Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Top Down Parsing

# Johannes Dellert Aleksandar Dimitrov

Seminar für Sprachwissenschaft, Universität Tübingen

December 2006

< □ > < 同 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

# Outline

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

# Summary

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

# **Top-Down Parsing**

# CYK and Unger parser are non-directional methods

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Top-Down Parsing

- CYK and Unger parser are non-directional methods
- They need the whole input sentence before beginning to parse

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

### Summary

# Top-Down Parsing

- CYK and Unger parser are non-directional methods
- They need the whole input sentence before beginning to parse

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

 Today we introduce a directional top-down parsing method

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Desce

#### Summary

# Top-Down Parsing

- CYK and Unger parser are non-directional methods
- They need the whole input sentence before beginning to parse
- Today we introduce a directional top-down parsing method
- This is what the term 'Top-down Parsing' usually refers to

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# General approach

# Rederive the word starting at the input symbol

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQ@

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# General approach

Rederive the word starting at the input symbol

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Build the tree from the top

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

- Rederive the word starting at the input symbol
- Build the tree from the top

General approach

Collect 'ideas' on how the tree might be continued

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Rederive the word starting at the input symbol

Build the tree from the top

General approach

Collect 'ideas' on how the tree might be continued

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

If the tree is 'full' and all the input is in the tree, parsing was successful

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

# Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

# Intuitive example

Assume the following grammar:

- $\mathsf{S} \to \mathsf{NP} \; \mathsf{VP}$
- $\mathsf{NP} \to \mathsf{D} \mathsf{N}$
- $\mathsf{VP} \to \mathsf{VT} \; \mathsf{NP} \;|\; \mathsf{VI} \; \mathsf{PP}$
- $\mathsf{PP} \to \mathsf{P} \; \mathsf{NP}$
- $\mathsf{D} \to \mathsf{der} \mid \mathsf{die}$

$$\mathsf{N} \to \mathsf{Mond} \mid \mathsf{Wiese}$$

$$\mathsf{VI} \to \mathsf{scheint}$$

- $VT \to \text{bescheint}$
- $\mathsf{P} \to \mathsf{auf}$

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

## Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

Now let us parse the sentence 'der Mond scheint auf die Wiese'

▲ロト ▲帰 ト ▲ ヨ ト ▲ ヨ ト ● の Q ()

First 'tree idea':

Intuitive example

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

### Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

 Now let us parse the sentence 'der Mond scheint auf die Wiese'

▲ロト ▲帰 ト ▲ ヨ ト ▲ ヨ ト ● の Q ()

First 'tree idea':

Intuitive example

S

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

### Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

# Intuitive example

'Tree idea' is expanded via leftmost derivations:

S NP VP

### ◆□▶ ◆□▶ ◆□▶ ◆□▶ = 三 のへで

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

### Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Intuitive example

'Tree idea' is expanded via leftmost derivations:





#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

### Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Intuitive example

# 'Tree idea' is expanded via leftmost derivations:



▲ロト ▲圖 ト ▲ ヨト ▲ ヨト ― ヨー つくぐ

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

### Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Tree begins to match input and is expanded:

Intuitive example



◆□▶ ◆□▶ ◆□▶ ◆□▶ = 三 のへで

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

### Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# ▶ Tree begins to match input and is expanded:

Intuitive example



#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

### Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# ▶ Tree begins to match input and is expanded:

Intuitive example



◆□▶ ◆□▶ ◆□▶ ◆□▶ = 三 のへで

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and

Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# Intuitive example

# Nondeterminism: Two different possible trees.



イロト 不得 トイヨト イヨト

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and

Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Desce

Summary

# Intuitive example



#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

### Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# Intuitive example

► No scan possible for first tree; remaining tree gets expanded



#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

# Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Intuitive example

# Expanding the predicted tree



#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

# Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

#### Summary

# Intuitive example

# Expanding the predicted tree



◆□ > ◆□ > ◆豆 > ◆豆 > ・豆 = のへ⊙

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

# Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Intuitive example

# Expanding the predicted tree



▲ロト ▲圖 ト ▲ ヨト ▲ ヨト ― ヨー つくぐ

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

# Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Intuitive example

# Expanding the predicted tree



・ロト ・ 一下・ ・ ヨト・

-20

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

# Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

#### Summary

# Intuitive example

# Expanding the predicted tree



・ロト ・ 一下・ ・ ヨト・

-20

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

# Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Intuitive example

# Expanding the predicted tree



・ロト ・ 一下・ ・ ヨト・

-20

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and

Peatures and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

# Intuitive example

Expanding the predicted tree
S



◆□ > ◆□ > ◆豆 > ◆豆 > ・豆 ・ の Q @ >

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# Intuitive example

Tree completed



◆□> ◆□> ◆目> ◆目> ◆目> 三三 のへで

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# General features of the top down method

# ▶ The parser makes predictions about the input.

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# General features of the top down method

- The parser makes predictions about the input.
- ► The left-most prediction is usually processed first.

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# General features of the top down method

- The parser makes predictions about the input.
- ► The left-most prediction is usually processed first.
- Terminals in the prediction are matched against the input.

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# General features of the top down method

- The parser makes predictions about the input.
- ► The left-most prediction is usually processed first.
- Terminals in the prediction are matched against the input.
- Non-Terminals are replaced by one of the right hand sides.

Operations

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example

Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# For Bottom-Up-Parsing, we got to know SHIFT and REDUCE

▲ロト ▲帰 ト ▲ ヨ ト ▲ ヨ ト ● の Q ()

Operations

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example

Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

- For Bottom-Up-Parsing, we got to know SHIFT and REDUCE
- The corresponding operations for Top-Down-Parsing are called PREDICT and SCAN
Operations

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example

Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

## Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

- For Bottom-Up-Parsing, we got to know SHIFT and REDUCE
- The corresponding operations for Top-Down-Parsing are called PREDICT and SCAN
- PREDICT replaces a non-terminal in the sentential form with the right hand side of a corresponding rule:

Operations

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example

Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

- For Bottom-Up-Parsing, we got to know SHIFT and REDUCE
- The corresponding operations for Top-Down-Parsing are called PREDICT and SCAN
- PREDICT replaces a non-terminal in the sentential form with the right hand side of a corresponding rule:
- $\blacktriangleright$  e.g. der Mond VP  $\rightarrow$  der Mond V PP for a rule VP  $\rightarrow$  V PP

Operations

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example

Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

- For Bottom-Up-Parsing, we got to know SHIFT and REDUCE
- The corresponding operations for Top-Down-Parsing are called PREDICT and SCAN
- PREDICT replaces a non-terminal in the sentential form with the right hand side of a corresponding rule:
- $\blacktriangleright$  e.g. der Mond VP  $\rightarrow$  der Mond V PP for a rule VP  $\rightarrow$  V PP
- SCAN matches a terminal in the sentential form with a symbol on the input string

Operations

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example

Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

- For Bottom-Up-Parsing, we got to know SHIFT and REDUCE
- The corresponding operations for Top-Down-Parsing are called PREDICT and SCAN
- PREDICT replaces a non-terminal in the sentential form with the right hand side of a corresponding rule:
- $\blacktriangleright$  e.g. der Mond VP  $\rightarrow$  der Mond V PP for a rule VP  $\rightarrow$  V PP
- SCAN matches a terminal in the sentential form with a symbol on the input string

・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

 $\blacktriangleright$  e.g. der N VP  $\rightarrow$  N VP, 'der' matched in the input string

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

 Parsing Schemata are a formal way of describing parsing methods

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Parsing Schema - Basics

Johannes Dellert, Aleksandar Dimitrov

## Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

# Summary

 Parsing Schemata are a formal way of describing parsing methods

< □ > < 同 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

▶ they are independent of the actual implementation

Parsing Schema - Basics

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Parsing Schema - Basics

- Parsing Schemata are a formal way of describing parsing methods
- ▶ they are independent of the actual implementation
- Every recognized subtree (or tree hypothesis) is stored as an item

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

- Parsing Schemata are a formal way of describing parsing methods
- they are independent of the actual implementation
- Every recognized subtree (or tree hypothesis) is stored as an item

• Items look like this:  $[\bullet\beta, j]$ 

Parsing Schema - Basics

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# Parsing Schemata are a formal way of describing parsing methods

- they are independent of the actual implementation
- Every recognized subtree (or tree hypothesis) is stored as an item
- Items look like this:  $[\bullet\beta, j]$

Parsing Schema - Basics

Meaning: β parts of the tree to be 'filled', j current position in input string

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

# Parsing Schemata are a formal way of describing parsing methods

- they are independent of the actual implementation
- Every recognized subtree (or tree hypothesis) is stored as an item
- Items look like this:  $[\bullet\beta, j]$

Parsing Schema - Basics

- Meaning: β parts of the tree to be 'filled', j current position in input string
- ► We start with [•S, 0] because the whole tree has to be built and we have not yet scanned anything from the input string

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

# Parsing Schemata are a formal way of describing parsing methods

- they are independent of the actual implementation
- Every recognized subtree (or tree hypothesis) is stored as an item
- Items look like this:  $[\bullet\beta, j]$

Parsing Schema - Basics

- Meaning: β parts of the tree to be 'filled', j current position in input string
- ► We start with [•S, 0] because the whole tree has to be built and we have not yet scanned anything from the input string
- Our goal item will be [•, n] meaning that the tree is complete and the whole input of length n is scanned

### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Parsing Schema - Inference Rules

# How do we formalize the scanning step?

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Parsing Schema - Inference Rules

How do we formalize the scanning step?

$$\begin{bmatrix} \bullet w_{j+1}\beta, j \end{bmatrix} \\ \begin{bmatrix} \bullet\beta, j+1 \end{bmatrix}$$

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

(1)

How do we formalize the prediction step?

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Parsing Schema - Inference Rules

How do we formalize the scanning step?

$$\frac{[\bullet w_{j+1}\beta, j]}{[\bullet\beta, j+1]} \tag{1}$$

How do we formalize the prediction step?

$$\frac{[\bullet B\beta, j]}{[\bullet\gamma\beta, j]}B \to \gamma$$
(2)

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

1

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

### Summary

# Now a complete derivation - with the Schema

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

# $[\bullet S, 0]$ INITIALIZE

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

1

2

## Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Now a complete derivation - with the Schema

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

# $[\bullet S, 0]$ INITIALIZE $[\bullet NP VP, 0]$ PREDICT from 1

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

1

2

3

# Motivation

Why Top-Down? Why Not?

## Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

# Summary

# Now a complete derivation - with the Schema

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

 $\begin{bmatrix} \bullet S, 0 \end{bmatrix} \quad \begin{array}{l} \mathsf{INITIALIZE} \\ \begin{bmatrix} \bullet NP \ VP, 0 \end{bmatrix} \quad \begin{array}{l} \mathsf{PREDICT from 1} \\ \hline \bullet D \ N \ VP, 0 \end{bmatrix} \quad \begin{array}{l} \mathsf{PREDICT from 2} \end{array}$ 

# Now a complete derivation - with the Schema

Johannes Dellert. Aleksandar Dimitrov

Parsing Schema

Why Top-Down? Why Not?

PDA + GNF Depth-first Left recursion

1  $[\bullet S, 0]$ 2  $[\bullet NP VP, 0]$ 3  $[\bullet der \ N \ VP, 0]$ 4

INITIALIZE PREDICT from 1 [•D N VP,0] PREDICT from 2 PREDICT from 3

(日) (雪) (日) (日) (日)

# Now a complete derivation - with the Schema

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

## Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

# Summary

1	<b>[●</b> <i>S</i> ,0]
2	[● <i>NP VP</i> ,0]
3	[● <i>D N VP</i> ,0]
4	$[\bullet der \ N \ VP, 0]$
5	$[\bullet die \ N \ VP, 0]$

INITIALIZE PREDICT from 1 PREDICT from 2 PREDICT from 3 PREDICT from 3

# Now a complete derivation - with the Schema

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# [●*S*, 0] [●*NP VP*, 0] [●*D N VP*, 0] [●*der N VP*, 0] [●*die N VP*, 0] [●*N VP*, 1]

1 2

3

4

5

6

INITIALIZE PREDICT from 1 PREDICT from 2 PREDICT from 3 PREDICT from 3 SCAN from 4

・ロト ・ 一下・ ・ ヨト・

- 10

# Now a complete derivation - with the Schema

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

## Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# [●*S*, 0] [●*NP VP*, 0] [●*D N VP*, 0] [●*der N VP*, 0] [●*die N VP*, 0] [●*N VP*, 1] [●*Mond VP*, 1]

1

2

3

4

5

6

7

- INITIALIZE
  - PREDICT from 1
  - PREDICT from 2
  - PREDICT from 3
- PREDICT from 3
- SCAN from 4
- PREDICT from 6

・ロト ・ 一下・ ・ ヨト・

- 10

# Now a complete derivation - with the Schema

Johannes Dellert. Aleksandar Dimitrov

Parsing Schema

Why Top-Down? Why Not?

PDA + GNF Depth-first Left recursion

1	<b>[●</b> <i>S</i> ,0]
2	[● <i>NP VP</i> ,0]
3	$[\bullet D \ N \ VP, 0]$
4	$[\bullet der \ N \ VP, 0]$
5	[●die N VP,0]
6	$[\bullet N VP, 1]$
7	$[\bullet Mond VP, 1]$
8	$[\bullet Wiese VP, 1]$

INITIALIZE

PREDICT from 1

PREDICT from 2

PREDICT from 3

PREDICT from 3

SCAN from 4

PREDICT from 6

PREDICT from 6

# Now a complete derivation - with the Schema

Johannes Dellert. Aleksandar Dimitrov

Parsing Schema

Why Top-Down? Why Not?

PDA + GNF Depth-first Left recursion

1	<b>[●</b> <i>S</i> ,0]	I
2	[● <i>NP VP</i> ,0]	F
3	$[\bullet D \ N \ VP, 0]$	F
4	$[\bullet der \ N \ VP, 0]$	F
5	[●die N VP,0]	F
6	$[\bullet N VP, 1]$	S
7	$[\bullet Mond VP, 1]$	F
8	$[\bullet Wiese VP, 1]$	F
9	[● <i>VP</i> ,2]	S

# NITIALIZE

PREDICT from 1

- PREDICT from 2
- PREDICT from 3
- PREDICT from 3
- SCAN from 4
- PREDICT from 6
- PREDICT from 6

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

SCAN from 7

Johannes Dellert, Aleksandar Dimitrov

9

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

### Summary

# Now a complete derivation - with the Schema

 $[\bullet VP, 2]$  SCAN from 7

◆□▶ ◆□▶ ◆□▶ ◆□▶ ● □ ● ● ● ●

Johannes Dellert, Aleksandar Dimitrov

9

10

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

### Summary

# Now a complete derivation - with the Schema

 $[\bullet VP, 2]$  SCAN from 7  $[\bullet VT NP, 2]$  PREDICT from 9

## Johannes Dellert, Aleksandar Dimitrov

9

10

11

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

# Summary

# Now a complete derivation - with the Schema

[●*VP*, 2] SCAN from 7 [●*VT NP*, 2] PREDICT from 9 [●*VI PP*, 2] PREDICT from 9

# Johannes Dellert, Aleksandar Dimitrov

9 10 11

12

### Introduction

Intuitive example Features and Operations Parsing Schema

## Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

# Summary

# Now a complete derivation - with the Schema

[● <i>VP</i> ,2]	SCAN from 7
[● <i>VT NP</i> ,2]	PREDICT from 9
[● <i>VI PP</i> ,2]	PREDICT from 9
[•bescheint NP,2]	PREDICT from 10

# Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

## Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

### Summary

# Now a complete derivation - with the Schema

9	[● <i>VP</i> ,2]	SCAN from 7
10	[● <i>VT NP</i> ,2]	PREDICT from 9
11	[● <i>VI PP</i> ,2]	PREDICT from 9
12	[•bescheint NP,2]	PREDICT from 10
13	[•scheint PP,2]	PREDICT from 11

# Johannes Dellert, Aleksandar Dimitrov

1 1

1 1

1

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

# Summary

# Now a complete derivation - with the Schema

9	[● <i>VP</i> ,2]	SCAN from 7
0	[● <i>VT NP</i> ,2]	PREDICT from 9
1	[● <i>VI PP</i> ,2]	PREDICT from 9
2	[•bescheint NP,2]	PREDICT from 10
3	[●scheint PP,2]	PREDICT from 11
4	[● <i>PP</i> , 3]	SCAN from 13

# Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

## Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Now a complete derivation - with the Schema

9	[● <i>VP</i> , 2]	SCAN from 7
0	[● <i>VT NP</i> ,2]	PREDICT from 9
1	[● <i>VI PP</i> ,2]	PREDICT from 9
2	[•bescheint NP,2]	PREDICT from 10
3	[●scheint PP,2]	PREDICT from 11
4	[● <i>PP</i> , 3]	SCAN from 13
5	[● <i>P NP</i> ,3]	PREDICT from 14

# Now a complete derivation - with the Schema

# Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

## Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

# Summary

9	[● <i>VP</i> ,2]	SCAN from 7
10	$[\bullet VT NP, 2]$	PREDICT from 9
11	[● <i>VI PP</i> ,2]	PREDICT from 9
12	[•bescheint NP,2]	PREDICT from 10
13	[•scheint PP,2]	PREDICT from 11
14	[● <i>PP</i> ,3]	SCAN from 13
15	[● <i>P NP</i> , 3]	PREDICT from 14
16	[•auf NP,3]	PREDICT from 15

# Now a complete derivation - with the Schema

# Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

# Summary

9	[● <i>VP</i> ,2]	SCAN from 7
10	[● <i>VT NP</i> ,2]	PREDICT from 9
11	[● <i>VI PP</i> ,2]	PREDICT from 9
12	[•bescheint NP,2]	PREDICT from 10
13	[•scheint PP,2]	PREDICT from 11
14	[● <i>PP</i> ,3]	SCAN from 13
15	[● <i>P NP</i> ,3]	PREDICT from 14
16	[• <i>auf</i> NP,3]	PREDICT from 15
17	[● <i>NP</i> ,4]	SCAN from 16

# Now a complete derivation - with the Schema

# Johannes Dellert, Aleksandar Dimitrov

9 10 11

12 13

14 15

16

17 18

#### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

SCAN from 7
PREDICT from 9
PREDICT from 9
PREDICT from 10
PREDICT from 11
SCAN from 13
PREDICT from 14
PREDICT from 15
SCAN from 16
PREDICT from 17

# Now a complete derivation - with the Schema

# Johannes Dellert, Aleksandar Dimitrov

9 10 11

12 13

14 15

16

17 18 19

#### Introduction

Intuitive example Features and Operations Parsing Schema

## Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

[● <i>VP</i> , 2]	SCAN from 7
$[\bullet VT NP, 2]$	PREDICT from 9
[● <i>VI PP</i> ,2]	PREDICT from 9
[•bescheint NP,2]	PREDICT from 10
[●scheint PP,2]	PREDICT from 11
[● <i>PP</i> ,3]	SCAN from 13
[● <i>P NP</i> ,3]	PREDICT from 14
[● <i>auf</i> NP,3]	PREDICT from 15
[● <i>NP</i> , 4]	SCAN from 16
[● <i>D N</i> ,4]	PREDICT from 17
[● <i>der N</i> ,4]	PREDICT from 18

# Now a complete derivation - with the Schema

Johannes Dellert, Aleksandar Dimitrov

9 10 11

12 13

14 15

16

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

[● <i>VP</i> ,2]	SCAN from 7
$[\bullet VT NP, 2]$	PREDICT from 9
[● <i>VI PP</i> ,2]	PREDICT from 9
[•bescheint NP,2]	PREDICT from 10
[●scheint PP,2]	PREDICT from 11
[● <i>PP</i> ,3]	SCAN from 13
[● <i>P NP</i> ,3]	PREDICT from 14
[● <i>auf NP</i> ,3]	PREDICT from 15
[● <i>NP</i> ,4]	SCAN from 16
[● <i>D N</i> ,4]	PREDICT from 17
[● <i>der</i> N,4]	PREDICT from 18
[● <i>die N</i> ,4]	PREDICT from 18

# Now a complete derivation - with the Schema

Johannes Dellert, Aleksandar Dimitrov

9 10 11

12 13

14 15

16

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

[● <i>VP</i> ,2]	SCAN from 7
$[\bullet VT NP, 2]$	PREDICT from 9
[● <i>VI PP</i> ,2]	PREDICT from 9
[•bescheint NP,2]	PREDICT from 10
[•scheint PP,2]	PREDICT from 11
[● <i>PP</i> ,3]	SCAN from 13
[● <i>P NP</i> ,3]	PREDICT from 14
[● <i>auf NP</i> ,3]	PREDICT from 15
[● <i>NP</i> ,4]	SCAN from 16
[● <i>D N</i> ,4]	PREDICT from 17
[● <i>der N</i> ,4]	PREDICT from 18
[● <i>die N</i> ,4]	PREDICT from 18
[● <i>N</i> ,5]	SCAN from 20
# Now a complete derivation - with the Schema

Johannes Dellert, Aleksandar Dimitrov

9 10 11

12 13

14 15

16

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

[• <i>VP</i> ,2]	SCAN from 7
$[\bullet VT NP, 2]$	PREDICT from 9
[• <i>VI PP</i> ,2]	PREDICT from 9
[•bescheint NP,2]	PREDICT from 10
[•scheint PP,2]	PREDICT from 11
[● <i>PP</i> ,3]	SCAN from 13
[● <i>P NP</i> ,3]	PREDICT from 14
[● <i>auf NP</i> ,3]	PREDICT from 15
[● <i>NP</i> , 4]	SCAN from 16
[● <i>D N</i> ,4]	PREDICT from 17
[● <i>der N</i> ,4]	PREDICT from 18
[● <i>die</i> N,4]	PREDICT from 18
[● <i>N</i> ,5]	SCAN from 20
[● <i>Mond</i> ,5]	PREDICT from 21

◆□ > ◆□ > ◆豆 > ◆豆 > ・豆 ・ の Q @

# Now a complete derivation - with the Schema

Johannes Dellert, Aleksandar Dimitrov

9 10 11

12 13

14 15

16

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

[• <i>VP</i> ,2]	SCAN from 7
$[\bullet VI NP, 2]$	PREDICT from 9
[● <i>VI PP</i> ,2]	PREDICT from 9
[•bescheint NP,2]	PREDICT from 10
[●scheint PP,2]	PREDICT from 11
[● <i>PP</i> ,3]	SCAN from 13
[● <i>P NP</i> ,3]	PREDICT from 14
[● <i>auf NP</i> ,3]	PREDICT from 15
[● <i>NP</i> , 4]	SCAN from 16
[● <i>D N</i> ,4]	PREDICT from 17
[● <i>der N</i> ,4]	PREDICT from 18
[● <i>die N</i> ,4]	PREDICT from 18
[● <i>N</i> ,5]	SCAN from 20
[• <i>Mond</i> , 5]	PREDICT from 21
$[\bullet Wiese, 5]$	PREDICT from 21

# Now a complete derivation - with the Schema

Johannes Dellert, Aleksandar Dimitrov

9 10 11

12 13

14 15

16

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

[•VP,2] [•VT NP,2] [•VI PP,2] [•bescheint NP,2] [•scheint PP,2] [•PP,3] [•P NP,3] [•P NP,3] [•NP,4] [•D N,4] [•der N,4] [•die N,4] [•N,5]	SCAN from 7 PREDICT from 9 PREDICT from 9 PREDICT from 10 PREDICT from 11 SCAN from 13 PREDICT from 14 PREDICT from 15 SCAN from 16 PREDICT from 17 PREDICT from 18 PREDICT from 18 PREDICT from 18 SCAN from 20
$[\bullet are N, 4]$ $[\bullet N, 5]$	SCAN from 20
[● <i>Mond</i> ,5] [● <i>Wiese</i> ,5] [●,6]	PREDICT from 21 PREDICT from 21 SCAN from 23 - GOAL

◆□▶ ◆□▶ ◆□▶ ◆□▶ ● □ ● ● ● ●

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Advantages

# Offers real-time processing of input

◆□ > ◆□ > ◆豆 > ◆豆 > ・豆 ・ の Q @

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# Advantages

- Offers real-time processing of input
- Resembles human real-time parsing

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Advantages

- Offers real-time processing of input
- Resembles human real-time parsing
- Relatively easy to implement using stacks

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Advantages

- Offers real-time processing of input
- Resembles human real-time parsing
- Relatively easy to implement using stacks
- Efficient in comparison with Unger parser

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

### Summary

# Advantages

- Offers real-time processing of input
- Resembles human real-time parsing
- Relatively easy to implement using stacks
- Efficient in comparison with Unger parser

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Not less efficient than CYK

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Disadvantages

# LEFT RECURSION really is a problem (cf. implementation)

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

Summary

# Disadvantages

- LEFT RECURSION really is a problem (cf. implementation)
- might check very unlikely predictions first

Disadvantages

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# LEFT RECURSION really is a problem (cf. implementation)

might check very unlikely predictions first

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

no lookahead in primitive version

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Disadvantages

- LEFT RECURSION really is a problem (cf. implementation)
- might check very unlikely predictions first

- no lookahead in primitive version
- Most Parsers are Non-deterministic

### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Implementations

# Push Down Automata

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへで

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Implementations

# Push Down Automata

> PDA, GNF and top-down are directly related

▲ロト ▲帰 ト ▲ ヨ ト ▲ ヨ ト ● の Q ()

### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Implementations

# Push Down Automata

- PDA, GNF and top-down are directly related
- Their approach to the problem is very similar

### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Implementations

# Push Down Automata

- PDA, GNF and top-down are directly related
- Their approach to the problem is very similar

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Breadth first

### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Implementations

# Push Down Automata

- PDA, GNF and top-down are directly related
- Their approach to the problem is very similar

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

# Breadth first

Allow fast parsing (even on-line)

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Implementations

# Push Down Automata

- PDA, GNF and top-down are directly related
- Their approach to the problem is very similar

- Breadth first
  - Allow fast parsing (even on-line)
  - Use lots of memory

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

# Implementations

# Push Down Automata

- PDA, GNF and top-down are directly related
- Their approach to the problem is very similar

- Breadth first
  - Allow fast parsing (even on-line)
  - Use lots of memory
- Depth first

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

# Implementations

# Push Down Automata

- PDA, GNF and top-down are directly related
- Their approach to the problem is very similar

- Breadth first
  - Allow fast parsing (even on-line)
  - Use lots of memory
- Depth first
  - Also called backtracking

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

# Summary

# Implementations

# Push Down Automata

- > PDA, GNF and top-down are directly related
- Their approach to the problem is very similar

- Breadth first
  - Allow fast parsing (even on-line)
  - Use lots of memory
- Depth first
  - Also called backtracking
  - Are simple to write

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

# Summary

# Implementations

# Push Down Automata

- PDA, GNF and top-down are directly related
- Their approach to the problem is very similar

- Breadth first
  - Allow fast parsing (even on-line)
  - Use lots of memory
- Depth first
  - Also called backtracking
  - Are simple to write
  - Have certain problems with prefixes

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

# Summary

# Implementations

# Push Down Automata

- PDA, GNF and top-down are directly related
- Their approach to the problem is very similar
- Breadth first
  - Allow fast parsing (even on-line)
  - Use lots of memory
- Depth first
  - Also called backtracking
  - Are simple to write
  - Have certain problems with prefixes
- Recursive descent is a technique to implement a Depth first parser

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ◇ ◇ ◇

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF

Depth-first Left recursion Recursive Descent

Summary

# transitions of a pushdown automaton strongly resemble operations of a top-down parser:

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

PDA as an implementation model

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

 $\mathsf{PDA} + \mathsf{GNF}$ 

Breadth-first Depth-first Left recursion Recursive Descent

Summary

# PDA as an implementation model

transitions of a pushdown automaton strongly resemble operations of a top-down parser:

► 
$$\delta(q_0, \epsilon, A) = q_0, \epsilon, BC \longleftrightarrow$$
 PREDICT

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

 $\mathsf{PDA} + \mathsf{GNF}$ 

Breadth-first Depth-first Left recursion Recursive Descent

Summary

# PDA as an implementation model

transitions of a pushdown automaton strongly resemble operations of a top-down parser:

• 
$$\delta(q_0, \epsilon, A) = q_0, \epsilon, BC \longleftrightarrow \mathsf{PREDICT}$$

$$\blacktriangleright \ \delta(q_0, a, a) = q_0, \epsilon, \epsilon \longleftrightarrow \mathsf{SCAN}$$

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

 $\mathsf{PDA} + \mathsf{GNF}$ 

Breadth-first Depth-first Left recursion Recursive Descent

Summary

# PDA as an implementation model

transitions of a pushdown automaton strongly resemble operations of a top-down parser:

• 
$$\delta(q_0, \epsilon, A) = q_0, \epsilon, BC \longleftrightarrow \mathsf{PREDICT}$$

• 
$$\delta(q_0, a, a) = q_0, \epsilon, \epsilon \longleftrightarrow \mathsf{SCAN}$$

 $\blacktriangleright \to$  in implementations, every tree hypothesis contains a stack and an input position

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

 $\mathsf{PDA} + \mathsf{GNF}$ 

Breadth-first Depth-first Left recursion Recursive Descent

Summary

transitions of a pushdown automaton strongly resemble operations of a top-down parser:

• 
$$\delta(q_0, \epsilon, A) = q_0, \epsilon, BC \longleftrightarrow \mathsf{PREDICT}$$

• 
$$\delta(q_0, a, a) = q_0, \epsilon, \epsilon \longleftrightarrow \mathsf{SCAN}$$

PDA as an implementation model

 $\blacktriangleright \rightarrow$  in implementations, every tree hypothesis contains a stack and an input position

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ◇ ◇ ◇

• compare parsing schema item:  $[\bullet\beta, j]$ 

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

# $\mathsf{PDA} + \mathsf{GNF}$

Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

# Greibach Normal Form

# allows only productions of the following form:

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQ@

### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

# $\mathsf{PDA} + \mathsf{GNF}$

Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

# Greibach Normal Form

allows only productions of the following form:

< □ > < 同 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

•  $A \longrightarrow aB_1...B_k$  with  $k \ge 0$ 

# Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Greibach Normal Form

- allows only productions of the following form:
- $A \longrightarrow aB_1...B_k$  with  $k \ge 0$
- ▶ invented by American mathematician Sheila A. Greibach

### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Greibach Normal Form

- allows only productions of the following form:
- $A \longrightarrow aB_1...B_k$  with  $k \ge 0$
- ▶ invented by American mathematician Sheila A. Greibach

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

 incidentally, she was also first to propose top-down-parsing

### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Greibach Normal Form

- allows only productions of the following form:
- $A \longrightarrow aB_1...B_k$  with  $k \ge 0$
- ▶ invented by American mathematician Sheila A. Greibach
- incidentally, she was also first to propose top-down-parsing
- ▶ What's the relation? Why is GNF ideal for TD-parsing?

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ◇ ◇ ◇

# GNF and Top-Down-Parsing

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

# $\mathsf{PDA} + \mathsf{GNF}$

Breadth-first Depth-first Left recursion Recursive Descent

#### Summary

# • $A \longrightarrow aB_1...B_k$ with $k \ge 0$

# GNF and Top-Down-Parsing

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

# PDA + GNF

Breadth-first Depth-first Left recursion Recursive Descent

Summary

# • $A \longrightarrow aB_1...B_k$ with $k \ge 0$

 a grammar in GNF form reduces the amount of prediction steps needed

# GNF and Top-Down-Parsing

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

```
PDA + GNF
Breadth-first
Depth-first
Left recursion
Recursive Descent
```

Summary

# • $A \longrightarrow aB_1...B_k$ with $k \ge 0$

- a grammar in GNF form reduces the amount of prediction steps needed
- each prediction will result in a possible scanning step
# GNF and Top-Down-Parsing

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# • $A \longrightarrow aB_1...B_k$ with $k \ge 0$

- a grammar in GNF form reduces the amount of prediction steps needed
- each prediction will result in a possible scanning step
- a wrong prediction can be discarded already with the next scanning step

# GNF and Top-Down-Parsing

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# • $A \longrightarrow aB_1...B_k$ with $k \ge 0$

- a grammar in GNF form reduces the amount of prediction steps needed
- each prediction will result in a possible scanning step
- a wrong prediction can be discarded already with the next scanning step
- intelligent implementations would only make predictions that start with the next terminal in the input

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

# GNF and Top-Down-Parsing

Johannes Dellert, Aleksandar Dimitrov

# Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# • $A \longrightarrow aB_1...B_k$ with $k \ge 0$

- a grammar in GNF form reduces the amount of prediction steps needed
- each prediction will result in a possible scanning step
- a wrong prediction can be discarded already with the next scanning step
- intelligent implementations would only make predictions that start with the next terminal in the input

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

 $\rightarrow$  GNF is for TD what CNF is for BU

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF

#### Breadth-first Depth-first

Left recursion Recursive Descent

#### Summary

# Breadth-first

# ▶ First described by Greibach (1964)

◆□▶ ◆□▶ ◆□▶ ◆□▶ ▲□ ● ● ●

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF

#### Breadth-first Depth-first

Left recursion Recursive Descent

### Summary

First described by Greibach (1964)

Breadth-first

▶ Maintains a list of *possible* derivations ....

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

# Breadth-first

Depth-first Left recursion Recursive Descent

### Summary

- First described by Greibach (1964)
- ► Maintains a list of *possible* derivations ....

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

... which is kept in memory.

Breadth-first

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

### Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# Breadth-first

- First described by Greibach (1964)
- Maintains a list of *possible* derivations ....
- ... which is kept in memory.
- Increases size of the list with every prediction operation

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

### Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# First described by Greibach (1964)

- Maintains a list of *possible* derivations ...
- ... which is kept in memory.

Breadth-first

- Increases size of the list with every prediction operation
  - For every non-terminal *all* possible derivations are added.

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF

# Breadth-first

Depth-first Left recursion Recursive Descent

Summary

- First described by Greibach (1964)
- Maintains a list of *possible* derivations ...
- ... which is kept in memory.

Breadth-first

- Increases size of the list with every prediction operation
  - For every non-terminal *all* possible derivations are added.
  - A LL parser predicts until every left-hand symbol is a terminal.

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

# Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# ▶ First described by Greibach (1964)

- Maintains a list of *possible* derivations ....
- ... which is kept in memory.

Breadth-first

- Increases size of the list with every prediction operation
  - For every non-terminal *all* possible derivations are added.
  - A LL parser predicts until every left-hand symbol is a terminal.

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

Decreases size of the list with every matching operation

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

### Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# ▶ First described by Greibach (1964)

- Maintains a list of *possible* derivations ....
- ... which is kept in memory.

Breadth-first

- Increases size of the list with every prediction operation
  - For every non-terminal *all* possible derivations are added.
  - A LL parser predicts until every left-hand symbol is a terminal.
- Decreases size of the list with every matching operation
  - When a terminal does not match the input, the tree (prediction stack) is discarded.

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

### Breadth-first

Depth-first Left recursion Recursive Descent

# Summary

- First described by Greibach (1964)
- Maintains a list of *possible* derivations ....
- ... which is kept in memory.

Breadth-first

- Increases size of the list with every prediction operation
  - For every non-terminal *all* possible derivations are added.
  - A LL parser predicts until every left-hand symbol is a terminal.
- Decreases size of the list with every matching operation
  - When a terminal does not match the input, the tree (prediction stack) is discarded.

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

When do we stop parsing and accept?

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

# Breadth-first

Depth-first Left recursion Recursive Descent

# Summary

- First described by Greibach (1964)
- Maintains a list of *possible* derivations ....
- ... which is kept in memory.

Breadth-first

- Increases size of the list with every prediction operation
  - For every non-terminal *all* possible derivations are added.
  - A LL parser predicts until every left-hand symbol is a terminal.
- Decreases size of the list with every matching operation
  - When a terminal does not match the input, the tree (prediction stack) is discarded.
- When do we stop parsing and accept?
- We introduce the end-of-input marker #

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

# Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

# Breadth-first

Depth-first Left recursion Recursive Descent

# Summary

- First described by Greibach (1964)
- Maintains a list of *possible* derivations ....
- ... which is kept in memory.

Breadth-first

- Increases size of the list with every prediction operation
  - For every non-terminal *all* possible derivations are added.
  - A LL parser predicts until every left-hand symbol is a terminal.
- Decreases size of the list with every matching operation
  - When a terminal does not match the input, the tree (prediction stack) is discarded.
- When do we stop parsing and accept?
- We introduce the end-of-input marker #

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF

#### Breadth-first

Depth-first Left recursion Recursive Descent

#### Summary

# Advantages and Shortcomings

# Shortcomings

# Can be very efficient on time resources

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

# Implementation

PDA + GNF

#### Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# Advantages and Shortcomings

# Shortcomings

Can be very efficient on time resources

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Can parse in real time

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

### Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# Advantages and Shortcomings

# Shortcomings

- Can be very efficient on time resources
- Can parse in real time
- Always finds the best prediction (if written adequately)

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

### Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# Advantages and Shortcomings

# Shortcomings

- Can be very efficient on time resources
- Can parse in real time
- Always finds the best prediction (if written adequately)

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

Memory usage increases exponentially

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

# Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# Advantages and Shortcomings

# Shortcomings

- Can be very efficient on time resources
- Can parse in real time
- Always finds the best prediction (if written adequately)
- Memory usage increases exponentially
  - This can be reduced by Dynamic Programming techniques

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

► Suffers from LEFT RECURSION

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF

# Breadth-first

Depth-first Left recursion Recursive Descent

Summary

# Advantages and Shortcomings

# Shortcomings

- Can be very efficient on time resources
- Can parse in real time
- Always finds the best prediction (if written adequately)
- Memory usage increases exponentially
  - This can be reduced by Dynamic Programming techniques

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

► Suffers from LEFT RECURSION

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first **Depth-first** Left recursion Recursive Desce

#### Summary

# Depth-first/Backtracking

# Predicts until it hits a terminal

< □ > < 同 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Desce

Summary

# ${\sf Depth-first}/{\sf Backtracking}$

Predicts until it hits a terminal

< □ > < 同 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

If the terminal . . .

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first **Depth-first** Left recursion Recursive Descent

Summary

# Depth-first/Backtracking

- Predicts until it hits a terminal
- If the terminal . . .

matches: Success, go to the next higher branch and continue

< □ > < 同 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

# ${\sf Depth-first}/{\sf Backtracking}$

- Predicts until it hits a terminal
- If the terminal . . .

matches: Success, go to the next higher branch and continue !matches: pop everything from the stack to the next higher branch

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first **Depth-first** Left recursion Recursive Descer

Summary

# Depth-first/Backtracking

- Predicts until it hits a terminal
- If the terminal ...

matches: Success, go to the next higher branch and continue !matches: pop everything from the stack to the next higher branch

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Accepts when it hits the end-of-input symbol #

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first **Depth-first** Left recursion Recursive Descer

Summary

# Depth-first/Backtracking

- Predicts until it hits a terminal
- If the terminal ...

matches: Success, go to the next higher branch and continue !matches: pop everything from the stack to the next higher branch

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Accepts when it hits the end-of-input symbol #

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

#### Summary

# Advantages and Shortcomings

# Does not use that much memory

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

#### Summary

# Advantages and Shortcomings

Does not use that much memory

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Is easier to handle

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Desce

Summary

# Advantages and Shortcomings

- Does not use that much memory
- Is easier to handle
- Performance can be increased by using statistical parsing methods

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first **Depth-first** Left recursion Recursive Desc

Summary

# Advantages and Shortcomings

- Does not use that much memory
- Is easier to handle
- Performance can be increased by using statistical parsing methods

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Generally disallows for on-line parsing

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Desc

Summary

# Advantages and Shortcomings

- Does not use that much memory
- Is easier to handle
- Performance can be increased by using statistical parsing methods
- Generally disallows for on-line parsing
- Is rather slow (can take up to exponential time)

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first **Depth-first** Left recursion Recursive Desc

Summary

# Advantages and Shortcomings

- Does not use that much memory
- Is easier to handle
- Performance can be increased by using statistical parsing methods
- Generally disallows for on-line parsing
- Is rather slow (can take up to exponential time)

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

 The "accept first match" policy can lead to undergeneration

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first **Depth-first** Left recursion Recursive Desc

Summary

# Advantages and Shortcomings

- Does not use that much memory
- Is easier to handle
- Performance can be increased by using statistical parsing methods
- Generally disallows for on-line parsing
- Is rather slow (can take up to exponential time)

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

- The "accept first match" policy can lead to undergeneration
- Also suffers from left recursion

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first **Depth-first** Left recursion Recursive Desc

Summary

# Advantages and Shortcomings

- Does not use that much memory
- Is easier to handle
- Performance can be increased by using statistical parsing methods
- Generally disallows for on-line parsing
- Is rather slow (can take up to exponential time)

◆□▶ ◆□▶ ◆□▶ ◆□▶ → □ ● ○○○

- The "accept first match" policy can lead to undergeneration
- Also suffers from left recursion

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive exampl Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# Left recursion

An example grammar:  $S \rightarrow DP VP$  $DP \rightarrow D NP$  $NP \rightarrow N \mid AP \mid NP \mid NP \mid PP$  $VP \rightarrow V DP \mid V PP \mid VP PP$  $PP \rightarrow P DP$  $AP \rightarrow A$  $D \rightarrow der \mid die \mid das \mid den \mid dem$  $N \rightarrow Fernglas | Frau | Mann | Mond | Wiese$  $V \rightarrow scheint \mid sieht$  $A \rightarrow kleine \mid kleinen \mid grosse \mid grossen$  $P \rightarrow auf \mid mit$ 

An example derivation for the sentence
"Der Mann sieht die Frau mit dem Fernglas" ...

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Another derivation

that is quite problematic

Just following the grammar ...
Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Another derivation

that is quite problematic

Just following the grammar ...

S

◆□ > ◆□ > ◆臣 > ◆臣 > ○臣 ○ のへ⊙

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Another derivation

## that is quite problematic

Just following the grammar ...



Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Another derivation

## that is quite problematic





Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Another derivation

## that is quite problematic



Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# Another derivation

## that is quite problematic



・ロト ・ 雪 ト ・ ヨ ト

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# Another derivation

## that is quite problematic



・ロト ・ 雪 ト ・ ヨ ト

3

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Another derivation

## that is quite problematic





(日) (雪) (日) (日) (日)

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Another derivation

# that is quite problematic





(日) (雪) (日) (日) (日)

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# Another derivation

## that is quite problematic





Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Another derivation

## that is quite problematic



#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Left Recursion ctd.

# ▶ ... we will get a LEFT RECURSION

▲ロト ▲帰 ト ▲ ヨ ト ▲ ヨ ト ● の Q ()

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# Left Recursion ctd.

- ▶ ... we will get a LEFT RECURSION
- LEFT RECURSIONS can generate an infinite deal of garbage unless stopped from doing so

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# ▶ ... we will get a LEFT RECURSION

Left Recursion ctd

► LEFT RECURSIONS can generate an infinite deal of garbage unless stopped from doing so

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

▶ There are **two types** of left-recursion:

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# ▶ ... we will get a LEFT RECURSION

 LEFT RECURSIONS can generate an infinite deal of garbage unless stopped from doing so

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

- ► There are **two types** of left-recursion:
  - Direct left-recursion

Left Recursion ctd

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# ▶ ... we will get a LEFT RECURSION

 LEFT RECURSIONS can generate an infinite deal of garbage unless stopped from doing so

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

- ► There are **two types** of left-recursion:
  - Direct left-recursion

Example: NP  $\rightarrow$  NP PP

Left Recursion ctd

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# ▶ ... we will get a LEFT RECURSION

 LEFT RECURSIONS can generate an infinite deal of garbage unless stopped from doing so

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

- ► There are **two types** of left-recursion:
  - Direct left-recursion

Example:  $NP \rightarrow NP PP$ 

Left Recursion ctd

Indirect left-recursion

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# ▶ ... we will get a LEFT RECURSION

 LEFT RECURSIONS can generate an infinite deal of garbage unless stopped from doing so

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

- ▶ There are **two types** of left-recursion:
  - Direct left-recursion

Example: NP  $\rightarrow$  NP PP

Left Recursion ctd

Indirect left-recursion

Example:  $S \rightarrow NP VP$  $VP \rightarrow V' AP$  $V' \rightarrow VP PP$ 

Workarounds

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Keep track of the count of processed rules

▲ロト ▲帰 ト ▲ ヨ ト ▲ ヨ ト ● の Q ()

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Workarounds

# Keep track of the count of processed rules

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

 $\blacktriangleright$   $\rightarrow$  does not allow on-line parsing

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# Workarounds

Keep track of the count of processed rules

- $\blacktriangleright$   $\rightarrow$  does not allow on-line parsing
- Rewrite the grammar
  - No  $\epsilon$  and unit-rules

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# Workarounds

- Keep track of the count of processed rules
  - $\blacktriangleright$   $\rightarrow$  does not allow on-line parsing
- Rewrite the grammar
  - No 
    e
     and unit-rules
  - Split the *direct* left-recursive rules up:

- $\begin{array}{ll} \blacktriangleright & \mbox{We start with} \\ & \mbox{NP} \rightarrow \mbox{NP PP} \mid \mbox{N} \end{array}$
- And transform into:
  - $\begin{array}{l} \mathsf{N}' \rightarrow \mathsf{N} \\ \mathsf{N}'' \rightarrow \mathsf{PP} \\ \mathsf{N}''' \rightarrow \mathsf{N}'' \; \mathsf{N}''' \mid \mathsf{N}'' \\ \mathsf{NP} \rightarrow \mathsf{N}' \; \mathsf{N}''' \mid \mathsf{N}' \end{array}$

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Our example revised

Let us have a look at the previous grammar

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

```
Implementation
```

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summarv

# Our example revised

• Let us have a look at the previous grammar

```
\mathsf{S} \to \mathsf{DP} \; \mathsf{VP}
```

```
\mathsf{DP} \to \mathsf{D} \; \mathsf{NP}
```

```
\mathsf{NP} \to \mathsf{N} \mid \mathsf{AP} \; \mathsf{NP} \mid \mathsf{NP} \; \mathsf{PP}
```

```
\mathsf{VP} \to \mathsf{V} \; \mathsf{DP} \; | \; \mathsf{VPP} \; | \; \mathsf{VP} \; \mathsf{PP}
```

```
\mathsf{PP} \to \mathsf{P} \; \mathsf{DP}
```

```
\mathsf{AP}\to\mathsf{A}
```

```
\mathsf{D} \to \mathsf{der} \mid \mathsf{die} \mid \mathsf{das} \mid \mathsf{den} \mid \mathsf{dem}
```

```
N \rightarrow Fernglas \mid Frau \mid Mann \mid Mond \mid Wiese
```

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

```
V \to \mathsf{scheint} \ | \ \mathsf{sieht}
```

 $\mathsf{A} \to \mathsf{kleine} \mid \mathsf{kleinen} \mid \mathsf{grosse} \mid \mathsf{grossen}$ 

 $\mathsf{P} \to \mathsf{auf} \mid \mathsf{mit}$ 

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Becursive Descent

Summary

# Our example revised

• Let us have a look at the previous grammar

```
\mathsf{S}\to\mathsf{DP}\;\mathsf{VP}
```

```
\mathsf{DP}\to\mathsf{D}\;\mathsf{NP}
```

```
NP \rightarrow N \mid AP \mid NP \mid NP \mid PP
```

```
\mathsf{VP} \to \mathsf{V} \; \mathsf{DP} \; | \; \mathsf{V} \; \mathsf{PP} \; | \; \mathsf{VP} \; \mathsf{PP}
```

```
\mathsf{PP} \to \mathsf{P} \; \mathsf{DP}
```

```
\mathsf{AP}\to\mathsf{A}
```

```
\mathsf{D} \to \mathsf{der} \mid \mathsf{die} \mid \mathsf{das} \mid \mathsf{den} \mid \mathsf{dem}
```

 $N \rightarrow Fernglas \mid Frau \mid Mann \mid Mond \mid Wiese$ 

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

```
V \to \mathsf{scheint} \ \big| \ \mathsf{sieht}
```

 $\mathsf{A} \to \mathsf{kleine} \mid \mathsf{kleinen} \mid \mathsf{grosse} \mid \mathsf{grossen}$ 

 $\mathsf{P} \to \mathsf{auf} \mid \mathsf{mit}$ 

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Our example revised

• Let us have a look at the previous grammar

```
\mathsf{S} \to \mathsf{DP} \; \mathsf{VP}
```

```
\mathsf{DP} \to \mathsf{D} \; \mathsf{NP}
```

```
\mathsf{NP} \to \mathsf{N} \mid \mathsf{AP} \; \mathsf{NP} \mid \mathsf{NP} \; \mathsf{PP}
```

```
\mathsf{VP} \to \mathsf{V} \; \mathsf{DP} \; | \; \mathsf{VPP} \; | \; \mathsf{VP} \; \mathsf{PP}
```

```
\mathsf{PP} \to \mathsf{P} \; \mathsf{DP}
```

```
\mathsf{AP}\to\mathsf{A}
```

```
\mathsf{D} \to \mathsf{der} \mid \mathsf{die} \mid \mathsf{das} \mid \mathsf{den} \mid \mathsf{dem}
```

```
N \rightarrow Fernglas \mid Frau \mid Mann \mid Mond \mid Wiese
```

```
V \to \mathsf{scheint} \mid \mathsf{sieht}
```

 $\mathsf{A} \to \mathsf{kleine} \mid \mathsf{kleinen} \mid \mathsf{grosse} \mid \mathsf{grossen}$ 

 $\mathsf{P} \to \mathsf{auf} \mid \mathsf{mit}$ 

This will now be taken care of ...

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

#### Summary

# Let us have a look at the previous grammar

▲ロト ▲帰 ト ▲ ヨ ト ▲ ヨ ト ● の Q ()

► Here are the revised rules:

Our example revised

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

WOTIVATION

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summary

# Our example revised

Let us have a look at the previous grammar

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

• Here are the revised rules:

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

Implementation

PDA + GNF Breadth-first Depth-first Left recursion

Recursive Descent

Summarv

# Our example revised

Let us have a look at the previous grammar

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

• Here are the revised rules:

VP → V DP | V PP | VP PP Vh → V DP | V PP Vt → PP Vts → Vt Vts | Vt VP → Vh Vts | Vh
NP → N | AP NP | NP PP Nh → AP NP | N Nt → PP Nts → Nt Nts | Nt NP → Nh Nts | Nh

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down Why Not?

Implementation

PDA + GNF Breadth-first Depth-first

Left recursion Recursive Descent

# Our example revised

- Let us have a look at the previous grammar
- Here are the revised rules:

```
VP → V DP | V PP | VP PP
Vh → V DP | V PP
Vt → PP
Vts → Vt Vts | Vt
VP → Vh Vts | Vh
NP → N | AP NP | NP PP
Nh → AP NP | N
Nt → PP
Nts → Nt Nts | Nt
NP → Nh Nts | Nh
```

 This is about three times faster than the first workaround.

#### Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

#### Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion **Recursive Descent** 

#### Summary

# Grammars can be viewed as describing a program



# **Recursive Descent**

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# **Recursive Descent**

• Grammars can be viewed as describing a *program* 

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

 How to implement a grammar in our favorite programming language?
 One could . . .

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion **Recursive Descent** 

Summary

# Recursive Descent

- Grammars can be viewed as describing a *program*
- How to implement a grammar in our favorite programming language?
   One could ....

Automata: ... try to emulate an *automaton* that describes the language.

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion **Recursive Descent** 

Summary

# Recursive Descent

- Grammars can be viewed as describing a *program*
- How to implement a grammar in our favorite programming language?
   One could ....

Automata: ... try to emulate an *automaton* that describes the language.

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

This is not very flexible.

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion **Recursive Descent** 

Summary

# Recursive Descent

- Grammars can be viewed as describing a *program*
- How to implement a grammar in our favorite programming language?
   One could ....

Automata: ... try to emulate an *automaton* that describes the language.

- This is not very flexible.
- May be inefficient when dealing with more complicated tasks.

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Recursive Descent

- Grammars can be viewed as describing a *program*
- How to implement a grammar in our favorite programming language?
   One could ....

Automata: ... try to emulate an *automaton* that describes the language.

- This is not very flexible.
- May be inefficient when dealing with more complicated tasks.

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Automation: ... use a parser-generator (also: a compiler-compiler)

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

#### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

# Recursive Descent

- Grammars can be viewed as describing a *program*
- How to implement a grammar in our favorite programming language?
   One could ....

Automata: ... try to emulate an *automaton* that describes the language.

- This is not very flexible.
- May be inefficient when dealing with more complicated tasks.

Automation: ... use a parser-generator (also: a compiler-compiler)

 May not cover our favorite programming language (or its latest version)
Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent

- Grammars can be viewed as describing a *program*
- How to implement a grammar in our favorite programming language?
  One could ....

Automata: ... try to emulate an *automaton* that describes the language.

- This is not very flexible.
- May be inefficient when dealing with more complicated tasks.

Automation: ... use a parser-generator (also: a compiler-compiler)

 May not cover our favorite programming language (or its latest version)

Could be less efficient

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent

- Grammars can be viewed as describing a *program*
- How to implement a grammar in our favorite programming language?
  One could ....

Automata: ... try to emulate an *automaton* that describes the language.

- This is not very flexible.
- May be inefficient when dealing with more complicated tasks.

Automation: ... use a parser-generator (also: a compiler-compiler)

 May not cover our favorite programming language (or its latest version)

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Could be less efficient

Non-auto\*: ... write a parser for every grammar at hand.

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent

- Grammars can be viewed as describing a *program*
- How to implement a grammar in our favorite programming language?
  One could ....

Automata: ... try to emulate an *automaton* that describes the language.

- This is not very flexible.
- May be inefficient when dealing with more complicated tasks.

Automation: ... use a parser-generator (also: a compiler-compiler)

 May not cover our favorite programming language (or its latest version)

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Could be less efficient

Non-auto\*: ... write a parser for every grammar at hand.

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

### Such parsers are implicitly deapth-first

Recursive Descent: HOWTO

1. Make up a function for each left-hand side

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent: HOWTO

- Such parsers are implicitly deapth-first
- 1. Make up a function for each left-hand side
  - ► The function body represents the right hand side

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent: HOWTO

- Such parsers are implicitly deapth-first
- - The function body represents the right hand side
  - All functions have to return true for the parse to succeed

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent: HOWTO

- Such parsers are implicitly deapth-first
- 1. Make up a function for each left-hand side
  - The function body represents the right hand side
  - All functions have to return true for the parse to succeed

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

2. Maintain the input as a global variable with a global pointer

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent: HOWTO

- Such parsers are implicitly deapth-first
- $1. \ {\rm Make} \ {\rm up} \ {\rm a}$  function for each left-hand side
  - The function body represents the right hand side
  - All functions have to return true for the parse to succeed
- 2. Maintain the input as a global variable with a global pointer

くしゃ 本面 アメヨア メヨア しゅうぐう

3. Have the methods call each other *recursively* 

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent: HOWTO

- Such parsers are implicitly deapth-first
- 1. Make up a function for each left-hand side
  - The function body represents the right hand side
  - All functions have to return true for the parse to succeed
- 2. Maintain the input as a global variable with a global pointer

- 3. Have the methods call each other *recursively*
- 4. The base case is when a rule hits a terminal

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent: HOWTO

- Such parsers are implicitly deapth-first
- 1. Make up a function for each left-hand side
  - The function body represents the right hand side
  - All functions have to return true for the parse to succeed
- 2. Maintain the input as a global variable with a global pointer

- 3. Have the methods call each other recursively
- 4. The base case is when a rule hits a terminal
- 5. Every rule must contain it's own *pointer* to the position it points to in the input sentence

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent: HOWTO

- Such parsers are implicitly deapth-first
- 1. Make up a function for each left-hand side
  - The function body represents the right hand side
  - All functions have to return true for the parse to succeed
- 2. Maintain the input as a global variable with a global pointer
- 3. Have the methods call each other recursively
- 4. The base case is when a rule hits a terminal
- 5. Every rule must contain it's own *pointer* to the position it points to in the input sentence
- 6. Maintain a stack of *matched predictions* to tell the derivation

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Recursive Descent: HOWTO

- Such parsers are implicitly deapth-first
- 1. Make up a function for each left-hand side
  - The function body represents the right hand side
  - All functions have to return true for the parse to succeed
- 2. Maintain the input as a global variable with a global pointer
- 3. Have the methods call each other recursively
- 4. The base case is when a rule hits a terminal
- 5. Every rule must contain it's own *pointer* to the position it points to in the input sentence
- 6. Maintain a stack of *matched predictions* to tell the derivation

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

## Certain Caveats

### Beware of left-recursion

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへで

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion **Recursive Descent** 

### Summary

Beware of left-recursion

Certain Caveats

Danger of undergeneration

▲ロト ▲帰 ト ▲ ヨ ト ▲ ヨ ト ● の Q ()

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

### Beware of left-recursion

Certain Caveats

- Danger of undergeneration
  - Works only for prefix-free grammars

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

## Certain Caveats

- Beware of left-recursion
- Danger of undergeneration
  - Works only for prefix-free grammars if

$$\mathsf{A} \to^* x$$
 and  $\mathsf{A} \to^* xy$ 

- this implies  $y = \epsilon$
- So one has to find a workaround for that either

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

## Certain Caveats

- Beware of left-recursion
- Danger of undergeneration
  - Works only for prefix-free grammars if

$$\mathsf{A} \to^* x$$
 and  $\mathsf{A} \to^* xy$ 

- this implies  $y = \epsilon$
- So one has to find a workaround for that either

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

Be a little depth first

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

## Summary

### LL top-down parsers can be used for on-line parsing

▲ロト ▲周ト ▲ヨト ▲ヨト ヨー のなべ

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

### Summary

# Summary

► LL top-down parsers can be used for on-line parsing

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

They are intuitive and generally quite fast

Johannes Dellert, Aleksandar Dimitrov

#### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Summary

- ► LL top-down parsers can be used for on-line parsing
- They are intuitive and generally quite fast
  - But the Breadth first parser is a memory hog

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descent

Summary

## Summary

- ► LL top-down parsers can be used for on-line parsing
- They are intuitive and generally quite fast
  - But the Breadth first parser is a memory hog

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

while the Depth first parser is too slow

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descen

### Summary

## Summary

- LL top-down parsers can be used for on-line parsing
- They are intuitive and generally quite fast
  - But the Breadth first parser is a memory hog
  - while the Depth first parser is too slow
- Greibach Normal Form allows for comfortable parsing

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

## Summary

- LL top-down parsers can be used for on-line parsing
- They are intuitive and generally quite fast
  - But the Breadth first parser is a memory hog
  - while the Depth first parser is too slow
- Greibach Normal Form allows for comfortable parsing

▲ロト ▲帰 ト ▲ヨト ▲ヨト - ヨ - の々ぐ

GNF is for top-down what CNF is for bottom-up

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

## Summary

- LL top-down parsers can be used for on-line parsing
- They are intuitive and generally quite fast
  - But the Breadth first parser is a memory hog
  - while the Depth first parser is too slow
- Greibach Normal Form allows for comfortable parsing
- GNF is for top-down what CNF is for bottom-up
- Recursive descent allows for easy implementation of a backtracking parser

Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

Summary

## Summary

- LL top-down parsers can be used for on-line parsing
- They are intuitive and generally quite fast
  - But the Breadth first parser is a memory hog
  - while the Depth first parser is too slow
- Greibach Normal Form allows for comfortable parsing
- GNF is for top-down what CNF is for bottom-up
- Recursive descent allows for easy implementation of a backtracking parser

### Johannes Dellert, Aleksandar Dimitrov

### Introduction

Intuitive example Features and Operations Parsing Schema

### Motivation

Why Top-Down? Why Not?

### Implementation

PDA + GNF Breadth-first Depth-first Left recursion Recursive Descer

### Summary

Thanks a lot.

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 三臣 - のへぐ