Descrevendo Sintaxe e Semântica
Parte 2

Prof. Dr. Eduardo Takeo Ueda
eduardo.tueda@sp.senac.br

Gramática de Atributos (1/6)

- Serve para descrever tanto a sintaxe quanto a semântica estática
- É um dispositivo usado para descrever melhor a estrutura de uma linguagem de programação do que o permitido pelas BNF
- Existem algumas características de linguagens que são difíceis de descrever usando a notação BNF
  - Compatibilidade de tipos:
    - A gramática ficaria muito grande
  - Variáveis que devem ser declaradas antes de serem utilizadas:
    - Impossivel
  - O end de um subprograma em ADA deve casar com o nome do subprograma

Gramática de Atributos (2/6)

- Estes problemas exemplificam a categoria de regras de linguagem chamada de regras de semântica estática
- Tem pouca relação com significado do programa:
  - Sintaxe ao invés de semântica
- A análise requerida para checar estas especificações podem ser feitas em tempo de compilação
- Gramática com atributos são gramáticas para as quais foram adicionadas atributos
- Atributos intrínsecos possuem valores determinados fora da árvore de derivação. O tipo de uma variável pode vir de uma tabela de símbolos

Gramática de Atributos (3/6)

- O nome no end de um procedimento ADA deve combinar com o nome do procedimento

- Regra sintática:
  - < proc_def > → **procedure** < proc_name >[1]
    < proc_body > **end** < proc_name >[2]**;**

- Regra semântica:
  - < proc_name >[1].string = <proc_name>[2].string

- Quando existe mais de uma ocorrência de um não-terminal, este é indexado com colchetes para diferencia-los

Gramática de Atributos (4/6)

1. Sintaxe: <assign> → <var> **=** <expr>
Semântica: <expr>.expected_type ← <var>.actual_type

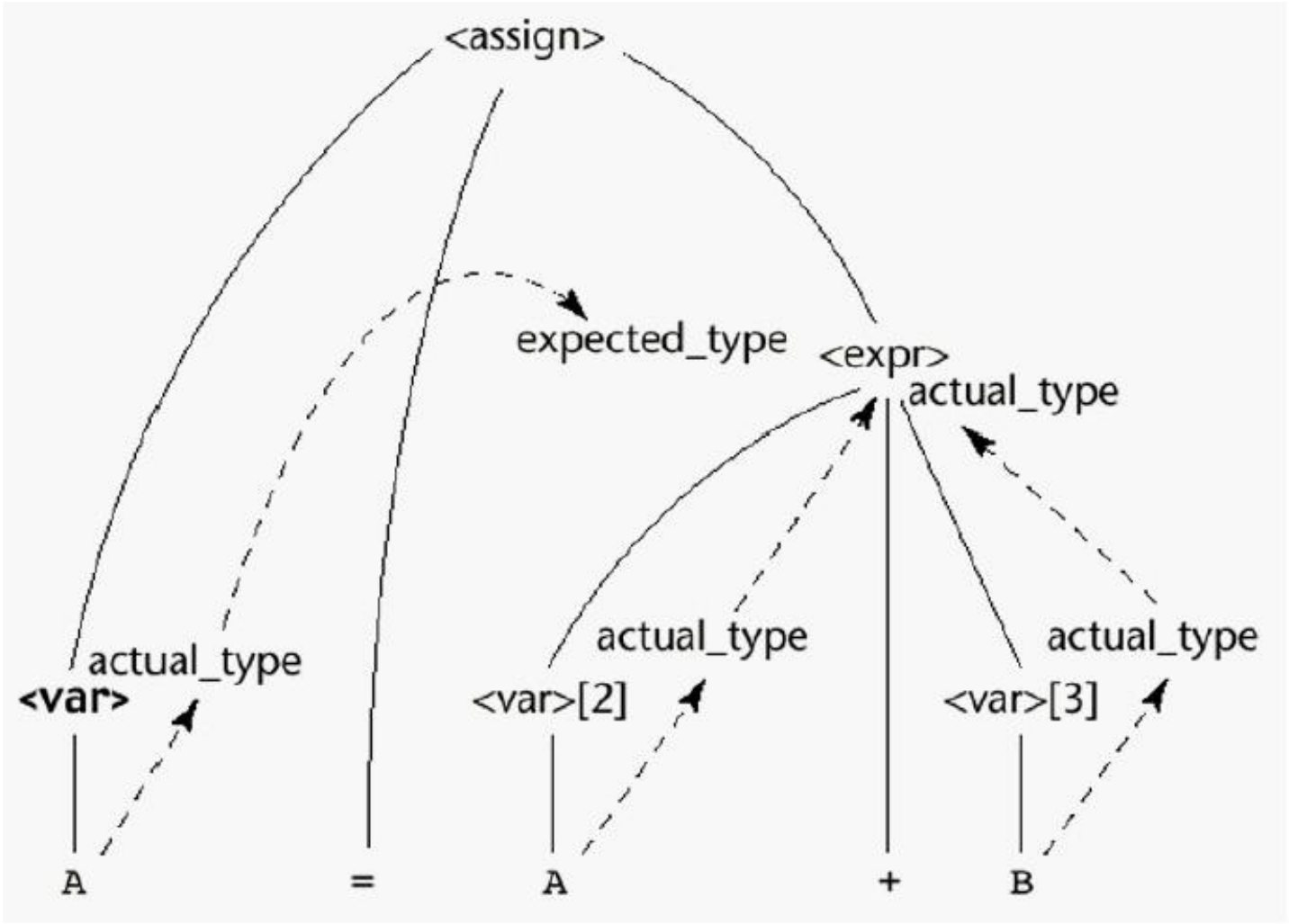
2. Sintaxe: <expr> → <var>[2] **+** <var>[3]
Semântica: <expr>.actual_type ← if (<var>[2].actual_type = int) &
(<var>[3].actual_type = int)
then int
else real
Predicado: <expr>.actual_type = <expr>.expected_type

3. Sintaxe: <expr> → <var>
Semântica: <expr>.actual_type ← <var>.actual_type
Predicado: <expr>.actual_type = <expr>.expected_type

4. Sintaxe: <var> → **A** | **B** | **C**
Semântica: <var>.actual_type ← look-up(<var>.string)

Obs.: look-up pesquisa determinado nome de variável na tabela de símbolos e retorna o tipo desta.

Gramática de Atributos (5/6)



Gramática de Atributos (6/6)

1. <var>.actual_type ← look-up(A) (regra 4)
2. <expr>.expected_type ← <var>.actual_type (regra 1)
3. <var>[2].actual_type ← look-up(A) (regra 4)
4. <var>[3].actual_type ← look-up(A) (regra 4)
5. <expr>.actual_type ← int ou real (regra 2)
6. <expr>.expected_type = <expr>.actual_type é TRUE ou FALSE (regra 2)

Semântica Dinâmica (1/2)

- Descrever a semântica dinâmica, ou significado, de expressões, instruções e unidades, não é uma tarefa fácil

- Não existe uma notação universalmente aceita para descrevê-la

- Por que descrever a semântica dinâmica:
  - Programadores precisam saber o que os elementos da linguagem fazem
  - Desenvolvedores de compiladores baseiam a semântica em descrições da linguagem natural

Semântica Dinâmica (2/2)

- É objeto de pesquisa encontrar um formalismo semântico que possa ser usado tanto por programadores quanto por desenvolvedores de compiladores

- A prova de corretude de programas confiam em alguma descrição formal da semântica da linguagem:
  - Semântica operacional
  - Semântica denotacional
  - Semântica axiomática

Semântica Operacional (1/5)

- A ideia por trás da semântica operacional é a descrição do significado de um programa pela execução de suas instruções em uma máquina real ou simulada

- As alterações que ocorrem no estado da máquina quando esta executa uma instrução define o significado desta instrução

- Exemplo:
  - Instrução em linguagem de máquina

Semântica Operacional (2/5)

- Descrever a semântica operacional de uma linguagem de alto nível requer a construção de um computador real ou simulado

- A semântica para uma linguagem de alto nível pode ser descrita usando um interpretador puro para esta linguagem

- Dois problemas:
  - Complexidade do hardware pode fazer com que as ações sejam difíceis de entender.
  - Só serviriam para um computador igualmente configurado

Semântica Operacional (3/5)

- Estes problemas são evitados pela troca do computador real por um computador virtual, implementado via simulação por software
- O conjunto de instruções poderia ser projetado de tal forma que a semântica de cada instrução fosse fácil de entender
- Requer a construção de dois componentes:
  - Tradutor para converter as instruções da linguagem de alto nível na linguagem intermediária
  - A máquina virtual para esta linguagem intermediária

Semântica Operacional (4/5)

- Este conceito e frequentemente usado em manuais de linguagens

- O leitor das descrições de semântica operacional e o computador virtual e é assumido ser capaz de executar corretamente as instruções e reconhecer os efeitos da execução

Semântica Operacional (5/5)

Instrução em C

```
for (expr1; expr2; expr3){
…
}
```

Semântica Operacional

```
expr1;
loop: if expr2 == 0 goto out;
…
expr3;
goto loop;
out: ....
```

Semântica Denotacional (1/6)

- Semântica denotacional é o método mais rigoroso para descrever o significado de programas
- E baseado na teoria das funções recursivas
- O conceito fundamental e a definição para cada entidade da linguagem tanto um objeto matemático quanto uma função que mapeia instâncias desta entidade em instâncias do objeto matemático
- Os objetos são tão rigorosamente definidos que eles representam o significado exato das entidades correspondentes

Semântica Denotacional (2/6)

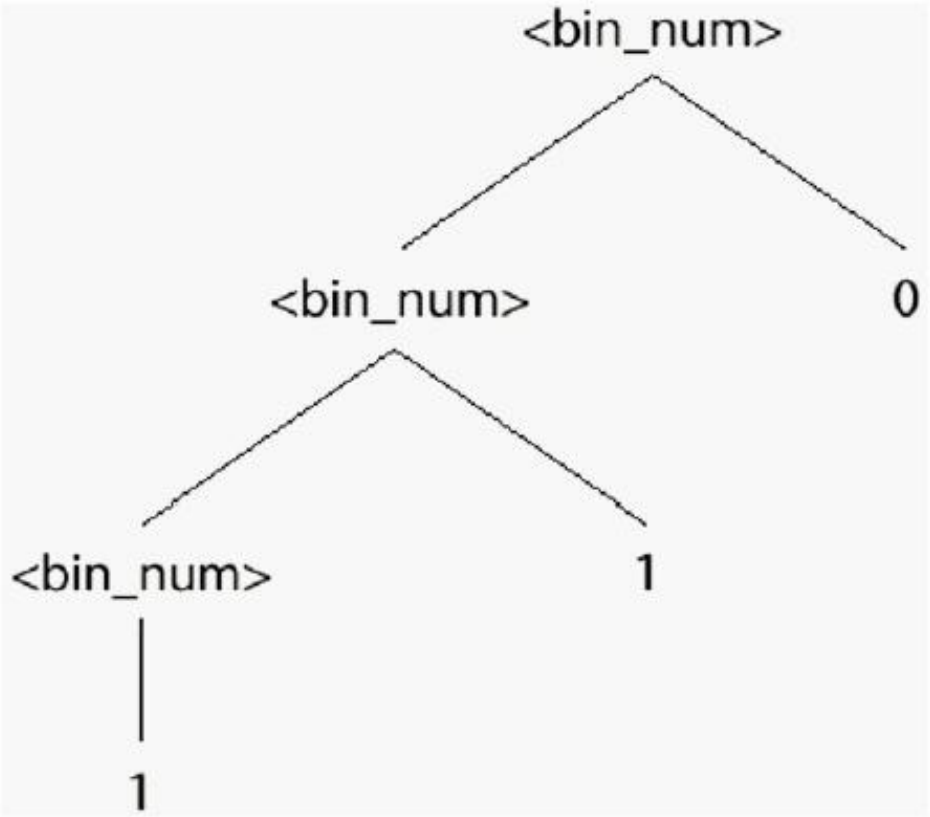
- A dificuldade deste método está na criação dos objetos e na função de mapeamento

- O método é chamado denotacional porque os objetos denotam o significado de suas correspondentes entidades sintáticas

- Exemplo:
  - A sintaxe de números binários podem ser descritas pelas seguintes regras gramaticais:
    - <bin_num> → 0 | 1 | <bin_num> 0 | <bin_num> 1

Semântica Denotacional (3/6)

- Uma parse tree para o numero binario 110 e:



Semântica Denotacional (4/6)

- Para descrever o significado de números binários usando a semântica denotacional e as regras gramaticais acima, o significado real e associado com cada regra que tem um simples símbolo terminal no lado direito. Objetos serão simples números decimais

- Faça o domínio de valores semânticos dos objetos ser N. Estes serão os objetos que queremos associar com números binários

- A função semântica, chamada Mbin mapeia a sintaxe abstrata, descrito pelas regras gramaticais, nos objetos de N

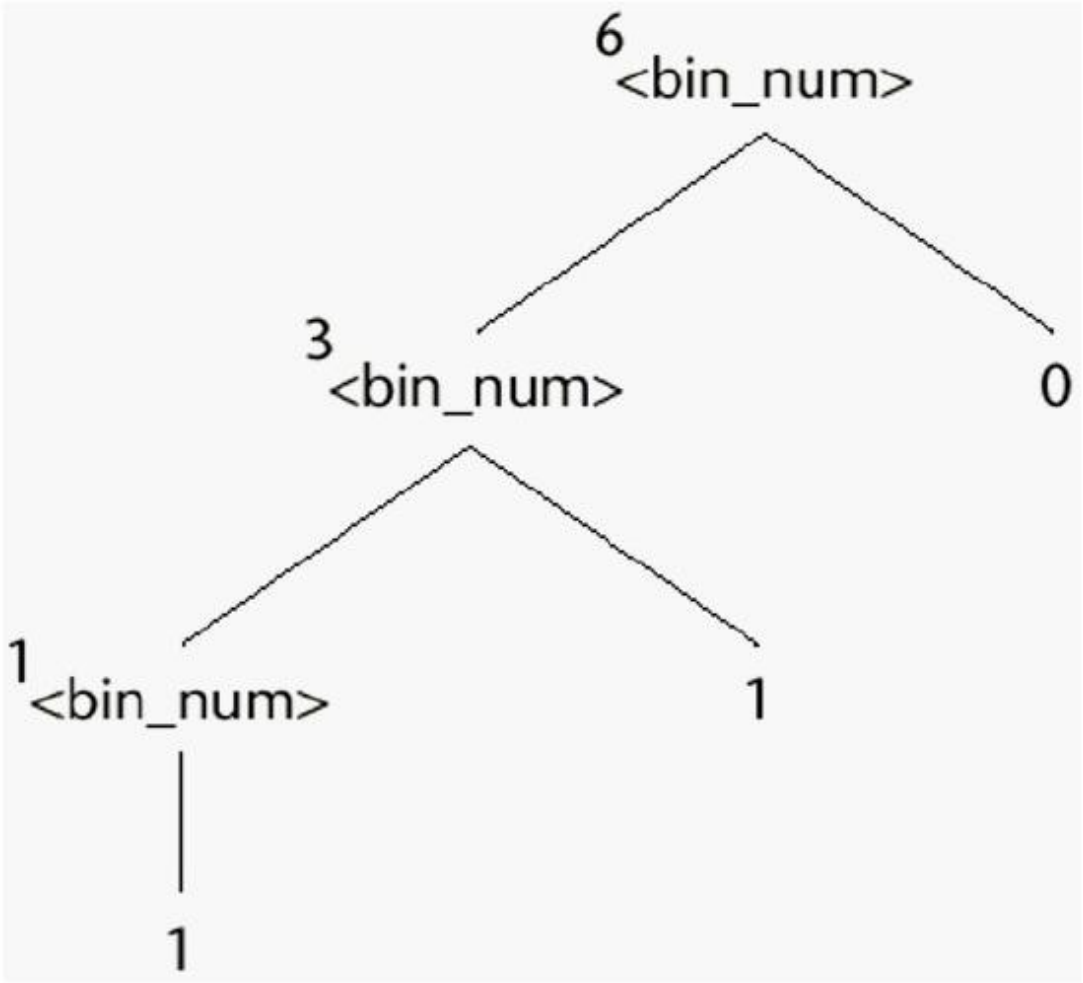
Semântica Denotacional (5/6)

- A função Mbin foi definida como:
  - Mbin(0) = 0
  - Mbin(1) = 1
  - Mbin(<bin_num> 0) = 2 * Mbin(<bin_num>)
  - Mbin(<bin_num> 1) = 2 * Mbin(<bin_num>) + 1

- O significado, ou objetos denotados (que no caso são números decimais), podem ser colocados junto aos nós da *parse tree*

- Entidades sintáticas abstratas são mapeadas em objetos matemáticos com significado concreto

Semântica Denotacional (6/6)



Semântica Axiomática (1/7)

- Semântica axiomática foi definida em conjunto com o desenvolvimento de um método para provar a corretude de programas
- Tais provas de corretude, mostram que um programa executa a computação descrita por sua especificação
- Cada instrução é precedida e seguida por uma expressão lógica que especifica restrições sobre variáveis de programa
- Estas expressões, ao invés de representar o estado de uma máquina abstrata, são usadas para representar o significado de instruções

Semântica Axiomática (2/7)

- Notação usada para descrever restrições e o cálculo de predicados
- Semântica axiomática e baseada em lógica matemática
- As expressões lógicas são chamadas predicados ou assertivas
- Uma assertiva imediatamente precedente de uma instrução descreve as restrições sobre as variáveis naquele ponto:
  - Chamadas de pré-condições

Semântica Axiomática (3/7)

- Uma assertiva imediatamente seguinte a uma instrução descreve uma nova restrição sobre estas, e outras, variáveis após a execução:
  - Chamadas de pós-condições
- Toda a instrução em um programa tem que ter as duas condições
- As pré-condições são computadas a partir das pós-condições

Semântica Axiomática (4/7)

- Exemplo:
  - sum = 2 * x + 1 {sum > 1}
- A pré-condição mais fraca é a pré-condição menos restritiva que garantirá a validade da pós-condição
  - {x > 10} e {x > 100} são exemplos de pré-condições válidas
- Entretanto:
  - {x > 0} e a pré-condição mais fraca

Semântica Axiomática (5/7)

- Para algumas instruções de programas, a computação da pré-condição mais fraca e simples e pode ser especificada por um axioma
- Na maioria dos casos, entretanto, só via regra de inferência
- Um axioma é uma instrução lógica que e assumida ser verdade
- Uma regra de inferência é um método de inferir a verdade de uma assertiva baseada em outras assertivas verdadeiras

Semântica Axiomática (6/7)

- Instrução de Atribuição:
  - Seja x := e uma instrução de atribuição geral e seja Q a pós-condição
  - A pré-condição é definida pelo seguinte axioma:
    - P = Qx→e, o que significa que P é computado como Q com todas as instâncias de x trocadas por e
  - Exemplo:
    - a := b / 2 – 1 {a > 10}
      - Calculando a pré-condição mais fraca:
        - b / 2 – 1 > 10
        - b > 22

Semântica Axiomática (7/7)

- Instrução de Atribuição:
  - A notação usual para especificar a semântica axiomática é:
    - {P} S {Q}
  - No caso de uma instrução de atribuição:
    - {Qx→e} S {Q}