Compiladores

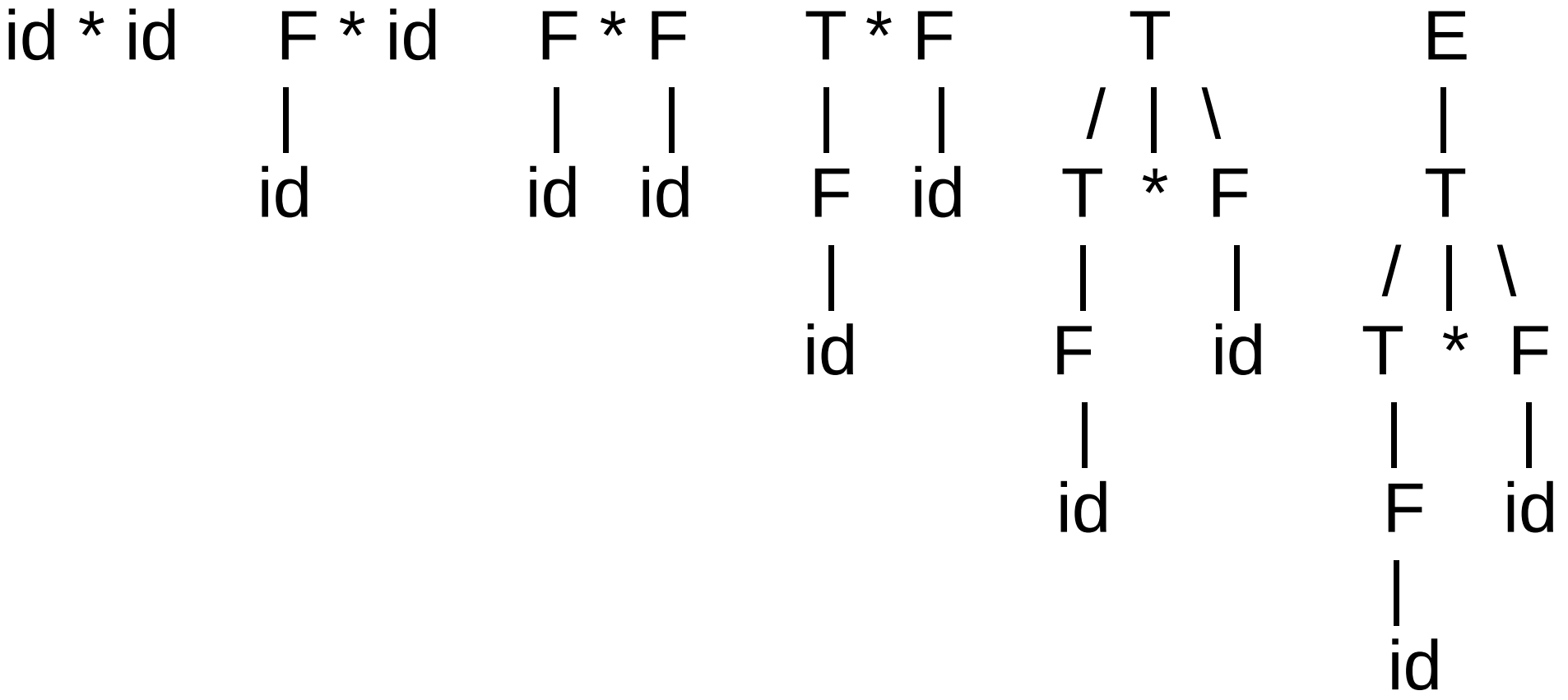
Análise Sintática Ascendente

Redução

Na análise sintática ascendente a árvore de derivação é construída a partir das folhas (cadeia de tokens) em direção a raiz (símbolo inicial da gramática) fazendo o processo inverso à derivação, o que é chamado de redução.

A redução consiste em identificar o lado direito de uma produção e substituí-lo pelo não-terminal à esquerda desta produção, o que em geral reduz o número de símbolos a serem analisados.

Redução



Handle

A análise sintática ascendente lê a cadeia de entrada da esquerda para a direita e constrói a árvore de derivação fazendo derivações mais à direita no sentido contrário.

Informalmente, um ***Handle*** de uma cadeia de símbolos é uma subcadeia que casa com o corpo de uma produção, e cuja a redução para o não-terminal à esquerda representa um passo de derivação mais à direita ao inverso.

Analisador Shift-Reduce

O analisador shift-reduce tem uma pilha com símbolos da gramática e um buffer de entrada contém os restante da cadeia a ser analisada, e um handle sempre aparece no topo da pilha antes de ser identificado como um handle.

Analisador Shift-Reduce

- No inicio a pilha está vazia e contém somente o final de cadeia ($);

- O analisador então passa a deslocar os símbolos da entrada para a pilha,

- ao identificar um handle no topo da pilha efetua a redução na pilha;

- repete esse ciclo até encontrar um erro , ou até alcançar $ no buffer de entrada e no top da pilha estiver o símbolo inicial da gramática.

Analisador Shift-Reduce

- O analisador efetua 4 operações:
  - Shift – desloca o próximo símbolo do buffer de entrada para o topo da pilha;
  - Reduce – identifica um handle no topo da pilha e o substitui pelo não-terminal da produção escolhida;
  - Accept – anuncia o sucesso na análise
  - Error – ao descobrir um erro sintático

Conflitos

Seja a seguinte gramática :
S → if expr then S |
S → if expr then S else S |
S → other

E durante a análise temos a seguinte configuração

Pilha	Entrada
$…. if expr then S	else ……..$

Não há como saber se if expr then S é um handle, não importa o que tenha abaixo dele na pilha.
Este é um exemplo de conflito shift/reduce onde não se sabe qual ação tomar. O Outro conflito é reduce/reduce.

Análise Sintática LR

Um do tipos de analisadores sintáticos mais utilizados é baseado no reconhecimento LR(k), onde:

- o L representa o sentido da leitura da cadeia de entrada, da esquerda para a direita – *Left-to-right*;
- o R representa a construção das derivações mais à direita no sentido reverso – *Rightmost*;
- o k representa os símbolos à frente do fluxo de entrada que auxiliam nas decisões.(aqui $k \leq 1$)

Analisadores LR(k)

Vamos estudar 3(três) analisadores LR, desde o mais simples, SLR e dois mais complexos, LR-Canônico e LALR.

Esses analisadores são baseados em um autômato de itens LR, itens LR(0) em SLR e itens LR(1) nos outros.

Analisadores LR(k)

- Funcionamento dos Analisadores LR(k)
  - Esses analisadores sintáticos são constituídos:
    - Uma fita de entrada contendo a cadeia de tokens;
    - Uma tabela sintática particionada em Action e Goto;
    - Uma pilha de execução; e
    - O programa controlador, o mesmo para todos os parsers.
  - A porção Action da tabela sintática determina quando uma das 4 possíveis acões deve ser aplicada.

Analisadores LR(k)

- As ações são:
  - Shift para o estado I (SI) – desloca para pilha de execução o token da entrada e o esta do I;
  - Reduce by A→ XYZ (RJ) – desempilha 2*|XYZ| símbolos do topo da pilha e empilha A; em seguida consulta a porção GOTO[pilha[top-1,top] que determina o novo estado do parser, que deve ser empilhado;
  - Accept (Acc) – encerra a análise com sucesso; e
  - Error – qualquer entrada vazia da tabela. Uma rotina para tratamento de erro é chamada.

Analisadores LR(k)

- Inicialmente, o analisador tem no topo da pilha o estado inicial 0(zero), e na fita de entrada toda a cadeia de tokens;
- Observando o estado I no topo da pilha e o token a na fita de entrada, o analisador consulta a porção action da tabela em Action[I,a], e executa determina ação;
- Repete esse processo até que a análise seja encerrada ao encontrar Acc na tabela.

Analisador SLR

- Item LR(0) de uma gramática
  - é uma produção da gramática com um ponto em alguma posição do seu lado direito (corpo).
  - A produção A → XYZ gera os seguintes itens:
    - A → .XYZ
    - A → X.YZ
    - A → XY.Z
    - A→ XYZ.
  - A produção A → ε gera o item A→ .

Analisador SLR

- Intuitivamente, um item indica quanto de uma produção já foi lido no processo de reconhecimento sintático.
  - Por exemplo: A → .XYZ indica o início da busca por uma cadeia derivável de XYZ na entrada. O item A → X.YZ indica que uma cadeia derivável de X já foi encontrada e a busca deve continuar para YZ. Assim, o item A → XYZ. indica o fim da busca, ou seja, XYZ já foi derivado e que a redução para A pode ser aplicada.

Analisador SLR

- Operações sobre Itens LR(0)

  Para construir o autômato LR(0) para o analisador SLR – Simple LR, é preciso trabalhar sobre os itens então as operações closure e goto são apresentadas a seguir.

Analisador SLR

- Operação Closure(I)
  - Se I é m conjunto de itens para uma gramática G, então Closure(I) é o conjunto de itens construído a partir de I pelas regras:

    1-Inicialmente, acrescente todo item de I a closure(I).

    2-Se $A \rightarrow \alpha.B\beta$ está em closure(i) e $B \rightarrow \Upsilon$ é uma produção, então adicione o item $B \rightarrow .\Upsilon$ em Closure(I), se ainda não está lá. Aplique essa regra até que nenhum outro item possa ser incluído no Closure(I).

Analisador SLR

- Operação Goto(I,X)
  - É uma função de transição onde:
    - I é um conjunto de itens; e
    - X é um símbolo da gramática.
  - É definida como o closure do conjunto de todos os itens [A $\rightarrow \alpha X.\beta$], tais que [A $\rightarrow \alpha.X\beta$] esteja em I.

Analisador SLR

- Construção do Autômato LR(0)
  - Os estados do autômato são formados por uma coleção de conjunto de itens LR(0) chamada de Coleção Canônica de Conjunto de Itens LR(0) para uma gramática estendida G'.
  - Estender uma gramática consiste em adicionar um novo símbolo inicial S' e uma nova produção onde o novo símbolo inicial produz o antigo: S' → S.

Analisador SLR

- Algoritmo para construção da Coleção Canônica de Itens LR(0).

```
void Itens(G'){
    C = closure({[S' → .S]});
    repeat
        for (cada conjunto de itens I em C)
            for (cada símbolo da gramática X)
                if ( GOTO(I,X) não é vazio e não está em C )
                    adicione GOTO(I,X) em C;
    until nenhum novo conjunto de itens seja adicionado em
    C em uma rodada;
}
```

Analisador SLR

Algoritmo Construção de uma tabela SLR

entrada: gramática estendida G'

saída: funções Action e Goto da tabela SLR

1- construa C={I0, I1, …, In }

2- o estado i é construído a partir do conjunto Ii, as ações são definidas da seguinte maneira:

a) Se o item [A→ α.aβ] está em Ii e GOTO(Ii,a)=Ij então Action[I,a] = shift j. Aqui a deve ser terminal.

b) Se o item [A→ α.] está em Ii, então Action[i,b] = reduce by A→ α, para todo b no follow(A). Aqui A não pode ser S'

c) Se o item [S'→ S.] estiver em Ii, então Action[i,$]=ACC.

Se quaisquer ações de conflito ocorrer das regras anteriores, a gramática não é SLR(1) e o processo é encerrado.

3- As transições Goto para o estado i para todos os não-terminais A, seguindo a regra: Se GOTO(Ii,A) = Ij, então Goto[i,A] = j.

4- Todas as entradas não definidas pelas regras 2 e 3 caracterizam error.

Tratamento de Erro

Um erro é detectado no analisador sintático LR(k) quando ao consultar a tabela sintática e a entrada da tabela é vazia.
Existem duas técnicas para construir o tratamento do erro:

- Modo Pânico; e
- Recuperação em Nível de Frase

Tratamento de Erro: Modo Pânico

A ideia é ignorar símbolos da entrada até encontrar um token no conjunto de tokens de sincronismo.
Sua eficácia depende da escolha do conjunto de tokens de sincronismo, que deve permitir que o analisador se recupere rapidamente dos erros.

Algumas das heurísticas para criação do conjunto de tokens de sincronismo são:

Tratamento de Erro: Modo Pânico

1. Incluir todos os símbolos do Follow(A) no conjunto de sincronização de A
2. Adicionar os símbolos que iniciam construções sintáticas de um nível superior, além do Follow(A)
3. Incluir o First(A) no conjunto de sincronização
4. Se um não-terminal A produz a cadeia vazia, então a produção $A \rightarrow \varepsilon$ pode ser usada com padrão e adiar a detecção do erro.
5. Se o terminal A no topo da pilha não casa com o token da entrada, uma solução simples é desempilhar o símbolo e emitir uma mensagem. Na verdade neste caso estamos assumindo que todos os outros tokens estão no conjunto de sincronização.

Tratamento de Erro:Modo Pânico

A recuperação de erro no modo pânico não resolve o problema das mensagens de erro, que devem ser precisas ao descrever o erro encontrado e indicar o local onde ocorre o erro.

Tratamento de Erro: em Nível de Frase

É implementado preenchendo cada entrada vazia da tabela com um apontador para uma rotina de tratamento do erro.

Essas rotinas podem :

- substituir, inserir e remover símbolos da cadeia de entrada e emitir mensagens de erros apropriadas.
- remover símbolos da pilha. (Não é recomendável inserir ou substituir símbolos na pilha pois isto pode levar a uma construção que não existe na gramática)