after computation the 2-nd part of algorithm 1*

	E	**T**	**F**	**i**	**(**	**)**	+	*
E	∅	<2>{+T<1>}	∅	∅	∅	∅	∅	∅
T	∅	*F<3>	<4>	∅	∅	∅	∅	∅
F	∅	∅	∅	<6>	E)<5>	∅	∅	∅

Tab. 2: *Mapping β after computation the 3-rd part of algorithm for A = E*

	E	***T***	***F***	***i***	**(**	**)**	**+**	*****
E	∅	∅	<4>{*F<3>}<2>{+T<1>}	∅	∅	∅	∅	∅
T	∅	∅	<4>{*F<3>}	∅	∅	∅	∅	∅
F	∅	∅	∅	<6>	E)<5>	∅	∅	∅

Tab. 3: *Mapping β after computation 3-rd part of algorithm for A = T*

	E	***T***	***F***	***i***	**(**	**)**	**+**	*****
E	∅	∅	∅	<6><4>{*F<3>}<2>{+T<1>}	E)<5><4>{*F<3>}<2>{+T<1>}	∅	∅	∅
T	∅	∅	∅	<6><4>{*F<3>}	E)<5><4>{*F<3>}	∅	∅	∅
F	∅	∅	∅	<6>	E)<5>	∅	∅	∅

Tab. 4: *Mapping β after computation 3-rd part of algorithm for A = F*

	i	**(**	**)**	**+**	*****
E	<6><4>{*F<3>}<2>{+T<1>}	E)<5><4>{*F<3>}<2>{+T<1>}	∅	∅	∅
T	<6><4>{*F<3>}	E)<5><4>{*F<3>}	∅	∅	∅
F	<6>	E)<5>	∅	∅	∅

Tab. 5: *Final mapping α for a grammar G_v*

3.2 SYNTACTIC ANALYSIS FOR STRING *i+i*

Let us parse *i+i*. The computation of next configuration we can see on the figures Fig. 1 - Fig. 3.

- Configuration C_1 = (*i+i#*, #, ∅, ∅, ∅); *a* = *i*, *b* = +, *v* = *i#*

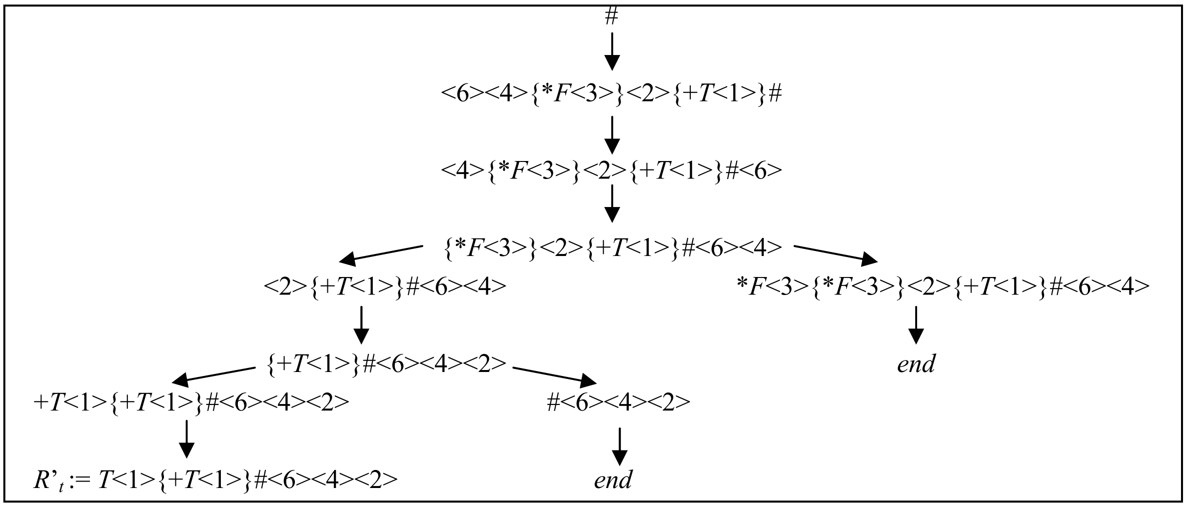

Fig. 1: *Computation of configuration C_2*

- Configuration C_2 = (+*i#*, ∅, ∅, ∅, *T*<1>{+*T*<1>}#<6><4><2>); *a* = +, *b* = *i*, *v* = #

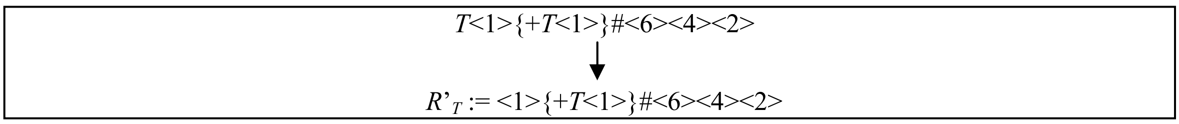

Fig. 2: *Computation of configuration C_3*

- Configuration C_3 = (*i*#, ∅, <1>{+*T*<1>}#<6><4><2>, ∅, ∅); $a = i$, $b = \#$, $v = \varepsilon$

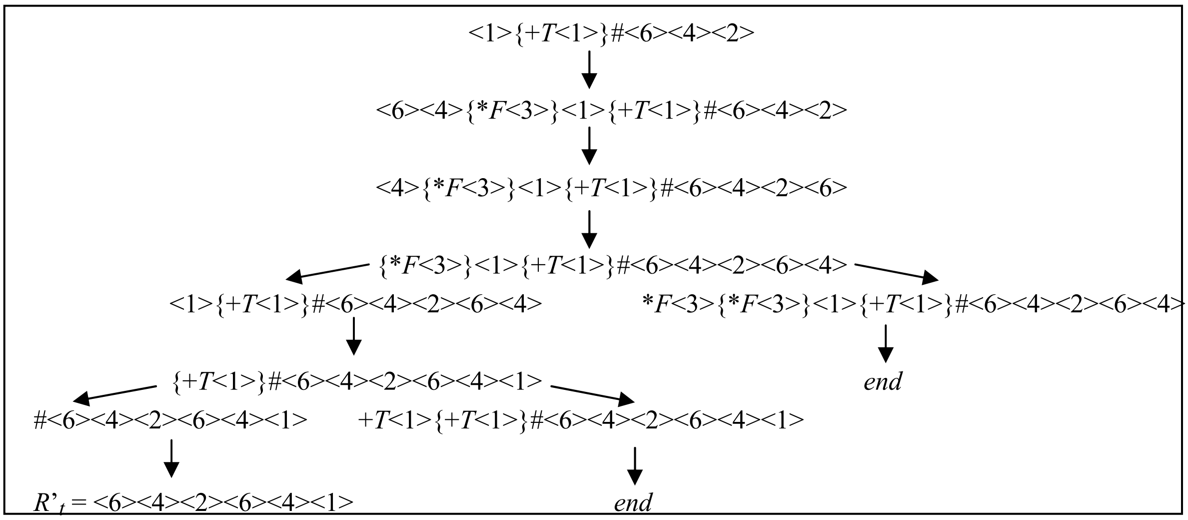

Fig. 3: *Computation of configuration* C_4

- Configuration C_4 = (#, ∅, ∅, ∅, <6><4><2><6><4><1>)

A configuration C_4 is final configuration. String *i*+*i* is generated in G_v. 146246 is the right parse.

REFERENCES

[1] Tremblay, J. P. and Sorenson, P. G.: The Theory and Practice of Compiler Writing, McGraw-Hill, New York, 1985

[2] Meduna, A. : Automata and Languages: Theory and Applications, Springer, London, 2000.

[3] Roman Lukáš: Semestral project, 2003