

Compiladores

Análise Semântica

Prof. Dr. Luiz Eduardo G. Martins
(adaptado por Profa Dra Ana Carolina Lorena)
UNIFESP

Análise Semântica

A definição da sintaxe de uma LP utiliza, geralmente, uma GLC, mas nem todas as LP podem ser descritas por GLC's

- Compatibilidade de tipos
- Regras de escopo
- Declaração antes de uso
- Etc.

Ex.: $a = b + c$;

É ilegal em C se alguma das variáveis não tiverem sido declaradas

Análise Semântica

Lista de erros sensíveis ao contexto:

- Identificador já declarado no escopo (nível) atual
- Tipo não definido
- Limite inferior > limite superior na declaração de vetores/matrizes
- Função não declarada (quando há parênteses)
- Função, variável, parâmetro, ou constante não definidas
- Incompatibilidade no número de parâmetros

Análise Semântica

- O objetivo geral da análise semântica é verificar se partes distintas do programa estão coerentes

Etapa da análise do código apoiada por heurísticas

- ▶ Difícil de ser formalizada por meio de gramáticas
- ▶ Associada a inter-relacionamentos entre partes distintas do código

–Heurística é um conjunto de regras e métodos que conduzem à descoberta, invenção ou resolução de problemas

Análise Semântica

- Existe uma formalização por **Gramática de Atributos**, mas em geral a análise semântica é feita de forma “manual”
 - É uma GLC estendida para fornecer sensibilidade a contexto por atributos ligados a terminais e não-terminais
 - Exemplos de atributos: tipo de dado de uma variável, valor de expressão ($x = a + b$ se traduz na regra $x.valor = a.valor + b.valor$)

Análise Semântica

- Gramática de atributos permite especificar:
 - Comportamento semântico de operações
 - Checagem de tipos
 - Manipulação de erros
 - Tradução do programa
- A **tabela de símbolos** assume papel fundamental e é usada para armazenar os atributos e para checagem de tipos

Análise Semântica

- Tabela de símbolos

- Estrutura de dados auxiliar criada para apoiar a análise semântica
- Essa estrutura normalmente é implementada como uma tabela de *hashing* (tempo constante), com tratamento de colisões por encadeamento externo
- Conteúdo usual da tabela de símbolos
 - Nome do identificador
 - Tipo do identificador (variável, função...)
 - Escopo da variável
 - Tipo de dados do identificador (int, float, void...)
 - Número da linha em que o identificador aparece no programa fonte

Análise Semântica

Exemplo: decl \rightarrow tipo var-lista

tipo \rightarrow int | float

var-lista \rightarrow id, var-lista | id

int x,y;

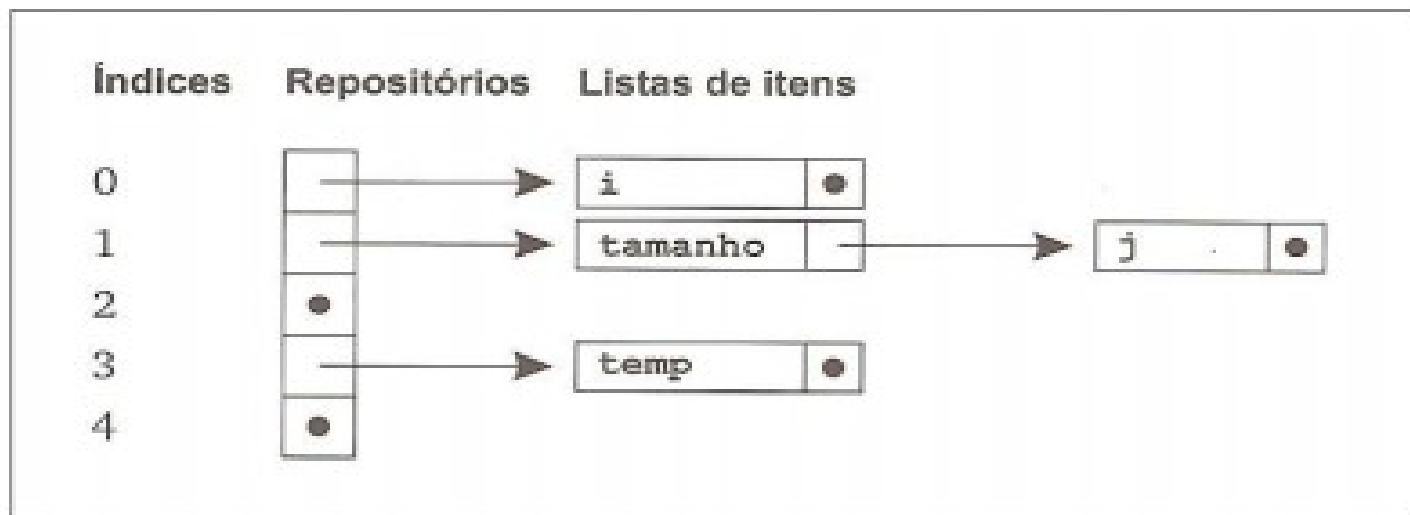
Regras gramaticais	Regras semânticas
decl \rightarrow tipo var-lista	var-lista.tipo_dado = tipo.tipo_dado
tipo \rightarrow int	tipo.tipo_dado = integer
tipo \rightarrow float	tipo.tipo_dado = real
var-lista ₁ \rightarrow id, var-lista ₂	id.tipo_dado = var-lista ₁ .tipo_dado var-lista ₂ .tipo_dado = var-lista ₁ .tipo_dado If busca(id)=FALSE then inserir(id,id.tipo_dado) else ERRO("identificador já declarado")
var-lista \rightarrow id	id.tipo_dado=var-lista.tipo_dado If busca(id)=FALSE then inserir(id,id.tipo_dado) else ERRO("identificador já declarado")

Análise Semântica

- Tabela de símbolos
 - Estratégia geral para popular a tabela de símbolos (considerando a estrutura de C-):
 - Pode ser feito durante a análise léxica, sintática ou semântica
 - Função de *hashing* deve gerar entrada para
 - Nome de função
 - » Calcular *hashing* usando apenas os caracteres do nome da função
 - Nome de variável
 - » Calcular *hashing* usando caracteres do nome da variável + nome da função
 - » Demanda controle do escopo atual

Análise Semântica

Exemplo de *hashing* com resolução de colisões para a inclusão dos identificadores *i*, *j*, *tamanho* e *temp*



Análise Semântica

- **Tabela de símbolos:** principais operações
 - **Busca**(Tab: TS; id: string; ref: Pont_entrada; declarado: boolean);
{busca id na Tab; retorna uma referência ref (ponteiro) para a entrada correspondente e um flag – declarado - para indicar se o nome já estava presente no escopo}
 - **Elimina**(Tab: TS; K:nível); {elimina todos os id que estão num dado nível K (escopo) }
 - **Insere**(Tab: TS; id: string; ref: Pont_entrada; declarado: boolean);
{insere id na TS; retorna um ponteiro para a entrada e um flag para indicar se o nome já estava presente naquele escopo}
 - **Declarado**(Tab: TS; id: string; K: nivel): boolean; {verifica se o id está declarado no nível K (corrente)}
 - **Seta_atributos**(Ref: Pont_entrada; AT: atributos);
 - **Obtem_atributos**(Ref: Pont_entrada; AT: atributos);

Análise Semântica

```
<exp>1 ::= <exp>2 div id  
  se busca(id)=falso  
    então ERRO("variável não declarada")  
  senão se exp2.tipo<>inteiro ou id.tipo<>inteiro  
    então ERRO("tipos inválidos para a operação")  
  se não ocorreu erro então  
    exp1.tipo=inteiro  
    exp1.val=exp2.val / id.val
```

```
procedimento atribuição(Seg)  
Início  
  se (simbolo=id)  
    então obtem_simbolo(cadeia,simbolo)  
      se busca(cadeia,simbolo,cat="var")=FALSE  
        então ERRO("variável não declarada")  
        senão tipo1:=recupera_tipo(cadeia,simbolo,cat="var");  
      senão ERRO(Seg+{simb_atrib});  
  se (simbolo=simb_atrib)  
    então obtem_simbolo(cadeia,simbolo)  
    senão ERRO(Seg+{id});  
  expressao(tipo2);  
  se tipo1<>tipo2 então ERRO("tipos incompatíveis na atribuição");  
  se (simbolo=simb_ponto-virgula)  
    então obtem_simbolo(cadeia,simbolo)  
    senão ERRO(Seg+P(comandos));  
fim
```

Análise Semântica

- Tabela de símbolos
 - Estratégia geral para popular a tabela de símbolos
 - Ao encontrar ID, verificar próximo *token*
 - Se for “(”, então ID é função
 - Senão, ID é variável
 - Se *token* anterior for “int”, “float” ou “void”, fazer entrada na tabela de símbolos
 - Senão, apenas atualizar número de linhas do ID na tabela de símbolos

Programa de entrada:

```
1 int gcd(int u, int v)
2 { if (v==0) return u;
3   else return gcd(v, u-u/v*v);
4 }
5 void main(void)
6 { int x; int y;
7   x = input(); y = input();
8   output(gcd(x,y));
9 }
```

Análise Semântica

- Principais tarefas realizadas durante a análise semântica estática (em tempo de compilação):
 - Verificação de declarações (e escopo)
 - Verificação de tipos
 - Verificação da unicidade de declaração de variáveis
 - Verificação de fluxo de controle

Análise Semântica

- Verificação de declarações
 - Verifica se as variáveis utilizadas no programa foram devidamente declaradas (quando a linguagem exigir)

```
#include <iostream>
using namespace std;

int main() {
    a = 10;
    cout << "Valor de a: " << a << endl;
}
```

Mensagem de erro

...cpp: In function 'int main()':

...cpp:5: error: 'a' was not declared in this scope

Análise Semântica

- Verificação de escopo
 - Controle do nível durante a compilação do programa
 - Quando se chama um procedimento (ou função), faz-se $nível = nível + 1$
 - Quando se sai de um procedimento (ou função), faz-se $nível = nível - 1$

Análise Semântica

- Verificação de declarações e escopo

```
#include <iostream>
using namespace std;
int main() {
    int a = 9;
    void mostra();
    mostra();
}
void mostra() {
    cout << "a: " << a << endl;
}
```

Mensagem de erro

...cpp: In function 'void mostra()':

...cpp:11: error: 'a' was not declared in this scope

Análise Semântica

- Verificação de tipos
 - Verifica se as variáveis declaradas estão sendo usadas de forma coerente, de acordo com o tipo especificado

```
#include <iostream>
using namespace std;

int main() {
    int a = 9;
    float b = 5;
    cout << "a%b: " << a%b << endl;
}
```

...cpp: In function 'int main()':

...cpp:7: error: invalid operands of types 'int' and 'float' to binary 'operator%'

Análise Semântica

- Verificação da unicidade de declaração de variáveis
 - Detecta duplicações em declarações de variáveis

Exemplo:

```
int main()  
{  
    int a;  
    float a;  
    ...  
}
```

Mensagem de erro:

In function 'main':

...: duplicate member 'a'

Análise Semântica

- Verificação de fluxo de controle
 - Detecta erros nas estruturas de controle do programa (*for, do, while, if else, switch case*)

Exemplo:

```
void exemplo(int j, int k)
{
    if (j == k) break;
    else continue;
}
```

Mensagem de erro:

In function 'exemplo':

...: break statement not within a loop or switch

...: continue statement not within a loop

Análise Semântica

- Principais erros semânticos que devem ser detectados no projeto de C-

(1)

```
void exemplo()  
{  
    int a; a = 0;  
    b = a;  
    ...  
}
```

“variável não
declarada”,
considerando que *b*
não seja global

Análise Semântica

- Principais erros semânticos que devem ser detectados no projeto de C-

(2)

```
void exemplo()  
{ ... }
```

```
void main(void)  
{  
    int a;  
    a = exemplo();  
    ...  
}
```

“atribuição inválida”, *a*
é do tipo `int` e
exemplo() não retorna
nada

Análise Semântica

- Principais erros semânticos que devem ser detectados no projeto de C-
(3)

```
void main(void)
{
    void a;
    ...
}
```

“declaração inválida de variável”, *void* só pode ser usado para declaração de função

Análise Semântica

- Principais erros semânticos que devem ser detectados no projeto de C-
(4)

```
void main(void)
{
    int a;
    int a;
    ...
}
```

“declaração inválida de
variável”, *a* já foi
declarada previamente

Análise Semântica

- Principais erros semânticos que devem ser detectados no projeto de C-
(5)

```
int fun1() { ... }  
int fun2() { ... }  
void main(void)  
{  
    int a;  
    a = fun3();  
    ...  
}
```

← “*fun3()*: chamada de função não declarada”

Análise Semântica

- Principais erros semânticos que devem ser detectados no projeto de C-
(6)

```
int fun1() { ... }  
int fun2() { ... }  
int fun3() { ... }
```


“função *main()* não declarada”

Análise Semântica

- Principais erros semânticos que devem ser detectados no projeto de C-
(7)

```
int fun1() { ... }  
int xyz() { ... }  
void main(void)  
{  
    int a;  
    int xyz;  
    ...  
}
```

“declaração inválida”, xyz já foi
declarado como nome de função

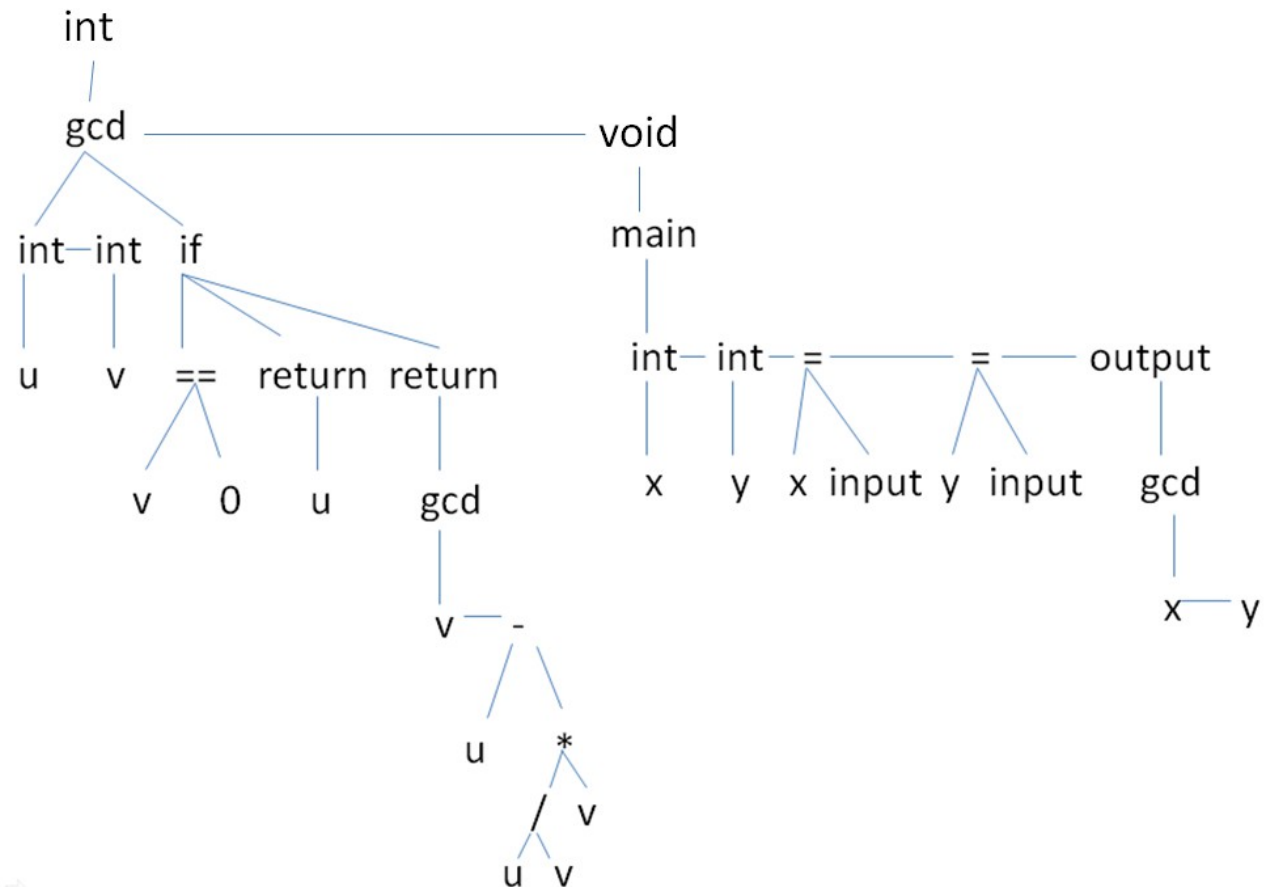
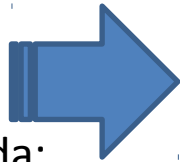


Análise Semântica

- Árvore sintática do programa para cálculo do gcd (Louden, apêndice A)

Programa de entrada:

```
1 int gcd(int u, int v)
2 { if (v==0) return u;
3   else return gcd(v, u-u/v*v);
4 }
5 void main(void)
6 { int x; int y;
7   x = input(); y = input();
8   output(gcd(x,y));
9 }
```



Análise Semântica

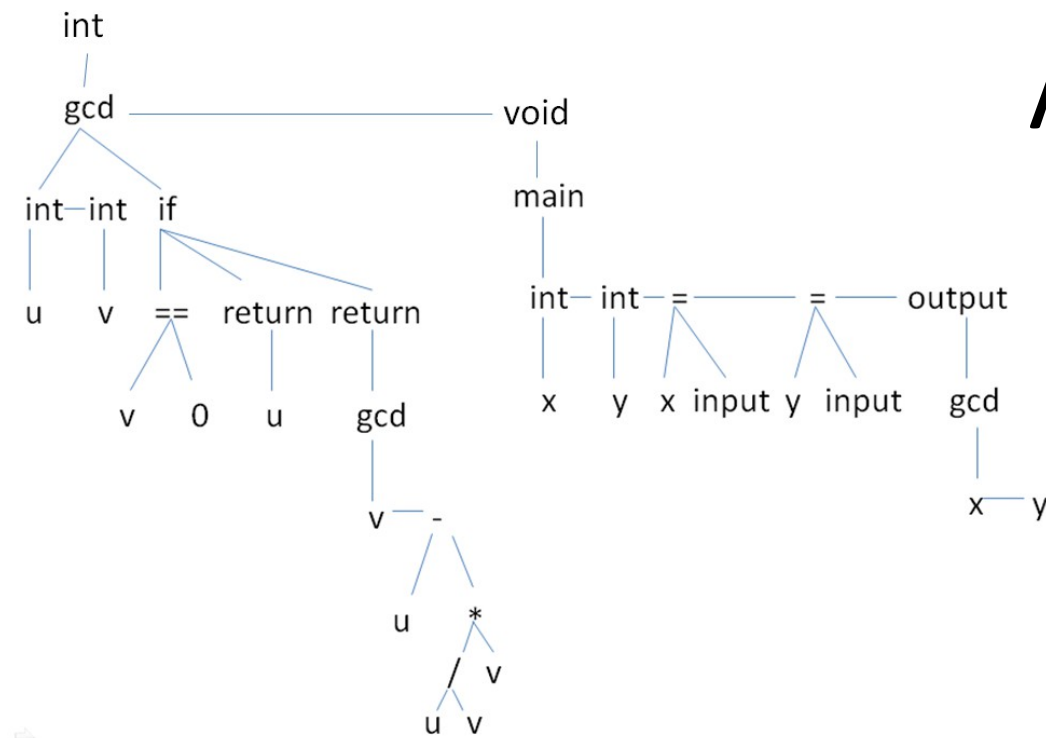


Tabela de Símbolos

Entrada	Nome ID	Escopo	Tipo ID	Tipo dado	n. Linha
...					
12	u	gcd	var	int	1,2,3
...					
136	main		fun	void	5
...					
200	gcd		fun	int	1,3,8

Programa de entrada:

```
1 int gcd(int u, int v)
2 { if (v==0) return u;
3   else return gcd(v, u-u/v*v);
4 }
5 void main(void)
6 { int x; int y;
7   x = input(); y = input();
8   output(gcd(x,y));
9 }
```

Análise Semântica

- Bibliografia consultada

RICARTE, I. **Introdução à Compilação**. Rio de Janeiro: Editora Campus/Elsevier, 2008. (cap. 5)

LOUDEN, K. C. **Compiladores: princípios e práticas**. São Paulo: Pioneira Thompson Learning, 2004. (cap. 6)

Slides Profs Tiago Pardo e Sandra Aluízio, ICMC-USP