PUC Minas

Pontifícia Universidade Católica de Minas Gerais
ICEI - Departamento de Ciência da Computação
Disciplina: Compiladores
Prof. Alexei Machado

Lista de Exercícios No. 2

1. Para cada atributo do esquema de tradução abaixo, indique se é herdado ou sintetizado. Dê o resultado da tradução dos strings aaabbcc e abccc.

S → {$A.a_1 := 0$} A {$B.b_2 := A.a_2$} B {$S.s_1 := A.a_2 + B.b_1$} {Escreva ($S.s_1$)}
A → a {$A_1.a_1 := A.a_1 + 1$} A_1 { $A.a_2 := A1.a_2 + A.a_1$}
A → a {$A.a_2 := A.a_1$}
B → b {$B_1.b_2 := B.b_2$} B1 {$B.b_1 := B_1.b_1 + 1$}
B → C {$B.b_1 := B.b_2 * C.c_1$}
C → c C_1 {$C.c_1 := C_1.c_1 + 1$}
C → c {$C.c_1 := 1$}

2. Crie um comando *MCOPY* semelhante ao *COPY* do DOS que copie uma lista de arquivos para um diretório destino. O comando deve ter a seguinte sintaxe: MCOPY *lista* > *destino*, onde *lista* é uma seqüência de identificadores e *destino* um único identificador. Escreva um tradutor que traduza o comando em uma série de comandos *COPY* equivalentes. Por exemplo, o comando *MCOPY x y > fontes* deverá produzir a saída *COPY x fontes*, *COPY y fontes*. A ordem dos comandos de saída pode ser alterada.

3. Dada a gramática abaixo, escreva um esquema de tradução preditívo que verifica se o tipo da expressão é compatível com o identificador. Números (num) podem ser inteiros ou reais. Identificadores podem ser inteiros ou reais e já foram previamente declarados (tipo é conhecido). Observe que um identificador real pode receber expressão inteira, mas não o contrário.

S → id := E ;
E → num | num + E

4. Escreva um tradutor predizível que avalie a derivada de uma função polinomial em um determinado valor de *x*. A expressão avaliada deve Ter a seguinte sintaxe: *f' (num), f(x) = polinômio ?*, onde *num* é uma constante inteira, *polinômio* é representado pela expressão regular $(+ \cup - \cup \lambda)\ T\ ((+ \cup -)\ T)^*$ e *T* pela expressão regular $(num \cup \lambda)\ x\ (num \cup \lambda) \cup num$. Teste o tradutor para strings *f'(1), f(x) =* $x^2 + 3x - 1$*?* e *f'(0), f(x) =* $-2x^3 - x$*?*, escrevendo as árvores anotadas correspondentes.

5. A empresa onde você trabalha está convertendo seus sistemas escritos em Pascal para a linguagem C. Para agilizar este processo, seu chefe criou uma comissão de analistas para desenvolver um conversor de programas de Pascal para C (e, lógico **você** está incluído). A equipe responsável por gerar a gramática para o conversor já produziu um versão com um conjunto reduzido de estruturas da linguagem Pascal e sua correspondência na linguagem C:

Gramática Pascal: (símbolo inicial < LCOM >)

< *LCOM* > → for < *ATRIB* > to < *FINAL* > do begin < *LCOM* > end; < *LCOM* >
→ < *ATRIB* > ; < *LCOM* >
→ λ
< *ATRIB* > → id : = const
< *FINAL* > → const

Gramática C correspondente:

< *LCOM* > → for "(" < *ATRIB* > ; < *FINAL* > ")" "{" < *LCOM* > "}" < *LCOM* >
→ < *ATRIB* > ; < *LCOM* >
→ λ
< *ATRIB* > → id = const
< *FINAL* > → id <= const ; id + +

a) Desenhe um autômato finito determinístico que implemente o analisador léxico para a gramática Pascal. Considere os identificadores constituídos de letras e dígitos e começados por letra; considere as constantes numéricas constituídas por dígitos sem sinal.
b) Transforme a **gramática Pascal** em um esquema de tradução que converta um programa-fonte escrito em Pascal para um programa-fonte correspondente em C, **sem alterar a gramática**.
c) Implemente, em Portugol, o tradutor para o problema. Considere a existência das funções *CasaToken* e *LeToken*.

6. Considere a gramática abaixo que define a sintaxe do comando *switch*:

S → *switch* (id) *begin* C D *end*
C → *case* const : { A; } C | λ
D → *else* : { A; } | λ
A → id := const

a) Escreva um esquema de tradução, sem alterar a gramática, para a verificação de tipos e unicidade. Considere a existência de uma tabela de símbolos acessada globalmente. Os tipos básicos são o caractere, inteiro e byte, sendo que inteiro recebe byte, mas não o contrário. O comando *switch* só está definido para números.
b) Escreva um esquema de tradução, sem alterar a gramática, para gerar código Assembly 80x86. Considere a existência de uma tabela de símbolos acessada globalmente. Os tipos básicos são o caractere, inteiro e byte, sendo que inteiro recebe byte, mas não o contrário. O comando *switch* só está definido para números. Utilize a tabela de instruções vista em sala e use um procedimento do tipo *Escreva* para a saída. Fora a tabela de símbolos e o registro léxico, não existem variáveis globais.

7. Baseado na tabela de instruções do Assembly 80x86, transforme a gramática abaixo em um esquema de tradução que gera código para manipulação de apontadores. A notação utilizada é a seguinte: ↑T significa apontador de T; @id significa endereço de id; id ↑ significa conteúdo apontado por id. Escreva o tradutor correspondente.

P → { D } *begin* { C } *end*.
D → VAR { id : Tipo ; }$^+$
Tipo → int | ↑Tipo
C → id := E ;
E → T { + T }
T → num | id A | @id
A → ↑ | λ

8. Deseja-se adicionar à linguagem C a possibilidade de fazer operações entre matrizes, de modo que uma matriz possa ser multiplicada por um escalar, 2 matrizes possam ser somadas ou multiplicadas. A gramática abaixo representa um subconjunto da linguagem C, acrescida destas operações:

S → `void main(void)` "{" {D;} {C;} "}"
D → int id V
V → "[" num "]" V | λ
C → id I = E
I → "[" E "]" | λ
E → T R
R → + T R | λ
T → F Z
Z → * F Z | λ
F → num | id I

Insira ações semânticas para a verificação de tipos da linguagem. As expressões de tipo das declarações devem ser inseridas na tabela de símbolos. Duas matrizes só podem ser somadas se possuírem mesmas dimensões. A multiplicação de matrizes só se aplica a matrizes de 2 dimensões onde o número de colunas da primeira for igual ao número de linhas da segunda. A matriz resultante tem o número de linhas da primeira e o número de colunas da segunda. A multiplicação de uma matriz por um escalar não altera suas dimensões. Fora a tabela de símbolos e o registro léxico, não existem variáveis globais.

9. A gramática abaixo descreve um subconjunto da linguagem C que contém declarações de variáveis inteiras e vetores, além de atribuições e do operador condicional "?". O comando de atribuição com operador condicional tem a sintaxe **id=exp1==exp2?exp3:exp4**. O identificador receberá o valor de **exp3** caso **exp1** seja igual a **exp2**, ou o valor de **exp4** caso contrário.

Pontifícia Universidade Católica de Minas Gerais
ICEI - Departamento de Ciência da Computação
Disciplina: Compiladores
Prof. Alexei Machado

P → void main(void) "{" {C | D} "}"
D → { int id ["[" const "]"]; }$^{+}$
C → id = E [== E ? E : E] ;
E → T { + T }
T → const | id ["[" E "]"]

a) Insira ações semânticas para a verificação de tipos da linguagem.
b) Baseado nas instruções do Assembly 80x86, transforme a gramática em um esquema de tradução que gera código para a linguagem.