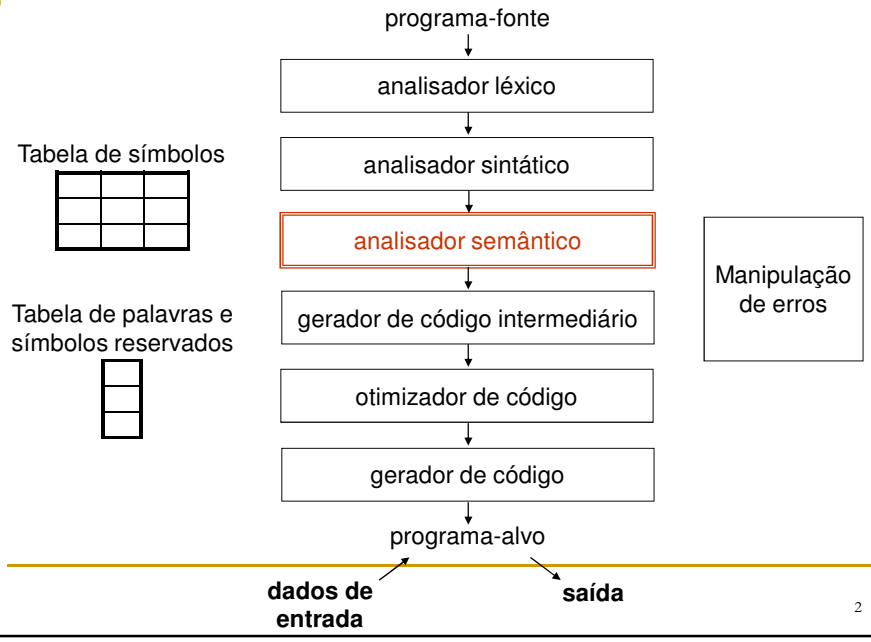
Análise semântica

Função, interação com o compilador
Tabela de símbolos
Análise semântica

Prof. Thiago A. S. Pardo
taspardo@icmc.usp.br

1

Estrutura geral de um compilador



2

Análise semântica

- Função: verificação do uso adequado
  - Análise contextual: declarações prévias de variáveis, procedimentos, etc.
  - Checagem de tipos
  - Coisas que vão além do domínio da sintaxe
    - Sensitividade ao contexto!
- Tipos de análise semântica
  - Estática, em tempo de compilação: linguagens tipadas, que exigem declarações
    - C, Pascal, etc.
  - Dinâmica, em tempo de execução: linguagens em que as variáveis são determinadas pelo contexto de uso
    - LISP, PROLOG

3

Análise semântica

- Devido às variações de especificação semântica das linguagens de programação, a análise semântica
  - Não é tão bem formalizada
  - Não existe um método ou modelo padrão de representação do conhecimento
  - Não existe um mapeamento claro da representação para o algoritmo correspondente
- Análise é artesanal, dependente da linguagem de programação

4

Análise semântica

- Semântica dirigida pela sintaxe
  - Conteúdo semântico fortemente relacionado à sintaxe do programa
  - Maioria das linguagens de programação modernas
- Em geral, a semântica de uma linguagem de programação não é especificada
  - O projetista do compilador tem que analisar e extrair a semântica

5

Formalização e implementação

- Assim como a sintaxe, a semântica precisa ser formalizada/descrita antes de ser implementada
  - Sintaxe: por exemplo, BNF → procedimentos recursivos
- Gramática de atributos é o formalismo de descrição da semântica comumente utilizado

6

Gramática de atributos

- Gramática de atributos
  - Método usualmente utilizado
  - Conjunto de atributos e regras semânticas para uma gramática
    - Cada regra sintática/gramatical pode ter regras semânticas associadas
  - Atributos associados aos símbolos gramaticais
    - Por exemplo, valor e escopo
      - x.valor, x.escopo
  - Regras semânticas que manipulam os atributos
    - Por exemplo, regra para somar os atributos valores de duas variáveis
      - x:=a+b, cuja regra é x.valor:=a.valor+b.valor

7

Gramática de atributos

- Atributos podem ser fixados durante a compilação ou a execução de um programa
  - A associação de um valor a um atributo é chamada "amarração" (ou vinculação) do atributo
  - Acontece em "tempo de amarração"
    - Em tempo de compilação, tem-se a amarração estática
    - Em tempo de execução, tem-se a amarração dinâmica

8

Gramática de atributos

- Exemplo de gramática de atributos

exp → exp + termo | exp - termo | termo
termo → termo * fator | termo div fator | fator
fator → (exp) | num

Regras gramaticais	Regras semânticas
$exp_1 \rightarrow exp_2$+termo	exp_1.val=exp_2.val+termo.val
$exp_1 \rightarrow exp_2$-termo	exp_1.val=exp_2.val-termo.val
exp→termo	exp.val=termo.val
$termo_1 \rightarrow termo_2$*fator	$termo_1$.val=$termo_2$.val*fator.val
$termo_1 \rightarrow termo_2$ div fator	$termo_1$.val=$termo_2$.val/fator.val
termo→fator	termo.val=fator.val
fator→(exp)	fator.val=exp.val
fator→num	fator.val=num.val

Gramática de atributos

- **Exercício**: escreva a gramática de atributos para a gramática abaixo

número → número dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

Gramática de atributos

- Exercício: escreva a gramática de atributos para a gramática abaixo

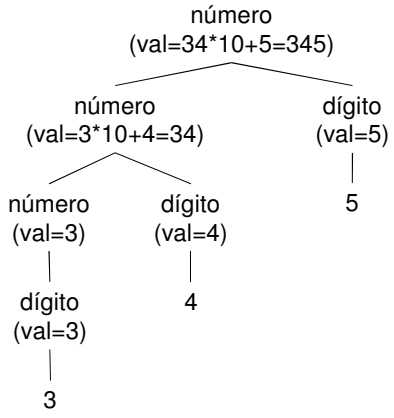
número → número dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

Regras gramaticais	Regras semânticas
$número_1$→$número_2$ dígito	$número_1$.val=$número_2$.val*10 + dígito.val
número→dígito	número.val=dígito.val
dígito→0	dígito.val=0
dígito→1	dígito.val=1
...	...
dígito→9	dígito.val=9

11

Gramática de atributos

- Árvore sintática com visualização da computação de atributos



12

Gramática de atributos

- **Exercício**: escreva a gramática de atributos para a gramática abaixo

  decl → tipo var-lista
  tipo → **int** | **float**
  var-lista → **id,** var-lista | **id**

13

Gramática de atributos

- Exercício: escreva a gramática de atributos para a gramática abaixo

  decl → tipo var-lista
  tipo → **int** | **float**
  var-lista → **id,** var-lista | **id**

Regras gramaticais	Regras semânticas
decl → tipo var-lista	var-lista.tipo_dado = tipo.tipo_dado
tipo → int	tipo.tipo_dado = integer
tipo → float	tipo.tipo_dado = real
var-lista_1 → id, var-lista_2	id.tipo_dado = var-lista_1.tipo_dado var-lista_2.tipo_dado = var-lista_1.tipo_dado
var-lista → id	id.tipo_dado=var-lista.tipo_dado

14

Gramática de atributos

- **Exercício**: construa a árvore sintática com cálculo dos atributos para a cadeia **float x, y**

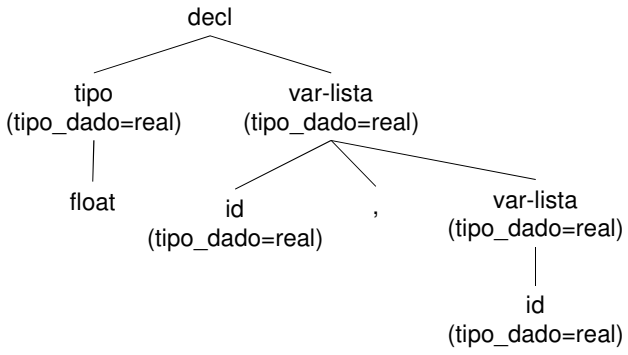
decl → tipo var-lista
tipo → **int** | **float**
var-lista → **id,** var-lista | **id**

Regras gramaticais	Regras semânticas
decl → tipo var-lista	var-lista.tipo_dado = tipo.tipo_dado
tipo → int	tipo.tipo_dado = integer
tipo → float	tipo.tipo_dado = real
$var\text{-}lista_1$ → id, $var\text{-}lista_2$	id.tipo_dado = $var\text{-}lista_1$.tipo_dado $var\text{-}lista_2$.tipo_dado = $var\text{-}lista_1$.tipo_dado
var-lista → id	id.tipo_dado=var-lista.tipo_dado

15

Gramática de atributos

- Exercício: construa a árvore sintática com cálculo dos atributos para a cadeia **float x, y**



16

Gramática de atributos

- Atenção
  - Nem todo símbolo gramatical tem atributos
  - Pode haver manipulação de mais de um atributo em uma mesma regra e para um mesmo símbolo
  - Pode não haver regras semânticas para uma regra sintática
- Em geral, a gramática de atributos de uma gramática pode especificar
  - Comportamento semântico das operações
  - Checagem de tipos
  - Manipulação de erros
  - Tradução do programa

Gramática de atributos

- Gramática para geração de números binários ou decimais, indicados pelos sufixos b ou d, respectivamente

número → num sufixo
sufixo → b | d
num → num dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

Gramática de atributos

- Escreva a gramática de atributos

número → num sufixo
sufixo → b | d
num → num dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

Gramática de atributos

Regras gramaticais	Regras semânticas
número → num sufixo	número.val = num.val num.base = sufixo.base
sufixo → b	sufixo.base = 2
sufixo → d	sufixo.base = 10
num_1 → num_2 dígito	num_1.val = **if** dígito.val = erro **or** num_2.val=erro **then** erro **else** num_2.val * num_1.base + dígito.val num_2.base = num_1.base dígito.base = num_1.base
num → dígito	num.val = dígito.val dígito.base = num.base
dígito → 0	dígito.val = 0
dígito → 1	dígito.val = 1
dígito → 2	dígito.val = **if** dígito.base=2 **then** erro **else** 2
...	...

Gramática de atributos

- Gramática para geração de números binários ou decimais, indicados pelos sufixos b ou d, respectivamente

  número → num sufixo
  sufixo → b | d
  num → num dígito | dígito
  dígito → 0|1|2|3|4|5|6|7|8|9

- A sintaxe permitiria o número **02b**, mas a semântica não

21

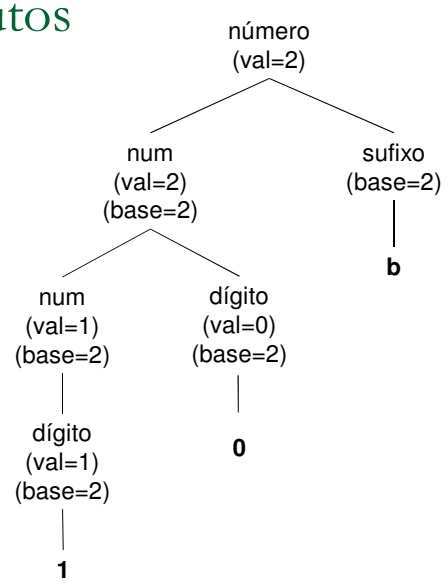
Gramática de atributos

- **Exercício**: construa a árvore sintática com cálculo dos atributos para a cadeia **10b**

22

Gramática de atributos

- 10b



23

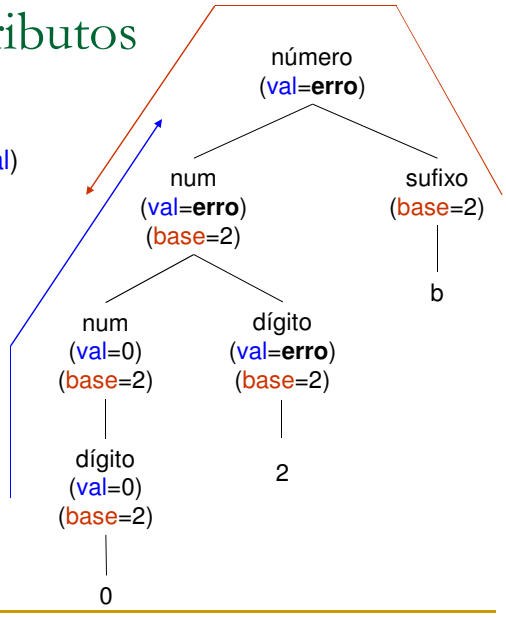
Gramática de atributos

- **Exercício**: construa a árvore sintática com cálculo dos atributos para a cadeia **02b**

24

Gramática de atributos

- Atenção
  - Alguns valores sobem (val)
  - Outros descem (base)



Como calcular os atributos de forma consistente?

Cômputo de atributos

- Com base na árvore sintática explícita
  - Grafos de dependência
    - Compilador de mais de uma passagem
- *Ad hoc*
  - Análise semântica "comandada" pela análise sintática
    - Compilador de uma única passagem

Cômputo de atributos

- Grafos de dependência
  - Especificam a ordem de cômputo dos atributos de cada regra gramatical em uma árvore sintática
    - Portanto, um grafo associado a cada regra gramatical
  - Para uma cadeia da linguagem, tem-se um grafo composto por todos os subgrafos

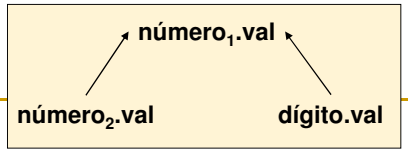
27

Cômputo de atributos

- Exemplo

número → número dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

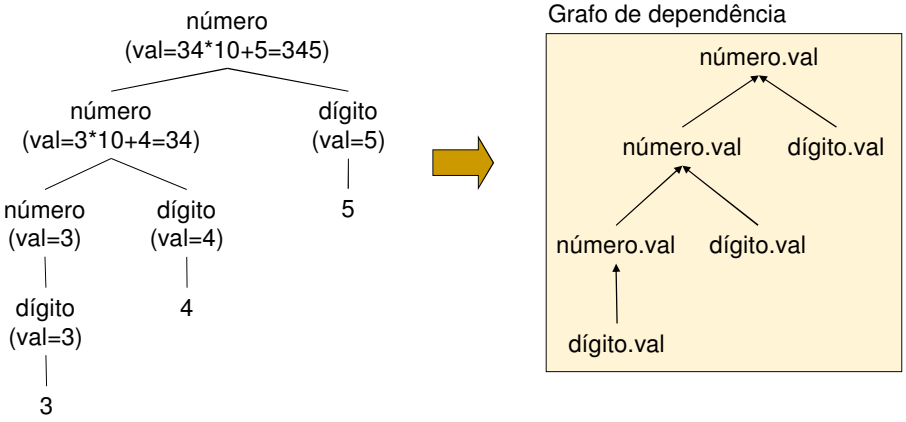
Regras gramaticais	*Regras semânticas*
número$_1$→número$_2$ dígito	**número$_1$.val→número$_2$.val*10 + dígito.val**
número→dígito	Número.val=dígito.val
dígito→0	dígito.val=0
dígito→1	dígito.val=1
...	...
dígito→9	dígito.val=2



28

Cômputo de atributos

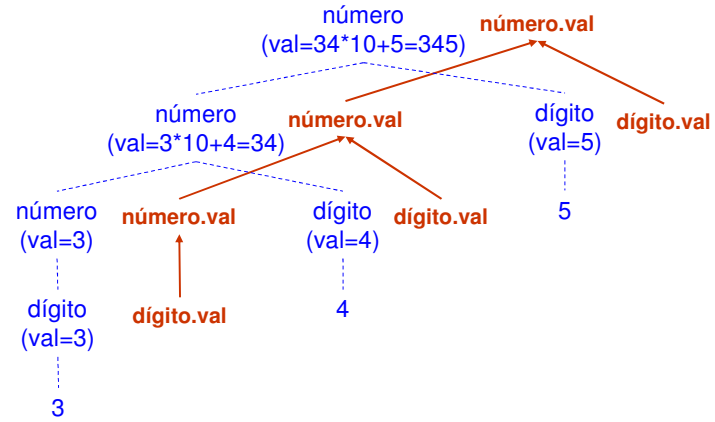
- Cadeia 345



29

Cômputo de atributos

- Grafo amarrado à árvore sintática



30

Cômputo de atributos

- Considere a gramática abaixo e sua gramática de atributos

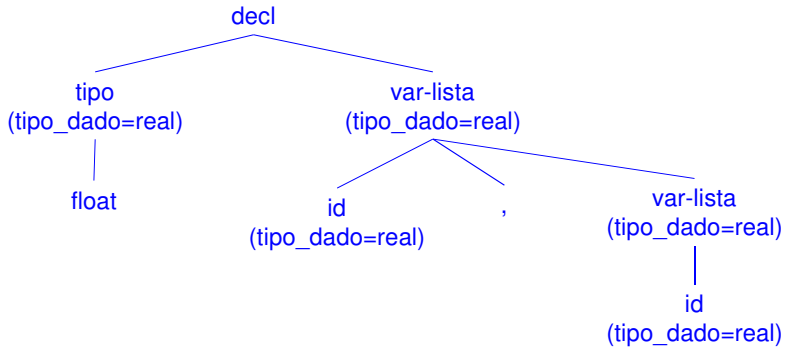
decl → tipo var-lista
tipo → **int** | **float**
var-lista → **id,** var-lista | **id**

Regras gramaticais	Regras semânticas
decl → tipo var-lista	var-lista.tipo_dado = tipo.tipo_dado
tipo → int	tipo.tipo_dado = integer
tipo → float	tipo.tipo_dado = real
$var\text{-}lista_1$ → id, $var\text{-}lista_2$	id.tipo_dado = $var\text{-}lista_1$.tipo_dado $var\text{-}lista_2$.tipo_dado = $var\text{-}lista_1$.tipo_dado
var-lista → id	id.tipo_dado=var-lista.tipo_dado

31

Cômputo de atributos

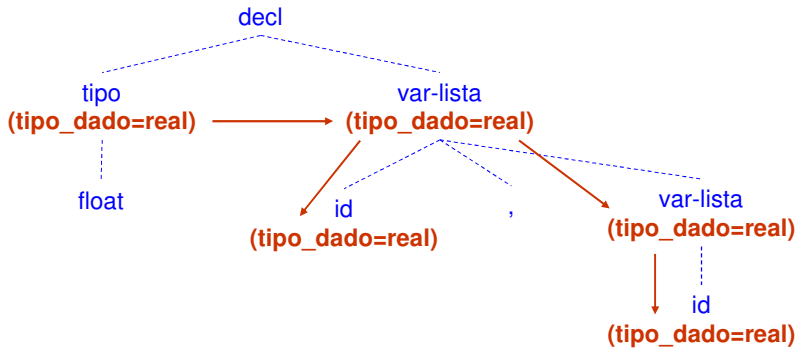
- Para a cadeia float x,y



32

Cômputo de atributos

- Para a cadeia float x,y



Cômputo de atributos

- **Exercício**: considere a gramática abaixo e sua gramática de atributos

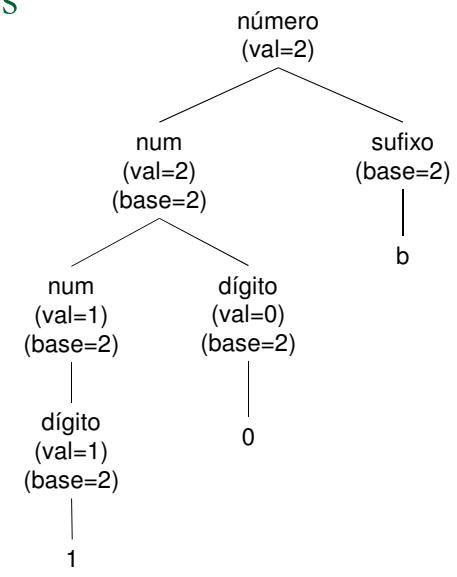
número → num sufixo
sufixo → b | d
num → num dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

Cômputo de atributos

Regras gramaticais	Regras semânticas
número → num sufixo	número.val = num.val num.base = sufixo.base
sufixo → b	sufixo.base = 2
sufixo → d	sufixo.base = 10
num_1 → num_2 dígito	num_1.val = **if** dígito.val = erro **or** num_2.val=erro **then** erro **else** num_2.val * num_1.base + dígito.val num_2.base = num_1.base dígito.base = num_1.base
num → dígito	num.val = dígito.val dígito.base = num.base
dígito → 0	dígito.val = 0
dígito → 1	dígito.val = 1
dígito → 2	dígito.val = **if** dígito.base=2 **then** erro **else** 2
...	...

Cômputo de atributos

- Dada a árvore sintática da cadeia 10b, construa o grafo de dependência



36

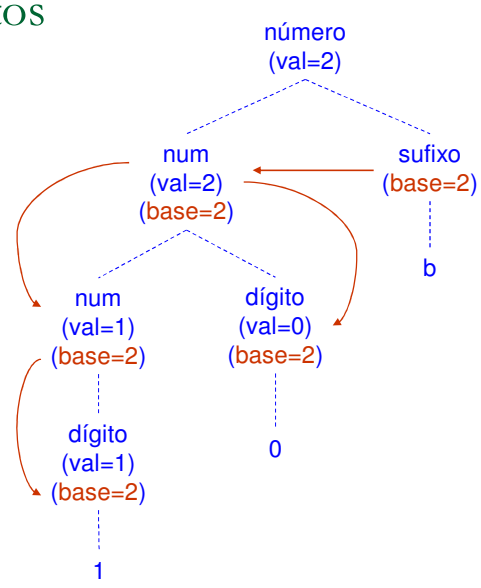
Cômputo de atributos

- Atributo base

número → num sufixo
num.base = sufixo.base

num_1 → num_2 dígito
num_2.base = num_1.base
dígito.base = num_1.base

num → dígito
dígito.base = num.base



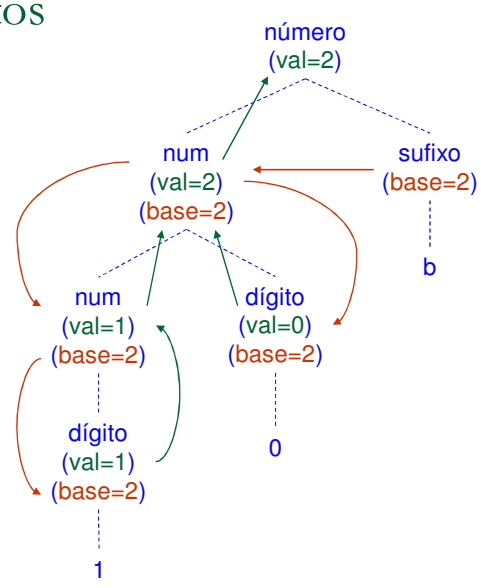
Cômputo de atributos

- Atributo base e atributo valor

número → num sufixo
número.val = num.val

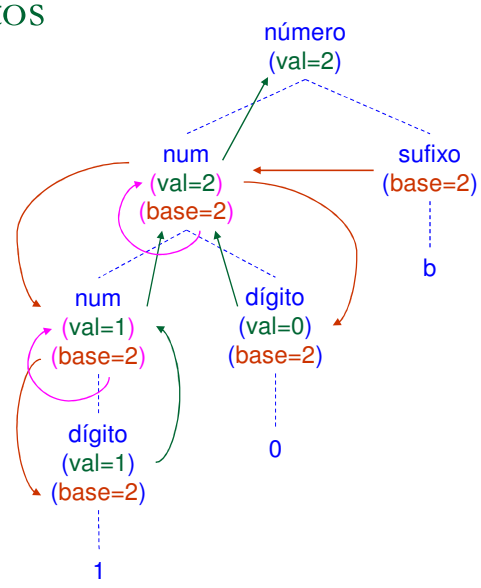
num_1 → num_2 dígito
num_1.val =
num_2.val * num_1.base +
dígito.val

num → dígito
num.val = dígito.val



Cômputo de atributos

- Atributo base e atributo valor e dependência do valor em relação à base



$num_1 \rightarrow num_2$ dígito

num_1.val =
if dígito.val = erro **or** num_2.val=erro **then** erro
else num_2.val * num_1.base + dígito.val

39

Cômputo de atributos

- Dois tipos de atributos
  - Atributos herdados: dependências de pais para filhos e/ou entre irmãos (p.ex., atributo *base* da gramática anterior)
  - Atributos sintetizados: dependências apontam de filhos para pais (p.ex., atributo *val* da gramática anterior)
    - Uma gramática que só tem atributos sintetizados é denominada gramática S-atribuída

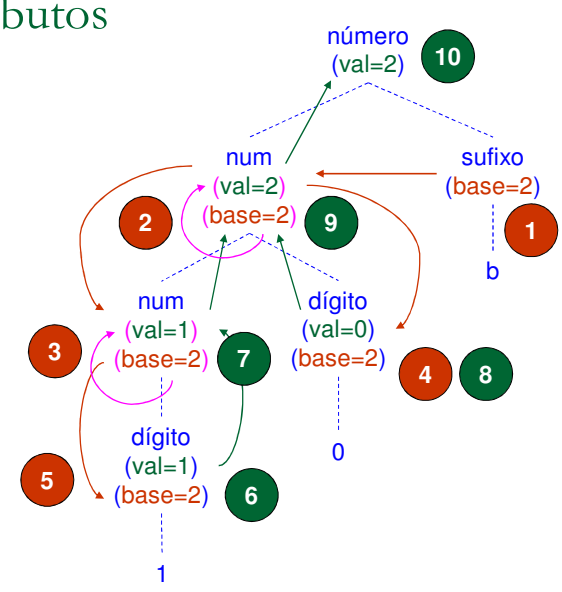
40

Cômputo de atributos

- Ordem de cômputo dos atributos
  - Os atributos que não dependem de outros atributos devem ser computados primeiro, logicamente
  - Opções
    - Ordenação topológica do grafo de dependências: método de árvore de análise sintática
    - Manualmente, determinado pelo projetista do compilador: método baseado em regras

Cômputo de atributos

- Ordenação topológica
  - Várias possibilidades e algoritmos
- Em geral, inicia-se pelos atributos independentes dos nós folha
- Dificuldades do método
  - Construção completa do grafo de dependência
  - Grafos acíclicos para gramáticas de atributos acíclicas (um algoritmo para tal verificação é exponencial em tempo)



Cômputo de atributos

- Método baseado em regras
  - Adotado em praticamente todos os compiladores
  - O projetista do compilador analisa a gramática de atributos e seus grafos de dependência e determina a ordem de cômputo dos atributos
    - Em geral, não é muito complicado de se fazer
  - Para cada regra gramatical, definição do percurso realizado no trecho correspondente na árvore sintática
    - Possibilidades: percurso em-ordem, pré-ordem, pós-ordem ou arbitrário

43

Cômputo de atributos

- Exemplos de percursos para uma árvore binária (relembrando ALG1)
  - Percurso em-ordem
    - Filho esquerdo, raiz (nó pai, isto é, lado esquerdo da regra em foco), filho direito
  - Percurso pré-ordem
    - Raiz, filho esquerdo, filho direito
  - Percurso pós-ordem
    - Filho esquerdo, filho direito, raiz

44

Cômputo de atributos

- Exemplo: sub-rotina para computar o atributo tipo_dado da gramática abaixo

decl → tipo var-lista
tipo → **int** | **float**
var-lista → **id,** var-lista | **id**

Regras gramaticais	Regras semânticas
decl → tipo var-lista	var-lista.tipo_dado = tipo.tipo_dado
tipo → int	tipo.tipo_dado = integer
tipo → float	tipo.tipo_dado = real
var-lista$_1$ → id, var-lista$_2$	id.tipo_dado = var-lista$_1$.tipo_dado var-lista$_2$.tipo_dado = var-lista$_1$.tipo_dado
var-lista → id	id.tipo_dado=var-lista.tipo_dado

45

Cômputo de atributos

- Exemplo: sub-rotina para computar o atributo tipo_dado da gramática abaixo

```
procedure AvalTipo(T: nó_árvore);
begin
    case nó de T of
    decl:
            AvalTipo(tipo);
            var-lista.tipo_dado:=tipo.tipo_dado;
            AvalTipo(var-lista);
    tipo:
            if filho de T=int then T.tipo_dado:=inteiro
            else T.tipo_dado:=real;
    var-lista:
            atribui T.tipo_dado a primeiro filho de T
            if terceiro filho de T não é NIL then
                    atribui T.tipo_dado a terceiro filho;
                    AvalTipo(terceiro filho de T);
    end;
end;
```

pré-ordem

Cômputo de atributos

- Opcionalmente, valores de atributos podem ser associados a parâmetros ou valores de retorno de sub-rotinas de cômputo de atributos em vez de serem armazenados nos nós de uma árvore sintática
  - Interessante para a situação em que muitos atributos são usados apenas temporariamente ou como suporte para cômputo de outros atributos
  - Normalmente, atributos herdados são passados via parâmetros e atributos sintetizados via valor de retorno

Cômputo de atributos

- Exemplo

número → num sufixo
sufixo → b | d
num → num dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

Cômputo de atributos

Regras gramaticais	Regras semânticas
número → num sufixo	número.val = num.val num.base = sufixo.base
sufixo → b	sufixo.base = 2
sufixo → d	sufixo.base = 10
num_1 → num_2 dígito	num_1.val = **if** dígito.val = erro **or** num_2.val=erro **then** erro **else** num_2.val * num_1.base + dígito.val num_2.base = num_1.base dígito.base = num_1.base
num → dígito	num.val = dígito.val dígito.base = num.base
dígito → 0	dígito.val = 0
dígito → 1	dígito.val = 1
dígito → 2	dígito.val = **if** dígito.base=2 **then** erro **else** 2
...	...

```
function AvalComBase(T: nó_árvore; base: inteiro): inteiro;
var temp, temp2: inteiro;
begin
    case nó de T of
        número:
                temp:=AvalComBase(filho à direita de T,0);
                return AvalComBase(filho à esquerda de T,temp);
        num:
                temp:=AvalComBase(filho à esquerda de T,base);
                if filho à direita de T não é NIL then
                        temp2:=AvalComBase(filho à direita de T,base);
                        if temp<>erro and temp2<>erro then
                                return base*temp+temp2
                        else return erro;
                else return temp;
        sufixo:
                if filho de T=b then return 2
                else return 10;
        dígito:
                if base=2 and filho de T>1 then return erro
                else return numval(filho de T);
    end;
end;
```