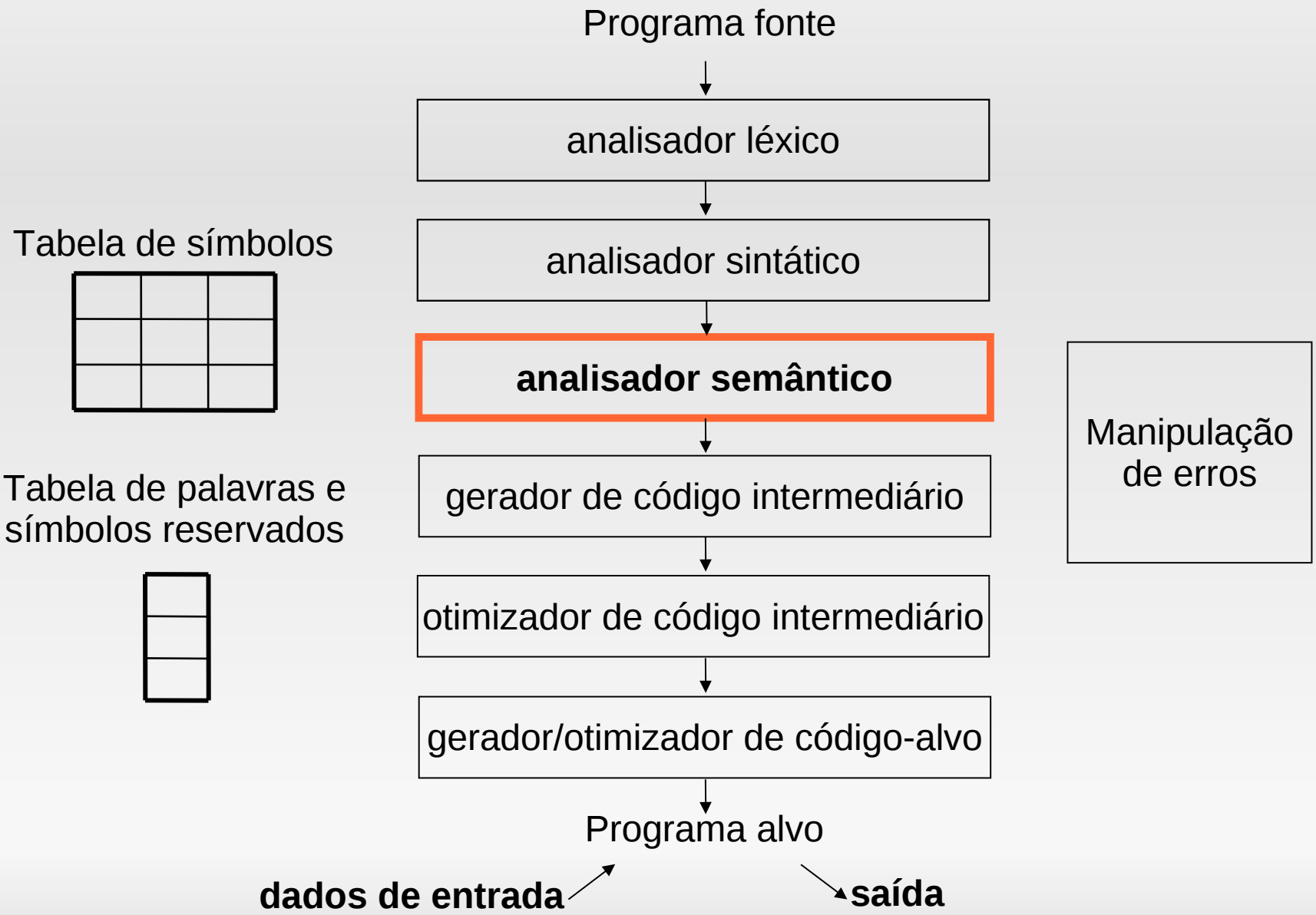
# Construção de Compiladores

## Análise Semântica – parte 1

Profa. Helena Caseli
helenacaseli@dc.ufscar.br

# Processo de Tradução



# Análise Semântica

- O que é?
  - Etapa na qual são realizadas verificações para assegurar que os componentes do programa se combinam de forma significativa
  - É tarefa do analisador semântico
    - Realizar verificações de tipo e declarações
    - Manipular a tabela de símbolos
    - → Garantir a ”corretude” de coisas que vão além do domínio da sintaxe
      - Sensitivade ao contexto

# Análise Semântica

- Tipos
  - 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, etc.

# Análise Semântica

- Como é feita?
  - Assim como a sintaxe, a semântica precisa ser formalizada/descrita antes de ser implementada
    - Análise sintática: regras → procedimentos recursivos
  - → Em geral a semântica de uma linguagem de programação não é especificada
    - O projetista do compilador tem que analisar a linguagem e extrair a semântica
  - Descrição
    - Gramática de atributos
    - Árvore de sintaxe abstrata

# Análise Semântica

- Como é feita?
  - Implementação
    - Semântica dirigida pela sintaxe
      - Conteúdo semântico fortemente relacionado à sintaxe do programa
      - Maioria das linguagens de programação modernas

# Análise Semântica

- Gramática de atributos
  - Atributo
    - Qualquer propriedade de uma construção de linguagem de programação
      - Tipo de dados de uma variável
      - Valor de uma expressão
      - Localização de uma variável na memória
      - Código-objeto de um procedimento
    - Amarração
      - Processo de computar o valor de um atributo e associar seu valor à construção da linguagem em questão
    - Tempo de amarração
      - Momento durante a compilação (amarração estática) ou execução (amarração dinâmica) em que ocorre a amarração de um atributo

# Análise Semântica

- Gramática de atributos
  - Tipos de atributos de acordo com o tempo de amarração
    - **Estáticos**
      - Podem ser amarrados antes da execução
        - Tipo de dados de uma variável estática (C, Pascal)
        - Código-objeto de um procedimento
    - **Dinâmicos**
      - **Só** podem ser amarrados durante a execução
        - Valor de uma expressão que não seja constante
        - Localização de uma estrutura de dados alocada dinamicamente

# Análise Semântica

- Gramática de atributos
  - Conjunto de **atributos** e **regras semânticas** para uma gramática
    - Cada regra sintática/gramatical tem uma regra semântica associada
    - **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 *valor* de duas variáveis e atribuir a uma terceira
        - x:=a+b, cuja regra é x.valor=a.valor+b.valor

# Análise Semântica

- Gramática de atributos
  - Princípio da semântica dirigida pela sintaxe
    - Dada a coleção de atributos $a_1, ..., a_k$
      - Para cada regra gramatical $X_o \rightarrow X_1 X_2 ... X_n$ , onde $X_o$ é um não-terminal e os outros $X_i$ são símbolos arbitrários
      - Os valores dos atributos $X_i.a_j$ de cada símbolo gramatical $X_i$ são relacionados aos valores dos atributos dos outros símbolos na regra

    - Se o mesmo símbolo $X_i$ aparecer mais de uma vez na regra gramatical, cada ocorrência deve ser diferenciada

# Análise Semântica

- Gramática de atributos
  - Exemplo

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

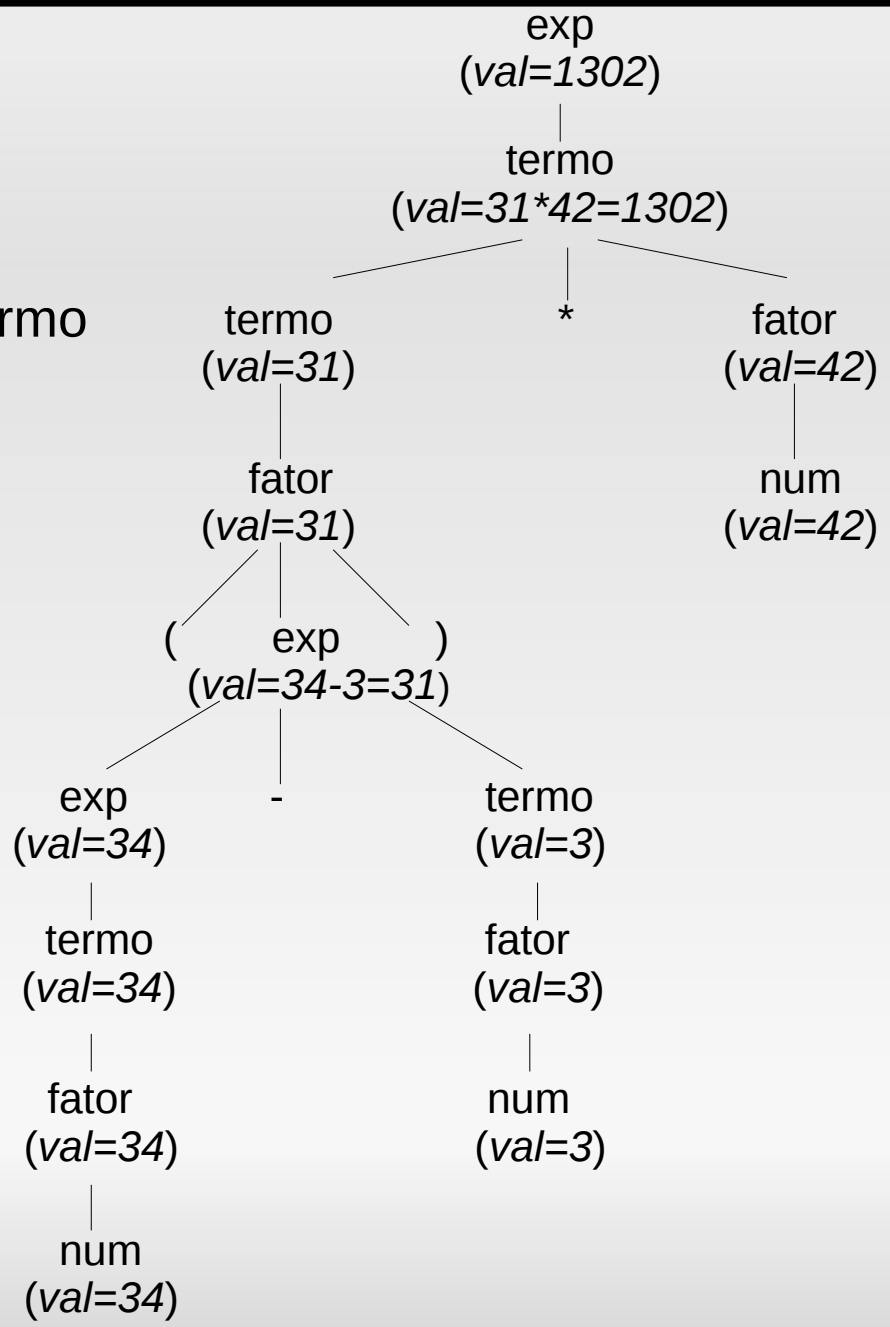
Como seria a árvore sintática com a computação de atributos para a cadeia **(34-3)*42**

| 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→fator | termo.val=fator.val |
| fator→(exp) | fator.val=exp.val |
| fator→num | fator.val=num.val |

# Análise Semântica

- Gramática de atributos
  - Exemplo

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

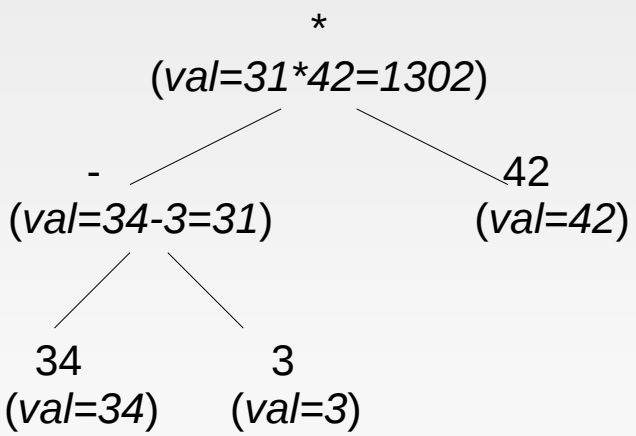


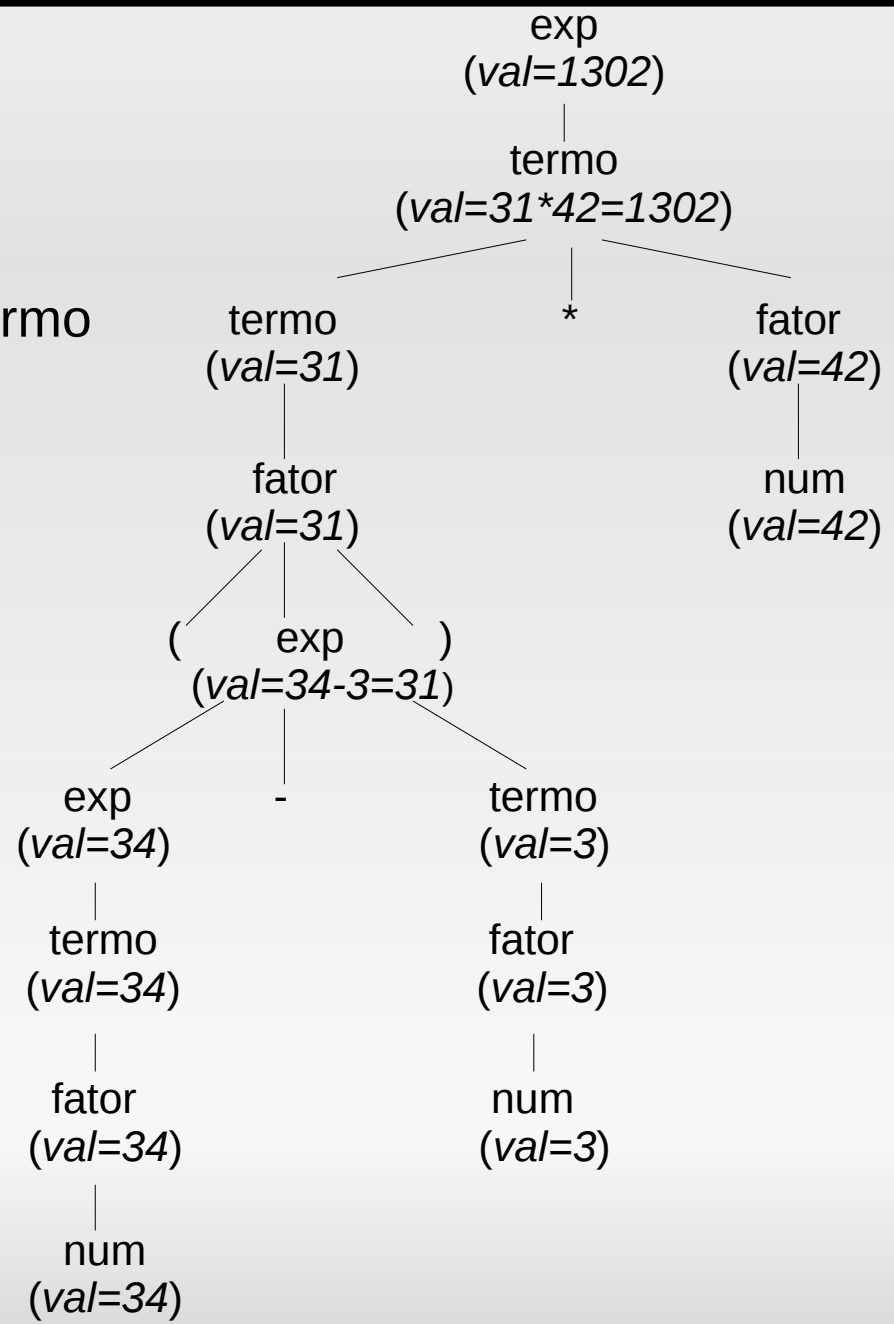
# Análise Semântica

- Gramática de atributos
  - Exemplo

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

Árvore de sintaxe abstrata



# Análise Semântica

- Gramática de atributos

A árvore abstrata pode ser definida pela gramática de atributos a seguir, na qual:
- mkOpNode – cria um nó para o 1o. parâmetro com filhos os outros 2 parâmetros
- mkNumNode – cria um nó folha com o valor do parâmetro
- num.lexval – é o valor numérico identificado pelo analisador léxico

| **Regras gramaticais** | **Regras semânticas** |
|---|---|
| $exp_1 \rightarrow exp_2$+termo | $exp_1$.árvore=mkOpNode(+,$exp_2$.árvore,termo.árvore) |
| $exp_1 \rightarrow exp_2$-termo | $exp_1$.árvore=mkOpNode(-,$exp_2$.árvore,termo.árvore) |
| exp→termo | exp.árvore=termo.árvore |
| $termo_1 \rightarrow termo_2$*fator | $termo_1$.árvore=mkOpNode(*,$termo_2$.árvore,fator.árvore) |
| termo→fator | termo.árvore=fator.árvore |
| fator→(exp) | fator.árvore=exp.árvore |
| fator→num | fator.árvore=mkNumNode(num.lexval) |

# Análise Semântica

- Exercício
  - Dada a gramática a seguir para números sem sinal
  - Escreva a gramática de atributos correspondente

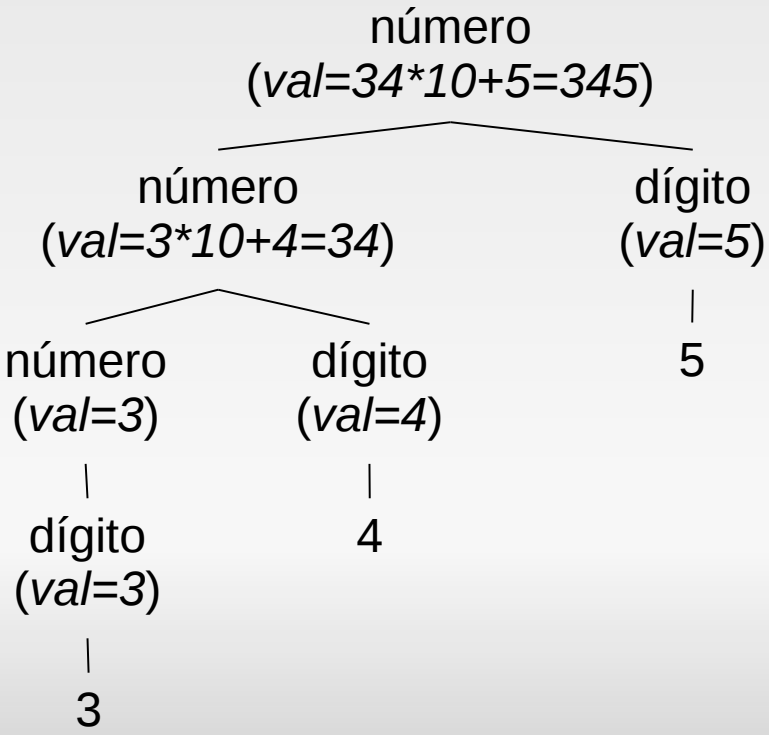
número → número dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

| Regras gramaticais | Regras semânticas |
|---|---|
| $número_1 \rightarrow número_2$ 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 |

# Análise Semântica

- Exercício
  - Dada a gramática a seguir para números sem sinal
  - Escreva a gramática de atributos correspondente

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




Árvore sintática com visualização da computação de atributos para a cadeia **345**

# Análise Semântica

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

# Análise Semântica

- Exercício
  - Dada a gramática para números binários ou decimais, indicados pelos sufixos b ou d, respectivamente
  - Escreva a gramática de atributos que dê o valor decimal correspondente
    - DICA – use 2 atributos: valor (val) e base

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

# Análise Semântica

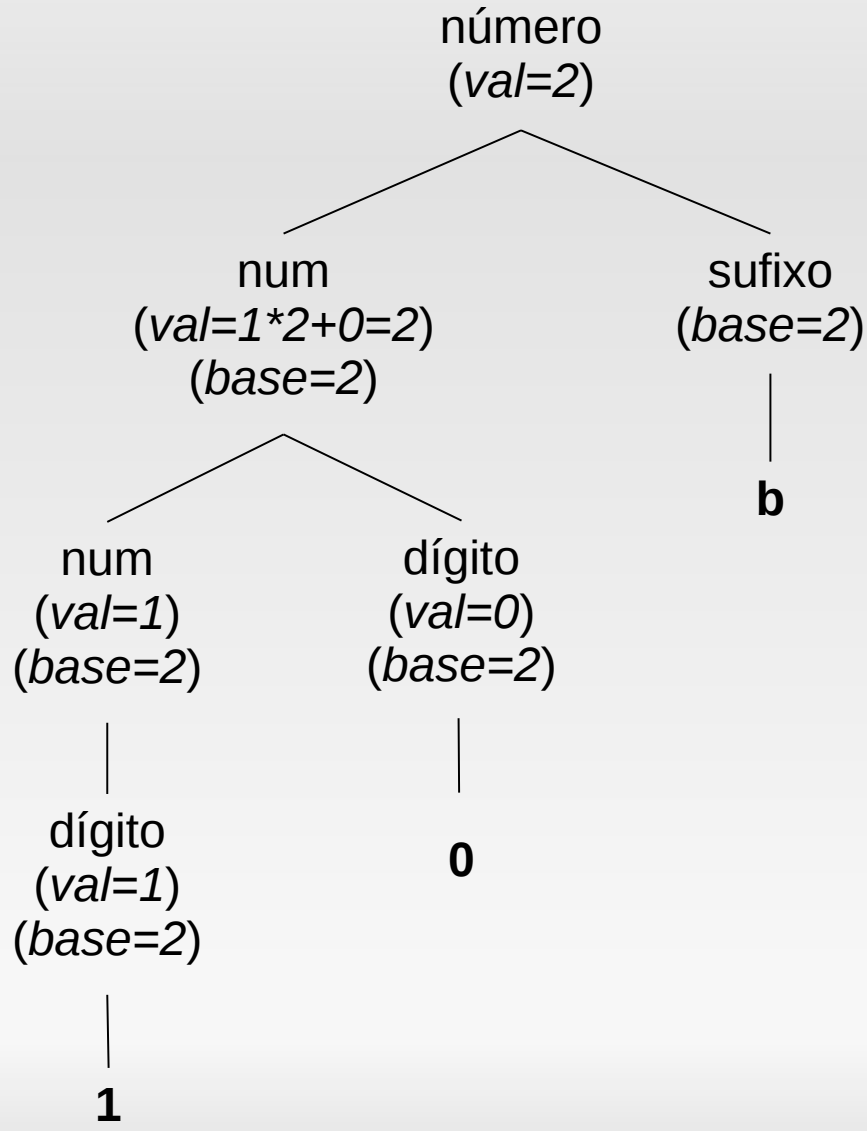
| Regras gramaticais | Regras semânticas |
|---|---|
| número → num sufixo | número.val = num.val<br>num.base = sufixo.base |
| sufixo → b | sufixo.base = 2 |
| sufixo → d | sufixo.base = 10 |
| $num_1$ → $num_2$ dígito | $num_1$.val =<br>**if** dígito.val = erro **or** $num_2$.val=erro **then** erro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

# Análise Semântica

- Exercício

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

Árvore sintática com cálculo dos atributos para a cadeia **10b**

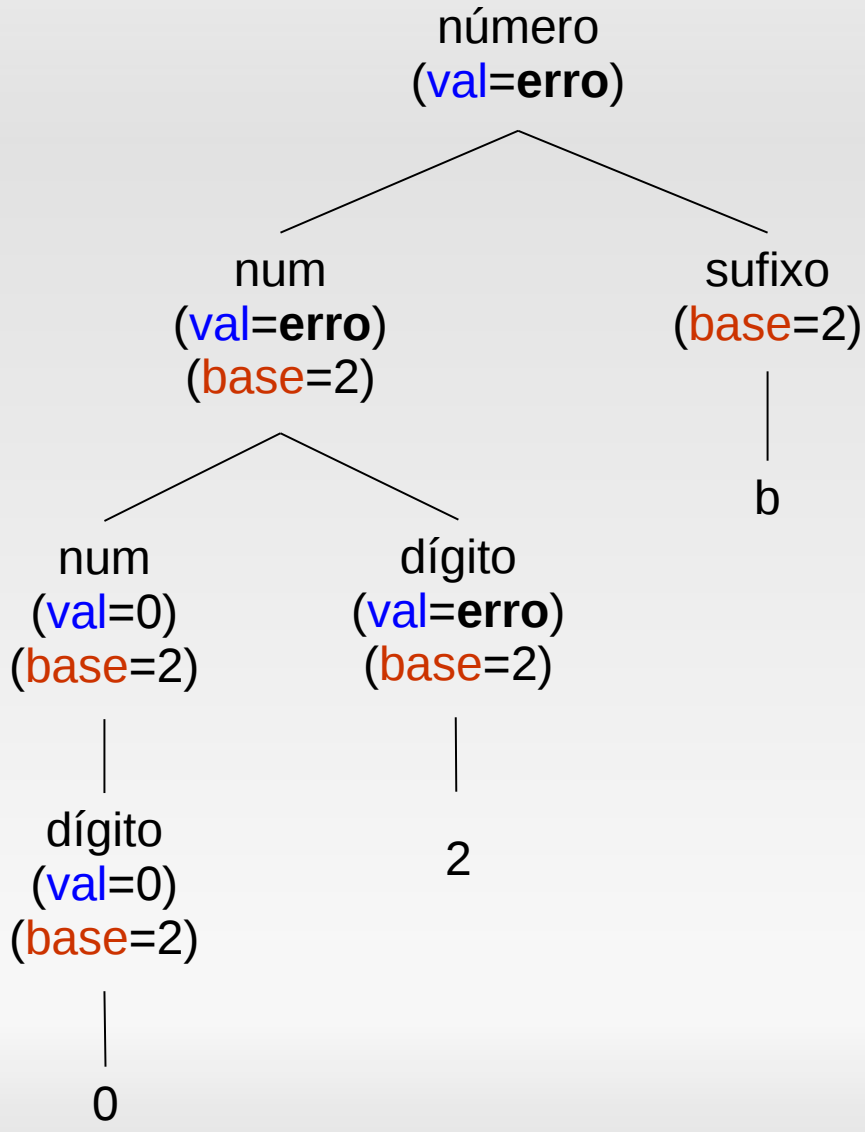


# Análise Semântica

- Exercício

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

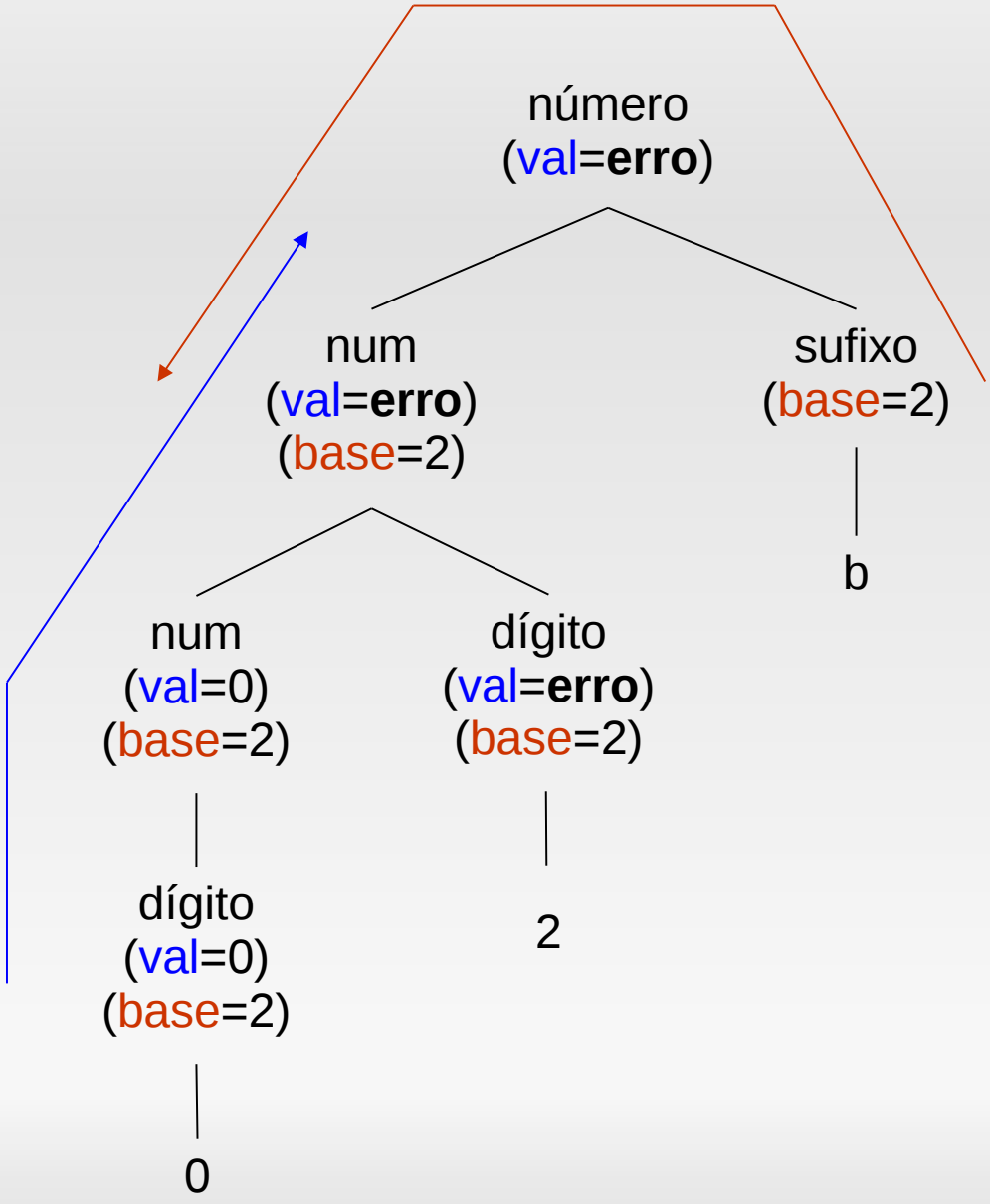


# Análise Semântica

- Exercício

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

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

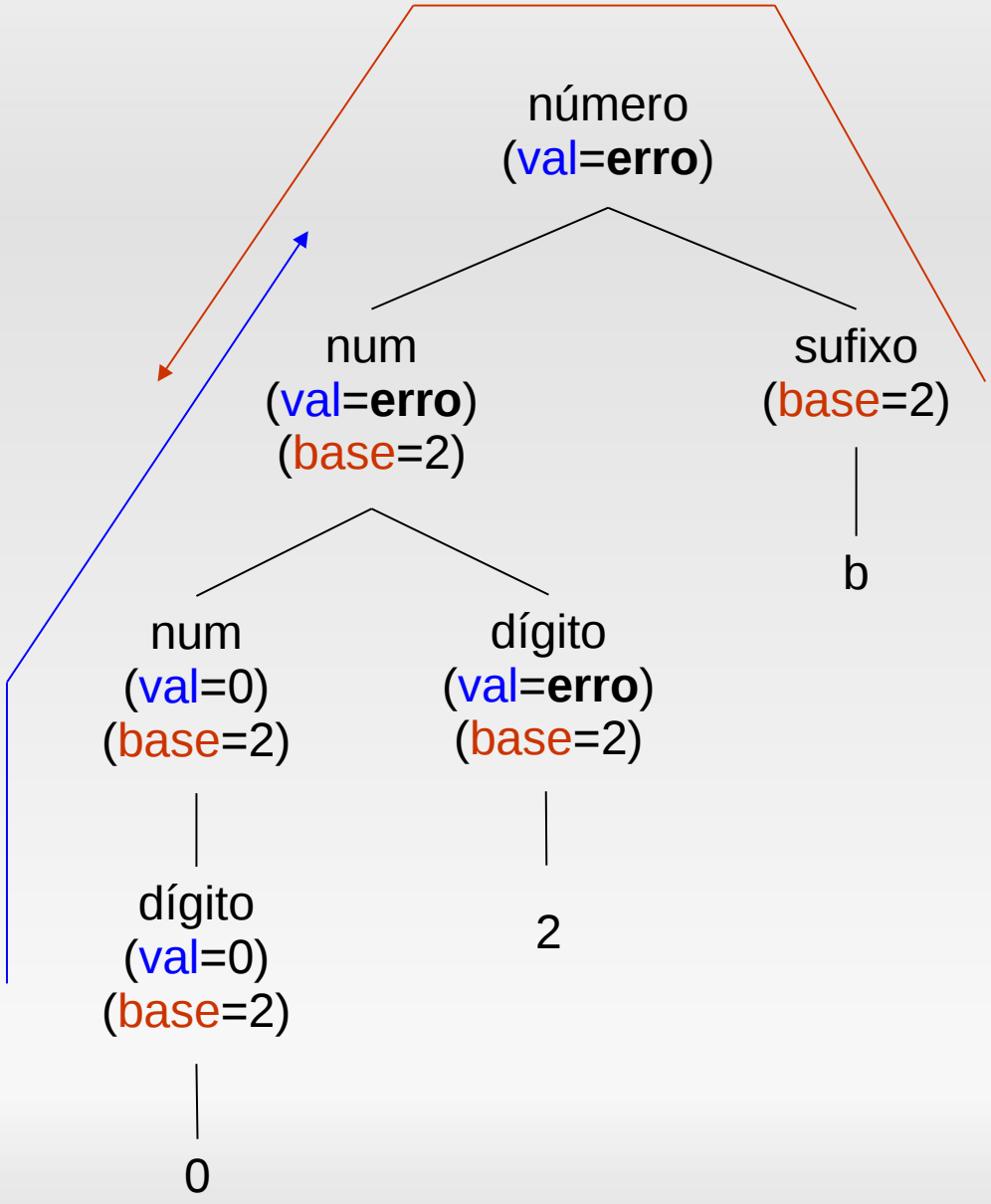


# Análise Semântica

- Exercício

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

Como calcular os atributos de forma consistente?



# Análise Semântica

- Algoritmos para computação de atributos
  - Grafos de dependência
    - Especificam a ordem de cálculo dos atributos de cada regra gramatical em uma árvore sintática
      - Indicam as dependências entre atributos
      - Um grafo associado a cada regra gramatical
      - Cada grafo tem um nó rotulado para cada atributo $X_i.a_j$ de cada símbolo na regra gramatical e para cada equação

        $$X_i.a_j = f(..., X_m.a_k, ...)$$

        existe um arco direcionado partindo de cada nó $X_m.a_k$ à direita para o nó $X_i.a_j$

    - Para uma cadeia da linguagem, tem-se um grafo composto por todos os subgrafos

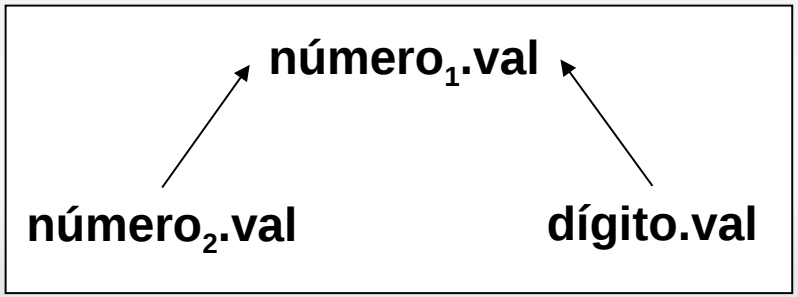
# Análise Semântica

- Exemplo

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

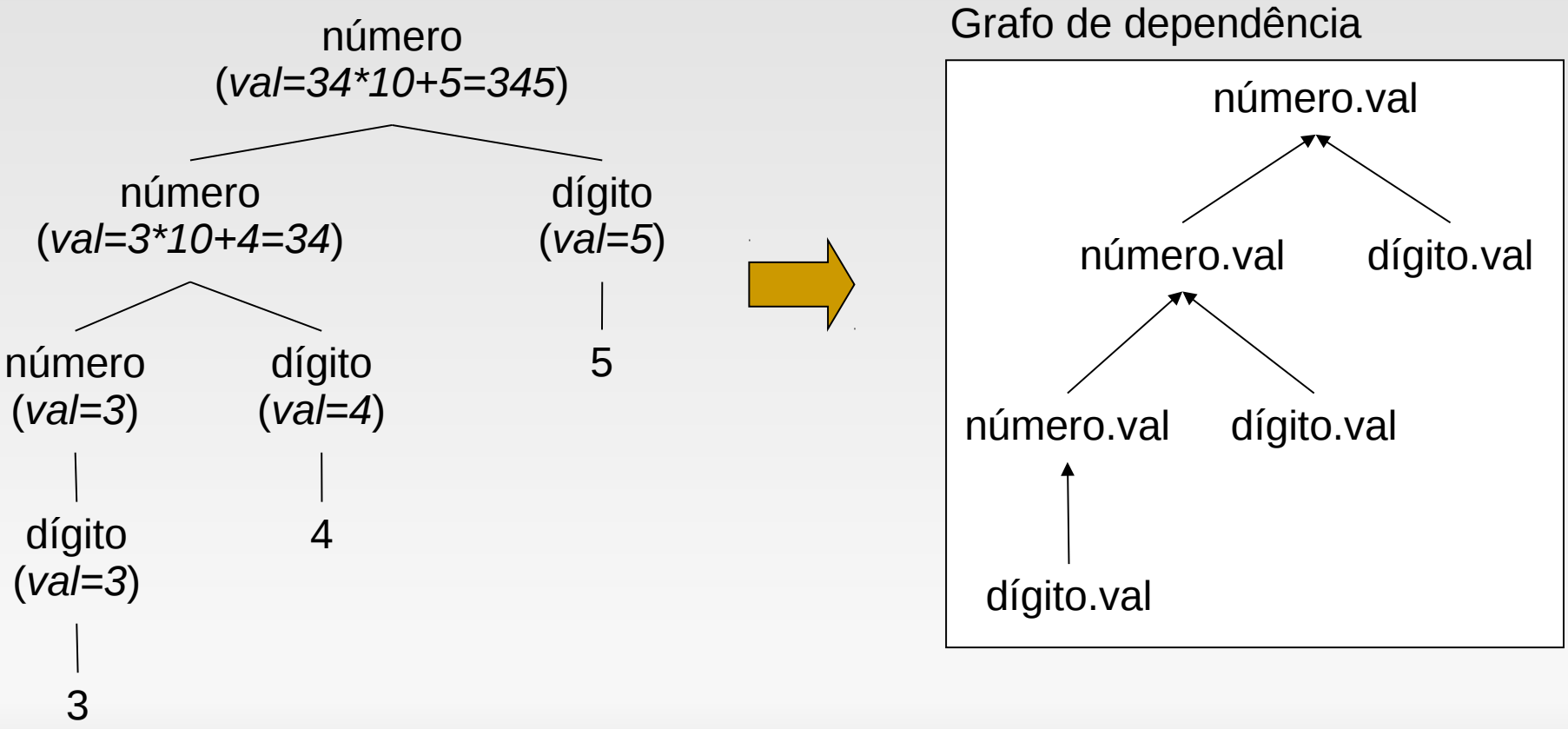
Qual seria o grafo de dependências para a cadeia 345?

| Regras gramaticais | Regras semânticas |
|---|---|
| $número_1 \rightarrow número_2$ 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 |



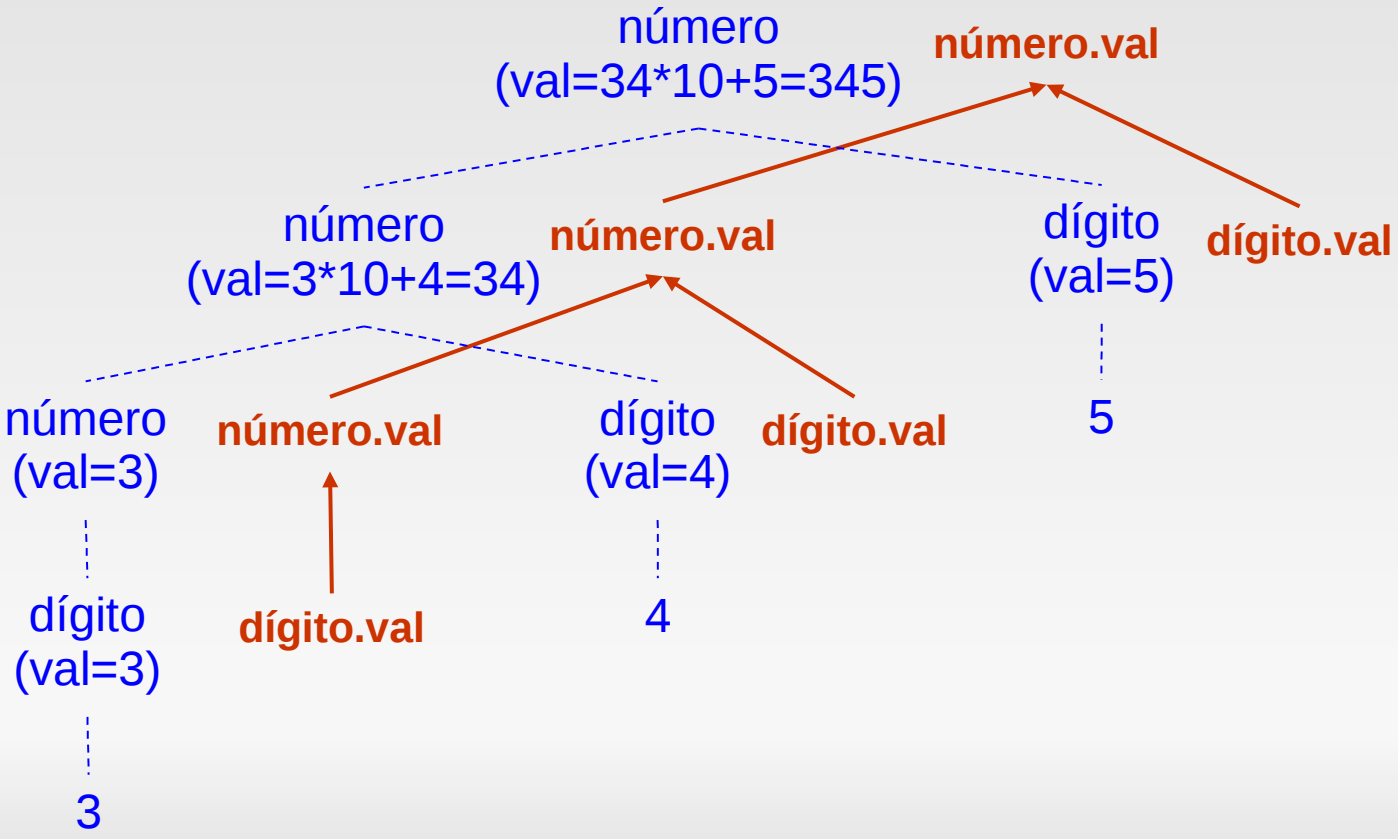
# Análise Semântica

- Exemplo
  - Grafo de dependências para a cadeia 345



# Análise Semântica

- Exemplo
  - Grafo de dependências para a cadeia 345 amarrado à árvore sintática



# Análise Semântica

- Exercício
  - Dada a gramática para números binários ou decimais, indicados pelos sufixos b ou d e a gramática de atributos correspondente

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

# Análise Semântica

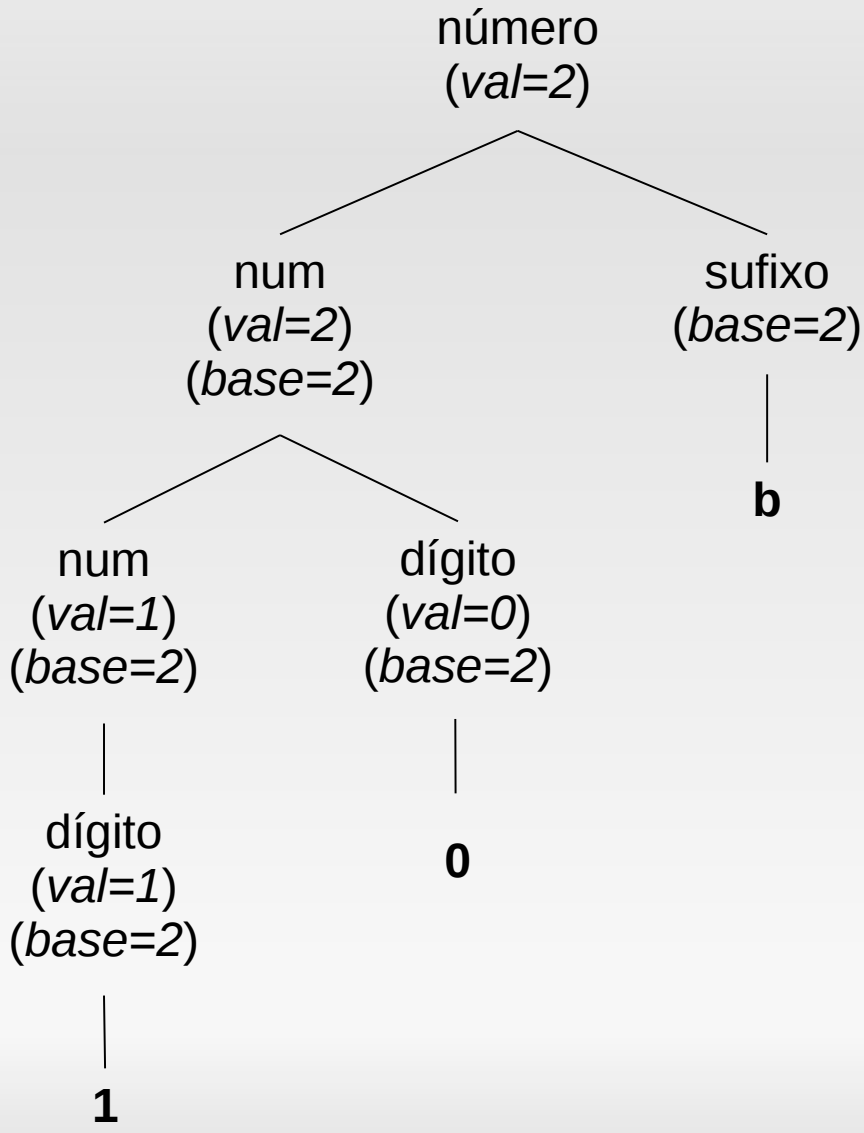
| Regras gramaticais | Regras semânticas |
|---|---|
| número → num sufixo | número.val = num.val<br>num.base = sufixo.base |
| sufixo → b | sufixo.base = 2 |
| sufixo → d | sufixo.base = 10 |
| $num_1$ → $num_2$ dígito | $num_1$.val =<br>**if** dígito.val = erro **or** $num_2$.val=erro **then** erro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

# Análise Semântica

- Exercício

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

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



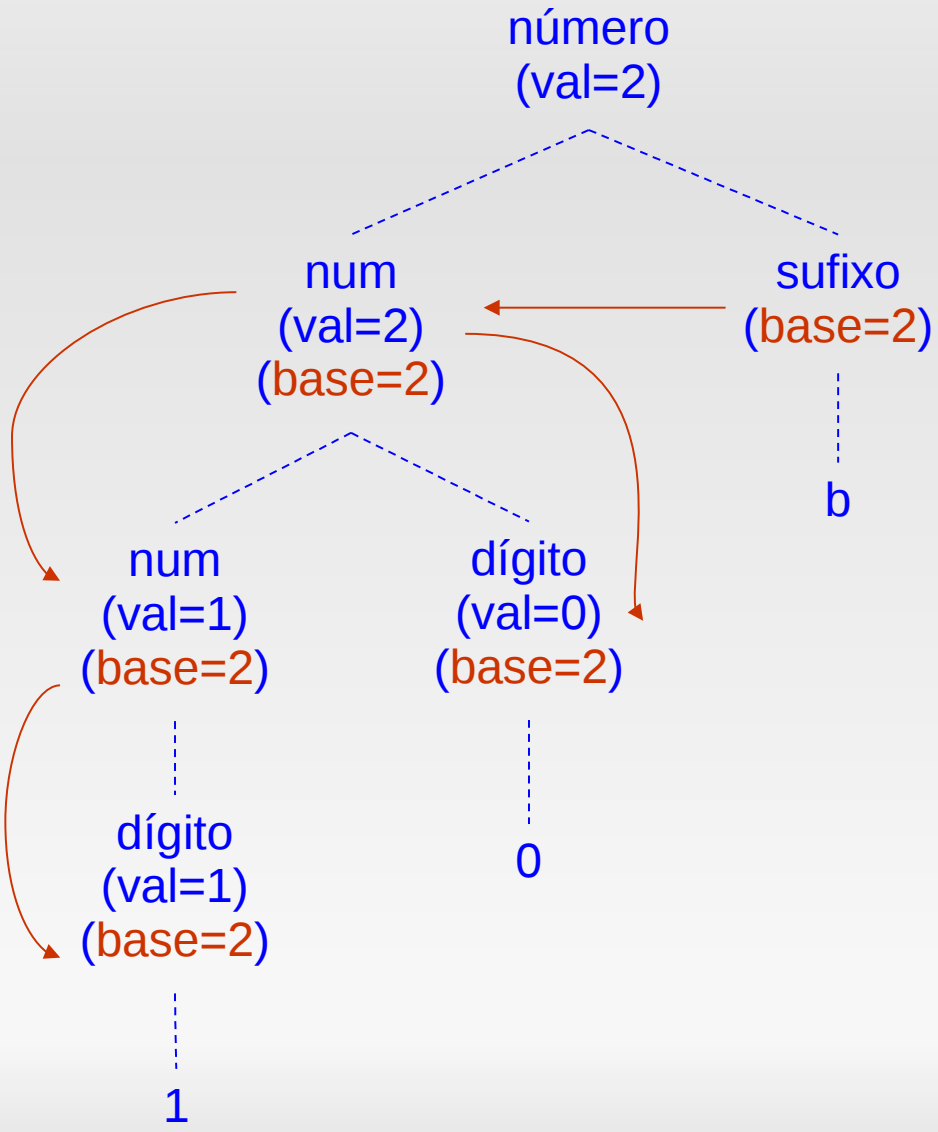
# Análise Semântica

- Exercício

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

Atributo base – regras semânticas

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



# Análise Semântica

- Exercício

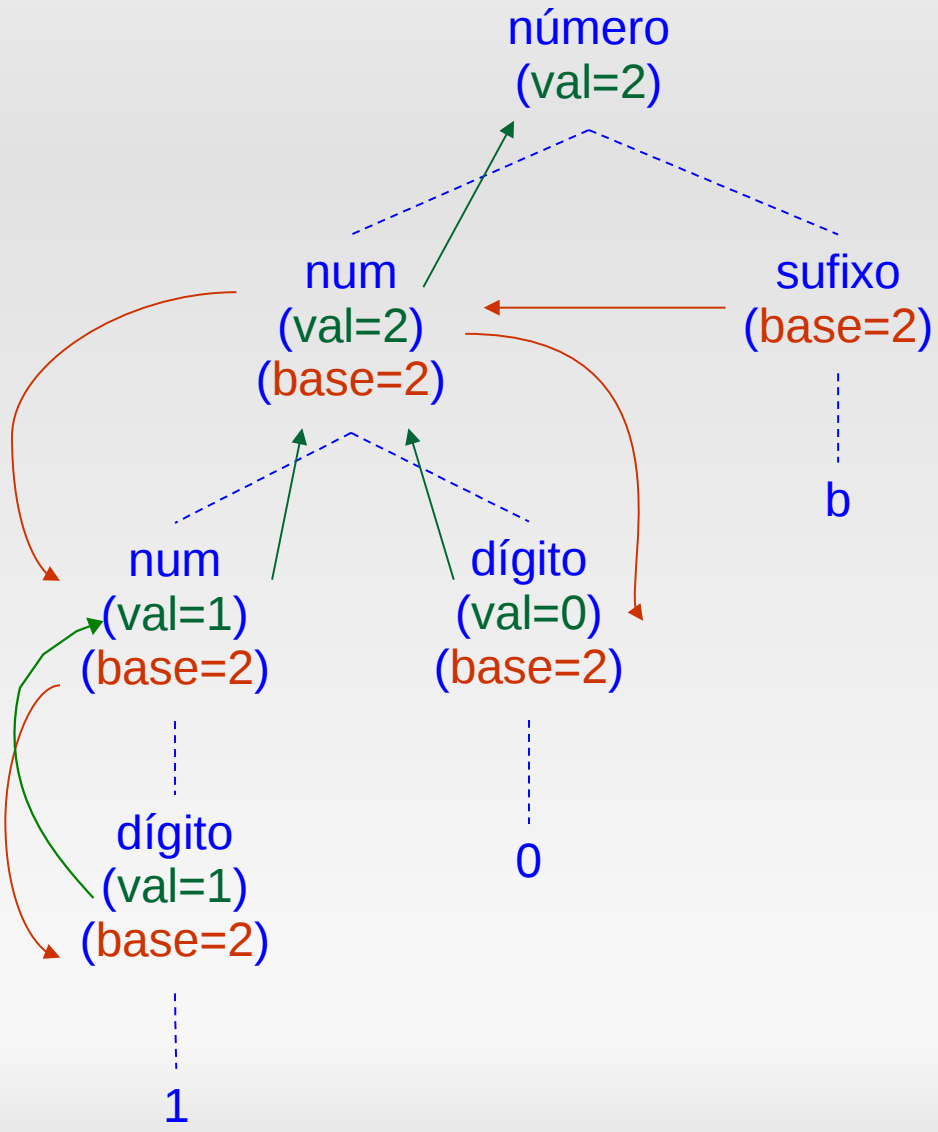
número → num sufixo
sufixo → b | d
num → num dígito | dígito
dígito → 0|1|2|3|4|5|6|7|8|9

**Atributo val – regras semânticas**

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



# Análise Semântica

- Exercício

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

Dependência de val em relação a base

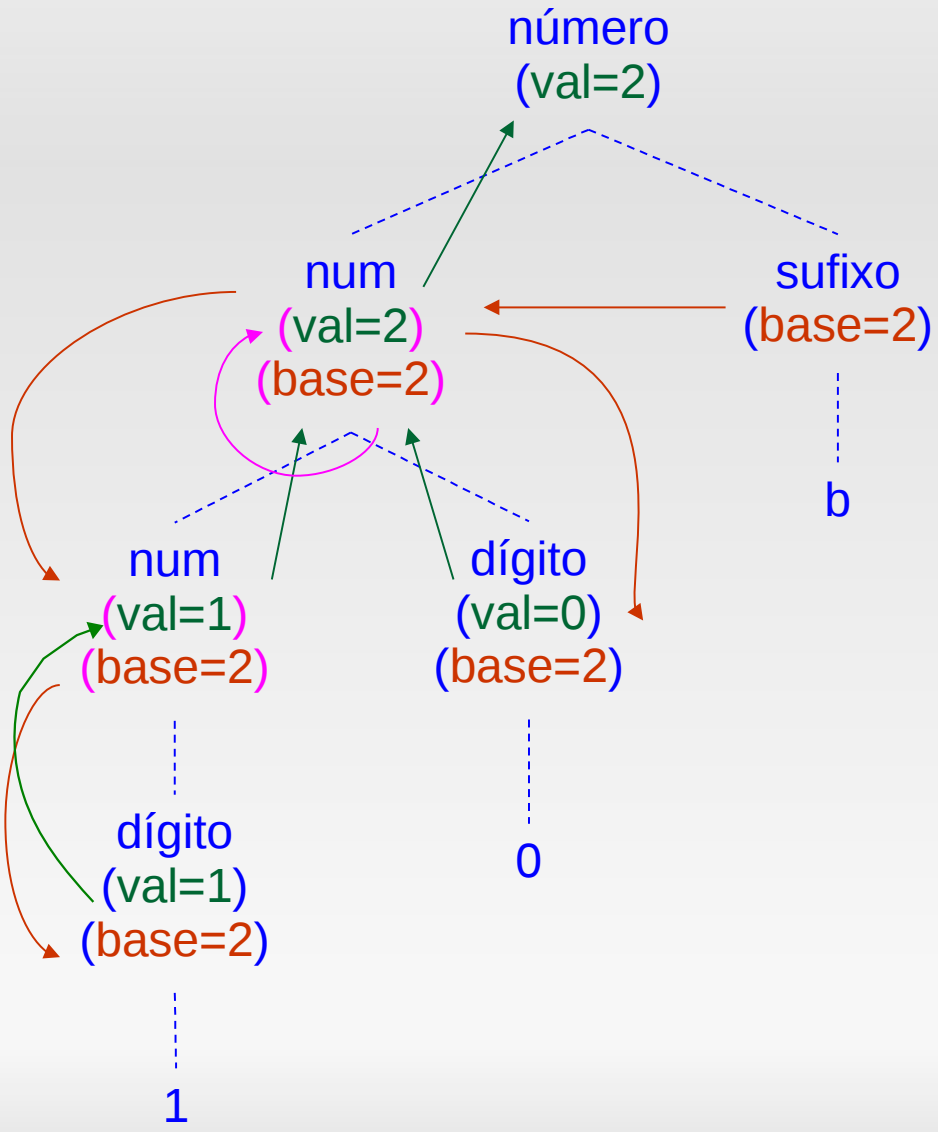
$num_1$ → $num_2$ dígito

**$num_1$.val** =

**if** dígito.val = erro **or** $num_2$.val=erro

**then** erro

**else** $num_2$.val*__$num_1$.base__+dígito.val



# Análise Semântica

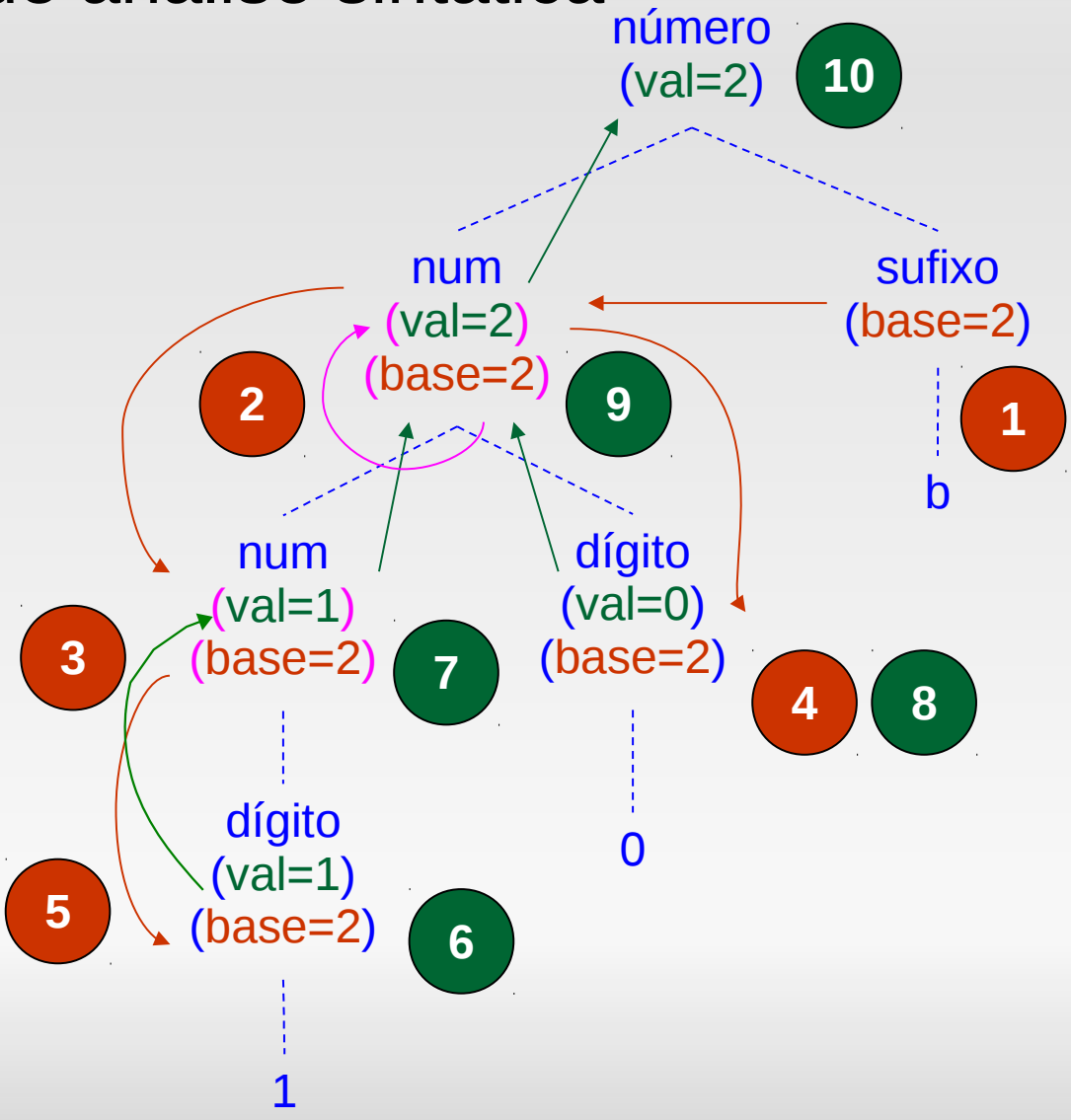
- Algoritmos para computação de atributos
  - Ordem de cálculo dos atributos
    - Os atributos que não dependem de outros atributos devem ser computados primeiro
    - 2 métodos
      - **Método de árvore de análise sintática**
        - Ordenação topológica do grafo de dependências
        - Em tempo de compilação
        - → Avalia atributos em gramáticas de atributos não circulares
      - **Método baseado em regras**
        - O projetista do compilador determina manualmente
        - Em tempo de construção do compilador
        - → Alternativa adotada em praticamente todos os compiladores

# Análise Semântica

- Método de árvore de análise sintática

**Ordenação topológica**

- Todos os atributos usados no cálculo do atributo atual já devem estar disponíveis
- Várias possibilidades e algoritmos
- Em geral, inicia-se pelos atributos independentes dos nós folha

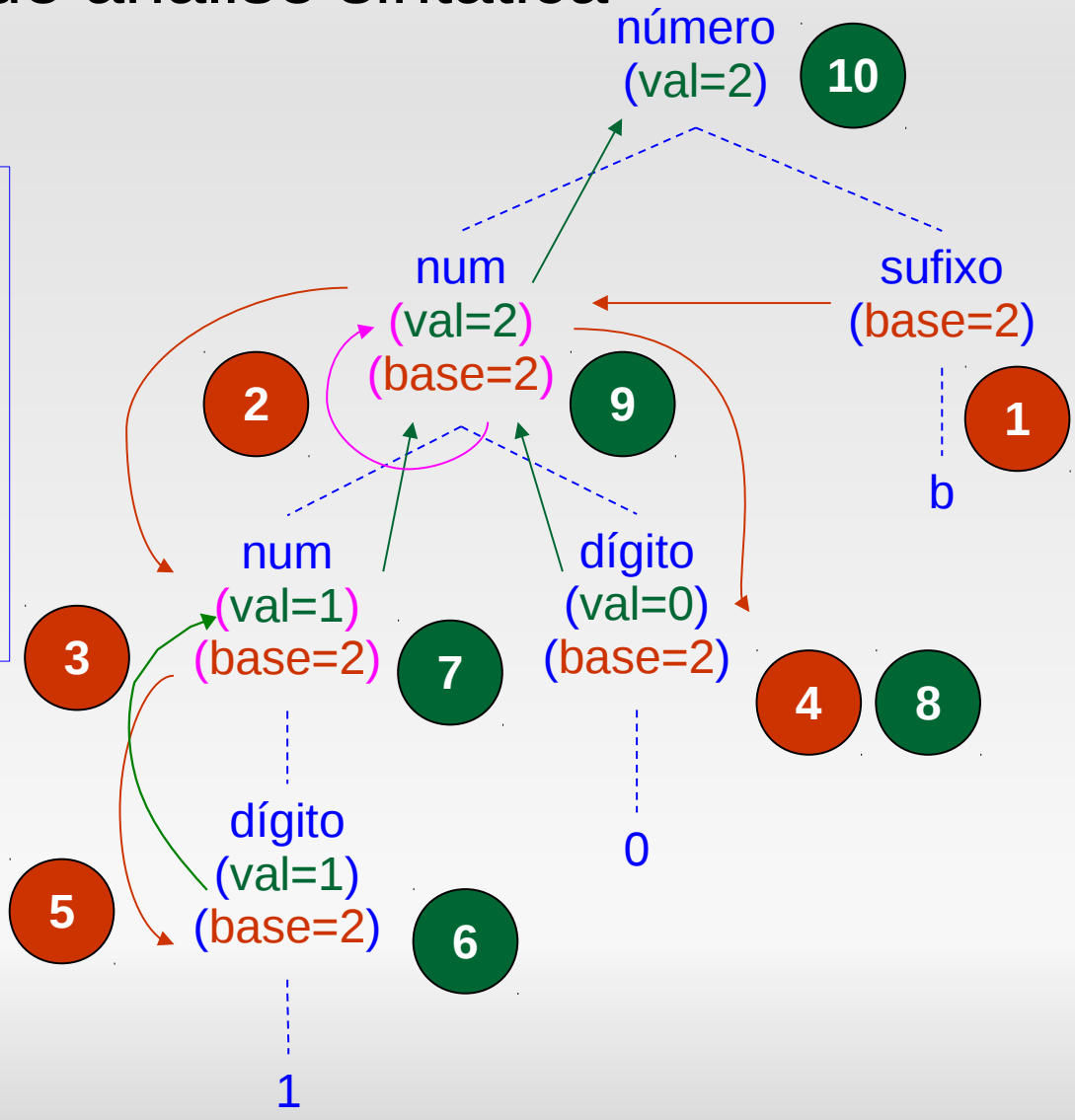


# Análise Semântica

- Método de árvore de análise sintática

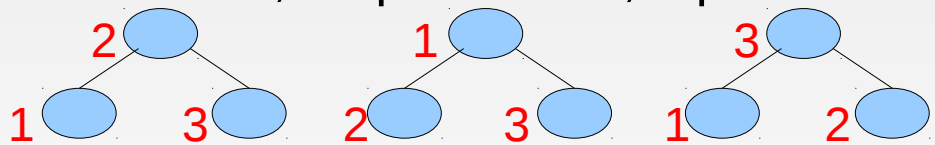
**Problemas**

- Construção complexa do grafo de dependência
- Grafos acíclicos para gramáticas de atributos acíclicas (um algoritmo para tal verificação tem tempo exponencial)



# Análise Semântica

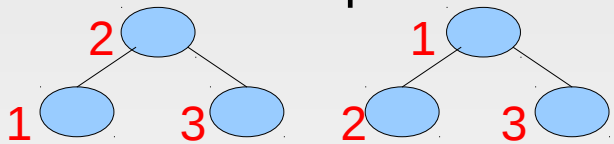
- Método baseado em regras
  - Adotado em praticamente todos os compiladores
  - Como funciona
    - O projetista do compilador analisa a gramática de atributos e seus grafos de dependência e determina a ordem de cálculo dos atributos
    - Em geral, não é muito complicado de se fazer
    - Para cada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



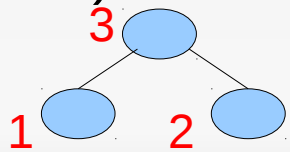

    - ➔ Diferentes percursos para diferentes tipos de atributos

# Análise Semântica

- Método baseado em regras
  - Dois tipos de atributos
    - Atributos herdados
      - Os valores são computados a partir dos valores dos atributos dos irmãos ou do pai (p.ex., atributo *base* da gramática anterior)
        - em-ordem e pré-ordem




    - Atributos sintetizados
      - Os valores são computados exclusivamente a partir dos valores dos atributos dos filhos (p.ex., atributo *val* da gramática anterior)
        - pós-ordem




    - ➔ Uma gramática que só tem atributos sintetizados é denominada gramática S-atribuída

# Análise Semântica

- Método baseado em regras
  - Questões de implementação
    - Opcionalmente, os valores de atributos podem ser associados a parâmetros ou valores de retorno de sub-rotinas de cálculo de atributos (ao invés 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 cálculo de outros atributos
      - Normalmente, atributos herdados são passados via parâmetros e atributos sintetizados via valor de retorno

# Análise Semântica

- Método baseado em regras
  - Exemplo

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

# Análise Semântica

| Regras gramaticais | Regras semânticas |
|---|---|
| número → num sufixo | número.val = num.val<br>num.base = sufixo.base |
| sufixo → b | sufixo.base = 2 |
| sufixo → d | sufixo.base = 10 |
| $num_1$ → $num_2$ dígito | $num_1$.val =<br>**if** dígito.val= erro **or** $num_2$.val=erro **then** erro<br>**else** $num_2$.val * $num_1$.base + dígito.val<br>$num_2$.base = $num_1$.base<br>dígito.base = $num_1$.base |
| num → dígito | num.val = dígito.val<br>dígito.base = num.base |
| dígito → 0 | dígito.val = 0 |
| ... | .. |
| dígito → 9 | dígito.val =<br>**if** dígito.base=2 **then** erro **else** 9 |

# Análise Semântica

```
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 lexval(filho de T) > 1 then return erro
                else return lexval(filho de T);
    end;
end;
```

Atributo sintetizado: *val*

Atributo herdado: *base*

# Análise Semântica

```
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 lexval(filho de T) > 1 then return erro
            else return lexval(filho de T);
    end;
end;
```

*val* é computado em pós-ordem

*base* é computado em pré-ordem

**Uma única passada**
- atributo herdado não depende de ninguém

# Análise Semântica

```
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 lexval(filho de T) > 1 then return erro
                else return lexval(filho de T);
    end;
end;
```

*base* é computado logo no início

# Análise Semântica

```
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 lexval(filho de T) > 1 then return erro
            else return lexval(filho de T);
    end;
end;
```

*lexval* é uma função que retorna o valor numérico identificado pelo analisador léxico

# Análise Semântica

- Cálculo de atributos X Análise sintática
  - Processamento da esquerda para a direita: LL e LR
    - Exige que os atributos sejam avaliados por um percurso da esquerda para a direita
      - Atributos sintetizados – OK
        - Os filhos podem ser processados em qualquer ordem
      - Atributos herdados – PROBLEMA
        - Implica em não existir dependências que apontem da direita para a esquerda (p. ex., *val* não pode ser computado até que o sufixo, que está no final da cadeia, seja conhecido)

# Análise Semântica

- Cálculo de atributos X Análise sintática
  - Processamento da esquerda para a direita: LL e LR
    - Gramáticas L-atribuídas
      - Restringem o uso de atributos herdados para permitir que as ações semânticas possam ser executadas durante a análise sintática em uma única passada
      - 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 (na mesma regra de produção)

    - → Uma gramática S-atribuída é L-atribuída

# Análise Semântica

- Método baseado em regras
  - Cálculo em uma única passada
    - Para gramáticas L-atribuídas
      - Combinação dos procedimentos pós-eval e pré-eval

```
procedimento Combinado-eval (T: nó-arvore);
inicio
   para cada filho C de T faça
      compute cada atributo herdado de C;
      Combinado-eval(C);
   compute cada atributo sintetizado de T;
fim;
```

  - Cálculo em mais passadas
    - Atributos herdados dependem dos atributos sintetizados
      - Situações complexas que exigem mais de uma passada