

2.7 - Tradução Orientada pela Sintaxe

- ♦ O esquema de tradução orientada pela sintaxe ("syntax-oriented translation") permite expressar a geração de código intermediário, diretamente em termos da estrutura sintática da linguagem fonte. Isso é conseguido associando-se ações semânticas às produções da gramática que define a linguagem fonte.
- ♦ **Exemplo:** $A \rightarrow XYZ \quad \{ \alpha \}$

A ação semântica α será executada toda vez que o analisador sintático reconhecer na sua entrada, uma subcadeia w que possui uma derivação da forma: $A \Rightarrow XYZ \Rightarrow^* w$.

Num analisador "bottom-up" a ação é realizada quando XYZ é reduzido para A . Num analisador sintático "top-down" a ação é realizada quando A , X , Y ou Z é expandido.
- ♦ Exemplos de ações semânticas:
 - calcular valores para as variáveis
 - gerar código intermediário
 - imprimir um diagnóstico de erro
 - incluir algo na tabela de símbolos
- ♦ Embora seja possível traduzir diretamente da linguagem fonte para código de máquina ou "assembly" por meio de um esquema orientado pela sintaxe, isso é uma tarefa difícil.

Tradução Orientada pela Sintaxe

- ♦ No caso de código de máquina ou "assembly" é necessário especificar os registradores nos quais serão armazenados os resultados parciais e o uso eficiente dos registradores é tarefa difícil.
- ♦ Ao usarmos uma linguagem intermediária teremos:
 - simplicidade de comandos (como no "assembly") pois cada comando envolve apenas uma operação,
 - sem precisar especificar os registradores.

E depois, passar da linguagem intermediária para "assembly" é fácil, dada a simplicidade dos comandos.

- ♦ Vamos considerar a gramática:

$$E \rightarrow T \mid E + T \mid E - T \mid -T$$

$$T \rightarrow F \mid T * F \mid T / F$$

$$F \rightarrow I \mid (E) \quad (\text{onde } I \text{ representa um identificador})$$

Vejamos como transformar uma expressão gerada por essa gramática numa expressão em notação polonesa, associando às regras de produção ações semânticas. A idéia é simples: vamos assumir que, por exemplo, quando formos reduzir $E^2 + T$ para E^1 , já tenhamos gerado o código correspondente a E^2 e o código correspondente a T , ou seja:

[código E^2] [código T]

Assim, o código para E^1 será: [código E^2] [código T] +

Tradução Orientada pela Sintaxe

- ♦ Sejam:

$pol[]$ - array que irá conter o código em notação polonesa

p - ponteiro para a primeira posição vaga em $pol[]$

$s[I]$ - nome (ou outra informação semântica) associado a I

Então:

produção	ação semântica
$E \rightarrow T$	
$E \rightarrow E + T$	$pol[p] := '+'; \quad p := p + 1$
$E \rightarrow E - T$	$pol[p] := '-'; \quad p := p + 1$
$E \rightarrow - T$	$pol[p] := 'minus'; \quad p := p + 1$
$T \rightarrow F$	
$T \rightarrow T * F$	$pol[p] := '*'; \quad p := p + 1$
$T \rightarrow T / F$	$pol[p] := '/'; \quad p := p + 1$
$F \rightarrow I$	$pol[p] := s[I]; \quad p := p + 1$
$F \rightarrow (E)$	

Tradução Orientada pela Sintaxe

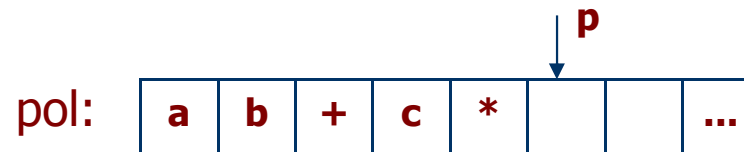
- ◆ Exemplo: Seja a expressão $(a + b) * c$

entrada	pilha sintática	produção usada	expressão polonesa
$(a + b) * c$			
$a + b) * c$	(
$+ b) * c$	(I_a		
	(F	$F \rightarrow I$	a
	(T	$T \rightarrow F$	
	(E	$E \rightarrow T$	
$b) * c$	(E +		
$) * c$	(E + I_b		
	(E + F	$F \rightarrow I$	a b
	(E + T	$T \rightarrow F$	
	(E	$E \rightarrow E + T$	a b +
	(E		
$* c$	(E)		
$* c$	F	$F \rightarrow (E)$	
$* c$	T	$T \rightarrow F$	
c	T *		

Tradução Orientada pela Sintaxe

entrada	pilha sintática	produção usada	expressão polonesa
	T * Ic		
	T * F	F → I	a b + c
	T	T → T * F	a b + c *
	E	E → T	

- ♦ ou seja,



- ♦ Vamos ver agora, ainda considerando a gramática para expressões aritméticas, como seriam as rotinas semânticas caso desejássemos transformar as expressões em quádruplas.
- ♦ Uma expressão como: $a * (b + c)$ quando transformada em quádruplas resulta em: $(+, b, c, T1)$

$(*, a, T1, T2)$

onde T_i ($i = 1, 2, \dots$) representam posições temporárias de armazenamento. Vamos supor que a informação semântica associada a um não-terminal X é representada por $X.SEM$ e que o procedimento $QUAD(w, x, y, z)$ gera uma quádrupla constituída pelos 4 parâmetros.

Tradução Orientada pela Sintaxe

- Então as ações semânticas para a geração de quádruplas serão:

note que a informação semântica associada a um não-terminal é basicamente o nome de uma posição temporária

produção	ação semântica
$E \rightarrow T$	$E.SEM := T.SEM$
$E \rightarrow E + T$	$i := i+1; E^1.SEM := Ti; QUAD(+, E^2.SEM, T.SEM, E^1.SEM)$
$E \rightarrow E - T$	$i := i+1; E^1.SEM := Ti; QUAD(-, E^2.SEM, T.SEM, E^1.SEM)$
$E \rightarrow - T$	$i := i+1; E^1.SEM := Ti; QUAD(minus, E^2.SEM, T.SEM, E^1.SEM)$
$T \rightarrow F$	$T.SEM := F.SEM$
$T \rightarrow T * F$	$i := i+1; T^1.SEM := Ti; QUAD(*, T^2.SEM, F.SEM, T^1.SEM)$
$T \rightarrow T / F$	$i := i+1; T^1.SEM := Ti; QUAD(/, T^2.SEM, F.SEM, T^1.SEM)$
$F \rightarrow I$	$F.SEM := I.SEM$
$F \rightarrow (E)$	$F.SEM := E.SEM$

mesma notação para identificadores (informação semântica pode ser o endereço na tabela de símbolos)

Tradução Orientada pela Sintaxe

- Exemplo: Seja a entrada $(a + b) * c$

produção usada na redução	ação semântica tomada
F → I	F.SEM := I.SEM := A
T → F	T.SEM := A
E → T	E.SEM := A
F → I	F.SEM := I.SEM := B
T → F	T.SEM := B
E → E + T	E.SEM := T1; QUAD(+, A, B, T1)
F → (E)	F.SEM := T1
T → F	T.SEM := T1
F → I	F.SEM := I.SEM := C
T → T * F	T.SEM := T2; QUAD(*, T1, C, T2)
E → T	E.SEM := T2

- Portanto, as quádruplas geradas são: $(+, A, B, T1)$
 $(*, T1, C, T2)$

Tradução Orientada pela Sintaxe

Processamento semântico com um analisador "top-down"

- ♦ Vamos ver agora como as rotinas semânticas podem ser associadas a um analisador "top-down". Vamos, em primeiro lugar, modificar a gramática para as expressões aritméticas, tornando-a conveniente para a análise descendente:

$$E \rightarrow [-] T \{ (+ | -) T \}$$

$$T \rightarrow F \{ (* | /) F \}$$

$$F \rightarrow I | (E)$$

onde, na primeira e segunda produções, usamos os meta-símbolos:

[] - ocorrência opcional

() - ocorrência alternativa

{ } - ocorrência em zero ou mais vezes

(notar que os () na terceira produção não são meta-símbolos).

- ♦ Vamos supor que desejamos gerar quádruplas. A informação semântica necessária para cada não-terminal será passada via parâmetro para o procedimento (possivelmente) recursivo associado àquele não-terminal. Teremos então:

Tradução Orientada pela Sintaxe

```
procedure E(string X);
begin
  string Y,Z,OP;
  OP := NEXTSIMB;
  if OP = '-' then
    begin
      SCAN;
      T(Y); j := j+1; QUAD(OP,0,Y,Tj);
      Y := Tj;
    end
  else T(Y);
  OP := NEXTSIMB;
  while OP = '-' or OP = '+' do
    begin
      SCAN;
      T(Z); j := j+1; QUAD(OP,Y,Z,Tj);
      Y := Tj;
      OP := NEXTSIMB;
    end;
  X := Y;
end;
```

$$E \rightarrow [-]T\{ (+ | -)T \}$$
$$T \rightarrow F\{ (* | /) F \}$$

```
procedure T(string X);
begin
  string Y,Z,OP;
  F(Y);
  OP := NEXTSIMB;
  while OP = '*' or op = '/' do
    begin
      SCAN;
      F(Z); j := j+1; QUAD(OP,Y,Z,Tj);
      Y := Tj;
      OP := NEXTSIMB;
    end;
  X := Y;
end;
```

Tradução Orientada pela Sintaxe

$F \rightarrow I \mid (E)$

```
procedure F(string X);
begin
  if NEXTSIMB = I then
    begin
      X := I.SEM;
      SCAN;
    end
  else
    if NEXTSIMB ≠ '(' then ERRO
    else
      begin
        SCAN;
        E(X);
        if NEXTSIMB ≠ '(' then ERRO
        else
          SCAN;
        end;
      end;
    end;
```

- ♦ Vamos ver agora a tradução orientada pela sintaxe de alguns comandos de uma linguagem de programação. Vamos admitir que desejamos gerar quádruplas e que iremos usar um analisador "bottom-up".

Tradução Orientada pela Sintaxe

Tradução de comandos de atribuição

- Seja a seguinte gramática, descrevendo os comandos de atribuição:

$A \rightarrow \langle \text{id} \rangle := E$

$E \rightarrow E \langle \text{op} \rangle E \mid - E \mid (E) \mid \langle \text{id} \rangle$

Vamos admitir que a atribuição ($A := B \langle \text{op} \rangle C$)

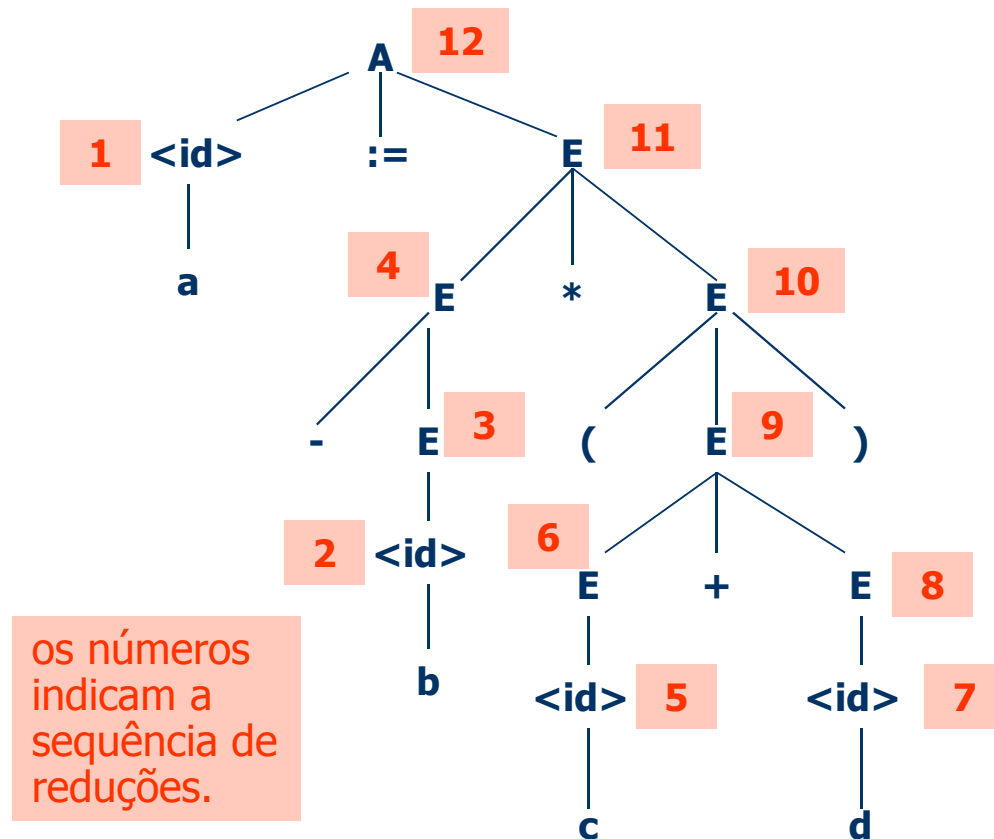
é representada pela quádrupla ($\langle \text{op} \rangle, B, C, A$)

Então:

produção	ação semântica
$A \rightarrow \langle \text{id} \rangle := E$	$(:=, E.SEM, , \langle \text{id} \rangle)$
$E \rightarrow E^1 \langle \text{op} \rangle E^2$	$(\langle \text{op} \rangle, E^1.SEM, E^2.SEM, E.SEM)$
$E \rightarrow - E^1$	$(-, 0, E^1.SEM, E.SEM)$
$E \rightarrow (E^1)$	$E.SEM := E^1.SEM$
$E \rightarrow \langle \text{id} \rangle$	$E.SEM := \langle \text{id} \rangle.SEM$

Tradução Orientada pela Sintaxe

Exemplo: Seja a entrada $a := - b * (c + d)$



Quádruplas geradas: $(-, 0, b, T1)$
 $(+, c, d, T2)$
 $(*, T1, T2, T3)$
 $(:=, T3, , a)$

redução	ação semântica tomada
1	$\langle id \rangle.SEM := a$
2	$\langle id \rangle.SEM := b$
3	$E.SEM := b$
4	$(-, 0, b, E.SEM)$ $(E.SEM = T1)$
5	$\langle id \rangle.SEM := c$
6	$E.SEM := c$
7	$\langle id \rangle.SEM := d$
8	$E.SEM := d$
9	$(+, c, d, E.SEM)$ $(E.SEM = T2)$
10	$E.SEM := T2$
11	$(*, T1, T2, E.SEM)$ $(E.SEM = T3)$
12	$(:=, T3, , a)$

Tradução Orientada pela Sintaxe

Tradução de expressões booleanas

- ♦ Seja a seguinte gramática, descrevendo expressões booleanas:

$E \rightarrow E \text{ or } E \mid E \text{ and } E \mid \text{not } E \mid (E) \mid \langle \text{id} \rangle \mid \langle \text{id} \rangle \langle \text{relop} \rangle \langle \text{id} \rangle$

onde $\langle \text{relop} \rangle$ pode ser $<$, $<=$, $=$, $<>$, $>$, $>=$.

Se o valor da expressão booleana é representado numericamente (por exemplo, 0 para false e 1 para true), a tradução pode ser feita de forma similar à tradução de expressões aritméticas.

- ♦ Exemplo: a expressão relacional $A < B$ é equivalente a:

if $A < B$ then 1 else 0

e pode, portanto, ser traduzida para (usando código de 3 endereços):

1. if $A < B$ goto (4)
2. $T := 0$
3. goto (5)
4. $T := 1$

- ♦ Pensando dessa maneira, as ações semânticas associadas às produções:

$E \rightarrow E \text{ or } E$

e

$E \rightarrow \langle \text{id} \rangle \langle \text{relop} \rangle \langle \text{id} \rangle$

são dadas por:

Tradução Orientada pela Sintaxe

produção	ação semântica
$E \rightarrow E^1 \text{ or } E^2$	<pre>T := NEWTEMP(); (OR, E¹.SEM, E².SEM, T); E.SEM := T</pre>
$E \rightarrow \langle \text{id} \rangle^1 \langle \text{relop} \rangle \langle \text{id} \rangle^2$	<pre>T := NEWTEMP(); (IF<relop>, <id>¹, <id>², NQUAD + 3); (:=, 0, , T); (GOTO, , , NQUAD + 2); (:=, 1, , T); E.SEM := T</pre>

- ♦ onde: NEWTEMP() é um procedimento para gerar um nome temporário e NQUAD é o número da quádrupla sendo gerada.
- ♦ Uma outra maneira de representar valores de expressões booleanas é através de fluxo de controle. Dessa maneira podemos, como veremos a seguir, mesmo sem avaliar a expressão booleana inteira, tomar a ação semântica correta.
- ♦ Exemplo: na expressão A or B, se A = true então podemos concluir (e tomar a ação pertinente) que o valor da expressão é true sem precisar avaliar B. Analogamente, a expressão A and B é false se A = false, independente de B.

Tradução Orientada pela Sintaxe

- ♦ Vamos considerar a tradução de expressões booleanas no contexto de comandos como:

if E then S1 else S2

while E do S

Nesses comandos podemos associar à expressão booleana E duas saídas: a saída E.TRUE e a saída E.FALSE.

- ♦ Exemplo: Seja o comando:

if (A < B or C < D) X = Y + Z

Esse comando será traduzido como:

1. IF A < B GOTO (4)
2. IF C < D GOTO (4)
3. GOTO (6)
4. T1 := Y + Z
5. X := T1
- 6.

e, nesse caso, E.TRUE = { 4 } e E.FALSE = { 6 }.

- ♦ Vamos considerar a tradução para quádruplas de expressões booleanas em contextos como este.

Tradução Orientada pela Sintaxe

- ♦ Um problema que pode ocorrer é o seguinte:

Podemos ainda não ter gerado a quádrupla para a qual um desvio deve ser feito. Por exemplo, ao gerar a quádrupla (1) do exemplo anterior, o desvio é para uma quádrupla ainda não gerada (qual seria o número dessa quádrupla?). Em casos assim, o procedimento será:

- a) deixar "em branco" o número da quádrupla para a qual o desvio será feito;
- b) armazenar a quádrupla atual em uma lista de quádruplas ainda incompletas.

Quando as quádruplas forem geradas, a lista é recuperada e suas quádruplas completadas.

- ♦ Vamos assumir a existência de três funções:

MAKELIST(i)	cria uma lista contendo o índice i.
MERGE(p_1 , p_2)	cria uma nova lista que é a união das listas apontadas por p_1 e p_2 .
BACKPATCH(p, i)	coloca o índice i nos espaços em branco das quádruplas da lista p (para completar as quádruplas).

Tradução Orientada pela Sintaxe

- ♦ Vamos introduzir um novo não-terminal M na gramática que irá servir de marcador para indicar quando devemos "remendar" ("backpatch") uma quádrupla. Esse novo não-terminal terá a seguinte produção e ação semântica associadas:

produção	ação semântica
$M \rightarrow \Lambda$	$M.QUAD := NQUAD$

não altera a linguagem

armazena o número da quádrupla para a qual o desvio deve ser feito

- ♦ Observação: Uma outra maneira de fazer a "marcação" é quebrar as produções nos locais que irão requerer "backpatch". Esse esquema é utilizado em: **GRIES** - "Compiler construction for digital computers" e tem a vantagem de poder ser facilmente implementado.
- ♦ Introduzindo M teremos a gramática:

$E \rightarrow E^1 \text{ or } M E^2 \mid E^1 \text{ and } M E^2 \mid \text{not } E^1 \mid (E^1) \mid \langle \text{id} \rangle \mid \langle \text{id} \rangle^1 \langle \text{relop} \rangle \langle \text{id} \rangle^2$
 $M \rightarrow \Lambda$

Tradução Orientada pela Sintaxe

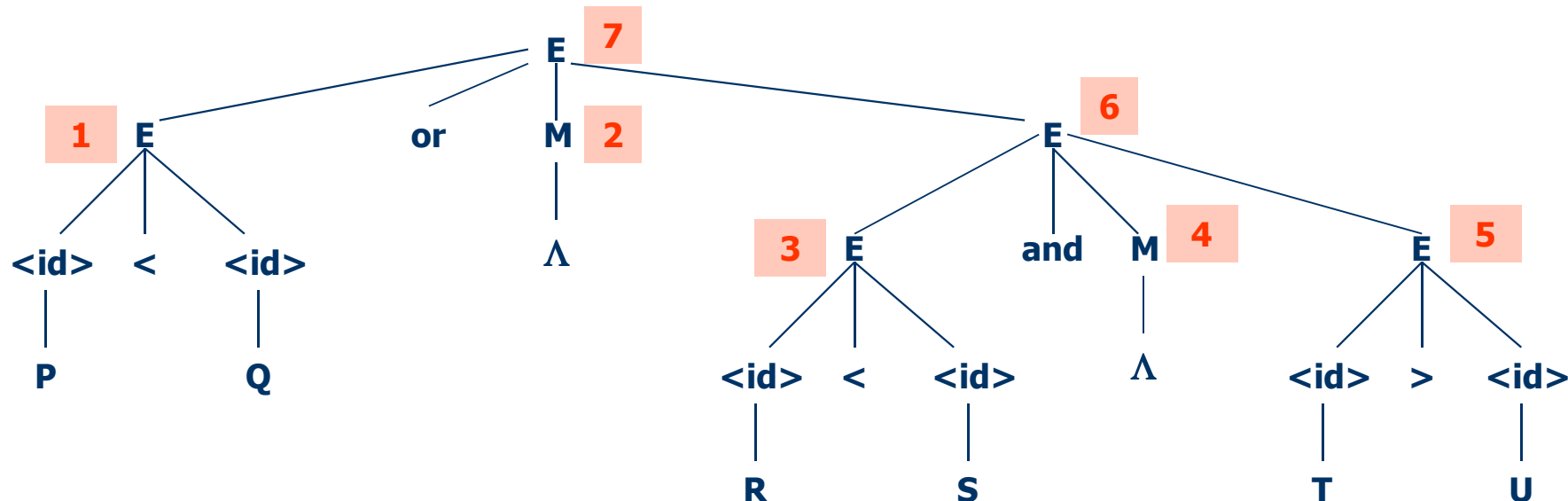
- ♦ O esquema de tradução será:

produção	ação semântica
$E \rightarrow E^1 \text{ or } M E^2$	<code>BACKPATCH(E¹.FALSE, M.QUAD)</code> <code>E.TRUE := MERGE(E¹.TRUE, E².TRUE)</code> <code>E.FALSE := E².FALSE</code>
$E \rightarrow E^1 \text{ and } M E^2$	<code>BACKPATCH(E¹.TRUE, M.QUAD)</code> <code>E.TRUE := E².TRUE</code> <code>E.FALSE := MERGE(E¹.FALSE, E².FALSE)</code>
$E \rightarrow \text{not } E^1$	<code>E.TRUE := E¹.FALSE</code> <code>E.FALSE := E¹.TRUE</code>
$E \rightarrow (E^1)$	<code>E.TRUE := E¹.TRUE</code> <code>E.FALSE := E¹.FALSE</code>
$E \rightarrow \langle \text{id} \rangle$	<code>E.TRUE := MAKELIST(NQUAD)</code> <code>E.FALSE := MAKELIST(NQUAD+1)</code> <code>GEN(IF <id>.SEM GOTO ____);</code> <code>GEN(GOTO ____);</code>

Tradução Orientada pela Sintaxe

produção	ação semântica
$E \rightarrow \langle id \rangle^1 \langle relop \rangle \langle id \rangle^2$	$E.TRUE := MAKELIST(NQUAD)$ $E.FALSE := MAKELIST(NQUAD+1)$ $GEN(IF \langle id \rangle^1.SEM \langle relop \rangle \langle id \rangle^2.SEM GOTO ___);$ $GEN(GOTO ___);$
$M \rightarrow \Lambda$	$M.QUAD := NQUAD$

- Exemplo: Seja a expressão: $P < Q$ or $(R < S$ and $T > U)$.
A árvore sintática para essa expressão será:



Tradução Orientada pela Sintaxe

- ♦ Vamos supor que, inicialmente, NQUAD = 100.

<u>redução 1:</u>	100 IF P < Q GOTO ____	E.TRUE = { 100 }
	101 GOTO ____	E.FALSE = { 101 }
<u>redução 2:</u>	M.QUAD = 102	
<u>redução 3:</u>	102 IF R < S GOTO ____	E.TRUE = { 102 }
	103 GOTO ____	E.FALSE = { 103 }
<u>redução 4:</u>	M.QUAD = 104	
<u>redução 5:</u>	104 IF T > U GOTO ____	E.TRUE = { 104 }
	105 GOTO ____	E.FALSE = { 105 }
<u>redução 6:</u>	BACKPATCH({ 102 }, 104)	
	E.TRUE = { 104 }	
	E.FALSE = { 103, 105 }	
<u>redução 7:</u>	BACKPATCH({ 101 }, 102)	
	E.TRUE = { 100, 104 }	
	E.FALSE = { 103, 105 }	

```
100 IF P < Q GOTO ____
101 GOTO 102
102 IF R < S GOTO 104
103 GOTO ____
104 IF T > U GOTO ____
105 GOTO ____
```

onde, como $E.TRUE = \{ 100, 104 \}$, as quádruplas 100 e 104 serão preenchidas com o número da quádrupla correspondente à ação tomada no caso da expressão ser verdadeira, e como $E.FALSE = \{ 103, 105 \}$, a expressão é falsa se e somente se as quádruplas 103 e 105 forem atingidas (notar que o código gerado não é otimizado).

Tradução Orientada pela Sintaxe

Tradução de comandos de controle de fluxo

- ♦ O comando de controle de fluxo mais elementar é o GOTO L. O compilador, ao encontrar um GOTO L, deve procurar por L na tabela de símbolos. Se o desvio é "para cima", L já deve ter sido encontrado e na tabela de símbolos deverá existir o número da primeira quádrupla gerada para o comando de rótulo L. Nesse caso, deve ser gerado um GOTO para esse endereço. Se o desvio é "para baixo", essa pode ser a primeira ocorrência de rótulo L e, nesse caso, L deve ser colocado na tabela de símbolos. De qualquer forma, se o desvio é "para baixo" podemos apenas gerar uma quádrupla GOTO ____, sem especificar o endereço de destino e adicionar essa quádrupla à lista cujo destino é L (um ponteiro para essa lista deverá existir para L na tabela de símbolos).
- ♦ A sintaxe dos comandos rotulados é dada por:

$S \rightarrow \text{LABEL} : S$

$\text{LABEL} \rightarrow \langle \text{id} \rangle$

A ação semântica de $\text{LABEL} \rightarrow \langle \text{id} \rangle$ será:

- a) colocar $\langle \text{id} \rangle$ na tabela de símbolos (caso ainda não esteja lá)
- b) salvar o número da quádrupla referenciada por esse rótulo (NQUAD)
- c) remendar a lista de quádruplas GOTO que tem $\langle \text{id} \rangle$ como destino.

Tradução Orientada pela Sintaxe

- ♦ Vamos considerar a seguinte gramática para construções de controle:
 1. $S \rightarrow \text{IF } E \text{ THEN } S \mid$
 2. $\text{IF } E \text{ THEN } S \text{ ELSE } S \mid$
 3. $\text{WHILE } E \text{ DO } S \mid$
 4. $\text{BEGIN } L \text{ END } \mid$
 5. A
 6. $L \rightarrow L ; S \mid$
 7. S
- ♦ O não-terminal E possui os campos de tradução E.TRUE e E.FALSE como vimos anteriormente. L e S também irão precisar de uma lista de quádruplas incompletas (que precisarão ser remendadas por BACKPATCH). L e S terão campos de tradução L.NEXT e S.NEXT que serão ponteiros para uma lista de todos os desvios condicionais ou incondicionais para a quádrupla que irá seguir L ou S na ordem de execução.
- ♦ Exemplo: **while E do S** - o código testa E e se true, executa S. Após executar S o código desvia para o teste de E e não para a quádrupla seguinte ao código para S (notar que S também pode ser um **while** e que poderá haver muitos níveis de "aninhamento"). Para isso, introduzimos o "marcador":

Tradução Orientada pela Sintaxe

$S \rightarrow \text{while } M^1 \text{ E do } M^2 S^1$

$M \rightarrow \Lambda$

Dessa forma, quando reduzirmos $\text{while } M^1 \text{ E do } M^2 S^1$ para S , um "remendo" é feito na lista $S^1.NEXT$ para o destino $M^1.QUAD$ e um "remendo" é feito na lista $E.TRUE$ com destino $M^2.QUAD$.

- ♦ Vejamos outro caso: **if E then S^1 else S^2** . Depois de gerar código para S^1 não temos idéia para onde desviar pois a quádrupla seguinte ao código de S^2 só será conhecida após terminarmos de gerar código para S^2 . Nesse caso, $S^1.NEXT$ e $S^2.NEXT$ resolverão o problema, por meio do uso de marcadores. Notar que para o **if** devemos ter um marcador antes de S^1 ($E.TRUE$) e um após S^1 (desvio a ser feito após S^1 ter sido executado), além de um marcador antes de S^2 ($E.FALSE$).
- ♦ Temos portanto a seguinte gramática modificada e ações semânticas:

produção	ação semântica
$S \rightarrow \text{if } E \text{ then } M^1 S^1 N \text{ else } M^2 S^2$	$\text{BACKPATCH}(E.TRUE, M^1.QUAD)$ $\text{BACKPATCH}(E.FALSE, M^2.QUAD)$ $S.NEXT := \text{MERGE}(S^1.NEXT, N.NEXT, S^2.NEXT)$

Tradução Orientada pela Sintaxe

produção	ação semântica
$N \rightarrow \Lambda$	$N.NEXT := MAKELIST(NQUAD)$ $GEN(GOTO ___)$
$M \rightarrow \Lambda$	$M.QUAD := NQUAD$
$S \rightarrow \text{if } E \text{ then } M S^1$	$BACKPATCH(E.TRUE, M.QUAD)$ $S.NEXT := MERGE(E.FALSE, S1.NEXT)$
$S \rightarrow \text{while } M^1 E \text{ do } M^2 S^1$	$BACKPATCH(E.TRUE, M2.QUAD)$ $BACKPATCH(S1.NEXT, M1.QUAD)$ $S.NEXT := E.FALSE$ $GEN(GOTO M1.QUAD)$
$S \rightarrow \text{begin } L \text{ end}$	$S.NEXT := L.NEXT$
$S \rightarrow A$	$S.NEXT := MAKELIST()$
$L \rightarrow L^1; M S$	$BACKPATCH(L1.NEXT.M.QUAD)$ $L.NEXT := S.NEXT$
$L \rightarrow S$	$L.NEXT := S.NEXT$

cria uma lista vazia

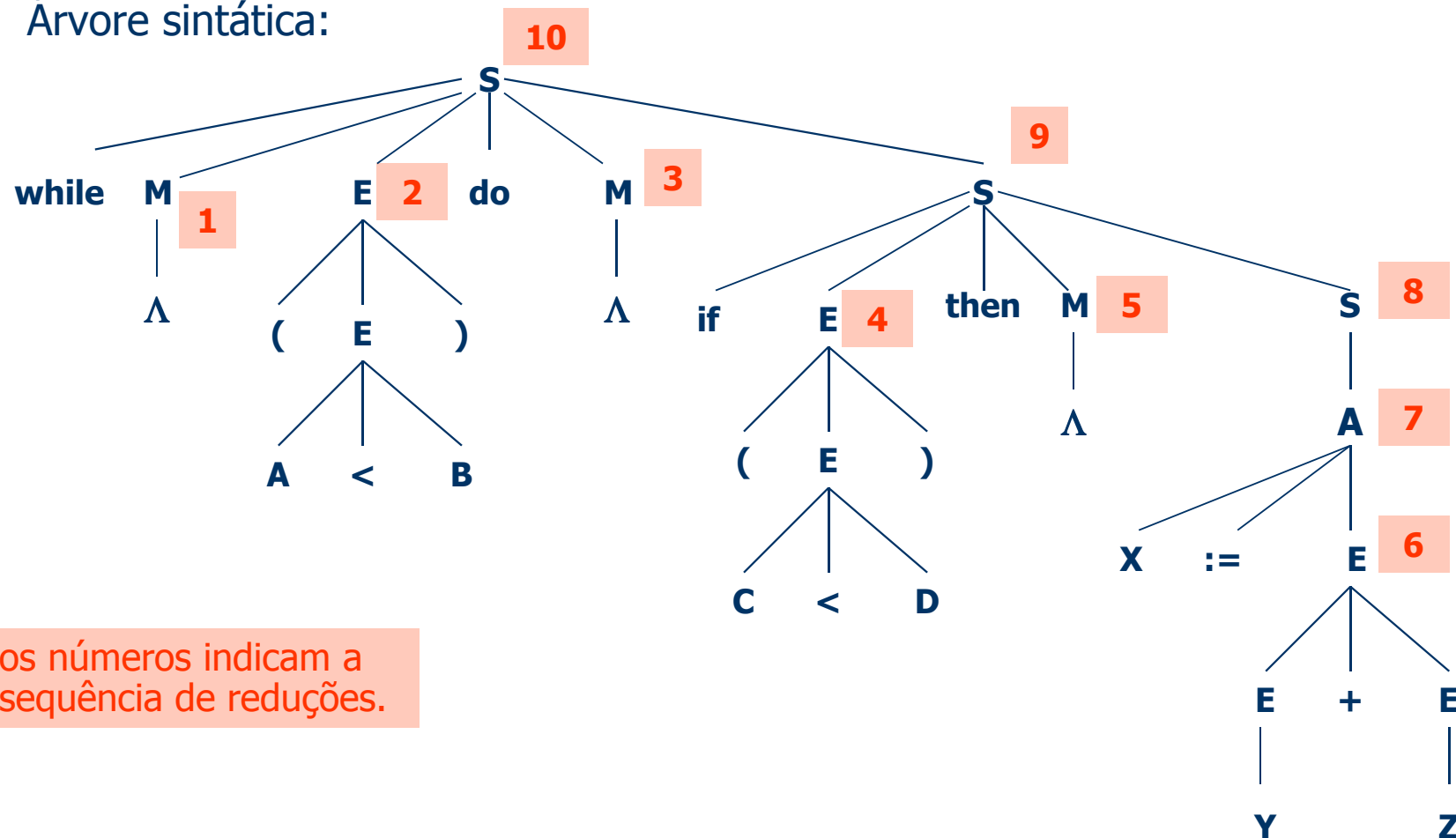
Tradução Orientada pela Sintaxe

- Exemplo: Seja o comando:

WHILE (A < B) DO

IF (C < D) THEN X := Y + Z

Árvore sintática:



os números indicam a
sequência de reduções.

Tradução Orientada pela Sintaxe

- Assumindo, inicialmente, NQUAD = 100.

redução	ação semântica tomada
1	M.QUAD := 100
2	100: IF A < B GOTO ____ E.TRUE := { 100 } 101: GOTO ____ E.FALSE := { 101 }
3	M.QUAD := 102
4	102: IF C < D GOTO ____ E.TRUE := { 102 } 103: GOTO ____ E.FALSE := { 103 }
5	M.QUAD := 104
6	104: T1 := Y + Z
7	105: X := T1
8	S.NEXT := { }
9	BACKPATCH({102}, 104) S.NEXT := { 103 }
10	BACKPATCH({100}, 102) BACKPATCH({103}, 100) S.NEXT := { 101 } 106: GOTO 100

Código gerado:

```

100: IF A < B GOTO 102
101: GOTO ____
102: IF C < D GOTO 104
103: GOTO 100
104: T1 := Y + Z
105: X := T1
106: GOTO 100
107:

```