# Table-Driven Parsing of Scattered Context Grammar

Jirák Ota

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 00 Brno, CZ
www.fit.vutbr.cz/∼ijirak

BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

March 18, 2010– Research Group of Formal Models, Brno

# Outline

# Outline

# Outline

# Outline

# Outline

- deterministic SCG parsing algorithm

# Motivation

- deterministic SCG parsing algorithm
- expansion only on the pushdown top

# Topic

## SCG

SCG G = (N,T,P,S)

- N - nonterminals
- T - terminals
- P - $(A_1, \ldots, A_n) \rightarrow (x_1, \ldots, x_n)$,
  $A_i \in N, x_i \in (N \cup T)^*, n \in \mathbb{N}, 1 \geq i \geq n$
- S - starting nonterminal

# Prerequisites

## SCG

SCG G = (N,T,P,S)

- N - nonterminals
- T - terminals
- P - $(A_1, \ldots, A_n) \rightarrow (x_1, \ldots, x_n)$,
  $A_i \in N, x_i \in (N \cup T)^*, n \in \mathbb{N}, 1 \geq i \geq n$
- S - starting nonterminal

## Derivation

- $u = x_1 A_1 x_2 A_2 \ldots x_n A_n x_{n+1}$, $v = x_1 w_1 x_2 w_2 \ldots x_n w_n x_{n+1}$,
- $x_i \in V^*, A_i \in V \backslash T, 1 \leq i \leq n$, for some $n \geq 1$.
- $u \Rightarrow v, x_1 A_1 x_2 A_2 \ldots x_n A_n x_{n+1} \Rightarrow x_1 w_1 x_2 w_2 \ldots x_n w_n x_{n+1}$ is a derivation.
- If $x_1 \in T^*, x_i \in (V \backslash \{A_i\})^*$, it is a leftmost derivation.

# Topic

# Existing Principles

- Regulated Pushdown Automata

# Existing Principles

- Regulated Pushdown Automata
- modified Deep Pushdown Automata

# Topic

## Indexing of SCG production rules

Let $G = (V, T, P, S)$ be an SCG,
$r = (A_1, A_2, \ldots, A_n) \to (x_1, x_2, \ldots, x_n) \in P, n \in \mathbb{N}$.

$$
\begin{aligned}
r[k] &= (A_k) \to (x_k), k \in \mathbb{N}, 1 \leq k \leq n, \\
r[k:] &= (A_k, \ldots, A_n) \to (x_k, \ldots, x_n), k \in \mathbb{N}, 1 \leq k \leq n, \\
r[k] = r[k:] &= \varepsilon, k \in \mathbb{N}, k > n.
\end{aligned}
$$

## Indexing of SCG production rules

Let $G = (V, T, P, S)$ be an SCG,
$r = (A_1, A_2, \ldots, A_n) \rightarrow (x_1, x_2, \ldots, x_n) \in P, n \in \mathbb{N}$.

$$\begin{aligned} r[k] &= (A_k) \rightarrow (x_k), k \in \mathbb{N}, 1 \leq k \leq n, \\ r[k:] &= (A_k, \ldots, A_n) \rightarrow (x_k, \ldots, x_n), k \in \mathbb{N}, 1 \leq k \leq n, \\ r[k] = r[k:] &= \varepsilon, k \in \mathbb{N}, k > n. \end{aligned}$$

## LL SCG

Let G = (N,T,P,S) be an SCG. $P_1$ is a multiset, such as
$P_1 = \{p[1] : p \in P\}$. Then, G is LL SCG if $G_1 = (N, T, P_1, S)$ is LL CFG.

**Predictive parsing LL-table**

- two-dimensional data structure
- index to one dimension is nonterminal `N`.
- index to second dimension is terminal `a`.
- LL-table[$N$, $a$] contains scg production rule

# Definitions 2/3

## Predictive parsing LL-table

- two-dimensional data structure
- index to one dimension is nonterminal `N`.
- index to second dimension is terminal `a`.
- LL-table[$N$, $a$] contains scg production rule

## Generation

- symbol
- rule

$$S \Rightarrow_1 A_1 B_1 C_1 [1] \Rightarrow_2 a_2 A_2 B_1 c_2 C_2 [2]$$

## Function Reversal()

Let x, be an input string $x = x_1 x_2 \dots x_n$. And g is a generation. Function `reversal` reverse string and add generation `g` to each symbol.

$$reversal(x, g) =< x_n, g > \cdots < x_2, g >< x_1, g >$$

## Function Reversal()

Let x, be an input string $x = x_1 x_2 \dots x_n$. And g is a generation. Function `reversal` reverse string and add generation `g` to each symbol.

$$reversal(x, g) =< x_n, g > \cdots < x_2, g >< x_1, g >$$

## Delay-Bag

- key - pair $\langle Nonterminal, generation \rangle$
- value - unprocessed part of a production rule
- $Delay - bag[X, g]$:
    - X - lhs
    - g - generation

## Parsing Algorithm

**input** : LL-table for G=(N,T,P,S); $x \in T^*$
**output**: Left parse of x if $x \in L(G)$; otherwise, `error`

algorithm initiation
**while** *pushdown is not empty* **do**
    cycle initiation
    **switch** $X$ **do**
        **case** $X=\$$: handling dollar
        **case** $X \in T$: handling terminals
        **case** $X \in N$: handling nonterminals
    **endsw**
**end**
delay-bag emptiness

## Algorithm Initiation

generation = 0;
`push` (⟨\$,0⟩);
`push` (⟨S,0⟩) onto the pushdown;

## Algorithm Initiation

generation = 0;
`push` (⟨$,0⟩);
`push` (⟨S,0⟩) onto the pushdown;

## Cycle Initiation

let ⟨X,g⟩ = the pushdown top and a = the current token

### Handling Dollar

**if** $a = \$$ **then** `break`; **else** `error`;

### Handling Dollar

**if** a =$ **then** `break`; **else** `error`;

### Handling Terminals

**if** *X*=a **then**
  `pop` (⟨X,g⟩);
  read next a from input string;
**else** `error`;

## Handling Nonterminals

```
if delay-bag [X, g] is not empty then
    p:⟨(X, X_2, ..., X_n) → (x, x_2, ..., x_n), g′⟩ = delay-bag [X, g + 1];
    replace ⟨X,g⟩ with reversal (x,g′) on the pushdown;
    delay-bag [X, g′].remove();
    delay-bag [X_2, g′]:=p(2:);
else
    if r:(X, X_2, ..., X_n) → (x, x_2, ..., x_n) ∈ LL-table(X,a) then
        generation ++;
        replace ⟨X,g⟩ with reversal(x,generation) on the pushdown;
        write r to output;
        delay-bag [X_2, generation] := r [2 :];
    else
        error;
    end
end
```

# Algorithm 5/5

**Delay-Bag Emptiness**

**if** delay-bag *is not empty* **then**
```
 |  error;
```
**else**
```
 |  success;
```
**end**

# Topic

SCG G = (N,T,P,S), N={S,A,B,C}, T={a,b,c},
P ={

$$\begin{aligned} 1 : (S) &\rightarrow (ABC), \\ 2 : (A, B, C) &\rightarrow (aA, bB, cC), \\ 3 : (A, B, C) &\rightarrow (\varepsilon, \varepsilon, \varepsilon) \end{aligned}$$

}

input string: *aabbcc*\$.

| | a | b | c | \$ |
|---|---|---|---|---|
| S | 1 | | | |
| A | 2 | 3 | | |
| B | | | | |
| C | | | | |

# Example 2/2

| Pushdown | input | Rule | Derivation | Delay-Bag |
|---|---|---|---|---|
| ⟨\$,0⟩ ⟨$\underline{\text{S}}$,0⟩ | $\underline{\text{a}}$abbcc\$ | 1 | 1) $\underline{S} \Rightarrow \underline{A}BC$ | |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨B,1⟩ ⟨$\underline{\text{A}}$,1⟩ | $\underline{\text{a}}$abbcc\$ | 2 | 2) $\Rightarrow a\underline{A}BC$ | 2:$(B, C) \rightarrow (bB, cC)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨B,1⟩ ⟨A,2⟩ ⟨$\underline{\text{a}}$,2⟩ | $\underline{\text{a}}$abbcc\$ | pop | | 2:$(B, C) \rightarrow (bB, cC)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨B,1⟩ ⟨$\underline{\text{A}}$,2⟩ | $\underline{\text{a}}$bbcc\$ | 2 | 3)$\Rightarrow aa\underline{A}BC$ | 2:$(B, C) \rightarrow (bB, cC)$<br>3:$(B, C) \rightarrow (bB, cC)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨B,1⟩ ⟨A,3⟩ ⟨$\underline{\text{a}}$,3⟩ | $\underline{\text{a}}$bbcc\$ | pop | | 2:$(B, C) \rightarrow (bB, cC)$<br>3:$(B, C) \rightarrow (bB, cC)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨B,1⟩ ⟨$\underline{\text{A}}$,3⟩ | $\underline{\text{b}}$bcc\$ | 3 | 4) $\Rightarrow aa\underline{B}C$ | 2:$(B, C) \rightarrow (bB, cC)$<br>3:$(B, C) \rightarrow (bB, cC)$<br>4:$(B, C) \rightarrow (\varepsilon, \varepsilon)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨$\underline{\text{B}}$,1⟩ | $\underline{\text{b}}$bcc\$ | d2 | $\Rightarrow aab\underline{B}C$ | 2:$(C) \rightarrow (cC)$<br>3:$(B, C) \rightarrow (bB, cC)$<br>4:$(B, C) \rightarrow (\varepsilon, \varepsilon)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨B,2⟩ ⟨$\underline{\text{b}}$,2⟩ | $\underline{\text{b}}$bcc\$ | pop | | 2:$(C) \rightarrow (cC)$<br>3:$(B, C) \rightarrow (bB, cC)$<br>4:$(B, C) \rightarrow (\varepsilon, \varepsilon)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨$\underline{\text{B}}$,2⟩ | $\underline{\text{b}}$cc\$ | d3 | $\Rightarrow aabb\underline{B}C$ | 2:$(C) \rightarrow (cC)$<br>3:$(C) \rightarrow (cC)$<br>4:$(B, C) \rightarrow (\varepsilon, \varepsilon)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨B,3⟩ ⟨$\underline{\text{b}}$,3⟩ | $\underline{\text{c}}$c\$ | pop | | 2:$(C) \rightarrow (cC)$<br>3:$(C) \rightarrow (cC)$<br>4:$(B, C) \rightarrow (\varepsilon, \varepsilon)$ |
| ⟨\$,0⟩ ⟨C,1⟩ ⟨$\underline{\text{B}}$,3⟩ | $\underline{\text{c}}$c\$ | d4 | $\Rightarrow aabb\underline{C}$ | 2:$(C) \rightarrow (cC)$<br>3:$(C) \rightarrow (cC)$<br>4:$(C) \rightarrow (\varepsilon)$ |
| ⟨\$,0⟩ ⟨$\underline{\text{C}}$,1⟩ | $\underline{\text{c}}$c\$ | d2 | $\Rightarrow aabbc\underline{C}$ | 3:$(C) \rightarrow (cC)$<br>4:$(C) \rightarrow (\varepsilon)$ |
| ⟨\$,0⟩ ⟨C,2⟩ ⟨$\underline{\text{c}}$,2⟩ | $\underline{\text{c}}$c\$ | pop | | 3:$(C) \rightarrow (cC)$<br>4:$(C) \rightarrow (\varepsilon)$ |
| ⟨\$,0⟩ ⟨$\underline{\text{C}}$,2⟩ | $\underline{\text{c}}$\$ | d3 | | 4:$(C) \rightarrow (\varepsilon)$ |
| ⟨\$,0⟩ ⟨C,3⟩ ⟨$\underline{\text{c}}$,3⟩ | $\underline{\text{c}}$\$ | pop | | 4:$(C) \rightarrow (\varepsilon)$ |
| ⟨\$,0⟩ ⟨$\underline{\text{C}}$,3⟩ | $\underline{\text{\\$}}$ | d4 | $\Rightarrow aabbcc$ | |
| ⟨$\underline{\text{\\$}}$,0⟩ | $\underline{\text{\\$}}$ | pop | success | |

## Conclusion

- table-driven parsing algorithm for SCG
- avoids expansion in the middle of pushdown
- using principle of a lazy-function evaluation

# Conclusion and Future Work

## Conclusion

- table-driven parsing algorithm for SCG
- avoids expansion in the middle of pushdown
- using principle of a lazy-function evaluation

## Open Questions and Future Work

- generative power of LL SCG
- efficient implementation of Delay-Bag

Thank you for your attention!

End