Universidade do Sul de Santa Catarina - UNISUL
Curso de Ciência da Computação **Campus de Tubarão**
Disciplina: Tradução de Linguagens de Programação
Professor: Charbel Szymanski **E-mail:** charbel@evoluma.com

Estratégia para implementar a análise semântica e geração de código
Aplicação na linguagem base LMS

1. Funções básicas da análise semântica – Breve resumo

- Verificar se as construções utilizadas no programa fonte estão semanticamente corretas, detectando e diagnosticando os erros semânticos encontrados.

  SEMÂNTICA ≅ COERÊNCIA ≅ SIGNIFICADO ≅ SENTIDO LÓGICO

- Extrair informações do programa fonte que permitam a geração de código intermediário

- Verificações Semânticas Usuais
  - Análise de escopo
    - Variáveis não declaradas
    - Múltiplas declarações de uma mesma variável
  - Compatibilidade de tipos
  - Coerência entre declaração e uso de identificadores
  - Correlação entre parâmetros formais e atuais
  - Referências não resolvidas
    - Procedimentos e desvios (no caso de uso de rótulos)

Mais especificamente, no caso da LMS teremos as verificações, por exemplo:

- Utilização dos identificadores segundo as regras semânticas da linguagem. Assim:
  - Análise de nomes de identificadores, constantes e procedures;
  - Procedures somente poderão ser referenciadas com o uso do comando CALL. O número de parâmetros da chamada deve ser igual ao número de parâmetros efetivos.
  - Constantes não podem ser redefinidas através de comandos de atribuição, nem mesmo referenciadas pelo comando READLN.
  - Lista de identificadores em comandos READLN e WRITELN devem ser nomes de variáveis.

2. Estratégia para a implementação - LMS (sem rótulos e sem variável indexada)

Os procedimentos relativos à análise semântica e à geração de código serão realizados concorrentemente, no processo de compilação em um só passo, de forma dirigida pela sintaxe. Teremos os procedimentos (na forma de ações) a serem disparados a partir da análise sintática.
Para tanto, embutiremos na gramática da LMS, de forma adequada, os símbolos das ações para avaliações semânticas e para geração de código intermediário.
Desta forma, na implementação, o algoritmo de análise sintática deve ser alterado para que, ao detectar um símbolo de ação na pilha de expansões (topo da pilha), seja invocada (executada) a ação correspondente.
Após a execução da ação, o símbolo deve ser retirado da pilha. Esta é uma das maneiras de implementar a análise e tradução para o código intermediário controlados pela sintaxe.

3. Geração de código intermediário

No processo de compilação, teremos a geração de código intermediário para uma máquina hipotética, definida para a disciplina (p/ linguagem LMS). A geração de código intermediário será realizada durante a análise semântica, conforme as ações semânticas definidas. Para maiores detalhes sobre a geração de código intermediário, consultar a descrição das ações semânticas, bem como, a definição da máquina hipotética e respectivo conjunto de instruções.

4. Gramática da linguagem LMS com ações semânticas embutidas (sem rótulos e sem variável indexada).

<PROGRAMA> ::= PROGRAM ID #100 ";" <BLOCO> "." #101 ;

<BLOCO> ::= <DCLCONST> <DCLVAR> #102 <DCLPROC><CORPO> ;

<LID> ::= ID #104 <REPID> ;

<REPID> ::= î ;

<REPID> ::= "," ID #104 <REPID> ;

<DCLCONST> ::= CONST ID #105 "=" INTEIRO #106 ";" <LDCONST> ;

<LDCONST> ::= î ;

<LDCONST> ::= ID #105 "=" INTEIRO #106 ";" <LDCONST> ;

<DCLCONST> ::= î ;

<DCLVAR> ::= VAR #107 <LID> ":" <TIPO> ";" <LDVAR> ;

<LDVAR> ::= î ;

<LDVAR> ::= <LID> ":" <TIPO> ";" <LDVAR> ;

<DCLVAR> ::= î ;

<TIPO> ::= INTEGER ;

<DCLPROC> ::= PROCEDURE ID #108 <DEFPAR> ";" #109 <BLOCO> ";" #110 <DCLPROC> ;

<DCLPROC> ::= î ;

<DEFPAR> ::= î ;

<DEFPAR> ::= "(" #111 <LID> ":" INTEGER ")" ;

<CORPO> ::= BEGIN <COMANDO> <REPCOMANDO> END ;

<REPCOMANDO> ::= î ;

<REPCOMANDO> ::= ";" <COMANDO> <REPCOMANDO> ;

<COMANDO> ::= ID #114 ":=" <EXPRESSAO> #115 ;

<COMANDO> ::= <CORPO> ;

<COMANDO> ::= î ;

<COMANDO> ::= CALL ID #116 <PARAMETROS> #117 ;

<PARAMETROS> ::= î ;

<PARAMETROS> ::= "(" <EXPRESSAO> #118 <REPPAR> ")" ;

<REPPAR> ::= î ;

<REPPAR> ::= "," <EXPRESSAO>#118 <REPPAR> ;

<COMANDO> ::= IF <EXPRESSAO> #120 THEN <COMANDO> <ELSEPARTE> #121 ;

<ELSEPARTE> ::= î ;

<ELSEPARTE> ::= #122 ELSE <COMANDO> ;

<COMANDO> ::= WHILE #123 <EXPRESSAO> #124 DO <COMANDO> #125 ;

<COMANDO> ::= REPEAT #126 <COMANDO> UNTIL <EXPRESSAO> #127 ;

<COMANDO> ::= READLN #128 "(" <VARIAVEL> <REPVARIAVEL> ")" ;

<VARIAVEL> ::= ID #129 ;

<REPVARIAVEL>::= î ;

<REPVARIAVEL>::= "," <VARIAVEL> <REPVARIAVEL> ;

<COMANDO> ::= WRITELN "(" <ITEMSAIDA> <REPITEM> ")" ;

<ITEMSAIDA> ::= LIT #130 ;

<ITEMSAIDA> ::= <EXPRESSAO> #131 ;

<REPITEM> ::= î ;

<REPITEM> ::= "," <ITEMSAIDA> <REPITEM> ;

<COMANDO> ::= CASE #132 <EXPRESSAO> OF <CONDCASE> END #133 ;

<CONDCASE> ::= INTEIRO <RPINTEIRO> ":" #134 <COMANDO> #135 <CONTCASE> ;

<RPINTEIRO> ::= "," #136 INTEIRO <RPINTEIRO> ;

<RPINTEIRO> ::= î ;

<CONTCASE> ::= î ;

<CONTCASE> ::= ";" <CONDCASE> ;

<COMANDO> ::= FOR ID #137 ":=" <EXPRESSAO> #138 TO <EXPRESSAO> #139 DO <COMANDO> #140 ;

<EXPRESSAO> ::= <EXPSIMP> <REPEXPSIMP> ;

<REPEXPSIMP> ::= î ;

<REPEXPSIMP> ::= "=" <EXPSIMP> #141 ;

<REPEXPSIMP> ::= "<" <EXPSIMP> #142 ;

<REPEXPSIMP> ::= ">" <EXPSIMP> #143 ;

<REPEXPSIMP> ::= ">=" <EXPSIMP> #144 ;

<REPEXPSIMP> ::= "<=" <EXPSIMP> #145 ;

<REPEXPSIMP> ::= "<>" <EXPSIMP> #146 ;

<EXPSIMP> ::= "+" <TERMO> <REPEXP> ;

<EXPSIMP> ::= "-" <TERMO> #147 <REPEXP> ;

<EXPSIMP> ::= <TERMO> <REPEXP> ;

<REPEXP> ::= "+" <TERMO> #148 <REPEXP> ;

<REPEXP> ::= "-" <TERMO> #149 <REPEXP> ;

<REPEXP> ::= OR <TERMO> #150 <REPEXP> ;

<REPEXP> ::= î ;

<TERMO> ::= <FATOR> <REPTERMO> ;

<REPTERMO> ::= î ;

<REPTERMO> ::= "*" <FATOR> #151 <REPTERMO> ;

<REPTERMO> ::= "/" <FATOR> #152 <REPTERMO> ;

<REPTERMO> ::= AND <FATOR> #153 <REPTERMO> ;

<FATOR> ::= INTEIRO #154 ;

<FATOR> ::= "(" <EXPRESSAO> ")" ;

<FATOR> ::= NOT <FATOR> #155 ;

<FATOR> ::= #156 <VARIAVEL> ;