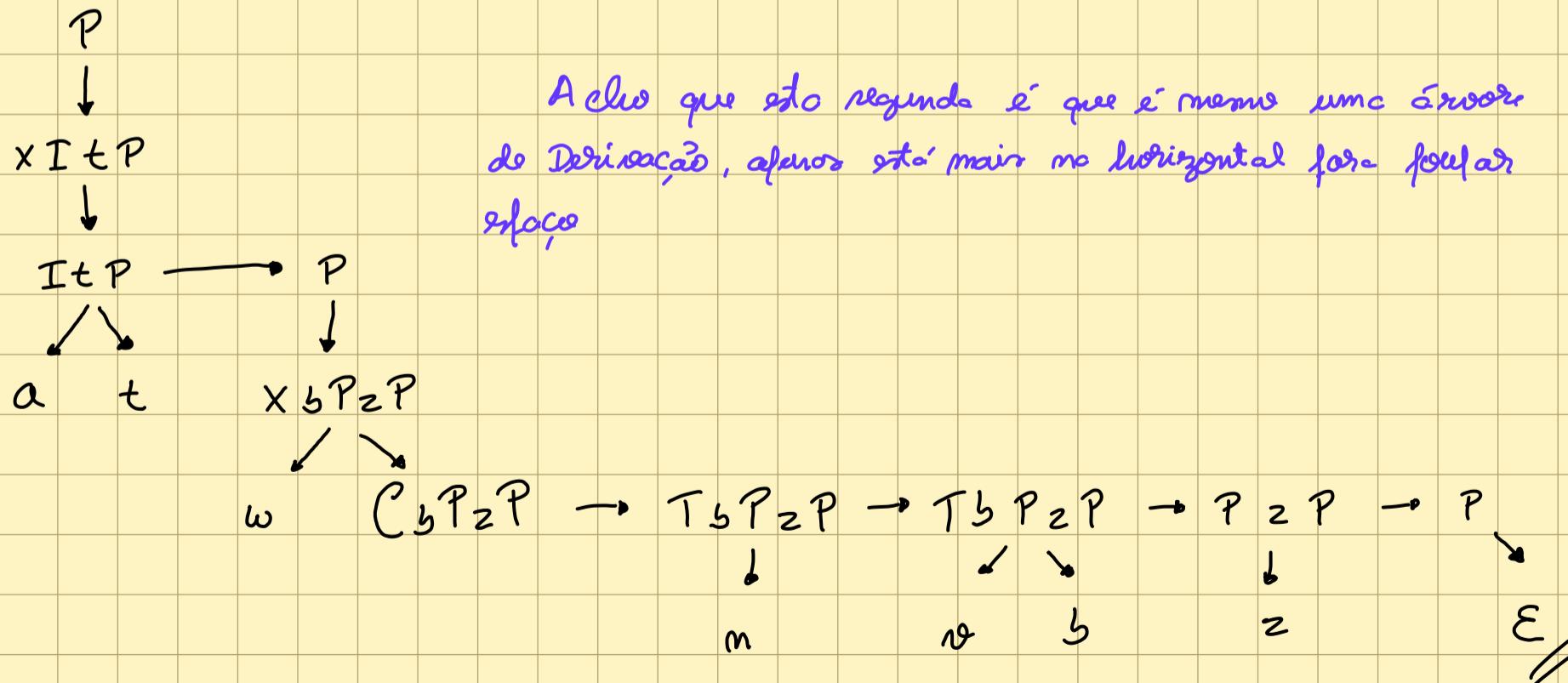
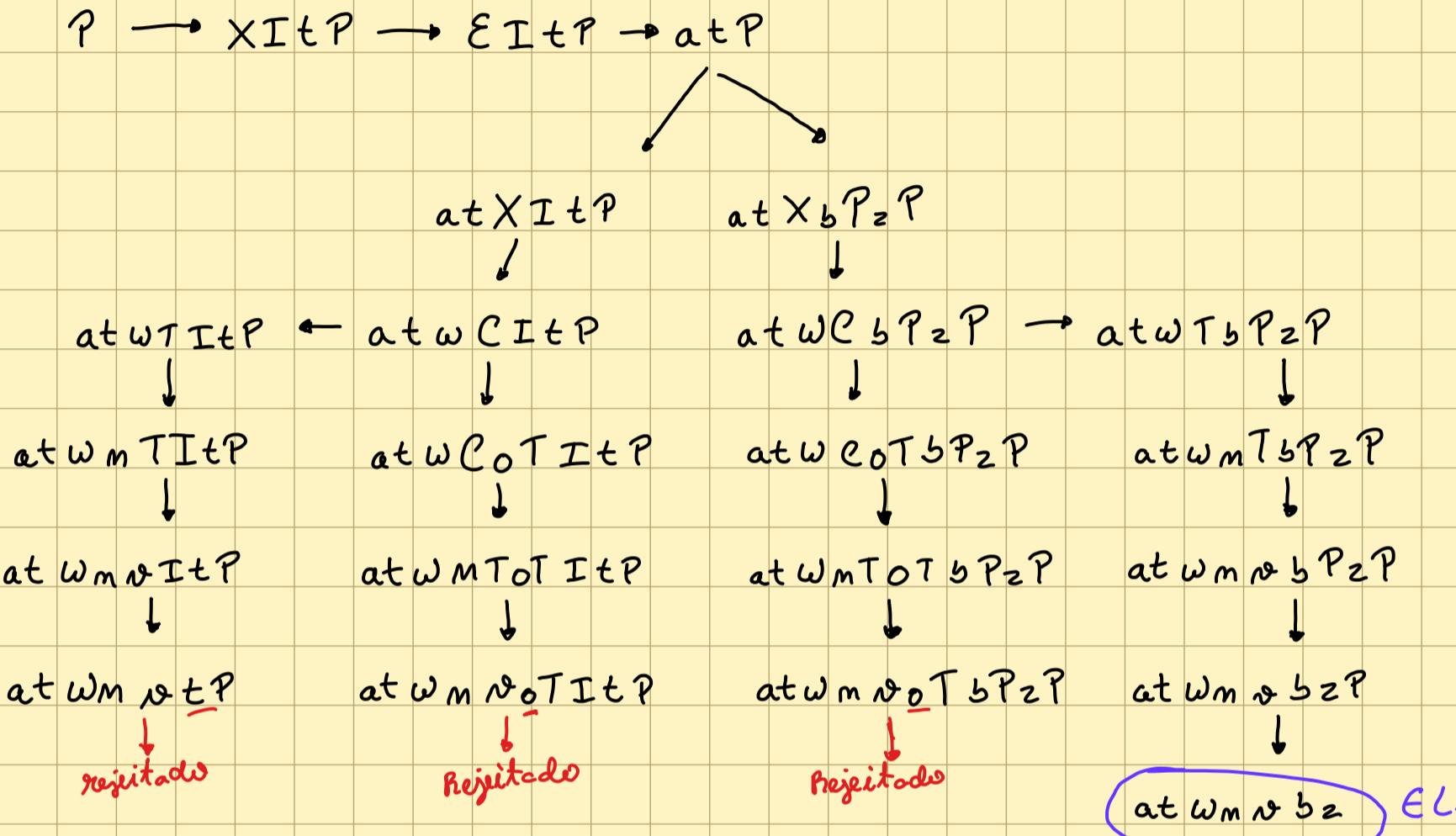


1. Sobre o alfabeto $T_1 = \{t, b, z, w, a, o, v, n\}$ considere a gramática G_1 dada a seguir e seja L_1 a linguagem por ela descrita.

$$\begin{array}{lcl} P & \rightarrow & \varepsilon \mid X \ I \ t \ P \mid X \ b \ P \ z \ P \\ X & \rightarrow & \varepsilon \mid w \ C \\ I & \rightarrow & \varepsilon \mid a \\ C & \rightarrow & T \mid C \circ T \\ T & \rightarrow & v \mid n \ T \end{array}$$

- [1,5] (a) Mostre que $atwnvbz \in L_1$.
 [1,5] (b) Avalie a veracidade da afirmação: $\{w, t\} \subset \text{first}(XItP)$. Apresente os passos intermédios e/ou o raciocínio adequados para suportar a sua resposta.
 [1,5] (c) Avalie a veracidade da afirmação: $t \in \text{follow}(T)$. Apresente os passos intermédios e/ou o raciocínio adequados para suportar a sua resposta.
 [2,0] (d) Calcule o conjunto $\text{predict}(P \rightarrow XItP)$. Apresente os passos intermédios e/ou o raciocínio adequados para suportar a sua resposta.
 [2,0] (e) As produções começadas por P e C tornam a gramática G_1 inadequada à implementação de um reconhecedor descendente com lookahead de 1. Altere-a de forma a obter uma equivalente que o permita.

a) 1º abordagem, com árvore de derivação (Descendente):



- 2º maneira, com Tabela de derivação (Arredondado)

Pilha	Entrada	Próxima ação
	at w m n b z \$	Deslocamento
a	t w m n b z \$	Deslocamento
a t	w m n b z \$	Deslocamento
at w	m n b z \$	Deslocamento
at w m	n b z \$	Deslocamento
at w m n	b z \$	Redução $T \rightarrow N$
at w m T	b \$	Redução $T \rightarrow mT$
at w T	b \$	Redução $C \rightarrow T$
at w C	b \$	Redução $X \rightarrow wC$
at X	b \$	Deslocamento
at X b	z \$	Deslocamento
at X b z	\$	Redução $P \rightarrow X \cup P \cup P$
at P	\$	Redução $P \rightarrow X \cup t \cup P$
P	\$	Deslocamento
\$		A saída $\in L_1$

b) Mostrar que $A \subseteq B$, como no teste anterior: $\forall x \in A : x \in B \wedge \exists x \in B : x \notin A$
 $\{w, t\} \subseteq \text{first}(X \cup t \cup P)$

- Abordagem por teoria dos conjuntos:

→ Verificar $\{w, t\}$

→ Calcular first

$X \cup t \cup P \Rightarrow wC \cup I \cup t \cup P \rightarrow w \in \text{first}(X \cup t \cup P)$

$X \cup t \cup P \Rightarrow I \cup t \cup P \Rightarrow t \cup P \rightarrow t \in \text{first}(X \cup t \cup P)$

$X \cup t \cup P \Rightarrow I \cup t \cup P \Rightarrow at \cup P \rightarrow a \in \text{first}(X \cup t \cup P)$

∴ Está provado que $\{w, t\}$ está contido no $\text{first}(X \cup t \cup P)$, no entanto, não são os únicos elementos do conjunto, pois $\{a\}$ também está, provando que estão no conjunto, mas não são o conjunto.

- Calcular logo o first , liga aplicação do algoritmo:

Partir

$$\begin{aligned} \text{first}(X \cup I \cup t \cup P) &= \text{first}(I \cup t \cup P) \cup \text{first}(wC \cup I \cup t \cup P) \\ &= \underbrace{\text{first}(t \cup P)}_{\{t\}} \cup \underbrace{\text{first}(a \cup t \cup P)}_{\{a\}} \cup \{w\} = \{w, t, a\} \subseteq \text{first}(X \cup I \cup t \cup P) \end{aligned}$$

$$\forall_{x \in A} : A \in B = \{w, t\} \in \text{first}(x|I+tP)$$

$$\exists_{x \in B} : B \notin A = \{a\} \in \text{first}(x|I+tP) \wedge \{a\} \notin \{w, t\},$$

c)

$$\begin{aligned} X|I+tP &\Rightarrow wC|I+tP \\ &\Rightarrow wT|I+tP \\ &\Rightarrow wTtP \rightarrow t \in \text{follow}(T), \text{ é possível} \\ &\quad \text{colocar } o t, \text{ loop a seguir ao } T \end{aligned}$$

- Usando o algoritmo de follow:
 - Quais produções têm o T à direita? C e T

$X|T \rightarrow mT$: Não ajuda pq já

$C \rightarrow T$: Em qualquer um dos casos, se valemos que $\text{follow}(T) \subseteq \text{follow}(C)$

$|C|OT$ que estiver à frente do C, vai ficar à frente do T, ou seja $\text{follow}(T) \supseteq \text{follow}(C)$

$f(T)$ inclui $f(c) : C \rightarrow \alpha T$

- Calcular o follow do C, fará nos produções com C do lado direito: X, C

$X \rightarrow wC$: $\text{follow}(c) \supseteq \text{follow}(x)$ Terminal

$C \rightarrow C|OT$ está à frente do C, rendendo $O \in \text{follow}(C)$, com $\text{follow}(C) \supseteq \{O\}$

- Calcular o follow de X, ou seja, produções com X do lado direito: 2 Prod. em P

$P \rightarrow X|I+tP$ Calcular o first do $I+tP$, que $\{a, t\} \rightarrow \text{follow}(x) \supseteq \{a, t\}$
 $|X|bPzP \rightarrow \text{follow}(x) \supseteq \{b\}$

Então, o $\text{follow}(x) = \{a, b, t\}$, o $\text{follow}(C) = \text{follow}(T) = \{a, b, t, O\}$,

D)

↳ Algoritmo Predict:

- Calcular o $\text{first}(X|I+tP) = \{w, t, a\}$
- $\epsilon \in \text{first}(x) = \text{Não!}$
- Predict = $\{w, t, a\}$

$$\begin{aligned} \text{Partir} \quad & wC \\ \text{first}(X|I+tP) &= \text{first}(I|tP) \cup \text{first}(w|C|I+tP) \\ & \quad \text{Terminal} \\ & \quad a \\ & \quad \{w\} \\ &= \underbrace{\text{first}(t|P)}_{\{\epsilon\}} \cup \underbrace{\text{first}(a|tP)}_{\{a\}} \cup \{w\} = \{w, t, a\} \subseteq \text{first}(X|I+tP) \end{aligned}$$

e) Reconhecedor Descendente: Não pode ter recursividade à esquerda, em contrário é implementado com LL(1)

Nota: Aqui não é como em Ante, não existe ordem de execução da regras, $A \mid B = B \mid A$, dai termos de usar outras técnicas para implementar essa ordem.

- Um Parser Descendente toma decisões com base no Predict, não sei porque é que esta gramática é inadequada:

$P \rightarrow X I t P$

$P \rightarrow X b P_2 P$

} Para tomar uma decisão, o Parser vai calcular o Predict dos dois, mas o Predict dos dois vai ter o W em comum. Com um lookahead de 1, ele não vai conseguir analisar mais tokens para tomar uma decisão e vai falhar! $\rightarrow \text{Predict}(P_1) \cap \text{Predict}(P_2)$, tem de ter \emptyset , senão não dá. Neste caso é $\{W\}$

$C \rightarrow T$

$| C_0 T$

} Tem recursividade à esquerda

- Termos de resolver problema nos P:

$P \rightarrow X I t P$

$| X b P_2 P$

$| \epsilon$

$P \rightarrow X \underbrace{(I t P \mid b P_2 P)}_{Z} | \epsilon$

$\boxed{\begin{array}{l} P \rightarrow X Z | \epsilon \\ Z \rightarrow I t P \mid b P_2 P \end{array}}$

} Generalizando a gramática por Fatorização à esquerda

- Resolver recursividade à esquerda em C:

$C \Rightarrow C_0 T \Rightarrow C_0 T_0 T$

$\Rightarrow^* C(O T)^m, m \geq 0$

$\Rightarrow T(\underbrace{O T}_W)^m, m \geq 0 \rightarrow$ fecho de OT

$C \rightarrow T W$

$W \rightarrow \epsilon \mid O T W$

Transformado em Recursividade à direita

\therefore Expressões em P sofrem de fatorização à esquerda, têm prefixes comuns e os em C tem recursividade à esquerda.

2. Considere o alfabeto $A = \{a, b, c\}$ e seja L_2 o conjunto de todas as expressões regulares definíveis sobre o alfabeto A . L_2 é uma linguagem independente do contexto definida sobre o alfabeto $T_2 = A \cup \{(), *, +\}$, em que $*$ representa o operador de fecho e $+$ o operador de escolha; operação de concatenação tem o operador implícito. Em termos de precedência, da mais alta para a mais baixa, estão as operações de fecho, concatenação e escolha. Os parêntesis podem ser usados para alterar a precedência por defeito.

[3,0] (.) Construa uma gramática independente do contexto que represente a linguagem L_2 .

Exemplo com expressões

númericas

$$: (N) + (N * N / N) - (N + N)$$

Parêntesis
implícitos

$$\begin{aligned} \text{Em Antlr: } E &\rightarrow E^* E \\ &\mid E^+ E \end{aligned}$$

Não funciona em Gramáticos

independente de contexto, pois não da ordem

→ É uma soma ou subtração de T (a ordem continua a não ser igual, porque a ' $-$ ' não é Comutativa)

↓

tem de ser resolvida

→ A multiplicação é Comutativa, mas a divisão não

de esquerda para o direito

$$\left[\begin{array}{c} T \\ (N) + (N * N / N) \\ E \end{array} \right] - (N + N) \quad \} \text{Garantir que a '+' é feita antes da '-'}$$

$$\left. \begin{array}{l} E \rightarrow E + T \\ \mid E - T \\ \mid T \end{array} \right\} \text{Associação de à esquerda forçada (os elementos à esquerda não são feitos primeiro)}$$

$$\begin{array}{c} T \rightarrow T * F \\ \mid T / F \\ \mid F \end{array}$$

$$\begin{array}{c} F \rightarrow N \\ \mid (E) \end{array}$$

- Respondendo à questão: Operações mas expressões regulares → fecho, concatenação, escolha

$$\begin{array}{c} E \rightarrow E + T \\ \mid T \end{array}$$

$$\begin{array}{c} T \rightarrow T * F \rightarrow \text{explícito} \\ \mid F \rightarrow \text{implícito} \\ \text{Concatenação} \end{array}$$

$$\begin{array}{c} F \rightarrow F * \\ \mid O \\ \text{Fechar} \end{array}$$

$$\begin{array}{c} O \rightarrow a \mid b \mid c \\ \mid d \mid (E) \\ \text{Termos ou Operadores} \end{array}$$

Nota: O menor precedente é o mais abaixo

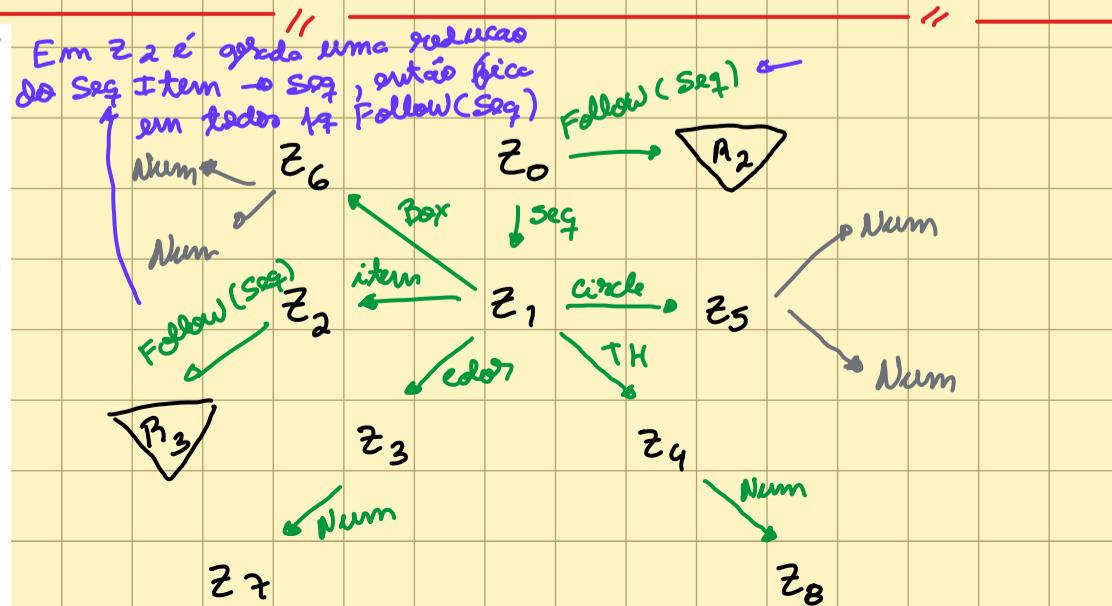
3. Sobre o alfabeto $T_3 = \{\text{NUM, BOX, CIRCLE, THICKNESS, COLOR, '(', ')'}\}$, considere a gramática G_3 dada a seguir e seja L_3 a linguagem por ela descrita.

$$\begin{array}{l} \text{draw} \rightarrow \text{seq} \\ \text{seq} \rightarrow \epsilon \\ \mid \text{seq item} \\ \text{item} \rightarrow \text{COLOR NUM} \\ \mid \text{THICKNESS NUM} \\ \mid \text{CIRCLE point NUM} \\ \mid \text{BOX point '(', seq ')'} \\ \text{point} \rightarrow \text{NUM NUM} \end{array}$$

4 Produção ($x_1 \dots x_4$)

Considere ainda o conjunto de estados (conjuntos de itens) usado na construção de um reconhecedor ascendente parcialmente apresentada a seguir, onde $\delta(Z_i, a)$ representa a função de transição de estado.

$$\begin{aligned} Z_0 &= \{\text{draw} \rightarrow \bullet \text{seq}, \text{seq} \rightarrow \bullet \text{seq item}\} \\ Z_1 &= \delta(Z_0, \text{seq}) = \{\text{draw} \rightarrow \text{seq} \bullet, \text{seq} \rightarrow \text{seq} \bullet \text{item}, \text{item} \rightarrow \bullet \text{COLOR NUM}, \text{item} \rightarrow \bullet \text{THICKNESS NUM}, \\ &\quad \text{item} \rightarrow \bullet \text{CIRCLE point NUM}, \text{item} \rightarrow \bullet \text{BOX point '(', seq ')'}\} \\ Z_2 &= \delta(Z_1, \text{item}) = \{\text{seq} \rightarrow \text{seq item}\} \rightarrow \text{redução} \\ Z_3 &= \delta(Z_1, \text{COLOR}) = \{\text{item} \rightarrow \text{COLOR} \bullet \text{NUM}\} \\ Z_4 &= \delta(Z_1, \text{THICKNESS}) = \{\text{item} \rightarrow \text{THICKNESS} \bullet \text{NUM}\} \\ Z_5 &= \delta(Z_1, \text{CIRCLE}) = \{\dots\} \\ Z_6 &= \delta(Z_1, \text{BOX}) = \{\dots\} \\ Z_7 &= \delta(Z_3, \text{NUM}) = \{\text{item} \rightarrow \text{COLOR NUM} \bullet\} \\ Z_8 &= \delta(Z_4, \text{NUM}) = \{\text{item} \rightarrow \text{THICKNESS NUM} \bullet\} \end{aligned}$$



[2,0] (a) Preencha as linhas da tabela de reconhecimento (parsing) para um reconhecedor ascendente relativamente aos estados Z_0 a Z_4 .

	NUM	BOX	CIRCLE	THICKNESS	COLOR	{	}	\$	draw	seq	item	point
Z_0		R_2	R_2	R_2	R_2			π_2	π_2		Z_1	
Z_1		S, π_3	S, π_3	S, π_4	S, π_3					ACC		Z_2
Z_2		R_3	R_3	R_3	π_3			π_3	π_3			
Z_3	S, π_7											
Z_4	S, π_8											

[2,0] (b) Determine os conjuntos de itens definidores dos estados Z_5 , Z_6 e de mais três, além dos apresentados.

- Pontos do lado direito \rightarrow Vai correr uma redução associada à produção que transforma Seq em E em Z_0 : Reduce Seq, E

b)

$$Z_5 = \delta(Z_1, \text{circle}) = \{ \text{item} \rightarrow \text{circle} \cdot \text{point Num} \}$$

U

$$\{ \text{point} \rightarrow \cdot \text{Num Num} \}$$

item $\rightarrow \cdot \text{CIRCLE point NUM}$ em Z_1 , estado a esquerda, dir. faz o Shift para a direita

$$Z_6 = \delta(Z_1, \text{Box}) = \{ \text{item} \rightarrow \text{Box} \cdot \text{point} \{ \text{Seq } \} \}$$

U

$$\{ \text{point} \rightarrow \cdot \text{Num Num} \}$$

Temos atenção às
Transições de estado (os símbolos,
que nos andamos com o "·" e
fazemos as derivações)

$$Z_9 = \delta(Z_5, \text{Num}) = \{ \text{point} \rightarrow \text{Num} \cdot \text{Num} \}$$

$$Z_{10} = \delta(Z_9, \text{Num}) = \{ \text{point} \rightarrow \text{Num Num} \cdot \}$$

$$Z_{11} = \delta(Z_6, \text{point}) = \{ \text{item} \rightarrow \text{Box point} \cdot \{ \text{Seq } \} \}$$

4. Considere novamente a gramática G_3 dada no exercício anterior. Uma palavra na linguagem dada por G_3 descreve um desenho definido por uma sequência das seguintes operações gráficas (item):

- COLOR NUM, que permite mudar a cor da caneta de desenho para a dada por NUM.
- THICKNESS NUM, que permite mudar a espessura da caneta de desenho para a dada por NUM.
- CIRCLE point NUM, que desenha um círculo centrado no ponto dado por point e com raio dado por NUM, usando a caneta de desenho ativa.
- BOX point { seq }, que cria um sub-desenho com um offset dado por point em relação ao desenho dentro do qual fica. O ponto (0,0) do sub-desenho é o ponto point do desenho onde está incluído.

Apenas o símbolo terminal NUM tem um atributo associado, designado v e que representa um número. O símbolo não terminal point representa as coordenadas X e Y de um ponto. A configuração inicial do sistema é caracterizada por cor 0, espessura 1 e offset (0,0). Finalmente, considere que dispõe da função drawCircle(x, y, r, c, t) que desenha uma circunferência centrada no ponto (x,y), com raio r, usando uma caneta de desenho com cor c e espessura t.

[1,5] (a) Trace a árvore de derivação da palavra

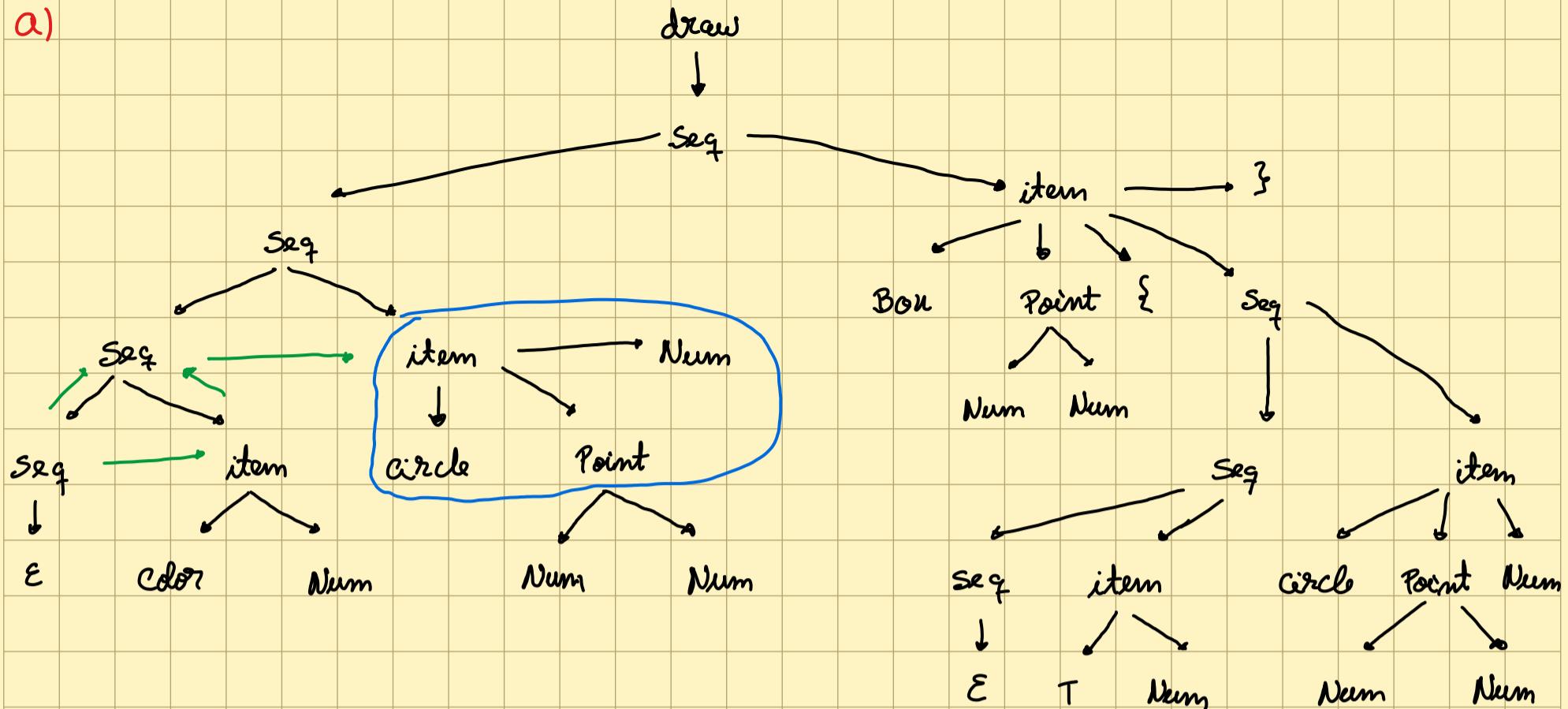
COLOR NUM CIRCLE NUM NUM BOX NUM NUM '{' THICKNESS NUM CIRCLE NUM NUM NUM '}'

Se quiser, ao traçar a árvore, pode abreviar a designação dos símbolos, usando n, ci, co, t, b, s, i e p em vez de NUM, CIRCLE, COLOR, THICKNESS, BOX, seq, item e point, respectivamente.

a) Terminais \rightarrow Ações: shift, reduce, accept, error
(Tabela actions)

Não-Terminais \rightarrow resultado de uma operação de Reduce + tem o push do valor (Tabela go-to)

a)



b)

- [3,0] (b) Complete a gramática de atributos abaixoo tal que ela adequadamente invoque a função `drawCircle` para cada circunferência incluída numa descrição em L_3 .

Production	Semantic rule
$\text{draw} \rightarrow \text{seq}$	$\text{Seq}.C = 0; \text{Seq}.t = 1; \text{Seq}.u = 0; \text{Seq}.y = 0$
$\text{seq} \rightarrow \epsilon$	_____ ?
$\text{seq} \rightarrow \text{seq item}$	$\text{Seq}_2.C = \cdot \text{Seq}_2.t = \text{Seq}_1.t$ item. $\text{item}.t = \text{Seq}_2.t$
$\text{item} \rightarrow \text{COLOR NUM}$	$\text{item}.C = \text{num}.V$
$\text{item} \rightarrow \text{THICKNESS NUM}$	$\text{item}.T = \text{num}.V$
$\text{item} \rightarrow \text{CIRCLE point NUM}$	<code>drawCircle(point.x, point.y, num.v, ... item.C, item.t)</code>
$\text{item} \rightarrow \text{BOX point } \{ \text{seq} \}$	$\text{Seq}.x = \text{Point}.x; \text{Seq}.y = \text{Point}.y$
$\text{point} \rightarrow \text{NUM NUM}$	$\text{point}.x = \text{num1}.V; \text{point}.y = \text{num2}.V$

- Assumir que o Visitor é void;
- É diferente do Antlr e permite atributos nos nós terminais;
- Atenção as variáveis formadas no anúncio
- Atenção aos valores default