Universidade Federal de São Carlos – Departamento de Computação
Construção de Compiladores e Construção de Compiladores 1
Profa. Helena Caseli

Quinta Lista de Exercícios – Análise Semântica

1) Considere a seguinte gramática simples de declarações de variáveis como na sintaxe de C:

<decl> ::= <tipo> <var-lista>
<tipo> ::= int | float
<var-lista> ::= id, <var-lista> | id

a) Construa a gramática de atributos para o atributo de tipo de dados, para o qual daremos o nome de **tipo_dados** para diferenciar do não-terminal tipo.

R.

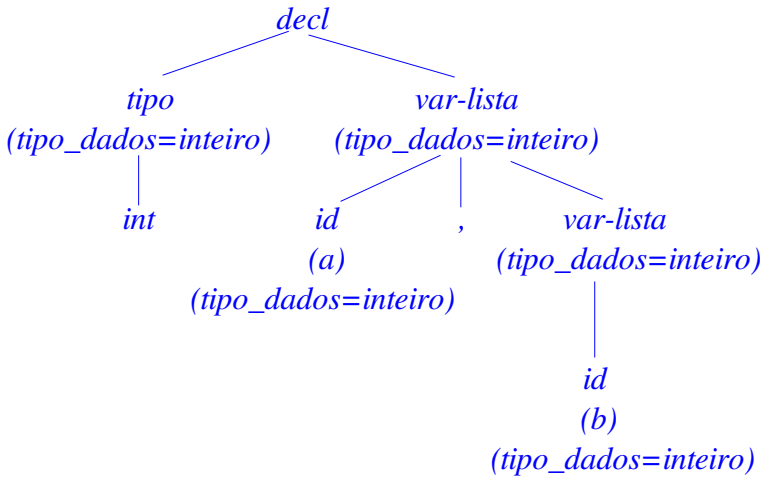
Regra gramatical	*Regras semânticas*
<decl> ::= <tipo> <var-lista>	*var-lista.tipo_dados = tipo.tipo_dados*
<tipo> ::= int	*tipo.tipo_dados = inteiro*
<tipo> ::= float	*tipo.tipo_dados = real*
<var-lista> ::= id, <var-lista>	*id.tipo_dados = var-lista$_1$.tipo_dados* *var-lista$_2$.tipo_dados = var-lista$_1$.tipo_dados*
<var-lista> ::= id	*id.tipo_dados = var-lista.tipo_dados*

Observações:

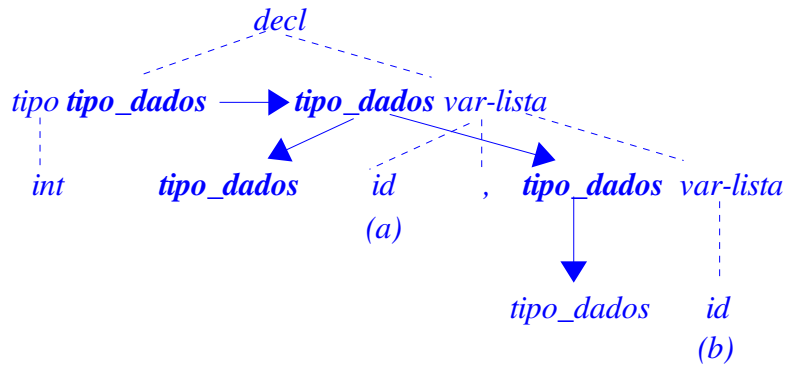
- *os valores de tipo_dados pertencem ao conjunto {inteiro, real} o que corresponde aos tokens int e float*
- *o não-terminal <tipo> tem um tipo_dados dado pelo token que ele representa*
- *esse tipo_dados corresponde ao tipo_dados de toda a var-lista pela equação associada à regra gramatical para <decl>*
- *cada id na lista tem esse mesmo tipo_dados pelas equações associadas à <var-lista>*
- *veja que não há uma equação que envolva o tipo_dados do não-terminal <decl> já que uma <decl> não precisa ter um tipo_dados (não é necessário que o valor de um atributo seja especificado para todos os símbolos gramaticais)*

b) Construa a árvore sintática com o cálculo dos atributos tipo_dados para a cadeia **int a, b**

R.



c) Desenhe o grafo de dependência para a cadeia **int a, b** amarrado à árvore sintática construída na letra b).
R.



2) O atributo tipo_dados da questão 1 é sintetizado ou herdado? Por que?

R. É herdado, pois ele tem dependência de pai para filho e entre irmãos.

3) Escreva um procedimento recursivo para calcular o atributo tipo_dados da questão 1, em todos os nós necessários.

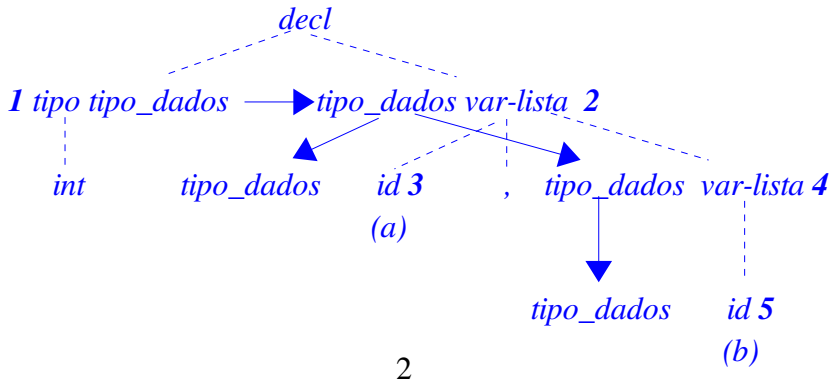
R.

```
procedure AvalTipo (T: no-arvore);
begin
        case tipo-no de T of
        decl:
                AvalTipo (tipo filho de T);
                Atribui tipo_dados de tipo filho de T a var-lista filho de T;
                AvalTipo (var-lista filho de T);
        tipo:
                if filho de T = int then T.tipo_dados := inteiro
                else T.tipo_dados = real;
        var-lista:
                atribui T.tipo_dados a primeiro filho de T;
                if terceiro filho de T não é nil then
                        atribui T.tipo_dados a terceiro filho;
                        AvalTipo (terceiro filho de T);
        end case;
end AvalTipo;
```

Observe a mistura de pré-ordem e em-ordem dependendo do tipo distinto de nó processado. Por exemplo, um nó <decl> requer que o tipo_dados de seu primeiro filho seja computado primeiro e, depois, atribuído ao segundo filho antes da chamada recursiva de AvalTipo naquele filho; esse processo é em-ordem. Um nó <var-lista>, no entanto, atribui tipo_dados aos seus filhos antes de qualquer chamada recursiva; isso é um processo em pré-ordem.

4) Enumere os nós da árvore construída na letra b) da questão 1 indicando a ordem de computação de tipo_dados de acordo com o algoritmo da questão 3. Que tipo de percurso é esse?

R. Combinação de percurso pré-ordem com em-ordem.



5) Considere a seguinte gramática simples de expressões com uma única operação, a divisão (/), e dois tipos de operandos: números inteiros (sequências de dígitos indicados pelo token num) e números de ponto flutuante (indicados pela sequência num.num).

<S> ::= <exp>

<exp> ::= <exp> / <exp> | num | num**.**num

a) Construa uma gramática de atributos capaz de interpretar de maneira distinta a operação divisão dependendo do tipo de operandos envolvidos: se pelo menos um for ponto flutuante, a divisão será de ponto flutuante; se todos forem inteiros então a divisão será inteira. Três atributos deverão ser calculados: um que indica se a expressão é de ponto flutuante (**éFlut**), outro para o tipo da expressão (**etipo**) e um último para armazenar o valor da expressão (**val**).

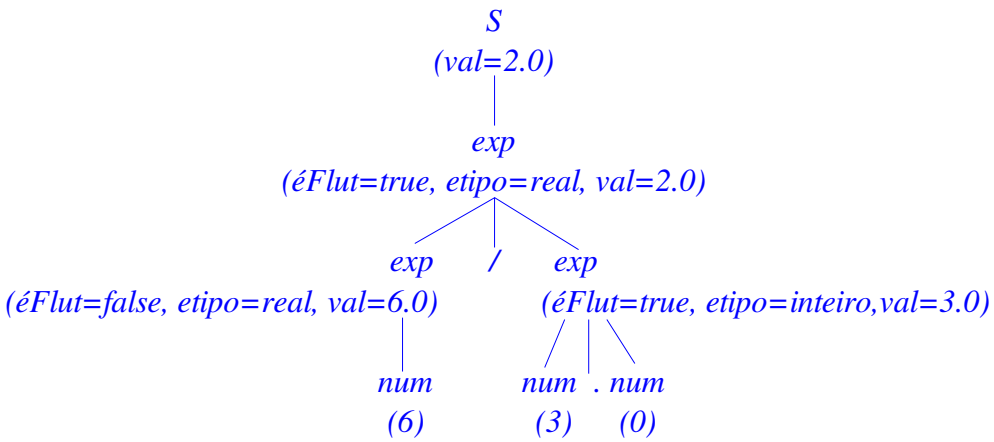
Use **div** para divisão de inteira e **/** para divisão de ponto flutuante, assim 5/2.0 = 1,25 (5 / 2.0) e 5/2 = 2 (5 div 2).

R.

Regra gramatical	*Regras semânticas*
$S \rightarrow exp$	$exp.etipo$ = ***if*** $exp.éFlut$ ***then*** *real* ***else*** *inteiro* $S.val = exp.val$
$exp_1 \rightarrow exp_2 / exp_3$	$exp_1.éFlut = exp_2.éFlut$ ***or*** $exp_3.éFlut$ $exp_2.etipo = exp_1.etipo$ $exp_3.etipo = exp_1.etipo$ $exp_1.val$ = ***if*** $exp_1.etipo$ = *inteiro* ***then*** $exp_2.val$ ***div*** $exp_3.val$ ***else*** $exp_2.val$ / $exp_3.val$
$exp \rightarrow num$	$exp.éFlut$ = ***false*** $exp.val$ = ***if*** $exp.etipo$ = *inteiro* ***then*** $num.lexval$ ***else*** *IntToFloat*($num.lexval$)
$exp \rightarrow num.num$	$exp.éFlut$ = ***true*** $exp.val = num.lexval$ **.** $num.lexval$

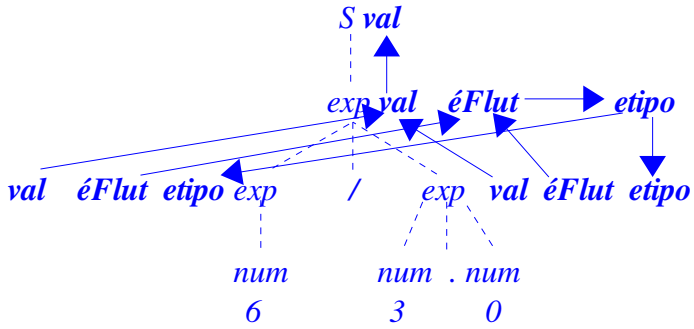
b) Construa a árvore sintática com o cálculo dos atributos para a cadeia **6/3.0**

R.



c) Desenhe o grafo de dependência para a cadeia **int a, b** amarrado à árvore sintática construída na letra b).

R.



6) De que tipo (sintetizado ou herdado) é cada um dos 3 atributos calculados na questão 5? Justifique sua resposta.

R. éFlut é sintetizado, pois seu valor é calculado nas folhas (como mostram as regras semânticas dos casos base exp → num, para a qual éFlut é true; e exp → num.num, para a qual éFlut é false) e propagado para os pais.

val também é sintetizado, pois seu valor é calculado nas folhas (como mostram as regras semânticas dos casos base exp → num e exp → num.num com base no valor numérico retornado pelo analisador léxico, lexval) e propagado para os pais.

etipo, por sua vez, é herdado, pois seu valor é atribuído logo no símbolo inicial da gramática com base no valor de outro atributo, éFlut, e propagado para os nós filhos.

7) Descreva com palavras (não precisa fazer o algoritmo) como seria o processo para calcular os 3 atributos da questão 5. Quantas passadas seriam necessárias para calculá-los e qual o percurso usado nesse cálculo?

R. Os atributos éFlut, etipo e val podem ser computados em duas passadas na árvore sintática. A primeira passada computa o atributo sintetizado éFlut por um percurso em pós-ordem. A segunda passada computa o atributo herdado etipo e o atributo sintetizado val em um percurso combinado em pré-ordem e pós-ordem.

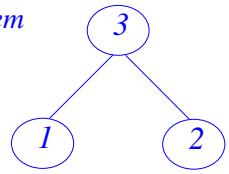
8) Diga quais são os dois tipos de gramáticas de atributos apresentados em aula explicando quais são as características de cada uma delas.

R. Os dois tipos são: Gramática S-atribuída e Gramática L-atribuída. Uma gramática S-atribuída é aquela que só possui atributos sintetizados, ou seja, para os quais os valores são computados exclusivamente a partir dos valores dos atributos filhos. Uma gramática L-atribuída, por sua vez, é aquela na qual a presença de atributos herdados é restringida para permitir que as ações semânticas possam ser executadas durante a análise sintática em uma única passada. Assim, em uma gramática L-atribuída, para um símbolo X no lado direito de uma regra de produção, a ação que calcula um atributo herdado de X deve aparecer à esquerda de X. Toda gramática S-atribuída é L-atribuída.

9) Considerando-se a árvore sintática para uma dada cadeia, diga quais são os percursos mais indicados nessa árvore para o cálculo de um atributo sintetizado e de um atributo herdado. Por que? Qual desses dois tipos de atributos é mais fácil de ser calculado, por que?

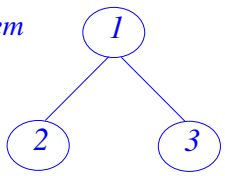
R.Um atributo sintetizado deve ser calculado usando o percurso em pós-ordem já que trata-se de um atributo para o qual o valor é calculado exclusivamente com base nos valores presentes em seus nós filhos e, portanto, os valores dos atributos dos filhos precisam ser conhecidos para se permitir o cálculo do atributo do nó pai.

pós-ordem

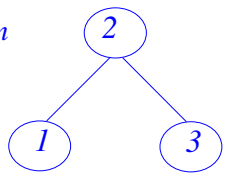


Um atributo herdado, por sua vez, é aquele para o qual o valor é calculado a partir dos valores dos atributos dos irmãos ou do pai e, nesse caso, o percurso mais indicado é o pré-ordem ou uma combinação de pré-ordem com em-ordem. Por exemplo, na ilustração abaixo, considerando-se que estamos calculando o valor do atributo herdado para o filho da direita, seguindo o percurso em pré-ordem ou em-ordem os valores de irmão e pai já seriam conhecidos.

Pré-ordem



em ordem



Os atributos são mais fáceis de serem calculados uma vez que se baseiam (a princípio) nas folhas (itens lexicais) e vão se propagando para a raiz. Já os atributos herdados dependem da ordem de computação dos atributos dos filhos.

10) Diga quais são as três principais operações na Tabela de Símbolos, explique o que vem a ser cada uma delas e dê exemplos de momentos nos quais elas ocorrem.

R. Inserção – armazena informações fornecidas pelas declarações. Ocorre principalmente no momento de declaração de elementos, mas também pode-se considerar a atualização de valores para um elemento já inserido como parte da "inserção" e, assim, em um comando de atribuição, por exemplo, o valor da variável à esquerda de ":=" é inserido na Tabela de Símbolos na linha desse elemento e coluna "valor".

Busca – recupera informações associadas a um elemento declarado no programa quando esse elemento é utilizado. Ocorre antes da inserção de um elemento na Tabela de Símbolos para verificar se o mesmo já foi declarado previamente e toda vez que um elemento é acessado, seja para verificar seu escopo, seu tipo ou outra informação relevante para a computação em questão.

Remoção – remove (ou torna inacessível) a informação a respeito de um elemento declarado quando esse não é mais necessário. Ocorre ao final da execução de um procedimento para a remoção de suas declarações locais (procedimentos, variáveis) uma vez que essas não serão mais necessárias.