# INE5318
# Construção de Compiladores

## AULA 5: Tratamento de Erros

Ricardo Azambuja Silveira
INE-CTC-UFSC
E-Mail: silveira@inf.ufsc.br
URL: www.inf.ufsc.br/~silveira

# Tratamento de Erros

- Sabemos que ocorre um erro quando:
  - o símbolo corrente da entrada não corresponde ao *terminal* contido no topo da pilha OU
  - o símbolo corrente da entrada não possui produção correspondente a partir do *não-terminal* contido no topo da pilha.
- Tais erros podem ser recuperados através da modalidade do desespero, i.e., através do descarte de símbolos da entrada até a localização de um *token* de sincronização.

# Tratamento de Erros: Modalidade Pânico

- A eficiência da recuperação de erros desta forma depende da escolha de um conjunto adequado de *tokens* de sincronização.
- Não existem regras formais para tal escolha que pode ser baseada em técnicas heurísticas
- A recuperação de erros é provável na maioria dos casos embora não possa ser assegurada completamente.

# Tratamento de Erros: Modalidade Pânico

- Considerando a ocorrência de um erro durante a expansão do *não-terminal* A:
  - **Técnica 1**:
    Como *tokens* de sincronização usam-se todos os símbolos em folow(A).
    Descartam-se os símbolos de entrada até encontrar-se um elemento de folow(A), quando também descartamos A da pilha
    É provável que a análise sintática possa prosseguir.

# Exemplo

- Dada a gramática:

  S  aAcd | bAef

  A  gh

  e a sentença agabcd

  - O conjunto FOLOW(A) = {c,e}
    - Uma chamada ao método S consome a da entrada
    - Uma chamada ao método A consome o g
    - O método A tenta achar na entrada um h, mas não encontra, acusando um erro sintático.
    - O método A deve consumir todos os caracteres da entrada até encontrar um menbro de FOLOW(A) quando então retorna para o método S e consome c

# Tratamento de Erros: Modalidade Pânico

- **Técnica 2**:
  Como somente os símbolos em folow(A) não são suficiente para a sincronização, adicionam-se os símbolos first das produções que se expandem em A.
  Descartam-se os símbolos de entrada até encontrar-se do conjunto de sincronização, quando também descartamos A.
  É provável que a análise sintática possa prosseguir.

# Tratamento de Erros: Modalidade Pânico

- **Técnica 3**:
  Se os símbolos em first(A) são adicionados ao conjunto de sincronização, poderá ser possível retomar a análise a partir de A se um símbolo que figura em first(A) estiver na entrada.
- **Técnica 4**:
  Se o não-terminal A puder gerar , tal produção pode ser usada como *default*. Pode-se assim adiar a detecção do erro sem haver possibilidade de perde-lo.

# Tratamento de Erros: Modalidade Pânico

- **Técnica 5**:
  Se um terminal no topo da pilha não pode ser reconhecido, pode-se remove-lo. Sugere-se a emissão de mensagem de aviso nestes casos.
  A análise provavelmente poderá prosseguir.
  Tal enfoque corresponde a adicionar todos os *tokens* possíveis como símbolos do conjunto de sincronização.

# Tratamento de Erros

- Toda e qualquer tentativa de recuperação de erros não deve provocar um laço infinito na análise sintática da entrada. Uma forma de proteção é assegurar-se que a pilha foi encurtada após a ação de correção.
- É sempre questionável a inserção ou alteração de símbolos na pilha quando tais operações “criam” construções inexistentes na linguagem sendo analisada.

# Mensagens de Erro

- Todas as mensagens de erro produzidas por um compilador devem ser informativas o suficiente para permitir a rápida localização e correção do erro.
- Sugere-se que cada entrada vazia da tabela sintática preditiva contenha apontadores para rotinas de tratamento de erro onde mensagens apropriadas serão fornecidas conforme o erro que as provocou.

# Geração da árvore sintática

- Ocorre na medida em que a análise sintática se processa
- Cada execução de um método associado a cada não-terminal cria um nó da árvore e associa a ele os nodos filhos
- Os nodos filhos correspondem aos tokens reconhecidos e pelos nodos criados pelas chamadas a outros método de não terminais

# Geração da árvore sintática

- Representação interna do programa após o parsing
- A árvore pode ser percorrida várias vezes nas demais fases do compilador, geralmente num processo de busca em profundidade:
  - Para fazer as verificações semânticas
  - Para gerar o código objeto ou intermediário

# Tabela de símbolos

- Estrutura construída durante a análise sintática
- Armazena informações relevantes sobre os identificadores encontrados no programa
  - Funções principais:
    - Verifica se um identificador já havia sido declarado
    - Verificaçao de tipos
    - Verificaçao de escôpo
    - Compatibilidade de parâmetros de métodos