Um compilador simples de uma passagem

Definição da sintaxe

Prof. Edson Alves

Faculdade UnB Gama

Definições

- Uma gramática deve descrever a estrutura hierárquica de seus elementos
- Por exemplo, o comando `if-else` da linguagem C, possui a forma

```
if (expressão) comando else comando
```

  a qual pode ser expressa como

$$cmd \rightarrow \textbf{if}\ (expr)\ cmd\ \textbf{else}\ cmd$$

- A expressão acima é uma regra de produção, onde a seta significa "pode ter a forma"
- Os elementos léxicos da produção (palavras-chaves, parêntesis) são chamados tokens ou terminais
- Variáveis como $expr$ e cmd representam sequências de tokens e são denominadas não-terminais

Componentes da linguagem livre de contexto

1. Um conjunto de tokens, denominados símbolos terminais
2. Um conjunto de não-terminais
3. Um conjunto de produções. Cada produção é definida por um não-terminal (lado esquerdo), seguido de uma seta, sucedida por uma sequência de tokens e/ou não-terminais (lado direito)
4. Designação de um dos não-terminais como símbolo de partida

Convenções de notação da gramática livre de contexto

- A gramática é especificada por uma lista de produções
- O símbolo de partida é definido como o não-terminal da primera produção listada
- Dígitos, símbolos e palavras em negrito são terminais
- Não-terminais são grafados em itálico
- Os demais símbolos são tokens
- Produções distintas de um mesmo não-terminal podem ser agrupadas por meio do caractere '|', que significa, neste contexto, “ou”

Exemplo de sintaxe para expressões infixas com adição e subtração

- Considere a seguinte gramática para expressões compostas por dígitos decimais e as operações de adição e subtração, em forma infixa:

$$\begin{aligned} expr &\rightarrow expr + digito \mid expr - digito \mid digito \\ digito &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$

- Os tokens desta gramática são os dez dígitos decimais e os caracteres `'+'` e `'-'`
- Os não-terminais são $expr$ e $digito$
- O símbolo de partida é o não-terminal $expr$

Cadeias de tokens

- Uma cadeia de tokens é uma sequência de zero ou mais tokens
- Uma cadeia que não contém nenhum token, grafada como ϵ, é denominada cadeia vazia
- Uma gramática deriva cadeias de tokens começando pelo símbolo de partida, substituíndo repetidamente um não-terminal pelo lado direito de uma produção deste não-terminal
- O conjunto de todas as cadeias de tokens possíveis gerados desta maneira corresponde a linguagem definida pela gramática

Exemplo de construção da expressão `1-2+3` por meio da gramática

1. `1` é $expr$, pois `1` é $digito$ (terceira alternativa para a produção de $expr$)
2. Pela segunda alternativa de produção de $expr$, `1-2` é também $expr$, pois `1` é $expr$ e `2` é $digito$
3. Por fim, pela primeira alternativa de produção de $expr$, `1-2+3` é $expr$, pois `1-2` é $expr$ e `3` é $digito$

Árvore gramatical

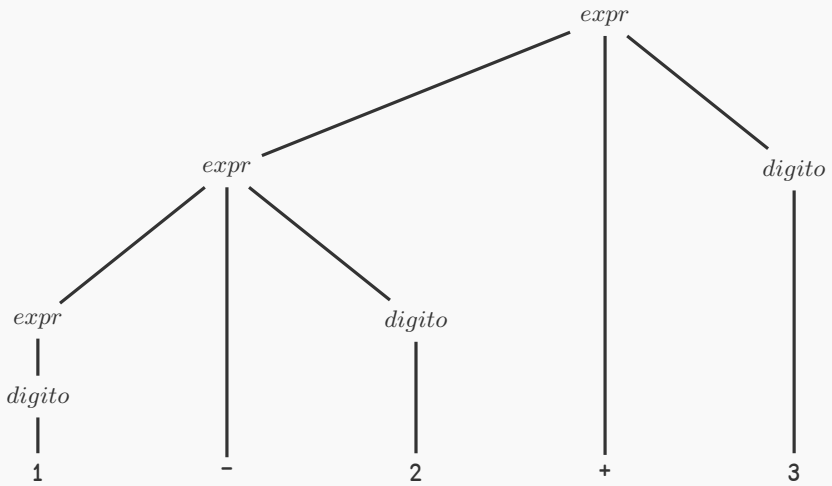
Dada uma gramática livre de contexto, uma árvore gramatical possui as seguintes propriedades:

1. A raiz é rotulada pelo símbolo de partida
2. Cada folha é rotulada por um token ou por ϵ
3. Cada nó interior é rotulado por um não-terminal
4. Se A é um não-terminal que rotula um nó interior e $X_1, X_2, \ldots, X_N$ são os rótulos de seus filhos (da esquerda para a direita), então

$$A \rightarrow X_1 \ X_2 \ \ldots X_N$$

é uma produção

Visualização da árvore gramatical da expressão 1-2+3

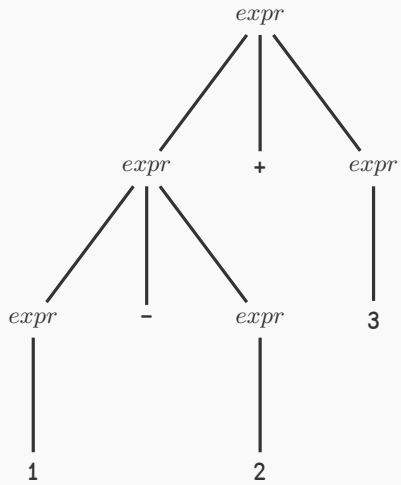


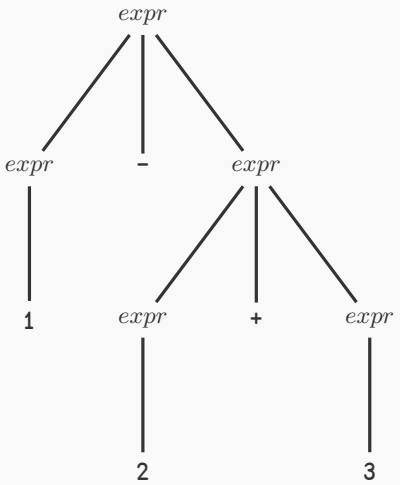
Características da árvore gramatical

- As folhas da árvore gramatical, quando lidas da esquerda para a direita, formam o produto da árvore, que é a cadeira gerada ou derivada a partir da raiz não-terminal
- O processo de encontrar uma árvore gramatical para uma dada cadeia de tokens é chmado de análise gramatical ou análise sintática daquela cadeia
- Uma gramática que permite a construção de duas ou mais árvores gramaticais distintas para uma mesma cadeia de tokens é denominada gramática ambígua
- A gramatica apresentada não é ambígua
- Contudo, se removida a distinção entre $expr$ e $digito$, a gramática passaria a ser ambígua:

$$expr \rightarrow expr + expr \mid expr - expr \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Exemplo de gramática ambígua





Associatividade de operadores

- Quando um operando está, simultaneamente, à esquerda e à direita de dois operadores (por exemplo, o dígito `2` na expressão `1-2+3`), é preciso decidir qual destes operadores receberá o operando
- Uma operação $\odot$ é associativa à esquerda se $a \odot b \odot c = (a \odot b) \odot c$
- Na maioria das linguagens de programação, os operadores aritméticos (`+`, `-`, `*` e `/`) são associativos à esquerda
- Uma operação $\oslash$ é associativa à direita se $a \oslash b \oslash c = a \oslash (b \oslash c)$
- Por exemplo, a atribuição (operador `=`) da linguagem C é associativa à direita: a expressão `a = b = c` equivale a expressão `a = (b = c)`
- Uma gramática possivel para esta atribuição seria:

$$\begin{array}{rcl} expr & \rightarrow & var = expr \mid var \\ var & \rightarrow & a \mid b \mid \ldots \mid z \end{array}$$

Precedência de operadores

- Algumas expressões da aritmética contém ambiguidades que não podem ser resolvidas apenas por meio da associatividade
- Por exemplo, qual seria o resultada expressão `1 + 2 * 3`? `9` ou `7`?
- Dizemos que o operador $\otimes$ tem maior precedência do que o operador $\oplus$ se $\otimes$ captura os operandos antes que $\oplus$ o faça
- Na aritmética, a multiplicação e a divisão tem maior precedência do que a adição e a subtração
- Se dois operadores tem mesma precedência, a associatividade determina a ordem que as operações serão realizadas

Construção de gramáticas com precedência de operadores

É possível construir uma gramática com precedência de operadores a partir dos seguintes passos:

1. Construa uma tabela com a associatividade e a precedência dos operadores, em ordem crescente de precedência (operadores com mesma precedência aparecem na mesma linha)

associatividade à esquerda	`+ -`
associatividade à esquerda	`* /`

2. Crie um não-terminal para cada nível ($expr$ e $termo$) e um não-terminal extra para as unidades básicas da expressão ($fator$)

$$fator \rightarrow \textbf{digito} \mid (expr)$$

Construção de gramáticas com precedência de operadores

3. Defina as produções para o último terminal criado para os níveis a partir dos operadores com maior precedência

$$\begin{array}{rcl} termo & \rightarrow & termo * fator \\ & | & termo \ / \ fator \\ & | & fator \end{array}$$

4. Faça o mesmo para os demais operadores, em ordem decrescente de precedência e crescente na lista de terminais criados para os níveis

$$\begin{array}{rcl} expr & \rightarrow & expr + termo \\ & | & expr - termo \\ & | & termo \end{array}$$

A presença de parêntesis na definição de $fator$ permite escrever expressões com níveis arbitrários de aninhamento, sendo que os parêntesis tem precedência sobre todos os operadores definidos.

Referências

1. **AHO**, Alfred V, **SETHI**, Ravi, **ULLMAN**, Jeffrey D. *Compiladores: Princípios, Técnicas e Ferramentas*, LTC Editora, 1995.