Universidade do Estado do Rio de Janeiro
Instituto de Matemática e Estatística
Departamento de Ciência da Computação
Compiladores

Relatório Técnico do Analisador Sintático

Aluno:
Marcio Sirimarco de Souza Junior - 201710076811
Mateus Henrique Freitas Maciel - 202220460111

Professor(a):
Lis Custódio

22 de novembro de 2025

1 Descrição Teórica do Programa

A análise semântica foi integrada ao analisador sintático por meio de ações semânticas embutidas nos mesmos procedimentos recursivos que tratam a gramática LL(1). O objetivo principal é realizar a análise contextual e a checagem de tipos, detectando erros como utilização de variáveis não declaradas, redeclaração de identificadores e incompatibilidade de tipos em expressões e atribuições.

A Tabela de Símbolos utilizada na etapa léxica foi estendida para armazenar os atributos necessários à análise semântica. Cada entrada da tabela contém o lexema do identificador, sua categoria (no trabalho atual, focado em variáveis) e o seu tipo estático (`int`, `float`, `string`, indefinido ou erro). Quando o analisador léxico encontra um novo identificador, ele o insere na tabela com tipo *indefinido*. Durante a análise sintática, ao processar uma declaração do tipo `<tipo> ID ;`, o analisador semântico atualiza a entrada correspondente ao `ID` com o tipo concreto definido (`int`, `float` ou `string`). Caso um identificador já possua tipo definido, a nova declaração é sinalizada como erro de redeclaração.

Sempre que um identificador é utilizado em contexto de variável (por exemplo, em um fator de expressão, em uma atribuição ou em um comando `read`), o analisador semântico consulta a Tabela de Símbolos para verificar se ele foi declarado previamente. Se a entrada ainda estiver com tipo indefinido, é emitido um erro semântico de variável não declarada. Em expressões aritméticas, cada nó sintático (*factor*, *term*, *expr*) propaga um atributo de tipo: literais numéricos são tipados como `int` ou `float` dependendo da presença de ponto decimal; cadeias de caracteres são tipadas como `string`; e operadores binários (`+`, `-`, `*`, `/`) combinam os tipos dos operandos por meio de uma função de inferência. Operações entre tipos não numéricos são sinalizadas como erro, e o tipo resultante é marcado como *erro* para evitar cascata em verificações subsequentes.

As atribuições são verificadas comparando o tipo da variável destino com o tipo da expressão à direita. São aceitas atribuições entre tipos iguais e a conversão implícita de `int` para `float`. A atribuição de um valor `float` para uma variável `int` é detectada explicitamente e relatada como erro de tipos incompatíveis, conforme o exemplo do enunciado (atribuição de um real a um inteiro). Comandos condicionais (`if` e `while`) também são verificados: a expressão dentro dos parênteses deve ser numérica (`int` ou `float`), caso contrário é gerado um erro indicando condição inválida.

Todas as mensagens de erro semântico indicam a linha e a coluna do lexema associado, utilizando a mesma função de impressão de contexto usada pelo analisador sintático. A análise semântica **não é interrompida no primeiro erro semântico**: enquanto a estrutura sintática do programa for válida, o analisador continua processando o restante do código e acumulando outros erros de tipos ou de declaração.

Entretanto, em caso de erro sintático grave o analisador entra em modo pânico e **a fase semântica é desativada a partir desse ponto**. Isso é feito para evitar que sejam reportados erros semânticos em trechos cuja estrutura sintática é incerta, o que poderia gerar uma cascata de mensagens pouco úteis ao usuário. Dessa forma, todos os erros semânticos são reportados apenas enquanto o programa permanece sintaticamente bem formado.

2 Casos de teste da análise semântica

A seguir são apresentados os principais arquivos de teste utilizados para validar o comportamento do analisador semântico, bem como as saídas geradas pelo compilador.

`semantico_erro1.txt` Exercita: redeclaração de variável e atribuição `float` para `int`.
Entrada:

```
inicio
  -- Erro 2: Redeclaracao
  int contador;
  float resultado;
  string contador; -- ERRO SEMANTICO (1)

  -- Erro 4: Atribuicao float -> int
  resultado = 100.5;
  contador = resultado; -- ERRO SEMANTICO (2)

fim
```

Saída obtida:

```
Erro linha 5, coluna 10:
  >   string contador; -- ERRO SEMANTICO (1)
             ^
ERRO SEMANTICO: Variavel 'contador' declarada mais de uma vez.

Erro linha 9, coluna 3:
  >   contador = resultado; -- ERRO SEMANTICO (2)
      ^
ERRO SEMANTICO: Tipos incompativeis na atribuicao para 'contador' (variavel: int, expressao: float).

[FALHA] Foram encontrados erros sintaticos e/ou semanticos.
```

Análise: O primeiro erro mostra a detecção de **variável declarada mais de uma vez**; o segundo mostra a verificação de tipos em atribuições, rejeitando **atribuição de `float` para `int`**, conforme o requisito de "atribuição de um real a um inteiro".

`semantico_erro2.txt` Exercita: uso de variável não declarada e interação com erro sintático em `read`. O segundo erro semântico planejado não é reportado, pois a análise semântica é desativada após o primeiro erro sintático.
Entrada:

```
inicio
  -- Erro 1: Nao declarada
  int a;
  a = b + 10; -- ERRO SEMANTICO (1)

  -- Erro 3: Parametro invalido
  -- 'read' espera uma variavel (CAT_VARIAVEL),
  -- mas 'print' e uma funcao (CAT_FUNCAO).
  read(print); -- ERRO SEMANTICO (2)
fim
```

Saída obtida:

```
Erro linha 4, coluna 7:
  >   a = b + 10; -- ERRO SEMANTICO (1)
          ^
ERRO SEMANTICO: Variavel 'b' nao declarada.

Erro linha 9, coluna 8:
  >   read(print); -- ERRO SEMANTICO (2)
            ^
ERRO SINTATICO: Esperava IDENTIFICADOR.
  Token recebido: 'print' (inesperado)
  Iniciando modo panico... Sincronizando com { ';' }
  Sincronizacao encontrada. Continuando analise no token ';'.

Erro linha 9, coluna 14:
  >   read(print); -- ERRO SEMANTICO (2)
                 ^
ERRO SINTATICO: esperado ')' antes do token ';' (recuperacao por insercao).

[FALHA] Foram encontrados erros sintaticos e/ou semanticos.
```

Análise: O teste confirma a detecção de **variável não declarada** (`b`). Na chamada `read(print)`, o erro sintático impede que o erro semântico "parâmetro inválido" seja reportado, ilustrando a política do compilador: após um erro sintático que exige recuperação em modo pânico, a análise semântica é desativada para evitar erros em cascata.

`semantico_erro3.txt` Exercita: variável não declarada, atribuição `float` para `int` e uso de variável não declarada em `print`. Não há erros sintáticos, portanto todos os erros semânticos são reportados.

Entrada:

```
inicio
  int x;
  int z;

  x = y; -- ERRO SEMANTICO (1)

  z = 10.5; -- ERRO SEMANTICO (2)

  print(w); -- ERRO SEMANTICO (3)

fim
```

Saída obtida:

```
Erro linha 5, coluna 7:
  >   x = y; -- ERRO SEMANTICO (1)
          ^
ERRO SEMANTICO: Variavel 'y' nao declarada.

Erro linha 7, coluna 3:
  >   z = 10.5; -- ERRO SEMANTICO (2)
      ^
ERRO SEMANTICO: Tipos incompativeis na atribuicao para 'z' (variavel: int, expressao: float).

Erro linha 9, coluna 9:
  >   print(w); -- ERRO SEMANTICO (3)
            ^
ERRO SEMANTICO: Variavel 'w' nao declarada.
```

Análise: Esse teste cobre simultaneamente **variáveis não declaradas** (`y` e `w`) e **atribuição de `float` a `int`** (`z`). Como não há erros sintáticos, o analisador consegue reportar todos os erros semânticos no mesmo arquivo, mostrando que ele não para no primeiro erro.

`semantico_erro4.txt` Exercita: redeclaração, atribuição `float` para `int`, uso de variável não declarada em atribuição e em `read`. Também ilustra que a mesma variável não declarada pode gerar mais de um erro semântico, dependendo dos pontos de uso.

Entrada:

```
inicio
  int var_int;
  float var_float;

  int var_int; -- ERRO TIPO 2: Redeclaracao

  var_float = 1.5;

  -- ERRO TIPO 4: Atribuicao float -> int
  var_int = var_float;

  -- ERRO TIPO 1: Nao declarada
  var_int = var_nao_declarada;

  -- ERRO TIPO 3: Parametro invalido
  read(var_float); -- (ok)
  read(var_int);   -- (ok)
  read(var_nao_declarada); -- ERRO TIPO 1 (de novo) e TIPO 3

fim
```

Saída obtida:

```
Erro linha 5, coluna 7:
  >   int var_int; -- ERRO TIPO 2: Redeclaracao
          ^
ERRO SEMANTICO: Variavel 'var_int' declarada mais de uma vez.

Erro linha 10, coluna 3:
  >   var_int = var_float;
      ^
ERRO SEMANTICO: Tipos incompativeis na atribuicao para 'var_int' (variavel: int, expressao: float).

Erro linha 13, coluna 13:
  >   var_int = var_nao_declarada;
                ^
ERRO SEMANTICO: Variavel 'var_nao_declarada' nao declarada.
```

```
Erro linha 18, coluna 8:
  >   read(var_nao_declarada); -- ERRO TIPO 1 (de novo) e TIPO 3
          ^
ERRO SEMANTICO: Variavel 'var_nao_declarada' nao declarada.
```

Análise: Este arquivo cobre quase todos os tipos de erro pedidos no enunciado: **redeclaração** (`var_int`), **atribuição de `float` para `int`** (`var_int = var_float`), **uso de variável não declarada** (`var_nao_declarada`) e parâmetro inválido em `read`. A mesma variável não declarada gera erros em contextos diferentes (atribuição e leitura).

`semantico_sucesso1.txt` Exercita um programa completo sem erros, com declarações, laço `while`, condicional `if/else`, coerção de tipos em expressões aritméticas e uso de `print` e `read`.

Entrada:

```
inicio
  -- Teste de sucesso completo (sintatico e semantico)
  int i;
  float total;
  string nome;

  read(nome);
  read(i);

  total = 0.0;

  -- Teste de coercao de tipo (float = float + int)
  while (i) {
    total = total + i;
    i = i - 1;
    print(total);
  }

  -- Teste de condicional com float
  if (total) {
     print(nome);
  } else {
     print("Total foi zero.");
  }

fim
```

Saída obtida:

```
[SUCESSO] Analise Sintatica e Semantica concluida.
```

Análise: Demonstra o comportamento esperado em um programa correto, incluindo **coerção `int` → `float`** em `total = total + i`, uso de `while` e `if/else` com expressões numéricas e chamadas bem tipadas de `read` e `print`.

`sintatico_semantico_erro1.txt` Este teste mostra que, após um erro sintático na fase de declarações, a análise semântica é desativada. O erro semântico planejado na expressão `b = a + c;` não é reportado.

Entrada:

```
inicio
  int a

  -- ERRO SINTATICO (1) (falta ';')

  float b;

  a = 10;
  b = a + c; -- ERRO SEMANTICO (2)
fim
```

Saída obtida:

```
Erro linha 6, coluna 3:
  >   float b;
      ^
ERRO SINTATICO: esperado ';' antes do token 'float' (recuperacao por insercao).

[FALHA] Foram encontrados erros sintaticos e/ou semanticos.
```

Análise: O erro sintático na declaração de `a` força uma recuperação em modo pânico. A partir desse ponto, a fase semântica é desativada e o erro planejado na atribuição `b = a + c;` não é mais checado, ilustrando a estratégia de evitar erros semânticos em cascata após falhas estruturais.

`sintatico_semantico_erro2.txt` Neste teste, o primeiro erro reportado é semântico (`c` não declarada em `print(c);`). Em seguida ocorre um erro sintático na linha `a 10;`, o que ativa o modo pânico e desliga definitivamente a análise semântica. Por isso, o erro semântico planejado em `a = b;` dentro do bloco não é reportado, assim como o segundo uso de `c`.

Entrada:

```
inicio
  int a;
  float b;
  print(c);  -- ERRO SEMANTICO 2: 'c' nao declarada
  -- Erro Sintatico 1: ID sem '=' ou '('
  a 10;

  -- O parser deve recuperar aqui e continuar
  b = 2.0;

  {
    a = 1;

    -- Erros Semanticos (apos recuperacao)
    a = b;     -- ERRO SEMANTICO 1: float -> int
    print(c);  -- ERRO SEMANTICO 2: 'c' nao declarada (Nao e pega porque desligou o semantico)

  -- Erro Sintatico 2: falta '}' no bloco

fim
```

Saída obtida:

```
Erro linha 4, coluna 9:
  >   print(c);  -- ERRO SEMANTICO 2: 'c' nao declarada
            ^
ERRO SEMANTICO: Variavel 'c' nao declarada.

Erro linha 6, coluna 3:
  >   a 10;
      ^
ERRO SINTATICO: Depois do ID esperava '=' ou '('.
  Token recebido: IDENTIFICADOR (inesperado)
  Iniciando modo panico... Sincronizando com { IDENTIFICADOR 'read' 'print' 'if' 'while' '{' 'fim' '}' }
  Sincronizacao encontrada. Continuando analise no token IDENTIFICADOR.

Erro linha 20, coluna 1:
  > fim
    ^
ERRO SINTATICO: esperado '}' antes do token 'fim' (recuperacao por insercao).

[FALHA] Foram encontrados erros sintaticos e/ou semanticos.
```

Análise: Aqui vemos os dois comportamentos em conjunto: primeiro, a análise semântica detecta **variável não declarada** em `print(c);`; depois, um erro sintático em `a 10;` dispara o modo pânico e desliga o semântico. Os erros planejados dentro do bloco (`a = b;` e o segundo `print(c);`) não são mais checados, reforçando a decisão de não produzir mensagens semânticas sobre uma estrutura sintática inconsistente.

Esses casos ilustram a política adotada: enquanto o programa permanece sintaticamente correto, todos os erros semânticos possíveis são acumulados e reportados. A partir do momento em que ocorre um erro sintático que exige recuperação em modo pânico, a fase semântica é desativada, evitando a geração de erros em cascata sobre uma estrutura de programa potencialmente inválida.

2.1 Conclusão dos testes

Os casos de teste abordam os principais tipos de erros semânticos, e, enquanto o programa permanece sintaticamente correto, a análise semântica não é interrompida no primeiro erro: o compilador percorre todo o código e exibe todas as inconsistências encontradas antes do fim dos tokens.

Os testes também satisfazem os requisitos do enunciado para a linguagem implementada, cobrindo:

- variável não declarada;
- variável declarada mais de uma vez;
- atribuição de um valor `float` a uma variável `int`;
- incompatibilidades de tipo em expressões e comandos (adaptando o requisito de parâmetros formais e reais à ausência de procedimentos declarados na linguagem).

Além disso, foram incluídos casos em que ocorrem erros sintáticos antes ou durante a análise semântica. Nesses cenários, a ocorrência de um erro sintático que exige recuperação em modo pânico faz com que a fase semântica seja desativada, evitando erros em cascata sobre uma estrutura de programa potencialmente inválida.