

Section 4.6: Ambiguity of Grammars

In this section, we say what it means for a grammar to be ambiguous. We also consider a straightforward method for disambiguating some commonly occurring grammars.

Copyright © 2003–4 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 \LaTeX 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/>.

(4.6) Introduction

Suppose G is our grammar of arithmetic expressions:

$$E \rightarrow E\langle\text{plus}\rangle E \mid E\langle\text{times}\rangle E \mid \langle\text{openPar}\rangle E \langle\text{closPar}\rangle \mid \langle\text{id}\rangle.$$

Question: are there multiple ways of parsing the string $\langle\text{id}\rangle\langle\text{times}\rangle\langle\text{id}\rangle\langle\text{plus}\rangle\langle\text{id}\rangle$ according to this grammar?

Answer:

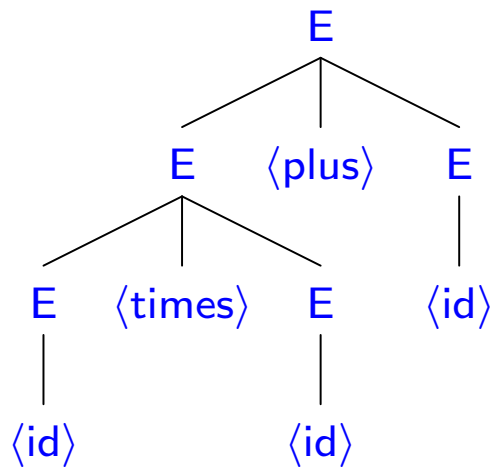
(4.6) Introduction

Suppose G is our grammar of arithmetic expressions:

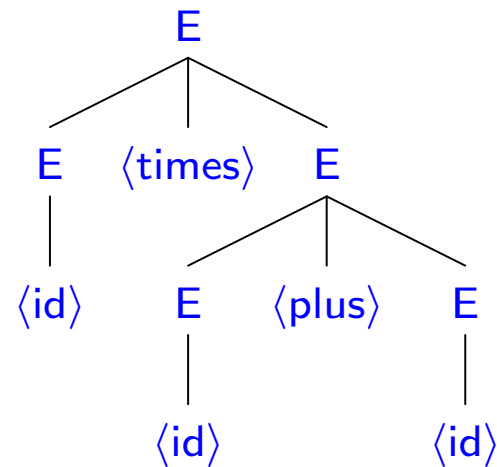
$$E \rightarrow E\langle\text{plus}\rangle E \mid E\langle\text{times}\rangle E \mid \langle\text{openPar}\rangle E \langle\text{closPar}\rangle \mid \langle\text{id}\rangle.$$

Question: are there multiple ways of parsing the string $\langle\text{id}\rangle\langle\text{times}\rangle\langle\text{id}\rangle\langle\text{plus}\rangle\langle\text{id}\rangle$ according to this grammar?

Answer: Yes:



(pt_1)



(pt_2)

(4.6) Introduction (Cont.)

In pt_1 , multiplication has higher precedence than addition; in pt_2 , the situation is reversed. Because there are multiple ways of parsing this string, we say that our grammar is “ambiguous”.

A grammar G is *ambiguous* iff there is a $w \in \mathbf{alphabet}(G)^*$ such that w is the yield of multiple valid parse trees for G whose root labels are s_G ; otherwise, G is *unambiguous*.

(4.6) Disambiguating Our Grammar of Arithmetic Expressions

Not every ambiguous grammar can be turned into an equivalent unambiguous one. However, we can use a simple technique to disambiguate our grammar of arithmetic expressions.

Since there are two binary operators in our language of arithmetic expressions, we have to decide:

(4.6) Disambiguating Our Grammar of Arithmetic Expressions

Not every ambiguous grammar can be turned into an equivalent unambiguous one. However, we can use a simple technique to disambiguate our grammar of arithmetic expressions.

Since there are two binary operators in our language of arithmetic expressions, we have to decide:

- whether multiplication has higher or lower precedence than addition;

(4.6) Disambiguating Our Grammar of Arithmetic Expressions

Not every ambiguous grammar can be turned into an equivalent unambiguous one. However, we can use a simple technique to disambiguate our grammar of arithmetic expressions.

Since there are two binary operators in our language of arithmetic expressions, we have to decide:

- whether multiplication has higher or lower precedence than addition;
- whether multiplication and addition are left or right associative.

(4.6) Disambiguating Our Grammar of Arithmetic Expressions

Not every ambiguous grammar can be turned into an equivalent unambiguous one. However, we can use a simple technique to disambiguate our grammar of arithmetic expressions.

Since there are two binary operators in our language of arithmetic expressions, we have to decide:

- whether multiplication has higher or lower precedence than addition;
- whether multiplication and addition are left or right associative.

As usual, we'll make multiplication have higher precedence than addition, and make both multiplication and addition be left associative.

(4.6) Example Disambiguation (Cont.)

As a first step towards disambiguating our grammar, we can form a new grammar with the three variables: **E** (expressions), **T** (terms) and **F** (factors), start variable **E** and productions:

$$E \rightarrow T \mid E\langle\text{plus}\rangle E,$$

$$T \rightarrow F \mid T\langle\text{times}\rangle T,$$

$$F \rightarrow \langle\text{id}\rangle \mid \langle\text{openPar}\rangle E \langle\text{closPar}\rangle.$$

The idea is that the lowest precedence operator “lives” at the highest level of the grammar, that the highest precedence operator lives at the middle level of the grammar, and that the basic expressions, including the parenthesized expressions, live at the lowest level of the grammar.

(4.6) Example Disambiguation (Cont.)

Now, there is only one way to parse the string $\langle \text{id} \rangle \langle \text{times} \rangle \langle \text{id} \rangle \langle \text{plus} \rangle \langle \text{id} \rangle$, since, if we begin by using the production $E \rightarrow T$, our yield will only include a $\langle \text{plus} \rangle$ if this symbol occurs within parentheses.

If we had more levels of precedence in our language, we would simply add more levels to our grammar.

(4.6) Example Disambiguation (Cont.)

On the other hand, there are still two ways of parsing the string $\langle \text{id} \rangle \langle \text{plus} \rangle \langle \text{id} \rangle \langle \text{plus} \rangle \langle \text{id} \rangle$: with left associativity or right associativity. To finish disambiguating our grammar, we must

(4.6) Example Disambiguation (Cont.)

On the other hand, there are still two ways of parsing the string $\langle \text{id} \rangle \langle \text{plus} \rangle \langle \text{id} \rangle \langle \text{plus} \rangle \langle \text{id} \rangle$: with left associativity or right associativity. To finish disambiguating our grammar, we must break the symmetry of the right-sides of the productions

$$E \rightarrow E \langle \text{plus} \rangle E,$$

$$T \rightarrow T \langle \text{times} \rangle T,$$

turning one of the E 's into T , and one of the T 's into F . To make our operators be left associative, we must

(4.6) Example Disambiguation (Cont.)

On the other hand, there are still two ways of parsing the string $\langle \text{id} \rangle \langle \text{plus} \rangle \langle \text{id} \rangle \langle \text{plus} \rangle \langle \text{id} \rangle$: with left associativity or right associativity. To finish disambiguating our grammar, we must break the symmetry of the right-sides of the productions

$$E \rightarrow E \langle \text{plus} \rangle E,$$

$$T \rightarrow T \langle \text{times} \rangle T,$$

turning one of the E 's into T , and one of the T 's into F . To make our operators be left associative, we must change the second E to T , and the second T to F ; right associativity would result from making the opposite choices.

(4.6) Example Disambiguation (Cont.)

Thus, our unambiguous grammar of arithmetic expressions is

$$E \rightarrow T \mid E\langle\text{plus}\rangle T,$$

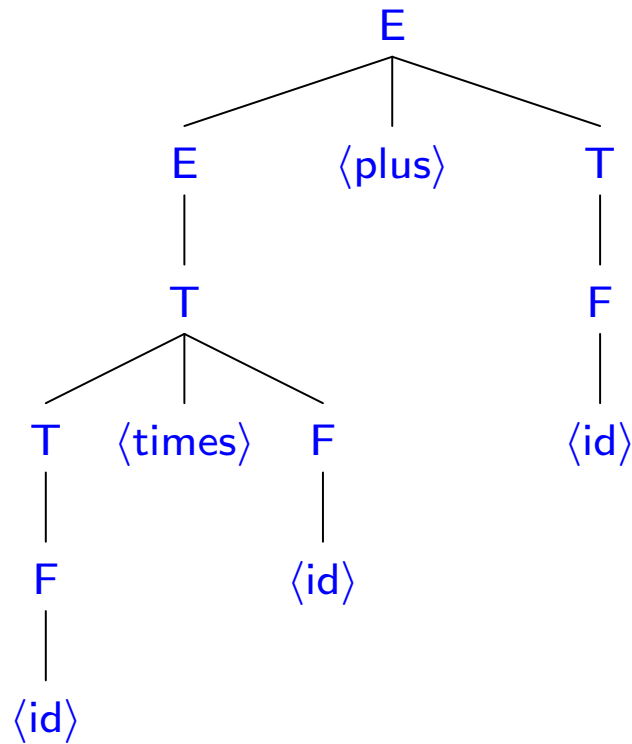
$$T \rightarrow F \mid T\langle\text{times}\rangle F,$$

$$F \rightarrow \langle\text{id}\rangle \mid \langle\text{openPar}\rangle E \langle\text{closPar}\rangle.$$

It can be proved that this grammar is indeed unambiguous, and that it is equivalent to the original grammar.

(4.6) Example Disambiguation (Cont.)

Now, the only parse of $\langle \text{id} \rangle \langle \text{times} \rangle \langle \text{id} \rangle \langle \text{plus} \rangle \langle \text{id} \rangle$ is



(4.6) Example Disambiguation (Cont.)

And, the only parse of $\langle id \rangle \langle plus \rangle \langle id \rangle \langle plus \rangle \langle id \rangle$ is

