Universidade Federal de Santa Catarina

Relatório T3 Construção de Compiladores

Isac de Souza Campos (17200449)
Arthur Philippi Bianco (17203358)
Enzo Coelho Albornoz (18100527)

1 Tarefa ASem

PONTO 1. Construção da árvore de expressão

Separamos as produções que derivam expressões aritméticas e chamamos o subconjunto de EXPA. Utilizaremos as produções da gramática que tiramos a recursão à esquerda no trabalho anterior. A seguir as produções:

***NUMEXPRESSION* -> TERM NUMEXPRESSION_**
REGRAS:

NUMEXPRESSION_.HER = TERM.SIN

NUMEXPRESSION.SIN = NUMEXPRESSION_.SIN

***NUMEXPRESSION_* -> ADDSUBTERM**
REGRAS:
ADDSUBTERM.HER = NUMEXPRESSION_.HER

NUMEXPRESSION.SIN = ADDSUBTERM.SIN

ADDSUBTERM -> ((+)TERM) NUMEXPRESSION_
REGRAS:

NUMEXPRESSION_.HER = NEW NO(+, ADDSUBTERM.HER, TERM.SIN)

ADDSUBTERM.SIN = NUMEXPRESSION_.SIN

***ADDSUBTERM* -> ((-)TERM) NUMEXPRESSION_**
REGRAS:

NUMEXPRESSION_.HER = NEW NO(-, ADDSUBTERM.HER, TERM.SIN)

ADDSUBTERM.SIN = NUMEXPRESSION_.SIN

***NUMEXPRESSION_* -> ε**
REGRAS:
NUMEXPRESSION_.SIN = NUMEXPRESSION_.HER

***TERM* -> UNARYEXPR TERM_**
REGRAS:

TERM_.HER = UNARYEXPR.SIN

TERM.SIN = TERM_.SIN

***TERM_* -> TERMULTDIVMOD**
REGRAS:

TERMULTDIVMOD.HER = TERM_.HER

TERM_.SIN = TERMULTDIVMOD.SIN

***TERM_* -> ε**
REGRAS:

TERM_.SIN = TERM_.HER

***TERMULTDIVMOD* -> (* | / | %) UNARYEXPR TERM_**
REGRAS:

TERM_.HER = NEW NO((* | / | %), TERMULTDIVMOD.HER, UNARYEXPR.SIN)

TERMULTDIVMOD.SIN = TERM_.SIN

***UNARYEXPR* -> ((+)) FACTOR**
REGRAS:

UNARYEXPR.SIN = FACTOR.SIN

***UNARYEXPR* -> ((-)) FACTOR**
REGRAS:

UNARYEXPR.SIN = FACTOR.SIN

***UNARYEXPR* -> FACTOR**
REGRAS:
UNARYEXPR.SIN = FACTOR.SIN

***FACTOR* -> int_constant**
REGRAS:
FACTOR.NO = NEW NO(int_constant.lvalue,,)

***FACTOR* -> float_constant**
REGRAS:
FACTOR.NO = NEW NO(float_constant.lvalue,,)

***FACTOR* -> string_constant**
REGRAS:
FACTOR.NO = NEW NO(string_constant.lvalue,,)

***FACTOR* -> null**
REGRAS:

FACTOR.NO = NEW NO(“null”,,)

***FACTOR* -> LVALUE**
REGRAS:
FACTOR.SIN = LVALUE.SIN

***FACTOR* -> (NUMEXPRESSION)**
REGRAS:
FACTOR.SIN = NUMEXPRESSION.SIN

***LVALUE* -> ident LVALUE_**
REGRAS:
LVALUE_.HER = NEW LEAF(SIMTAB(IDENT))
LVALUE.SIN = LVALUE_.SIN

***LVALUE_* -> NUMEXPRESSIONARRAY**
REGRAS:
LVALUE_.SIN = NUMEXPRESSIONARRAY.SIN

***LVALUE_* -> ε**
REGRAS:
LVALUE_.SIN = LVALUE_.HER

***NUMEXPRESSIONARRAY* -> [NUMEXPRESSION] NUMEXPRESSIONARRAY_**
REGRAS:

NUMEXPRESSIONARRAY_.HER = NEW NO(“NumExpressionArray”,
NUMEXPRESSION.SIN, NUMEXPRESSIONARRAY.HER)

NUMEXPRESSIONARRAY.SIN = NUMEXPRESSIONARRAY_.SIN

***NUMEXPRESSIONARRAY_* -> NUMEXPRESSIONARRAY**
REGRAS:
NUMEXPRESSIONARRAY_.SIN = NUMEXPRESSIONARRAY.SIN

NUMEXPRESSIONARRAY_ -> ε
NUMEXPRESSIONARRAY_.SIN = NUMEXPRESSIONARRAY_.HER

Com as regras semânticas finalizadas, devemos provar que esta SDD é L-atribuída. Para isso, observamos os atributos herdados e sintetizados e verificarmos a qual símbolo eles são associados. No caso dos herdados, esses devem ser associados ao seu irmão à esquerda ou seu pai (cabeça de produção). Já os sintetizados devem ser associados a um irmão à esquerda ou a ele mesmo.

Consideremos as regras semânticas para os atributos herdados, vistos na tabela 1:

Produção	Regra Semântica
NUMEXPRESSION -> TERM NUMEXPRESSION_	NUMEXPRESSION_.HER = TERM.SIN
NUMEXPRESSION_ -> ADDSUBTERM	ADDSUBTERM.HER = NUMEXPRESSION_.HER
ADDSUBTERM -> ((+)TERM) NUMEXPRESSION_	NUMEXPRESSION_.HER = NEW NO(+, ADDSUBTERM.HER, TERM.SIN)
ADDSUBTERM -> ((-)TERM) NUMEXPRESSION_	NUMEXPRESSION_.HER = NEW NO(-, ADDSUBTERM.HER, TERM.SIN)
TERM -> UNARYEXPR TERM_	TERM_.HER = UNARYEXPR.SIN
TERM_ -> TERMULTDIVMOD	TERMULTDIVMOD.HER = TERM_.HER
TERMULTDIVMOD -> (* | / | %) UNARYEXPR TERM_	TERM_.HER = NEW NO((* | / | %), TERMULTDIVMOD.HER, UNARYEXPR.SIN)
LVALUE -> ident LVALUE_	LVALUE_.HER = NEW LEAF(SIMTAB(IDENT))
NUMEXPRESSIONARRAY -> [NUMEXPRESSION] NUMEXPRESSIONARRAY_	NUMEXPRESSIONARRAY_.HER = NEW NO(“NumExpressionArray”, NUMEXPRESSION.SIN, NUMEXPRESSIONARRAY.HER)

Tabela 1: Produções da gramática EXPA e suas regras semânticas dos atributos herdados.

NUMEXPRESSIONARRAY_.HER = NEW NO(“NumExpressionArray”, NUMEXPRESSION.SIN, NUMEXPRESSIONARRAY.HER)

Como todos os atributos herdados vistos na tabela estão associados à cabeça de produção ou irmão à esquerda, esses casos estão de acordo com a definição de SDD's L-atribuídas.

Agora, vejamos as regras para os atributos sintetizados na tabela 2:

Produção	Regra Semântica
NUMEXPRESSION -> TERM NUMEXPRESSION_	NUMEXPRESSION.SIN = NUMEXPRESSION_.SIN
NUMEXPRESSION_ -> ADDSUBTERM	NUMEXPRESSION.SIN = ADDSUBTERM.SIN
ADDSUBTERM -> ((+)TERM) NUMEXPRESSION_	ADDSUBTERM.SIN = NUMEXPRESSION_.SIN
ADDSUBTERM -> ((-)TERM) NUMEXPRESSION_	ADDSUBTERM.SIN = NUMEXPRESSION_.SIN
NUMEXPRESSION_ -> ε	NUMEXPRESSION_.SIN = NUMEXPRESSION_.HER
TERM -> UNARYEXPR TERM_	TERM.SIN = TERM_.SIN
TERM_ -> TERMULTDIVMOD	TERM_.SIN = TERMULTDIVMOD.SIN
TERM_ -> ε	TERM_.SIN = TERM_.HER
TERMULTDIVMOD -> (* | / | %) UNARYEXPR TERM_	TERMULTDIVMOD.SIN = TERM_.SIN
UNARYEXPR -> ((+)) FACTOR	UNARYEXPR.SIN = FACTOR.SIN
UNARYEXPR -> ((-)) FACTOR	UNARYEXPR.SIN = FACTOR.SIN
UNARYEXPR -> FACTOR	UNARYEXPR.SIN = FACTOR.SIN
FACTOR -> LVALUE	FACTOR.SIN = LVALUE.SIN
FACTOR -> (NUMEXPRESSION)	FACTOR.SIN = NUMEXPRESSION.SIN
LVALUE -> ident LVALUE_	LVALUE.SIN = LVALUE_.SIN
*LVALUE_* -> NUMEXPRESSIONARRAY	LVALUE_.SIN = NUMEXPRESSIONARRAY.SIN
LVALUE_ -> ε	LVALUE_.SIN = LVALUE_.HER
NUMEXPRESSIONARRAY -> [NUMEXPRESSION] NUMEXPRESSIONARRAY_	NUMEXPRESSIONARRAY.SIN = NUMEXPRESSIONARRAY_.SIN

NUMEXPRESSIONARRAY_ -> NUMEXPRESSIONARRAY	NUMEXPRESSIONARRAY_.SIN = NUMEXPRESSIONARRAY.SIN
NUMEXPRESSIONARRAY_ -> ε	NUMEXPRESSIONARRAY_.SIN = NUMEXPRESSIONARRAY_.HER

Tabela 2: Produções da gramática EXPA e suas regras semânticas dos atributos sintetizados.

Os atributos sintetizados da tabela 2 estão todos associados ao seu irmão à esquerda ou a ele mesmo. Sendo assim, também respeitam a definição de SDD L-atribuída. Como todos os atributos herdados e sintetizados respeitam as definições, podemos afirmar que esta SDD é L-atribuída.

O próximo passo é, então, gerar a SDT da gramática com base nas informações obtidas até aqui. Assim, temos:

NUMEXPRESSION **-> TERM {NUMEXPRESSION_.HER = TERM.SIN} NUMEXPRESSION_ {NUMEXPRESSION.SIN = NUMEXPRESSION_.SIN}**

NUMEXPRESSION_ **-> {ADDSUBTERM.HER = NUMEXPRESSION_.HER} ADDSUBTERM {NUMEXPRESSION_.SIN = ADDSUBTERM.SIN}**

ADDSUBTERM -> ((+)TERM) {NUMEXPRESSION_.HER = NEW NO(+, ADDSUBTERM.HER, TERM.SIN)} NUMEXPRESSION_ {ADDSUBTERM.SIN = NUMEXPRESSION_.SIN}

ADDSUBTERM -> ((+)TERM) {NUMEXPRESSION_.HER = NEW NO(-, ADDSUBTERM.HER, TERM.SIN)} NUMEXPRESSION_ {ADDSUBTERM.SIN = NUMEXPRESSION_.SIN}

NUMEXPRESSION_ **-> ε {NUMEXPRESSION_.SIN = NUMEXPRESSION_.HER}**

TERM **-> UNARYEXPR {TERM_.HER = UNARYEXPR.SIN} TERM_ {TERM.SIN = TERM_.SIN}**

TERM_ **-> {TERMULTDIVMOD.HER = TERM_.HER} TERMULTDIVMOD {TERM_.SIN = TERMULTDIVMOD.SIN}**

TERM_ **-> ε {TERM_.SIN = TERM_.HER}**

```
TERMULTDIVMOD -> (* | / | %) UNARYEXPR {TERM_.HER = NEW NO((* | / | %),
TERMULTDIVMOD.HER, UNARYEXPR.SIN)} TERM_ {TERMULTDIVMOD.SIN = TERM_.SIN}

UNARYEXPR -> ((+)) FACTOR {UNARYEXPR.SIN = FACTOR.SIN}

UNARYEXPR -> ((-)) FACTOR {UNARYEXPR.SIN = FACTOR.SIN}

UNARYEXPR -> FACTOR {UNARYEXPR.SIN = FACTOR.SIN}

FACTOR -> int_constant {FACTOR.NO = NEW NO(int_constant.lvalue, , )}

FACTOR -> float_constant {FACTOR.NO = NEW NO(float_constant.lvalue, , )}

FACTOR -> string_constant {FACTOR.NO = NEW NO(string_constant.lvalue, , )}

FACTOR -> null {FACTOR.NO = NEW NO("null", , )}

FACTOR -> LVALUE {FACTOR.SIN = LVALUE.SIN}

FACTOR -> (NUMEXPRESSION) {FACTOR.SIN = NUMEXPRESSION.SIN}

LVALUE -> ident {LVALUE_.HER = NEW LEAF(SIMTAB(IDENT))} LVALUE_ {LVALUE.SIN =
LVALUE_.SIN}

LVALUE_ -> NUMEXPRESSIONARRAY {LVALUE_.SIN = NUMEXPRESSIONARRAY.SIN}

LVALUE_ -> ε {LVALUE_.SIN = LVALUE_.HER}

NUMEXPRESSIONARRAY -> [NUMEXPRESSION] {NUMEXPRESSIONARRAY_.HER = NEW
NO("NumExpressionArray", NUMEXPRESSION.SIN, NUMEXPRESSIONARRAY.HER)}
NUMEXPRESSIONARRAY_ {NUMEXPRESSIONARRAY.SIN = NUMEXPRESSIONARRAY_.SIN}

NUMEXPRESSIONARRAY_ -> NUMEXPRESSIONARRAY {NUMEXPRESSIONARRAY_.SIN =
NUMEXPRESSIONARRAY.SIN}

NUMEXPRESSIONARRAY_ -> ε {NUMEXPRESSIONARRAY_.SIN = NUMEXPRESSIONARRAY_.HER}
```

Esta é a SDT da gramática EXPA, que basicamente consiste em unir as produções com suas regras semânticas numa ordem que permita o fluxo acontecer com êxito, tendo todos os atributos necessários disponíveis quando forem requisitados.

Ponto 2. Inserção do tipo na tabela de símbolos

1. Separando produções que derivam declaração de variáveis DEC:

***VARDECL* -> (int|float|string)ident VARDECL_**

***VARDECL_* -> VARDECLARRAY**

***VARDECL_* -> ε**

***VARDECLARRAY* -> [int_constant] VARDECL_**

2. Construção de uma SDD L-Atribuída:

***VARDECL* -> (int|float|string)ident VARDECL_**
REGRAS:

VARDECL.tipo = int|float|string

VARDECL_.HER = NEW NO(“vardecl”, ident.sin, VARDECL.HER)

VARDECL.SIN = VARDECL_.SIN

***VARDECL_* -> VARDECLARRAY**
REGRAS:

VARDECLARRAY.HER = VARDECL_.HER

VARDECL_.SIN = VARDECLARRAY.SIN

***VARDECL_* -> ε**

REGRAS:
VARDECL_.SIN = VARDECL_.HER

***VARDECLARRAY* -> [int_constant] VARDECL_**
REGRAS:

```
VARDECLARRAY.VEC.APPEND(int_constant.lvalue)

VARDECL_.HER = VARDECLARRAY.VEC

VARDECLARRAY.VEC = VARDECL_.SIN
```

Assim como na gramática EXPA, agora analisaremos as associações dos atributos herdados e sintetizados, na tabela 3, para podermos confirmar que esta SDD é L-atribuída.

Produção	Regra Semântica
VARDECL -> (int|float|string)ident VARDECL_	VARDECL_.HER = NEW NO("vardecl", ident.SIN VARDECL.HER)
*VARDECL_* -> VARDECLARRAY	VARDECLARRAY.HER = VARDECL_.HER
VARDECLARRAY -> [int_constant] VARDECL_	VARDECL_.HER = VARDECLARRAY.VEC

Tabela 3: Produções da gramática DEC e suas regras semânticas dos atributos herdados.

De acordo com a definição de SDD's L-atribuídas já citadas na primeira gramática, esta SDD é L-atribuída de acordo com as associações dos atributos herdados. Agora os sintetizados podem ser vistos na tabela 4:

Produção	Regra Semântica
VARDECL -> (int|float|string)ident VARDECL_	VARDECL.SIN = VARDECL_.SIN
*VARDECL_* -> VARDECLARRAY	VARDEC_.SIN = VARDECLARRAY.SIN
VARDECL_ -> ε	VARDECL_.SIN = VARDECL_.HER

Tabela 4: Produções da gramática DEC e suas regras semânticas dos atributos sintetizados.

Mais uma vez temos as regras respeitando a definição, nos permitindo confirmar que esta é uma SDD L-atribuída. Em seguida, vemos a SDT correspondente:

VARDECL **-> (int|float|string) {VARDECL.tipo = int|float|string} ident**
{VARDECL_.HER = NEW NO(“vardecl”, ident.SIN VARDECL.HER)
} VARDECL_ {VARDECL.SIN = VARDECL_.SIN}

VARDECL_ **-> {VARDECLARRAY.HER = VARDECL_.HER} VARDECLARRAY {VARDECL_.SIN = VARDECLARRAY.SIN}**

VARDECL_ **-> ε {VARDECL_.SIN = VARDECL_.HER}**

VARDECLARRAY **-> [int_constant] {VARDECLARRAY.VEC.APPEND(int_constant.lvalue), VARDECL_.HER = VARDECLARRAY.VEC} VARDECL_ {VARDECLARRAY.VEC = VARDECL_.SIN}**

Ponto 3. Verificação de tipos

Para resolvermos o problema de verificação de tipos, visitamos o pai e verificamos se o nodo da esquerda e o da direita são de tipos compatíveis dado a operação. Se houver string em qualquer um dos operandos, mas no outro não, então temos um erro de tipagem.

Ponto 4. Verificação de identificadores por escopo

Este problema foi resolvido da seguinte forma: no momento que encontramos a declaração de uma variável, buscamos na tabela de símbolos atual (deste escopo atual). Caso uma variável com o mesmo nome já esteja presente na tabela, lançamos um erro de redeclaração de variável, caso contrário adicionamos a variável na tabela e prosseguimos.