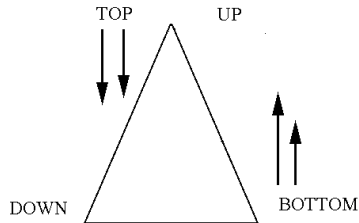


Top-down Parsing

Brian Malloy
Clemson University
School of Computing



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



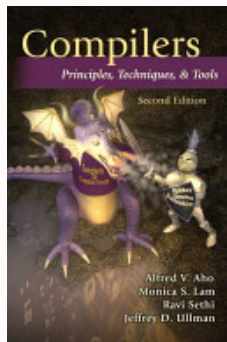
Slide 1 of 24

Go Back

Full Screen

Quit

1. Resources



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



Slide 2 of 24

Go Back

Full Screen

Quit

2. Introduction

- The goal of parsing is not the generation of sentences, but the recognition of them.
- This implies that the generating steps that lead to the construction must be deduced from the finished sentence.
- top-down is simplest parsing technique
- Simple implementations of top-down do not terminate in the presence of left recursion
- Tow-down with backtracking may have exponential complexity
- e.g.: recursive descent, and LL(k)



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



Slide 3 of 24

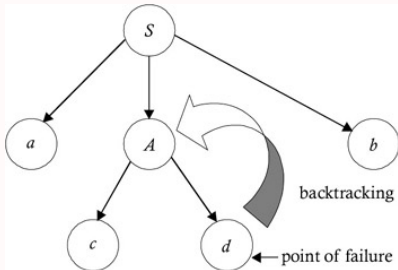
Go Back

Full Screen

Quit

2.1. How it works

- Start with **start** symbol
- Try to regenerate the sentence by applying productions.
- Determine production by looking at next terminal in sentence.



Slide 4 of 24

Go Back

Full Screen

Quit

2.2. Intuition

- Try to find a leftmost derivation,
- by searching for parse trees,
- using a top-down expansion of the grammar rules!
- Tokens are consumed from left to right.



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 5 of 24

Go Back

Full Screen

Quit



3. Recursive Descent Parsing

- Def: Top-down approach to syntax analysis where a set of recursive procedures is used to process the input.
- One procedure is associated with each non-terminal
- **Predictive** recursive descent: the look-ahead symbol must unambiguously determine the flow of control through the procedure body.
- The body of the procedure mimics the body of the chosen non-terminal.
- Left recursive grammars lead to infinite recursion and must be transformed

Resources

Introduction

Recursive Descent ...

Example: Balanced ...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table- ...



Slide 6 of 24

Go Back

Full Screen

Quit



4. Example: Balanced Parens

Consider the following grammar:

```
S : '(' S ')' S
```

One non-terminal in grammar \Rightarrow one function:

```
void S() const {  
    if ( lookahead == '(' ) {  
        match('(');  
        S();  
        match(')');  
        S();  
    }  
}
```

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



Slide 7 of 24

Go Back

Full Screen

Quit



5. Example:

Infix to Postfix translator

We can use the expression grammar:

```
expr  : expr '+' term
      | expr '-' term
      | term
term   | DIGIT
```

Except that left recursion \Rightarrow infinit loop!

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 8 of 24

Go Back

Full Screen

Quit



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...

5.1. Removal of Left Recursion

Given a grammar with left recursion:

$$\begin{array}{lcl} A & : & A \alpha \\ & | & A \beta \\ & | & \gamma \end{array}$$

We get the following with left recursion removed:

$$\begin{array}{lcl} A & : & \gamma R \\ R & : & \alpha R \\ & | & \beta R \\ & | & \lambda \end{array}$$


Slide 9 of 24

Go Back

Full Screen

Quit

5.2. Transformed Expression Grammar without left recursion

```
expr  : term rest
rest  : '+' term rest
      | '-' term rest
      |  $\lambda$ 
term  : DIGIT
```



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 10 of 24

Go Back

Full Screen

Quit

5.3. Transformed expression grammar with semantic actions

```
expr  : term rest

rest  : '+' term { print('+'); } rest
      | '-' term { print('-'); } rest
      |  $\lambda$ 

term  : DIGIT { print(DIGIT); }
```



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 11 of 24

Go Back

Full Screen

Quit



5.4. Pseudo-code for translator

```
void expr() {
    term(); rest();
}

void rest() {
    if (lookahead == '+') {
        match('+'); term(); print('+'); rest();
    }
    else if (lookahead == '-') {
        match('-'); term(); print('-'); rest();
    }
    else {} // do nothing
}

void term() {
    if (lookahead is a digit) {
        t = lookahead; match(lookahead); print(t);
    }
    else report("syntax error");
}
```

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 12 of 24

Go Back

Full Screen

Quit

6. Table-driven Parsing

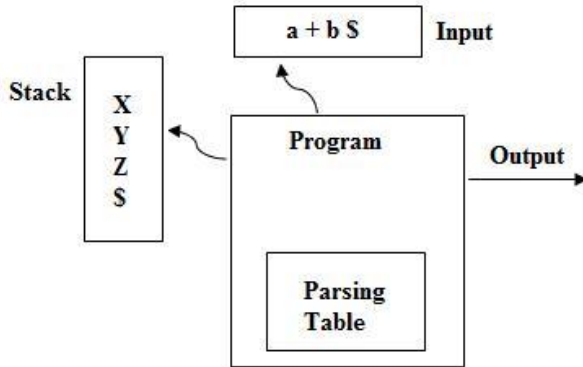


Fig 2.6 Model of a predictive parser

Resources

Introduction

Recursive Descent ...

Example: Balanced ...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table- ...



Slide 13 of 24

Go Back

Full Screen

Quit



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...

6.1. Writing a Grammar

1. Grammars can describe most, but not **all** of the syntax.
2. For example: CFG cannot specify that an ID must be declared before it's used.
3. Tokens accepted by parser form a **super-set** of the language.
4. Subsequent phases must analyze parser output to ensure compliance of rules not checked by parser (p. 209 of the dragon book).



Slide 14 of 24

Go Back

Full Screen

Quit



6.2. Re-Writing a Grammar

- Some grammars are not in the proper form for a given parsing algorithm.
- For example, some transformations are needed to enable top-down parsing:
 - Left factor
 - Left recursion
 - Ambiguity

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 15 of 24

Go Back

Full Screen

Quit



6.3. Consider expression grammar

$$E : E '+' E \mid E '*' E \mid (E) \mid ID$$

Eliminate ambiguity:

$$\begin{aligned} E &: E '+' T \mid T \\ T &: T '*' F \mid F \\ F &: (E) \mid ID \end{aligned}$$

Eliminate left recursion to get G' :

$$\begin{aligned} E &: T E' \\ E' &: '+' T E' \mid \epsilon \\ T &: F T' \\ T' &: '*' F T' \mid \epsilon \\ F &: (E) \mid ID \end{aligned}$$

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



Slide 16 of 24

Go Back

Full Screen

Quit



7. First and Follow

1. Useful for construction of both Top-down and Bottom-up parsers
2. The First and Follow is associated with a grammar G
3. In Top-down parsing, First and Follow help decide which production to apply, based on next input.
4. During recovery, sets of tokens in Follow can dictate how far ahead to skip.

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 17 of 24

Go Back

Full Screen

Quit



7.1. How First and Follow predict

- Consider production $A \Rightarrow \alpha \mid \beta$, where $\text{FIRST}(\alpha)$ and $\text{FIRST}(\beta)$ are disjoint sets.
- We can choose rhs of A by looking at next input symbol a , since a can be \in either $\text{FIRST}(\alpha)$ or $\text{FIRST}(\beta)$, but not both.

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



Slide 18 of 24

Go Back

Full Screen

Quit



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



Slide 19 of 24

Go Back

Full Screen

Quit

7.2. To compute FIRST(X)

Apply the following rules until no more terminals or ϵ can be added to any FIRST set:

1. If X is a terminal, then $\text{FIRST}(X) = \{X\}$.
2. If X is a non-terminal and $X \Rightarrow Y_1 Y_2 \dots Y_k$ is a production for some $k \geq 1$, then place a in $\text{FIRST}(X)$ if for some i , $a \in \text{FIRST}(Y_i)$, and ϵ is in all of $\text{FIRST}(Y_1), \dots, \text{FIRST}(Y_{i-1})$;
In other words: $Y_1 \dots Y_{i-1} \Rightarrow \epsilon$.
 - (a) If $\epsilon \in \text{FIRST}(Y_j)$ for all $j=1, 2, \dots, k$ then add ϵ to $\text{FIRST}(X)$.
 - (b) E.g., everything in $\text{FIRST}(Y_1)$ is $\in \text{FIRST}(X)$; if Y_1 does not derive ϵ , add no more.
3. If $X \rightarrow \epsilon$ is a production, add ϵ to $\text{FIRST}(X)$



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



Slide 20 of 24

Go Back

Full Screen

Quit

7.3. To compute FOLLOW(A)

For all non-terminals A, apply the following rules until nothing can be added to any FOLLOW sets:

1. Place \$ in FOLLOW(S), where S is start symbol and \$ is end marker.
2. If there is a production $A \rightarrow \alpha B \beta$, then everything in FIRST(β) except ϵ is in FOLLOW(B).
3. If there is a production $A \rightarrow \alpha B$, or a production $A \rightarrow \alpha B \beta$, where FIRST(β) contains ϵ , then everything in FOLLOW(A) is in FOLLOW(B).



Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...

7.4. Compute First and Follow for G'

1. $\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{ '(', \text{ID} \}$
2. $\text{FIRST}(E') = \{ '+', \epsilon \}$
3. $\text{FIRST}(T') = \{ '*', \epsilon \}$
4. $\text{FOLLOW}(E) = \text{FOLLOW}(E') = \{ ')', \$ \}$
5. $\text{FOLLOW}(T) = \text{FOLLOW}(T') = \{ '+', ')', \$ \}$
6. $\text{FOLLOW}(F) = \{ '+', '*', ')', \$ \}$



Slide 21 of 24

Go Back

Full Screen

Quit



7.5. Construction of parse table

- INPUT: Grammar G
- OUTPUT: Parse table M
- METHOD: For each production $A \rightarrow \alpha$:
 1. For each terminal α in $\text{First}(A)$, add $A \rightarrow \alpha$ to $M[A, a]$.
 2. If $\epsilon \in \text{FIRST}(\alpha)$, then for each terminal b in $\text{Follow}(A)$, add $A \rightarrow \alpha$ to $M[A, b]$.
If $\epsilon \in \text{FIRST}(\alpha)$ and $\$ \in \text{FOLLOW}(A)$, then add $A \rightarrow \alpha$ to $M[A, \$]$.
- If after performing above, there is no production at all in $M[A, \alpha]$, then set $M[A, \alpha]$ to error (empty entry in table)

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table...



Slide 22 of 24

Go Back

Full Screen

Quit



8. Algorithm for table-driven Parser

- **INPUT:** a string w , and a parse table M for grammar G .
- **OUTPUT:** if $w \in L(G)$, a leftmost derivation of w , or *error*.
- **METHOD:** Initially, the parser is in a configuration with $w\$$ in the input buffer and the start symbol S on the top of the stack, above $\$$.

The algorithm on next page uses table M to produce a predictive parse for the input.

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 23 of 24

Go Back

Full Screen

Quit



8.1. Pseudo-code for table-driven parser

```
set ip to point to first symbol w;  
set X to the top stack symbol;  
while ( X != $ ) { // stack is not empty  
    if ( X is a ) pop the stack and advance ip;  
    else if ( X is a terminal ) error();  
    else if ( M[X, a] is an error entry ) error();  
    else if ( M[X, a] = X→Y1 Y2...Yk ) {  
        output the production X→Y1 Y2...Yk;  
        pop the stack;  
        push Yk, Yk-1, ..., Y1 onto stack; Y1 on top;  
    }  
    set X to the top stack symbol;  
}
```

Resources

Introduction

Recursive Descent...

Example: Balanced...

Example: ...

Table-driven Parsing

First and Follow

Algorithm for table-...



Slide 24 of 24

Go Back

Full Screen

Quit