

Section 4.3: A Parsing Algorithm

In this section, we consider a simple, fairly inefficient parsing algorithm that works for all context-free grammars. Compilers courses cover efficient algorithms that work for various subsets of the context free grammars.

Copyright © 2003–5 Alley Stoughton

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

The L^AT_EX source of these slides, the associated book, and the distribution of the Forlan toolset are available on the WWW at <http://people.cis.ksu.edu/~stough/forlan/>.

1

(4.3) Introduction (Cont.)

The parsing algorithm takes in a grammar G and a string w , and attempts to find a minimally-sized parse tree pt such that:

- pt is valid for G ;
- $\text{rootLabel}(pt) = s_G$;
- $\text{yield}(pt) = w$.

If there is no such pt , then the algorithm reports failure.

2

(4.3) Determining Whether $w \in L(G)$

Let's start by considering an algorithm for checking whether $w \in L(G)$, for a string w and grammar G .

Let $A = Q_G \cup \text{alphabet}(w)$ and $B = \{x \in \text{Str} \mid x \text{ is a substring of } w\}$. We generate the least subset X of $A \times B$ such that:

- For all $a \in \text{alphabet}(w)$, $(a, a) \in X$;
- For all $q \in Q_G$, if $q \rightarrow \% \in P_G$, then $(q, \%) \in X$;
- For all $q \in Q_G$, $n \in \mathbb{N} - \{0\}$, $a_1, \dots, a_n \in A$ and $x_1, \dots, x_n \in B$, if
 - $q \rightarrow a_1 \cdots a_n \in P_G$,
 - for all $1 \leq i \leq n$, $(a_i, x_i) \in X$, and
 - $x_1 \cdots x_n \in B$,then $(q, x_1 \cdots x_n) \in X$.

Since $A \times B$ is finite, this process terminates.

3

(4.3) Determining Whether $w \in L(G)$ (Cont.)

For example, let G be the grammar

$$\begin{aligned}A &\rightarrow BC \mid CD, \\B &\rightarrow 0 \mid CB, \\C &\rightarrow 1 \mid DD, \\D &\rightarrow 0 \mid BC,\end{aligned}$$

and let $w = 0010$.

We have that:

- $(0, 0) \in X$;
- $(1, 1) \in X$;
- $(B, 0) \in X$, since $B \rightarrow 0 \in P_G$, $(0, 0) \in X$ and $0 \in B$;
- $(C, 1) \in X$, since $C \rightarrow 1 \in P_G$, $(1, 1) \in X$ and $1 \in B$;
- $(D, 0) \in X$, since $D \rightarrow 0 \in P_G$, $(0, 0) \in X$ and $0 \in B$;

4

(4.3) Determining Whether $w \in L(G)$ (Cont.)

- $(A, 01) \in X$, since $A \rightarrow BC \in P_G$, $(B, 0) \in X$, $(C, 1) \in X$ and $01 \in B$;
- $(A, 10) \in X$, since $A \rightarrow CD \in P_G$, $(C, 1) \in X$, $(D, 0) \in X$ and $10 \in B$;
- $(B, 10) \in X$, since $B \rightarrow CB \in P_G$, $(C, 1) \in X$, $(B, 0) \in X$ and $10 \in B$;
- $(C, 00) \in X$, since $C \rightarrow DD \in P_G$, $(D, 0) \in X$, $(D, 0) \in X$ and $00 \in B$;
- $(D, 01) \in X$, since $D \rightarrow BC \in P_G$, $(B, 0) \in X$, $(C, 1) \in X$ and $01 \in B$;

5

(4.3) Determining Whether $w \in L(G)$ (Cont.)

- $(C, 001) \in X$, since $C \rightarrow DD \in P_G$, $(D, 0) \in X$, $(D, 01) \in X$ and $0(01) \in B$;
- $(C, 010) \in X$, since $C \rightarrow DD \in P_G$, $(D, 01) \in X$, $(D, 0) \in X$ and $(01)0 \in B$;
- $(A, 0010) \in X$, since $A \rightarrow BC \in P_G$, $(B, 0) \in X$, $(C, 010) \in X$ and $0(010) \in B$;
- $(B, 0010) \in X$, since $B \rightarrow CB \in P_G$, $(C, 00) \in X$, $(B, 10) \in X$ and $(00)(10) \in B$;
- $(D, 0010) \in X$, since $D \rightarrow BC \in P_G$, $(B, 0) \in X$, $(C, 010) \in X$ and $0(010) \in B$;
- Nothing more can be added to X . To verify this, one must go through all the productions of G , checking that nothing can be added to X using a given production and the existing elements of X .

6

(4.3) Determining Whether $w \in L(G)$ (Cont.)

The following lemmas concerning X are easy to prove:

Lemma 4.3.1

For all $(a, x) \in X$, there is a $pt \in \mathbf{PT}$ such that

- pt is valid for G ,
- $\mathbf{rootLabel}(pt) = a$,
- $\mathbf{yield}(pt) = x$.

Lemma 4.3.2

For all $a \in A$ and $x \in B$, if there is a $pt \in \mathbf{PT}$ such that

- pt is valid for G ,
- $\mathbf{rootLabel}(pt) = a$,
- $\mathbf{yield}(pt) = x$,

then $(a, x) \in X$.

7

(4.3) Determining Whether $w \in L(G)$ (Cont.)

Thus, to determine if $w \in L(G)$, we just have to check whether $(s_G, w) \in X$.

In the case of our example grammar, we have that $w = 0010 \in L(G)$, since $(A, 0010) \in X$.

8

(4.3) A Parsing Algorithm

If we label each element (a, x) of our set X with a parse tree pt such that

- pt is valid for G ,
- $\text{rootLabel}(pt) = a$,
- $\text{yield}(pt) = x$,

then we can return the parse tree labeling (s_G, w) , if this pair is in X . Otherwise, we report failure.

With some more work, we can arrange that the parse trees returned by our parsing algorithm are minimally-sized, and this is what the official version of our parsing algorithm guarantees. This goal is a little tricky to achieve, since some pairs will first be labeled by parse trees that aren't minimally sized.

9

(4.3) Parsing in Forlan

The Forlan module `Gram` defines the functions

```
val parseStrFromSym : gram -> sym * str -> pt
val parseStr       : gram -> str -> pt
val generatedFromSym : gram -> sym * str -> bool
val generated      : gram -> str -> bool
```

The function `parseStrFromSym` implements our parsing algorithm. It takes in a grammar G , and returns a function that takes in a symbol/string pair (a, x) , and tries to find a minimally-sized parse tree that is valid for G , whose root label is a , and whose yield is x ; it issues an error message if this is impossible. The function `parseStr` works similarly, except that it looks for a parse tree whose root label is s_G .

The function `generatedFromSym` is like `parseStrFromSym` except that it returns `true` if a parse tree is found, and returns `false` (without issuing an error message) otherwise. And the function `generated` is similarly related to the function `parseStr`.

(4.3) Parsing in Forlan (Cont.)

Suppose that `gram` of type `gram` is bound to the grammar

$$\begin{aligned}A &\rightarrow BC \mid CD, \\B &\rightarrow 0 \mid CB, \\C &\rightarrow 1 \mid DD, \\D &\rightarrow 0 \mid BC.\end{aligned}$$

We can attempt to parse some strings according to this grammar, as follows.

11

(4.3) Forlan Parsing Examples

```
- fun test s =
=       PT.output("",
=           Gram.parseStr gram (Str.fromString s));
val test = fn : string -> unit
- test "0010";
A(B(0), C(D(B(0), C(1)), D(0)))
val it = () : unit
- test "0100";
A(C(D(B(0), C(1)), D(0)), D(0))
val it = () : unit
- test "0101";
no such parse exists

uncaught exception Error
- Gram.generated gram (Str.fromString "0100");
val it = true : bool
- Gram.generated gram (Str.fromString "0101");
val it = false : bool
```

12