

# Compiladores

## (IF688)

**Leopoldo Teixeira**  
**[imt@cin.ufpe.br](mailto:imt@cin.ufpe.br) | [@leopoldomt](#)**

# Exemplo

$$_1 S \rightarrow S; S$$

$$_2 S \rightarrow \mathbf{id} := E$$

$$_3 S \rightarrow \mathbf{print}(L)$$

$$_4 E \rightarrow \mathbf{id}$$

$$_5 E \rightarrow \mathbf{num}$$

$$_6 E \rightarrow E + E$$

$$_7 E \rightarrow ( S , E )$$

$$_8 L \rightarrow E$$

$$_9 L \rightarrow L , E$$

$_1 S \rightarrow S; S$	$_4 E \rightarrow \mathbf{id}$	$_7 E \rightarrow ( S, E )$
$_2 S \rightarrow \mathbf{id} := E$	$_5 E \rightarrow \mathbf{num}$	$_8 L \rightarrow E$
$_3 S \rightarrow \mathbf{print}(L)$	$_6 E \rightarrow E + E$	$_9 L \rightarrow L, E$

**a := 7 \$**

**a := 7; b := c + ( d := 5+6 , d ) \$**

	id	num	print	;	,	+	:=	(	)	\$	<i>S</i>	<i>E</i>	<i>L</i>
1	s4		s7								g2		
2				s3						a			
3	s4		s7								g5		
4							s6						
5				r1	r1					r1			
6	s20	s10						s8				g11	
7								s9					
8	s4		s7								g12		
9	s20	s10						s8				g15	g14
10				r5	r5	r5			r5	r5			
11				r2	r2	s16				r2			
12				s3	s18								
13				r3	r3					r3			
14					s19				s13				
15					r8				r8				
16	s20	s10						s8				g17	
17				r6	r6	s16			r6	r6			
18	s20	s10						s8				g21	
19	s20	s10						s8				g23	
20				r4	r4	r4			r4	r4			
21									s22				
22				r7	r7	r7			r7	r7			
23					r9	s16			r9				

# Problemas

- Problemas na gramática ou limitações da técnica escolhida podem levar a conflitos
- shift-reduce: não consegue decidir entre uma ação de shift (ou mais) ou reduce
- reduce-reduce: não tem como decidir entre duas ou mais ações de reduce, geralmente por ambiguidade ou algum bug na gramática

# Análise SLR

- Utiliza o conjunto FOLLOW para resolver conflitos
- Intuição: só reduz se o próximo token (lookahead) estiver no conjunto FOLLOW do não-terminal associado
- Na tabela de parsing, só inclui ação de reduce, caso o terminal esteja no conjunto FOLLOW

# Autômatos SLR

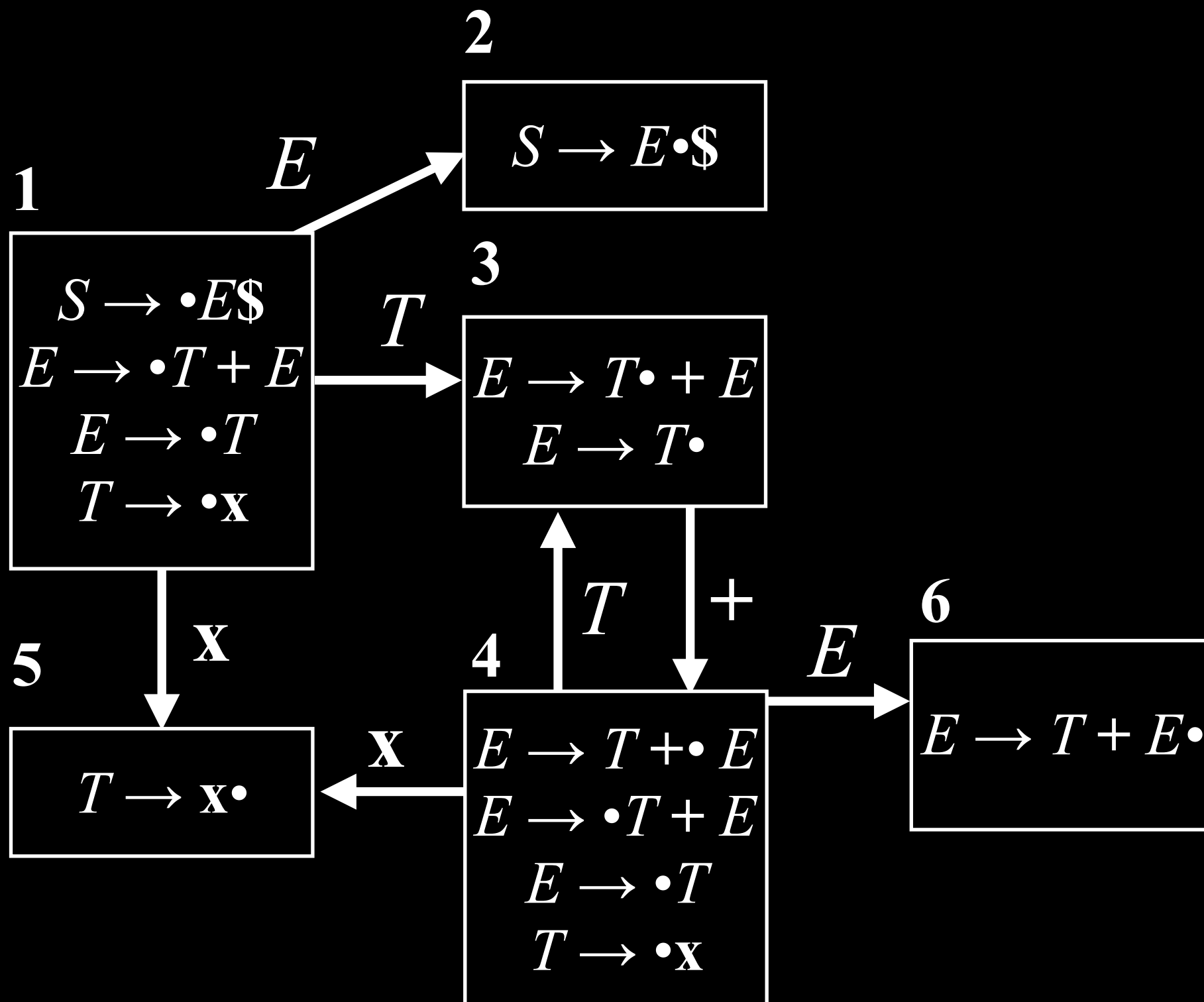
- Estados podem ter mais de um item de redução, caso conjuntos FOLLOW sejam distintos
- Estados podem misturar itens de shift com itens de redução, caso os terminais associados ao shift não estejam no FOLLOW dos itens de redução
- Isso não elimina todos os tipos de conflitos

$$_0 S' \rightarrow ES$$

$$_1 E \rightarrow T + E$$

$$_2 E \rightarrow T$$

$$_3 T \rightarrow \mathbf{x}$$





$$_0 S' \rightarrow ES$$

$$_1 E \rightarrow T + E$$

$$_2 E \rightarrow T$$

$$_3 T \rightarrow \mathbf{x}$$

	$\mathbf{x}$	$+$	$\$$	$E$	$T$
1	s5			g2	g3
2			a		
3		s4	r2		
4	s5			g6	g3
5		r3	r3		
6			r1		

# LR(1)

- Mais poderoso que SLR
- Suficiente para grande parte das linguagens de programação
- A noção de item é mais sofisticada, inclui o símbolo de lookahead

# Construção do autômato LR(1)

- Iniciar com estado contendo item  $S \rightarrow \bullet X [\$]$ , onde  $S$  é o símbolo inicial
- Compute o fecho do estado
  - Se  $S \rightarrow \alpha \bullet X \beta [t]$  pertence ao estado, adicione  $X \rightarrow \bullet \Phi [r]$  ao estado, para toda produção existente  $X \rightarrow \Phi$  e todo terminal  $r \in \text{FIRST}(\beta t)$
- Repita o passo abaixo até que nenhum novo estado seja adicionado
  - Se um estado contém a produção  $S \rightarrow \alpha \bullet x \beta [r]$  para o símbolo  $x$ , adicione uma transição deste estado para o estado contendo o fecho de  $S \rightarrow \alpha x \bullet \beta [r]$

# Estrutura dos autômatos

- Simulam dois processos simultaneamente
- Compreendem o autômato LR(0) para encontrar handles
- Um rastreador de tokens de lookahead, para determinar qual o lookahead atual
- Remover os lookaheads de um autômato LR(1) resulta em um autômato LR(0) muito maior para a mesma gramática

# Construindo autômato

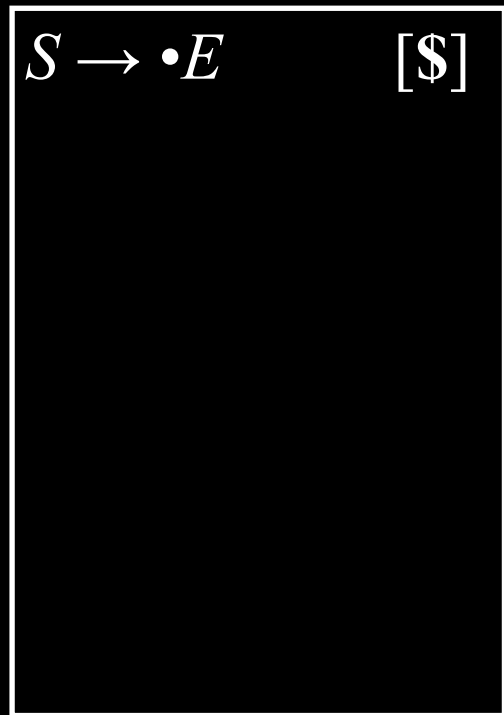
$$_0 S \rightarrow E$$

$$_1 E \rightarrow T$$

$$_3 T \rightarrow \mathbf{int}$$

$$_2 E \rightarrow E + T$$

$$_4 T \rightarrow (E)$$



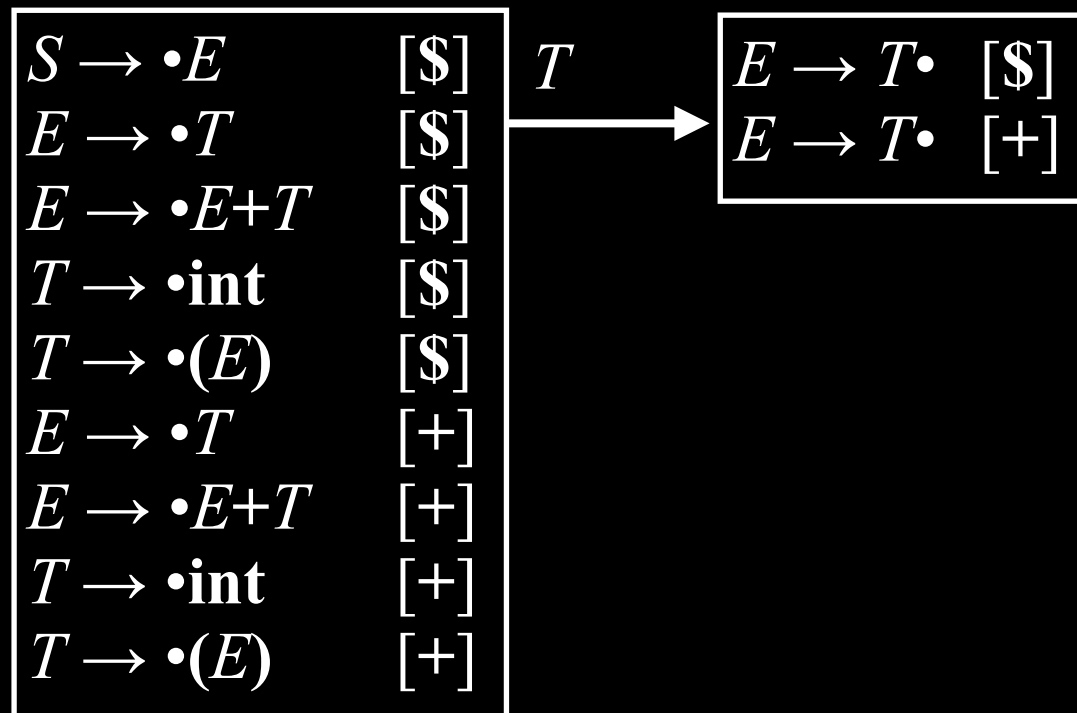
$S \rightarrow \bullet E$	$[\$]$
$E \rightarrow \bullet T$	$[\$]$
$E \rightarrow \bullet E + T$	$[\$]$

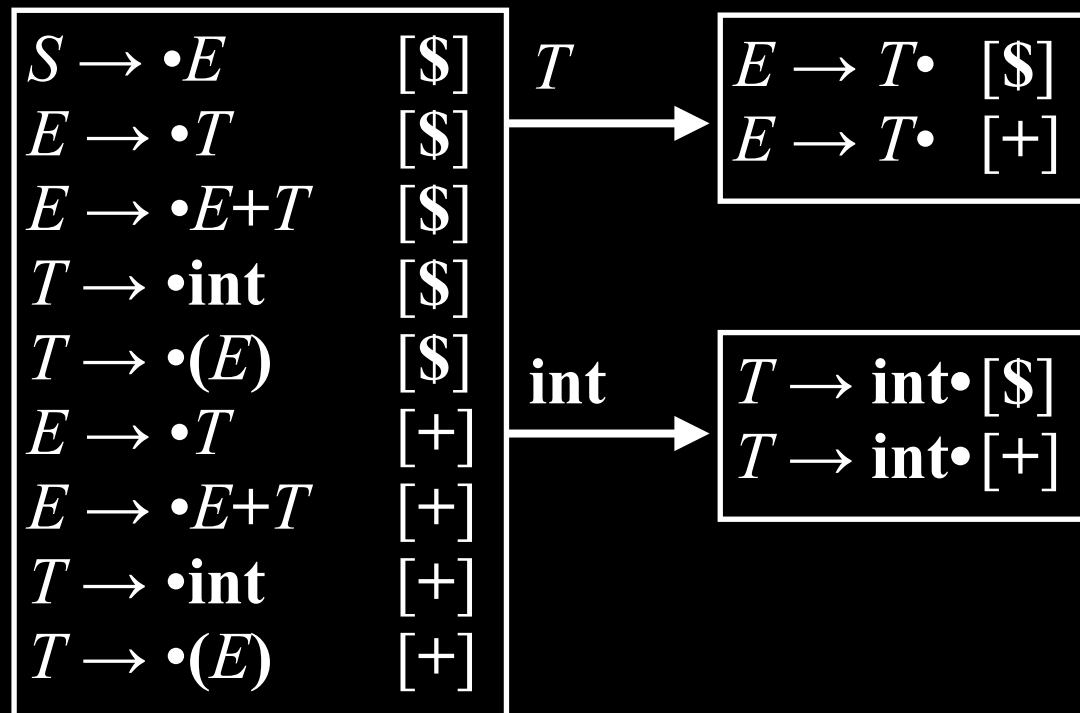
$S \rightarrow \bullet E$	$[\$]$
$E \rightarrow \bullet T$	$[\$]$
$E \rightarrow \bullet E + T$	$[\$]$
$T \rightarrow \bullet \text{int}$	$[\$]$
$T \rightarrow \bullet (E)$	$[\$]$

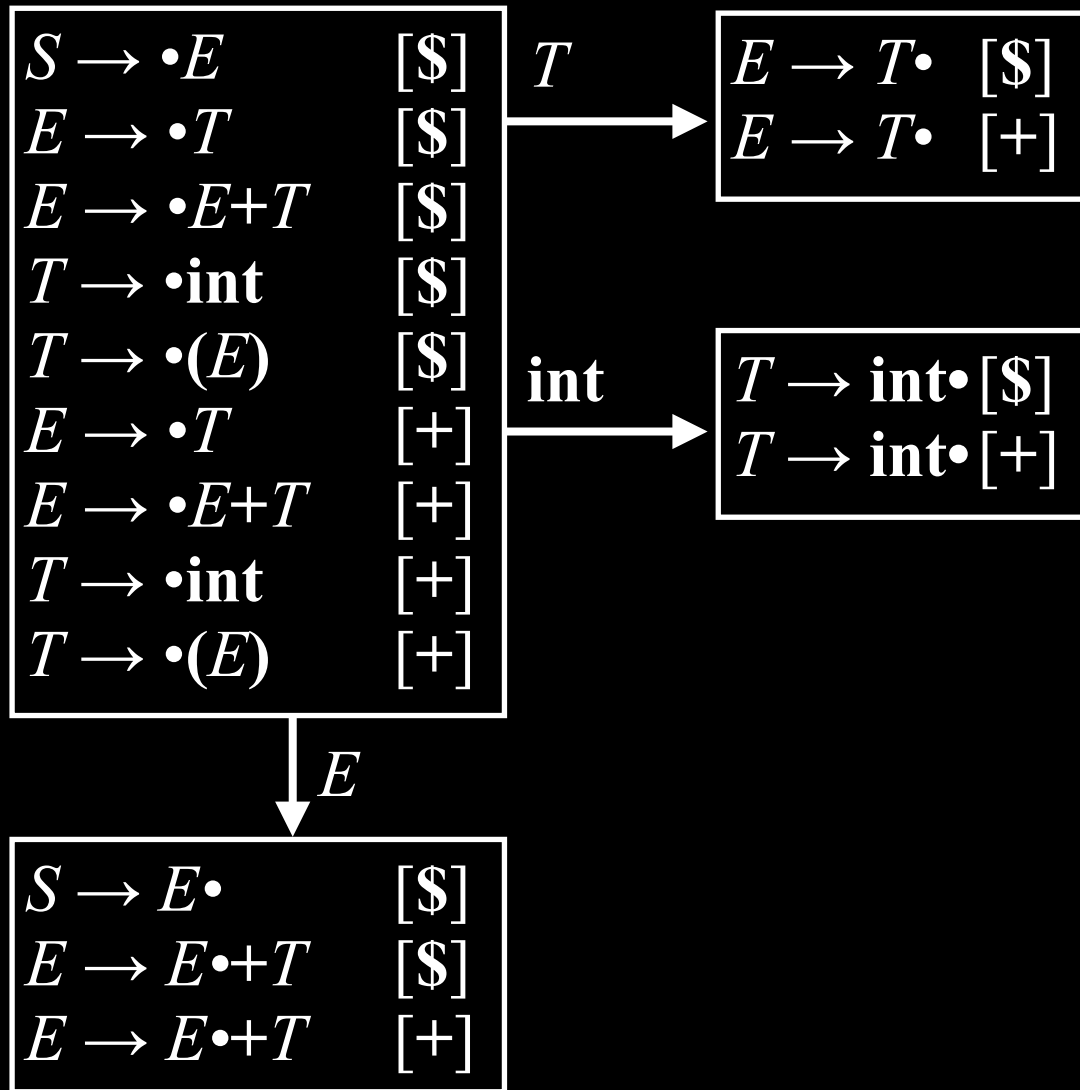


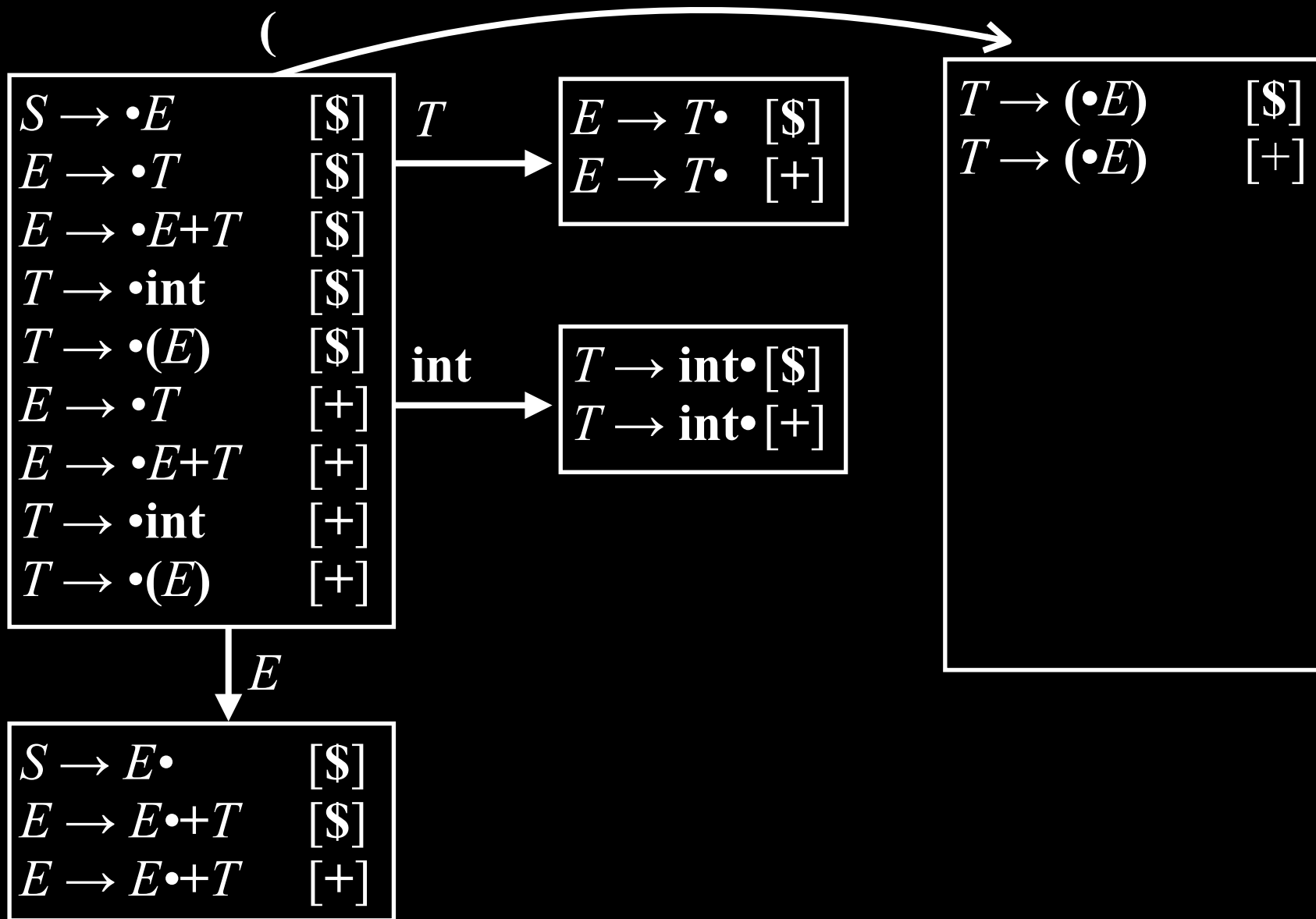
$S \rightarrow \bullet E$	$[\$]$
$E \rightarrow \bullet T$	$[\$]$
$E \rightarrow \bullet E + T$	$[\$]$
$T \rightarrow \bullet \mathbf{int}$	$[\$]$
$T \rightarrow \bullet (E)$	$[\$]$
$E \rightarrow \bullet T$	$[+]$
$E \rightarrow \bullet E + T$	$[+]$

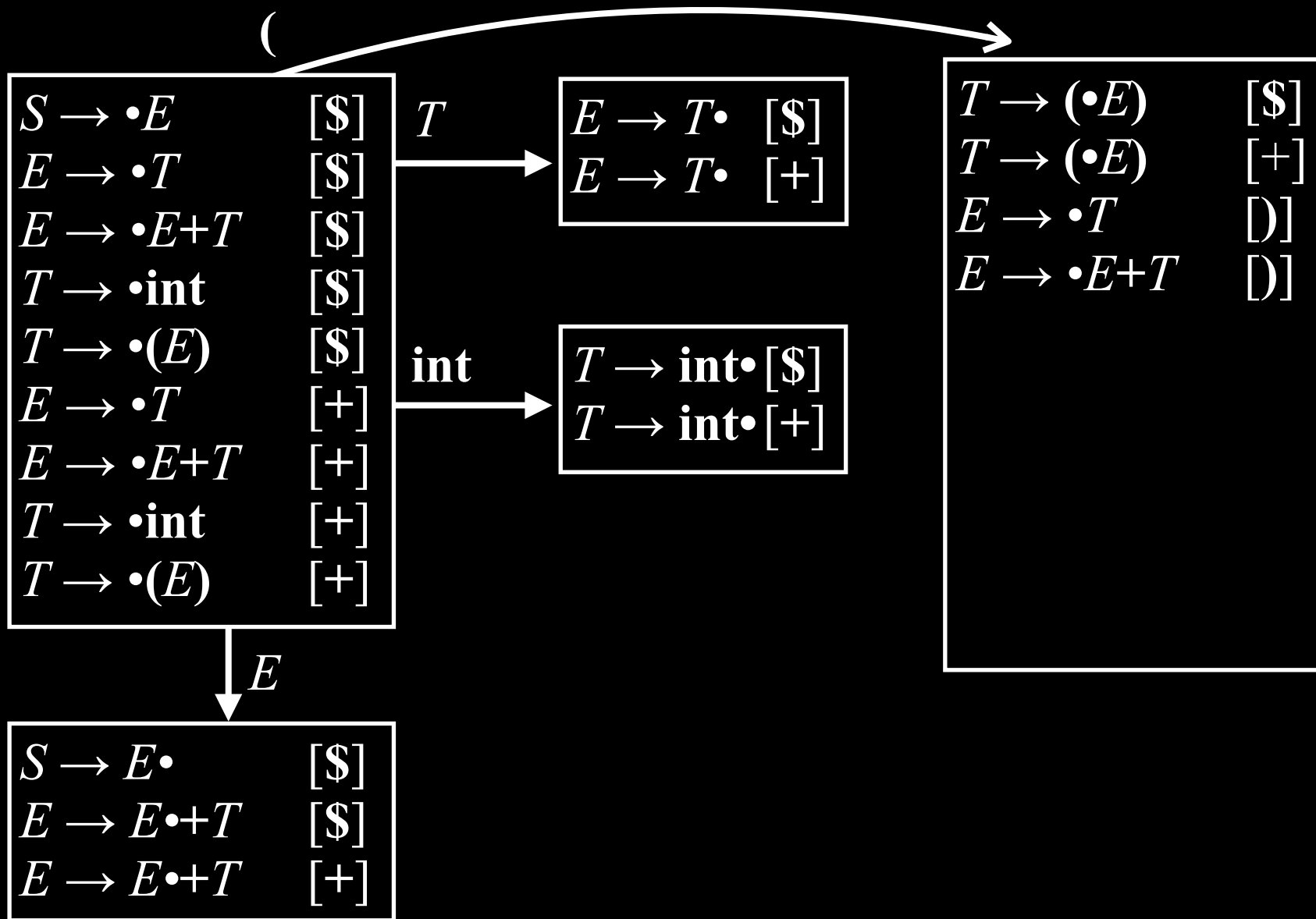
$S \rightarrow \bullet E$	[\$]
$E \rightarrow \bullet T$	[\$]
$E \rightarrow \bullet E + T$	[\$]
$T \rightarrow \bullet \mathbf{int}$	[\$]
$T \rightarrow \bullet (E)$	[\$]
$E \rightarrow \bullet T$	[+]
$E \rightarrow \bullet E + T$	[+]
$T \rightarrow \bullet \mathbf{int}$	[+]
$T \rightarrow \bullet (E)$	[+]

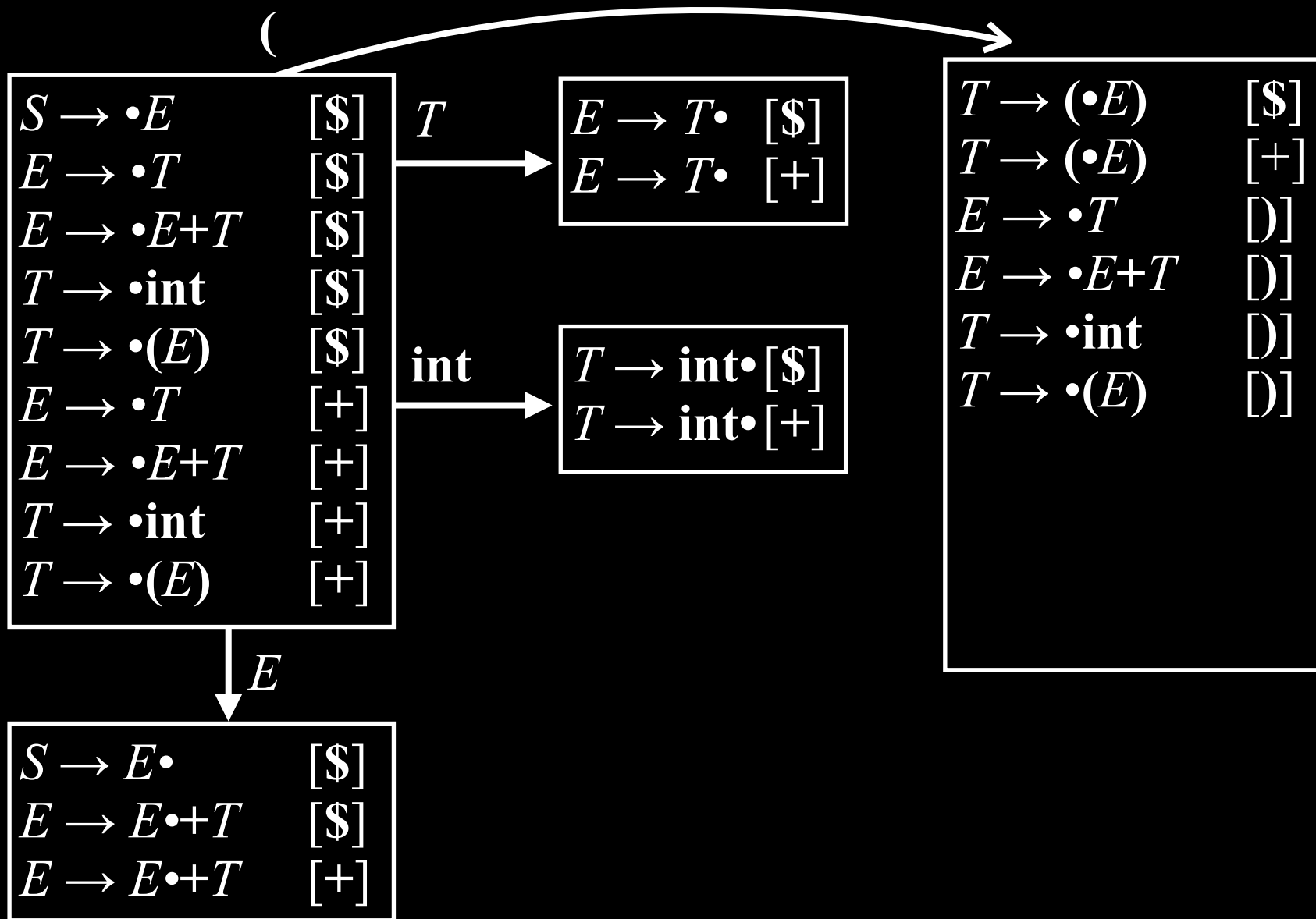




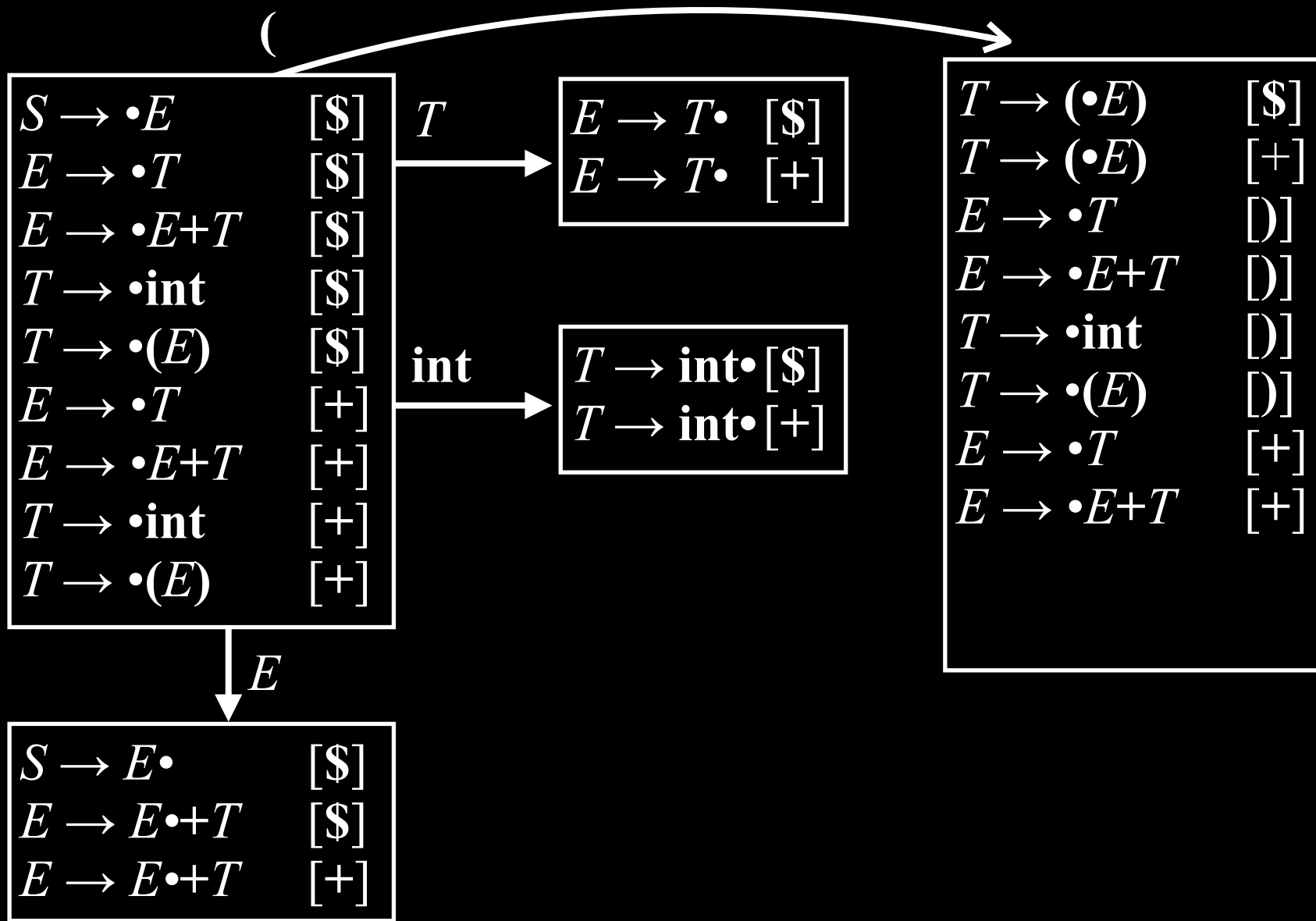


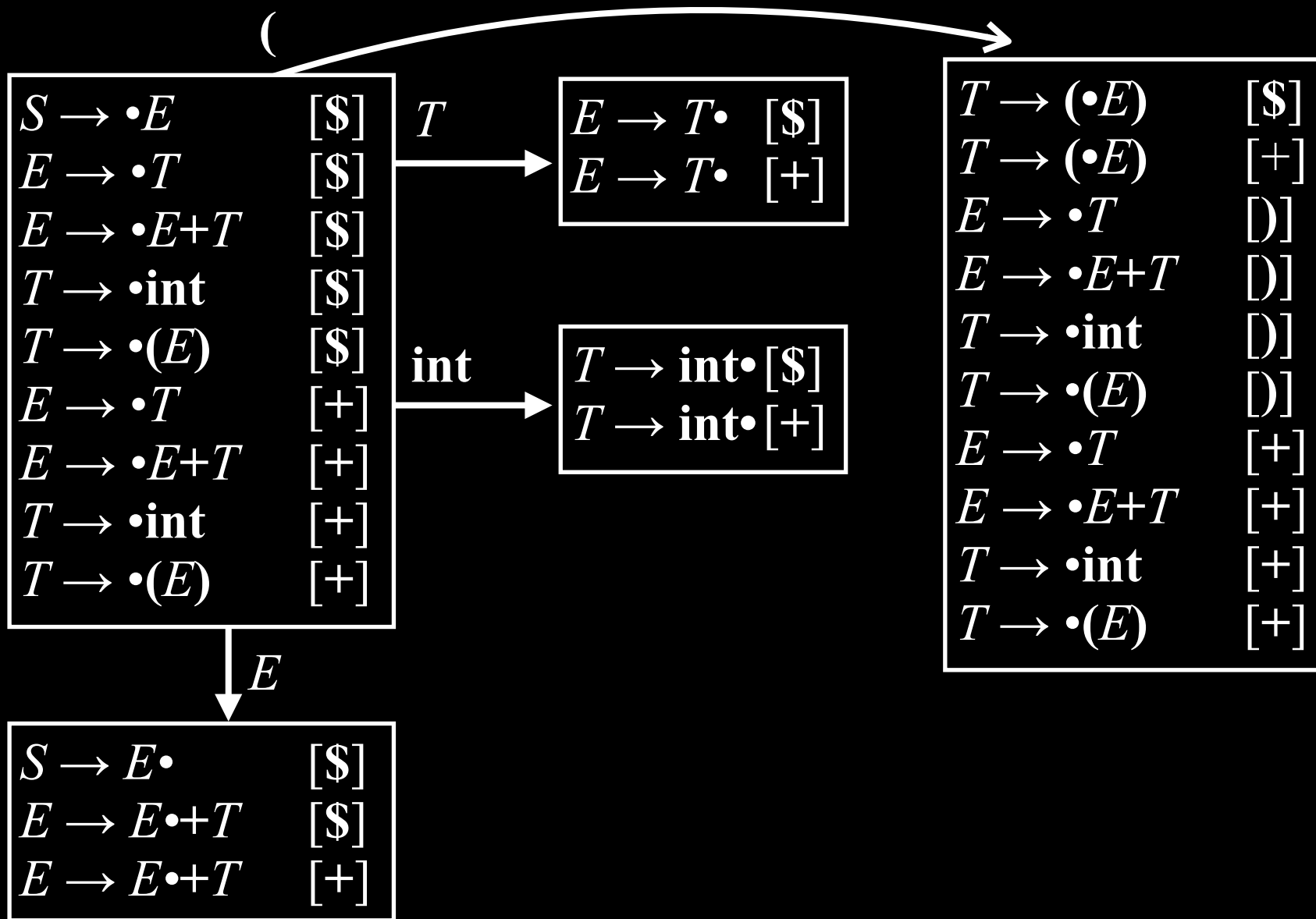


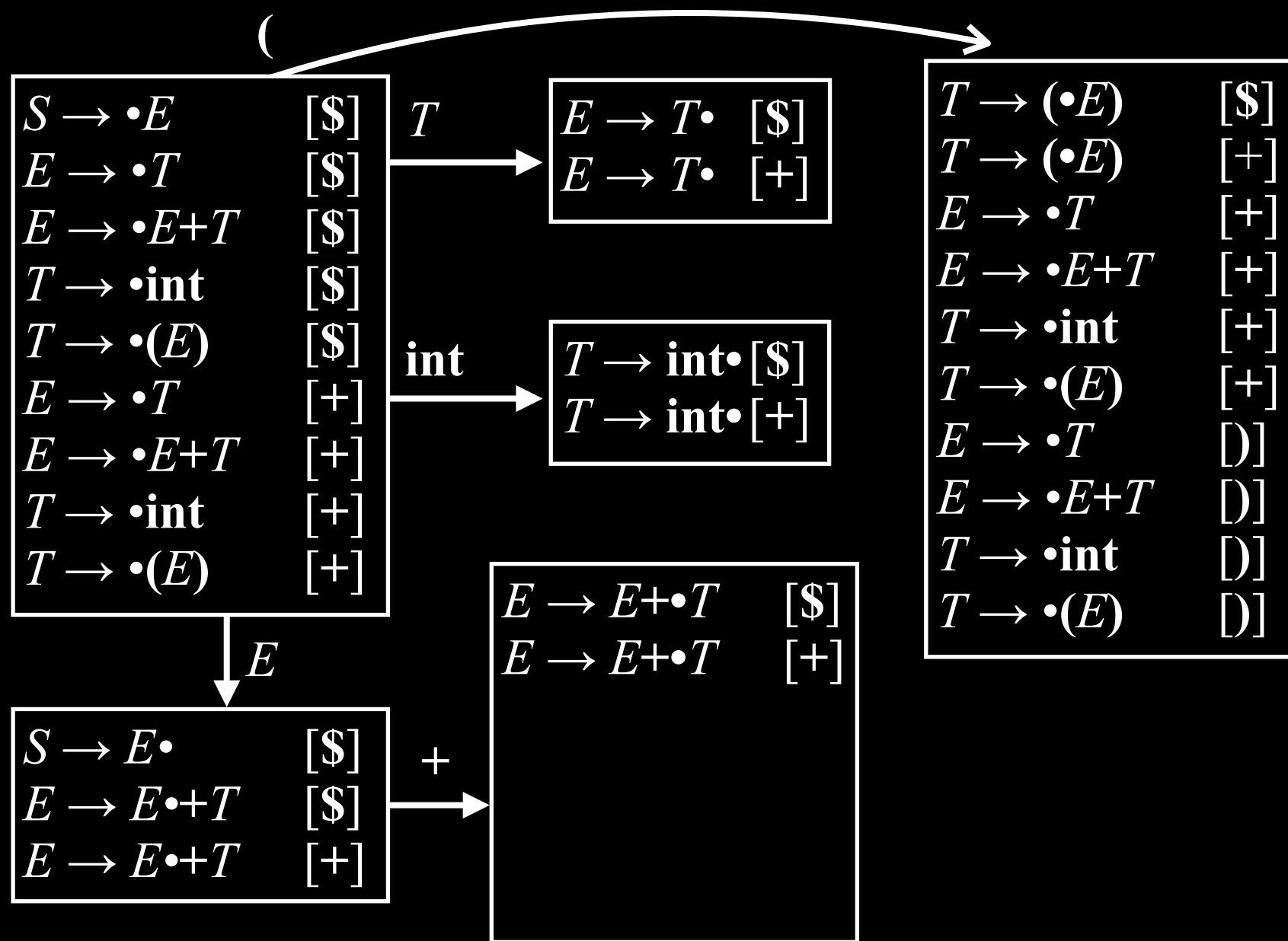


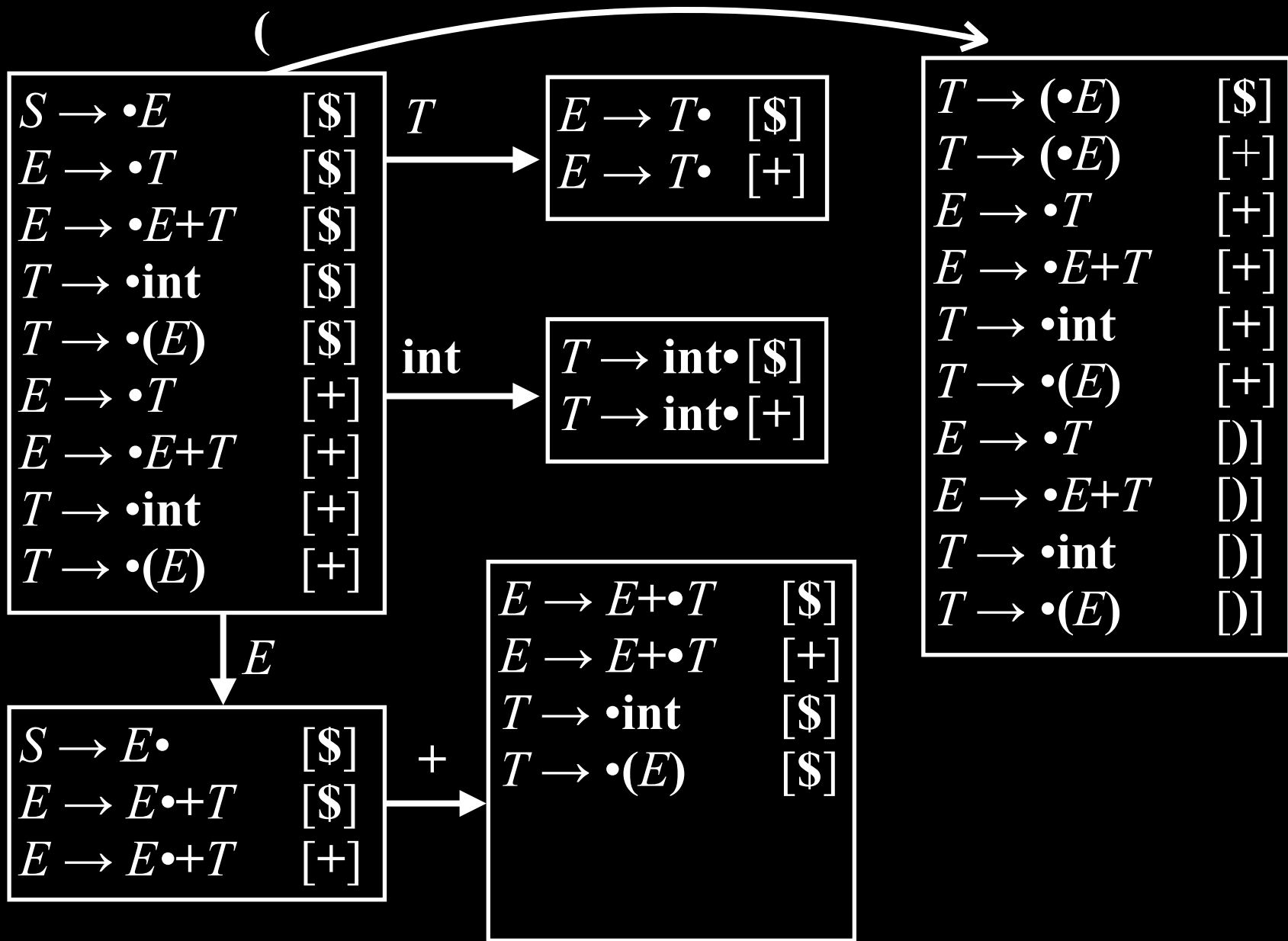


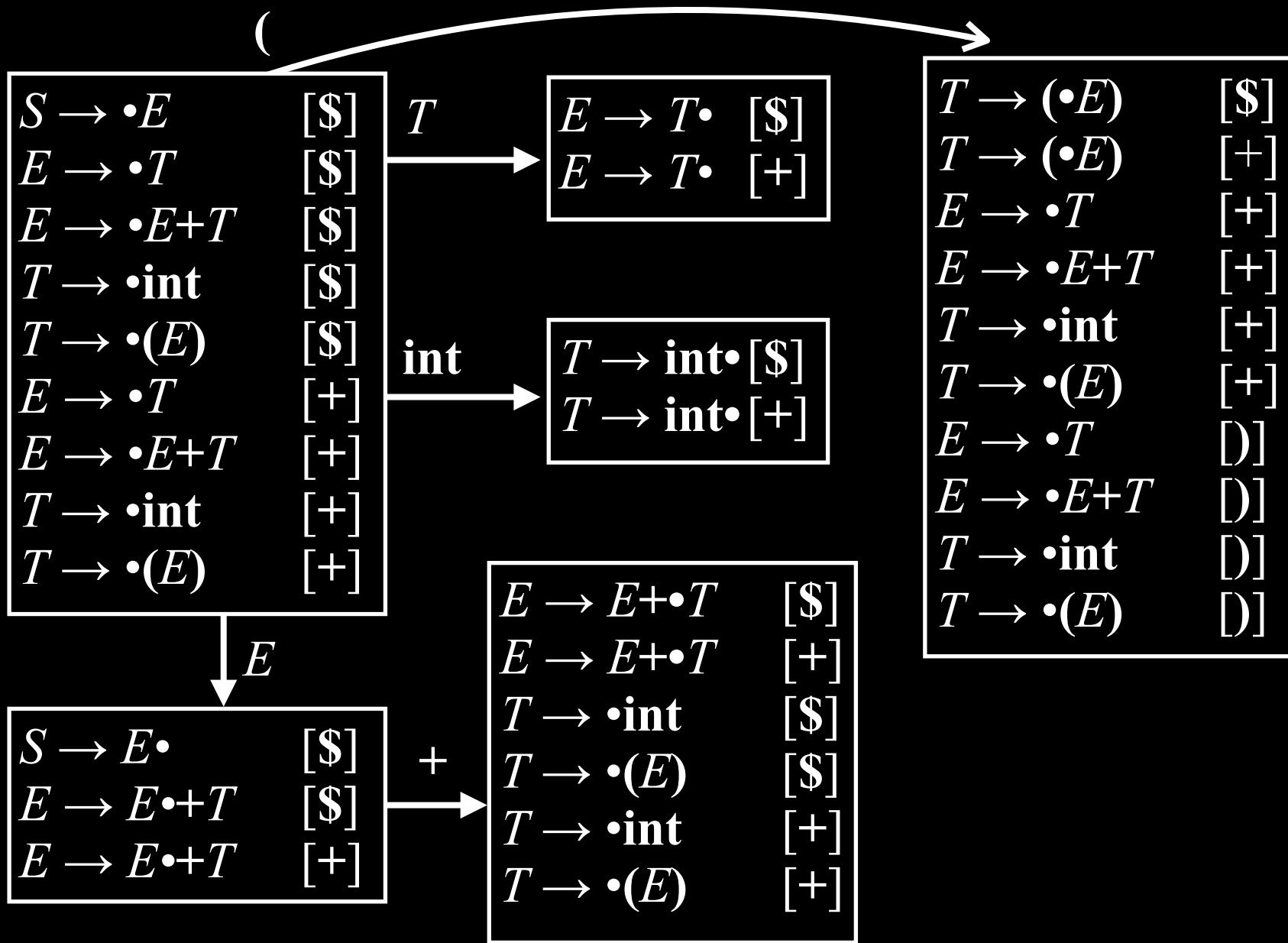


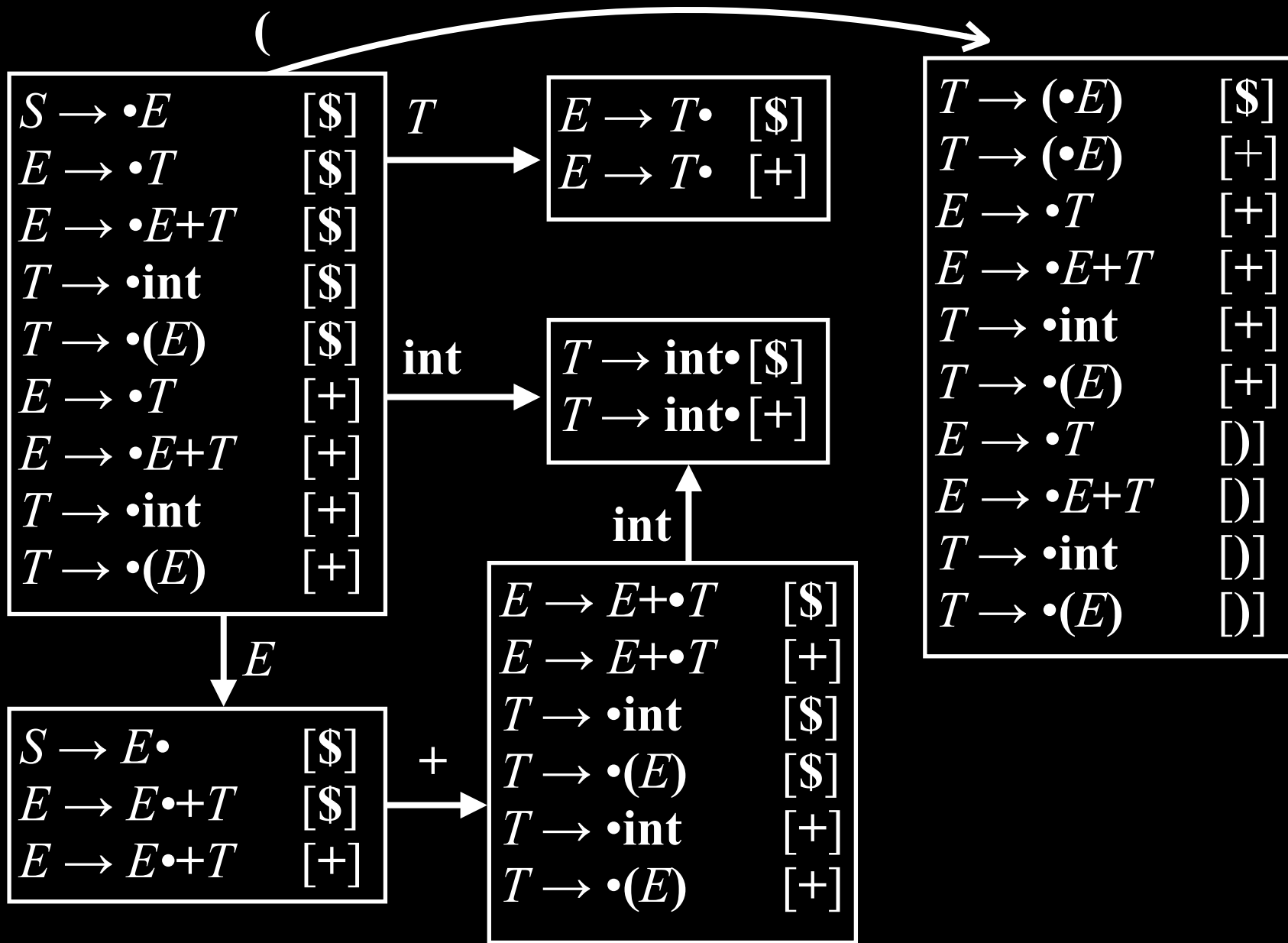


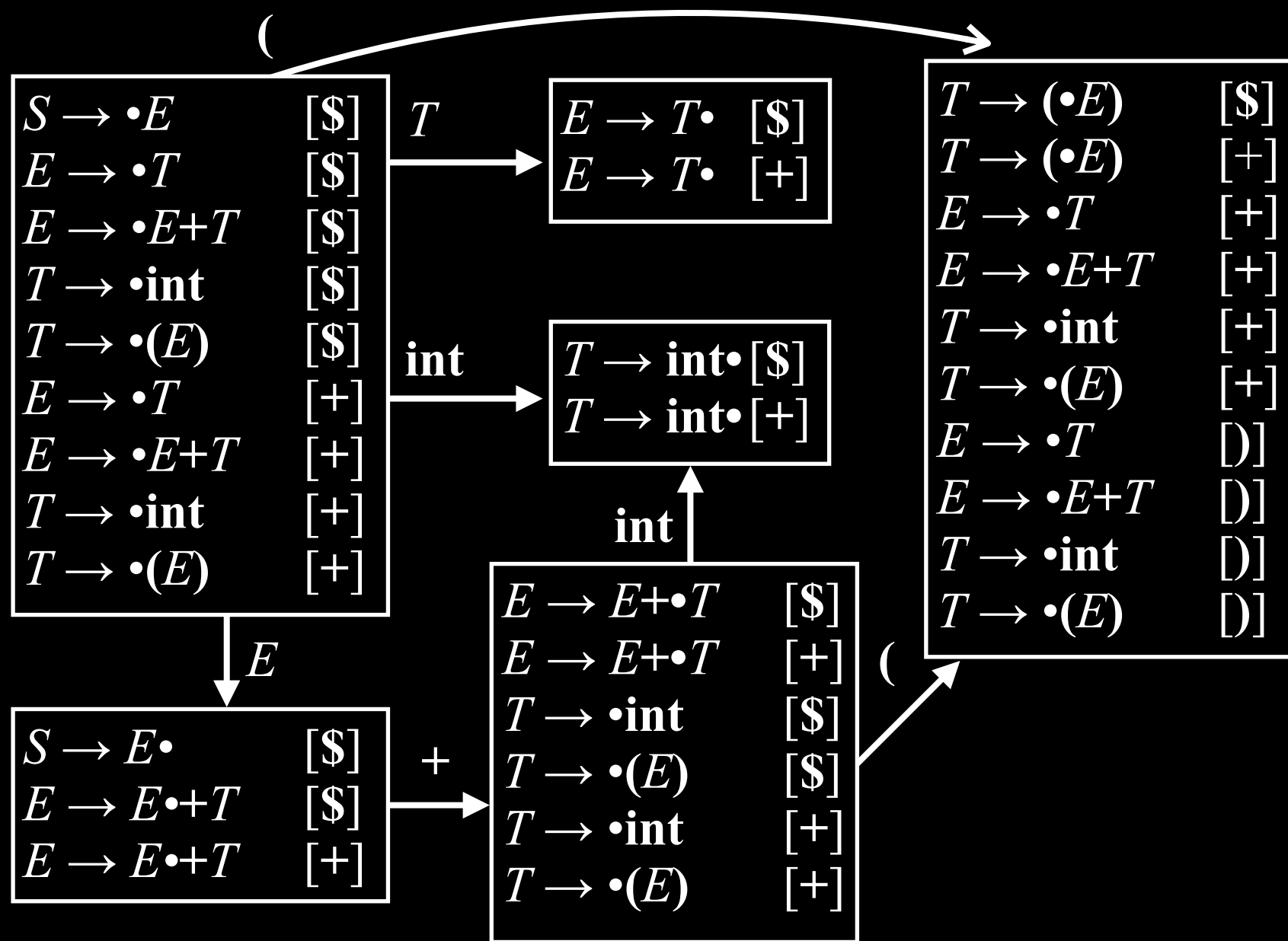


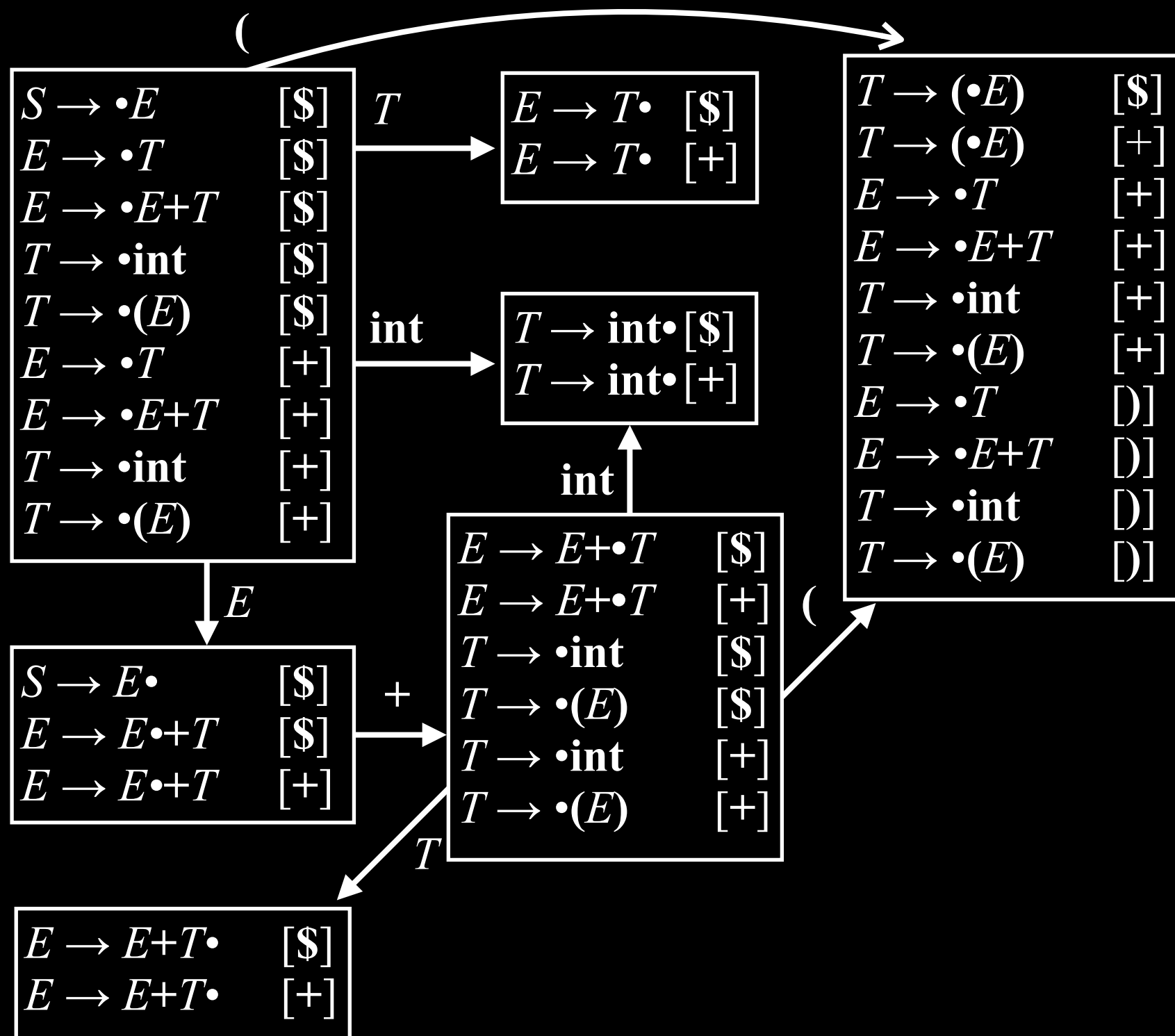




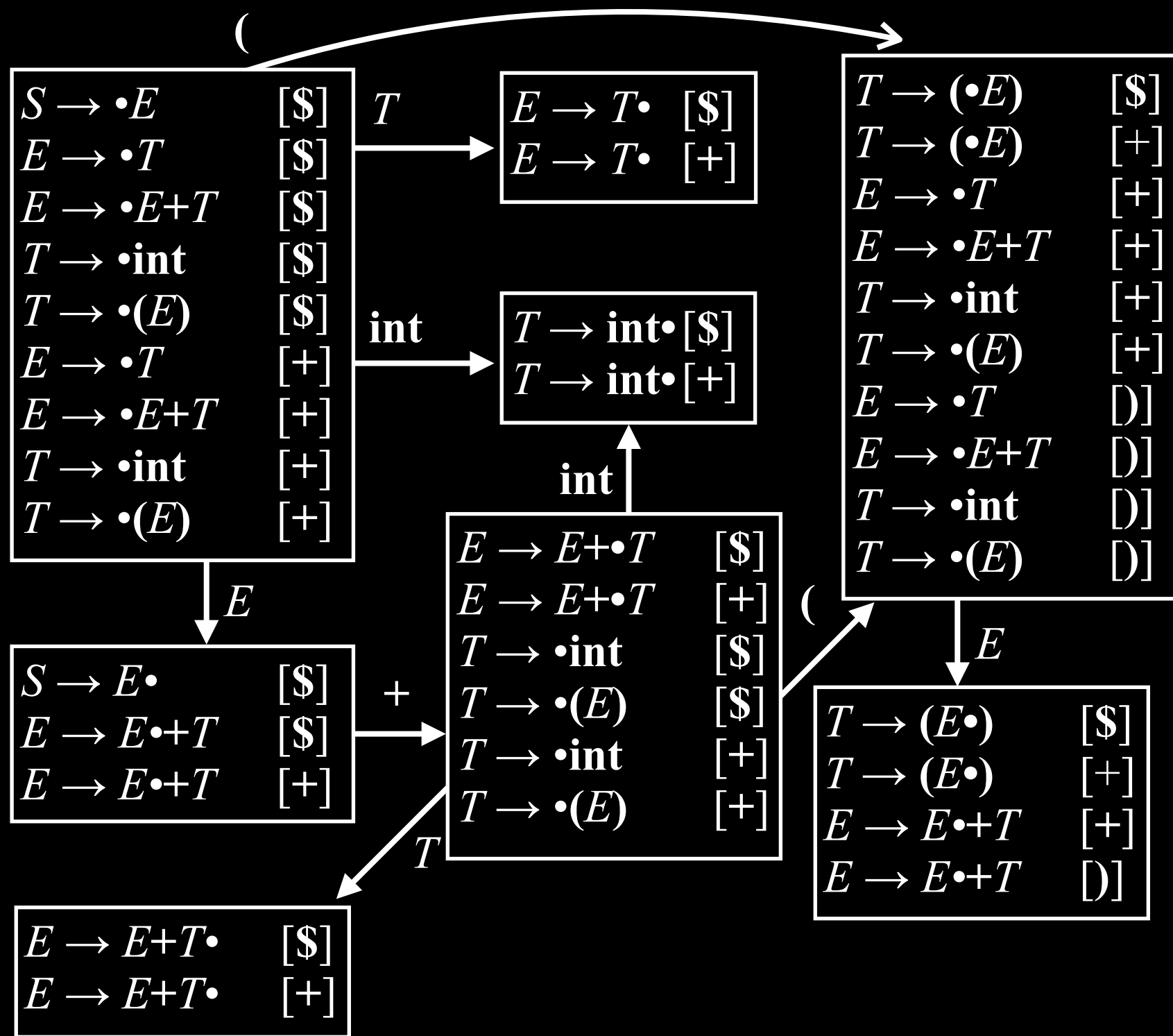


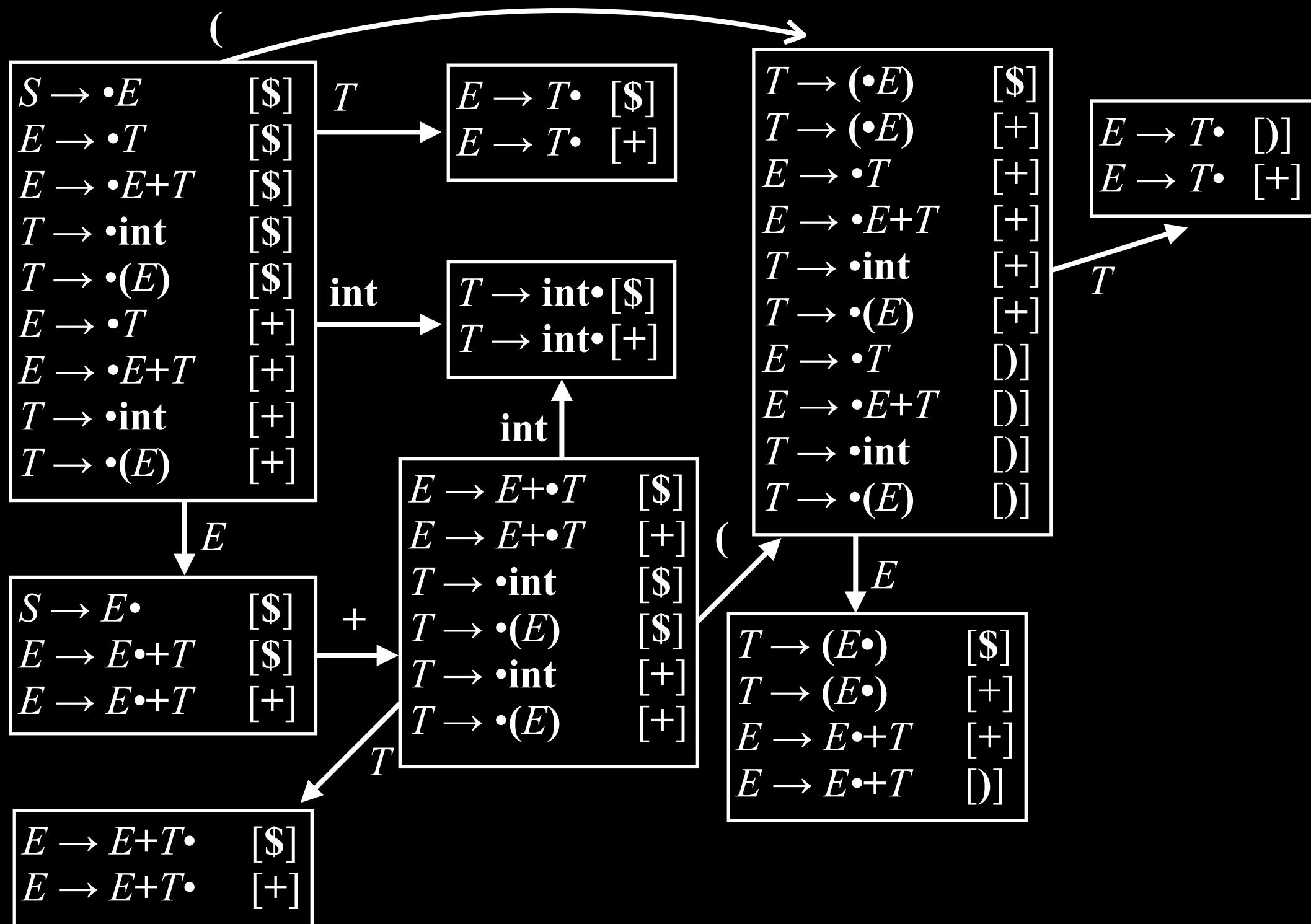


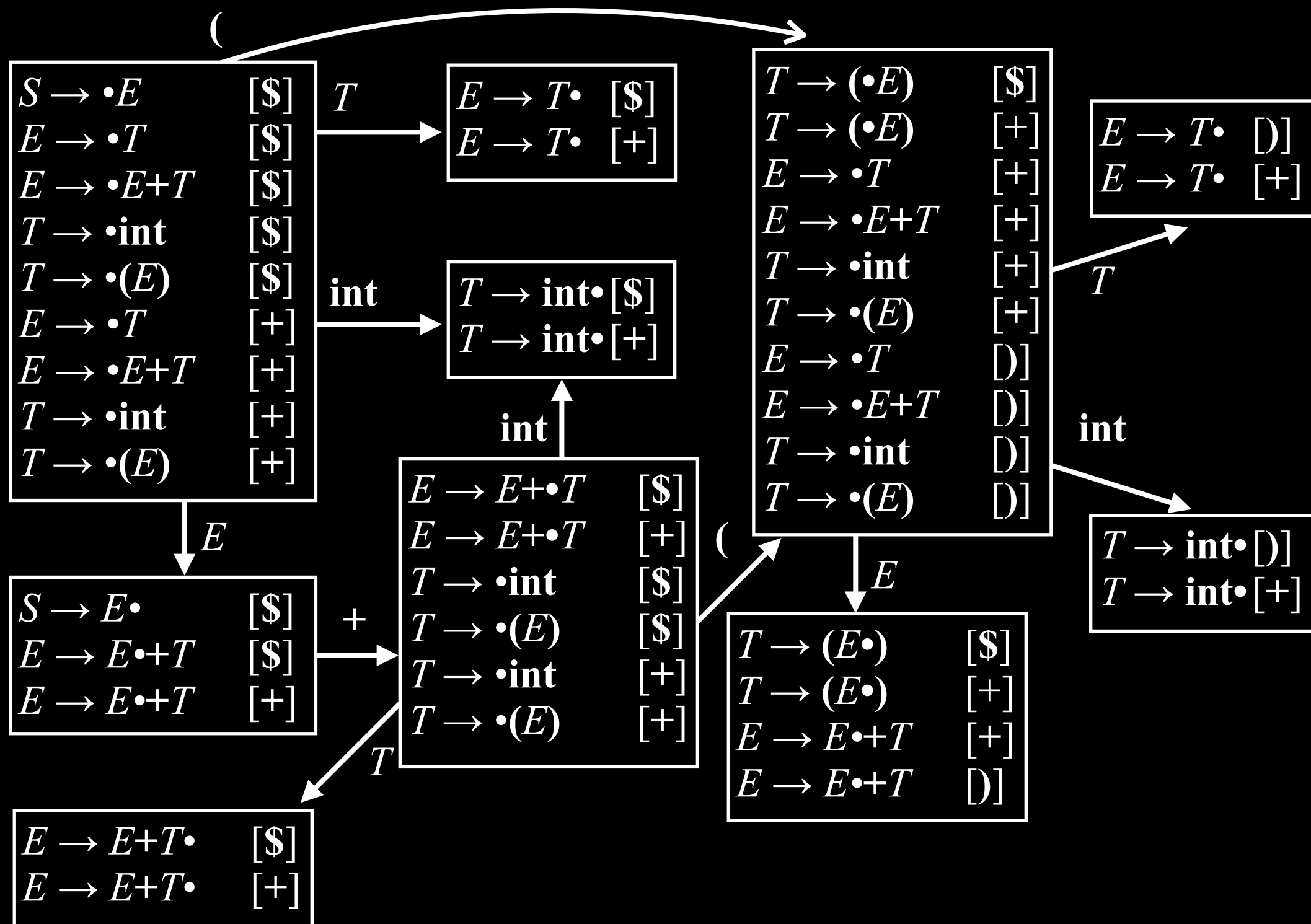


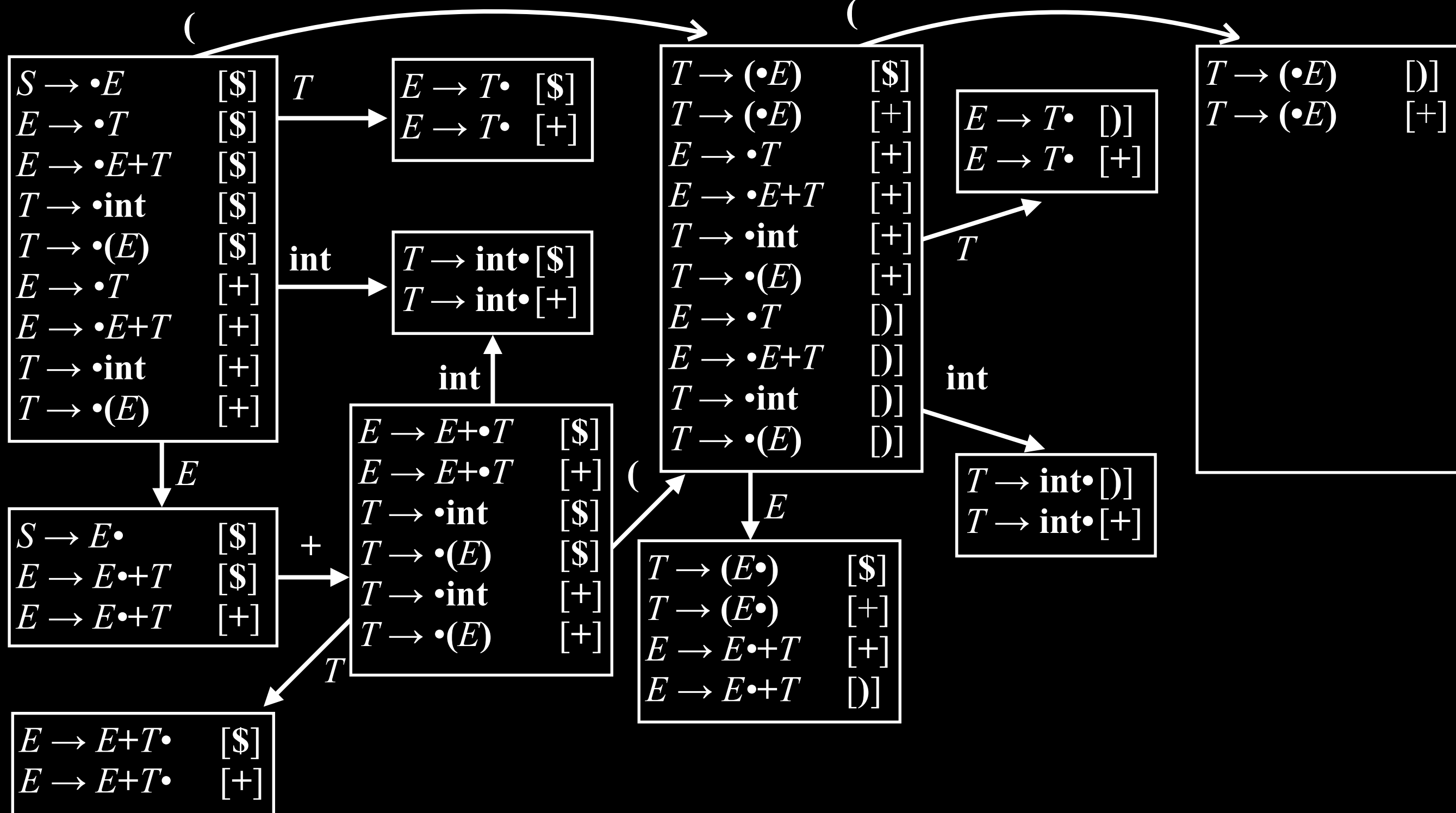


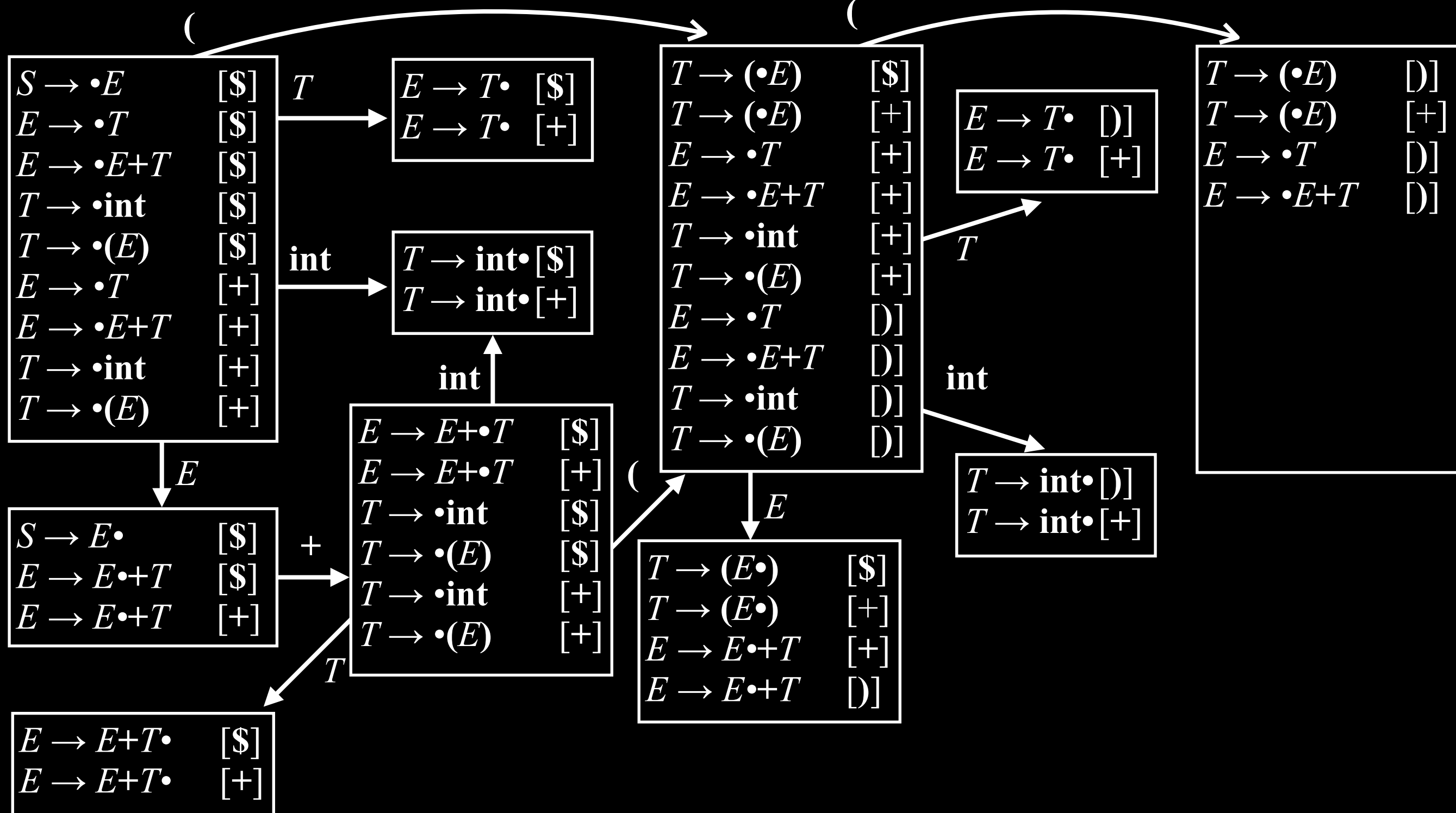


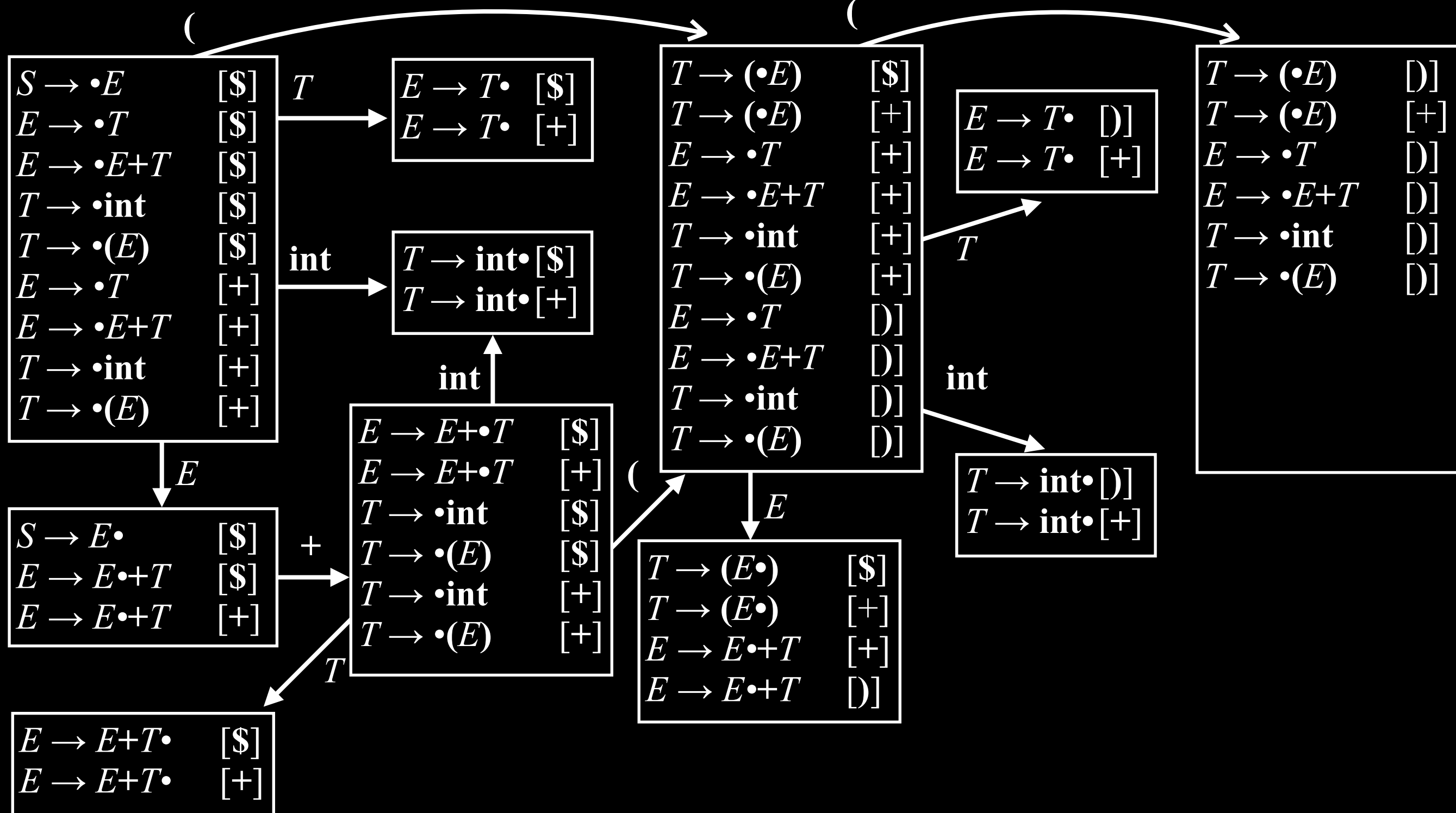


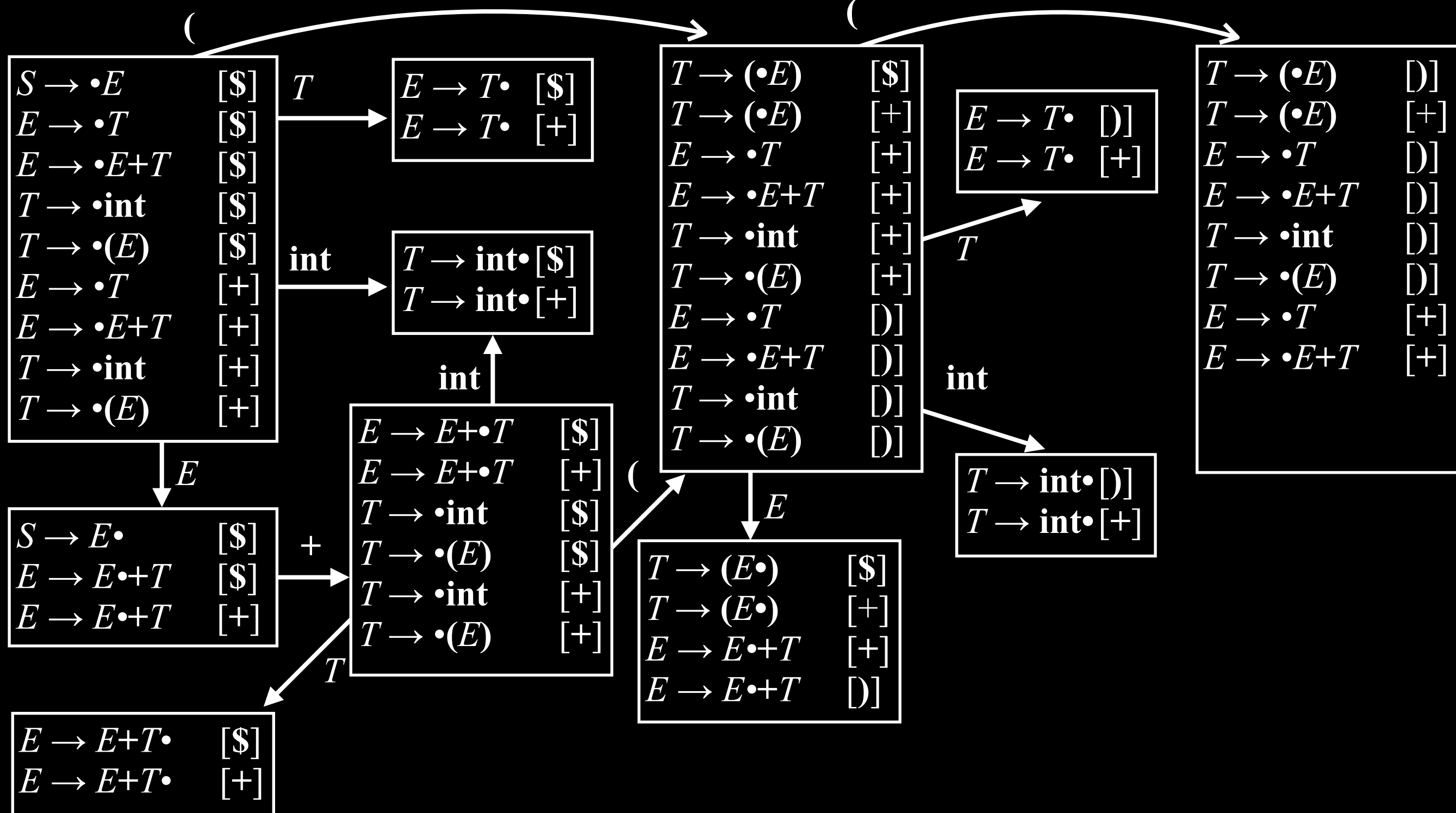


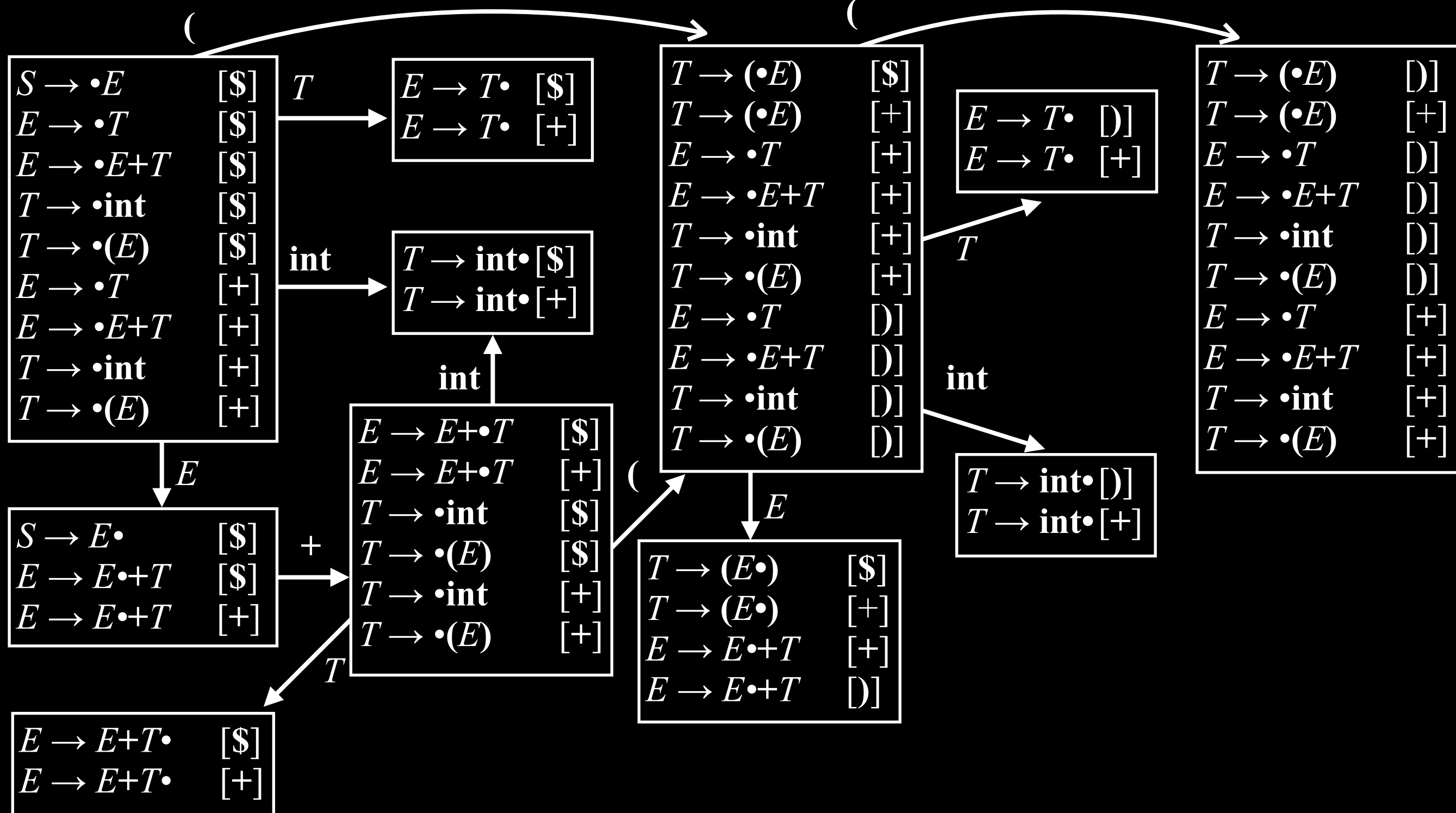




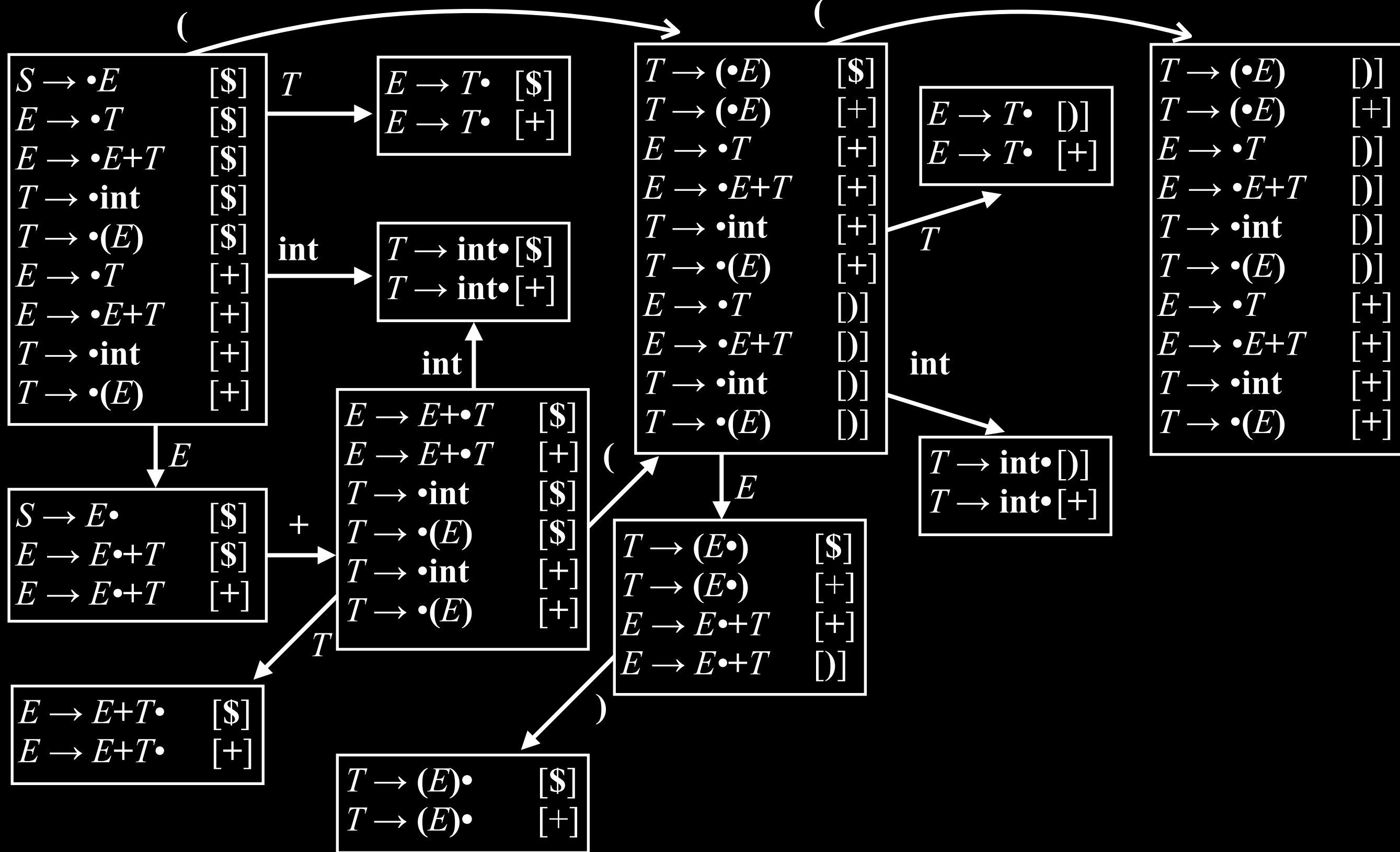


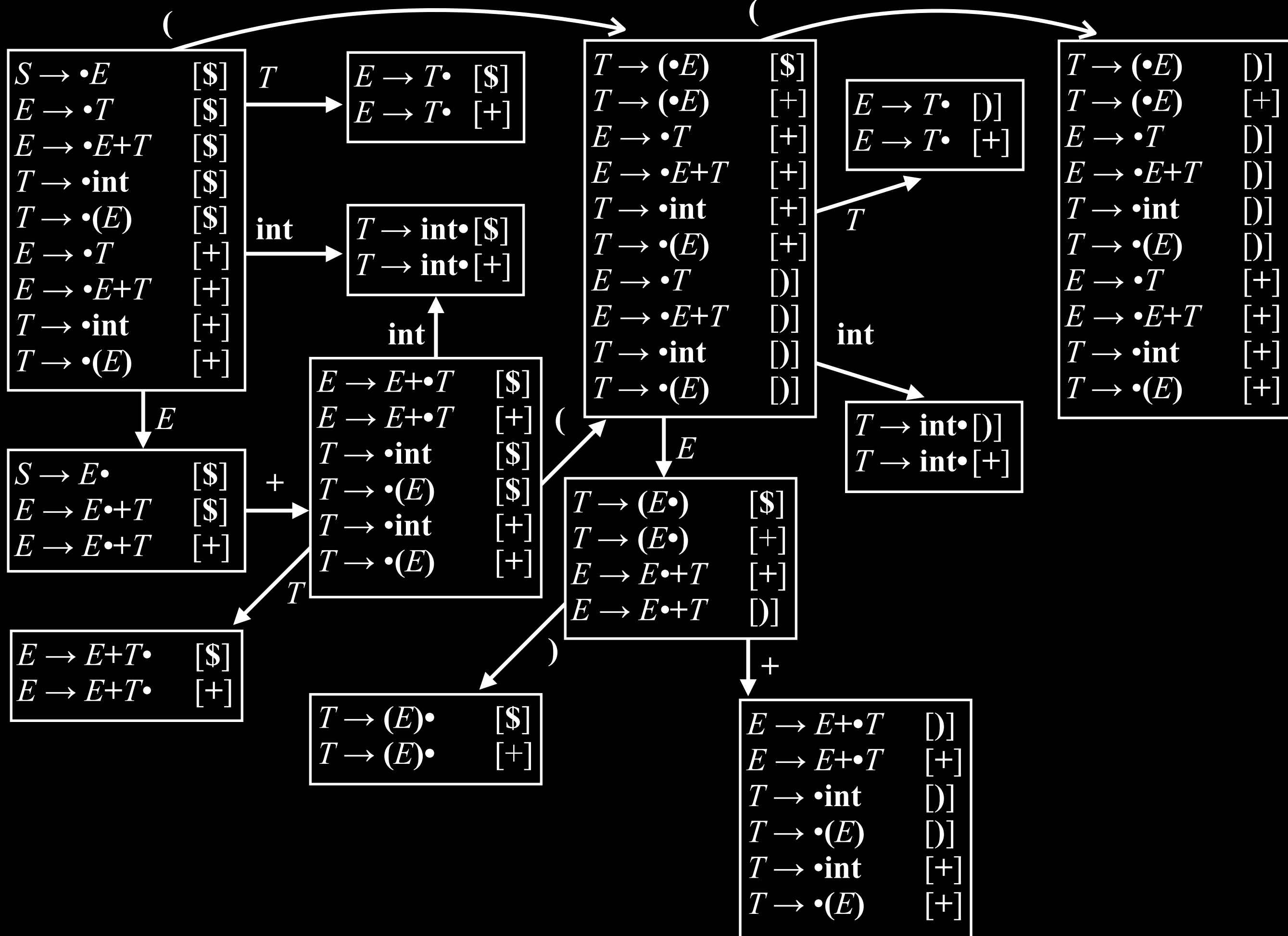


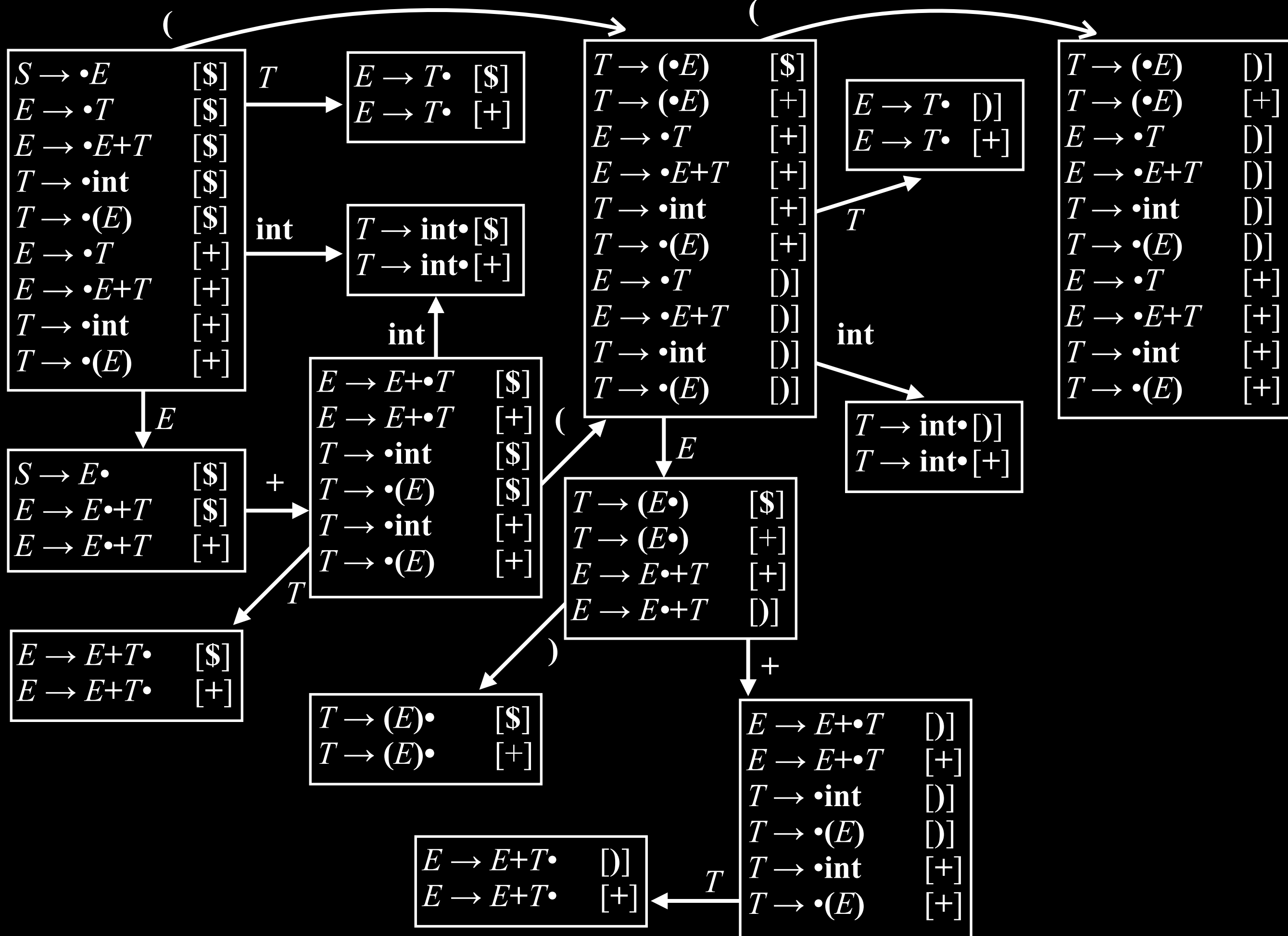


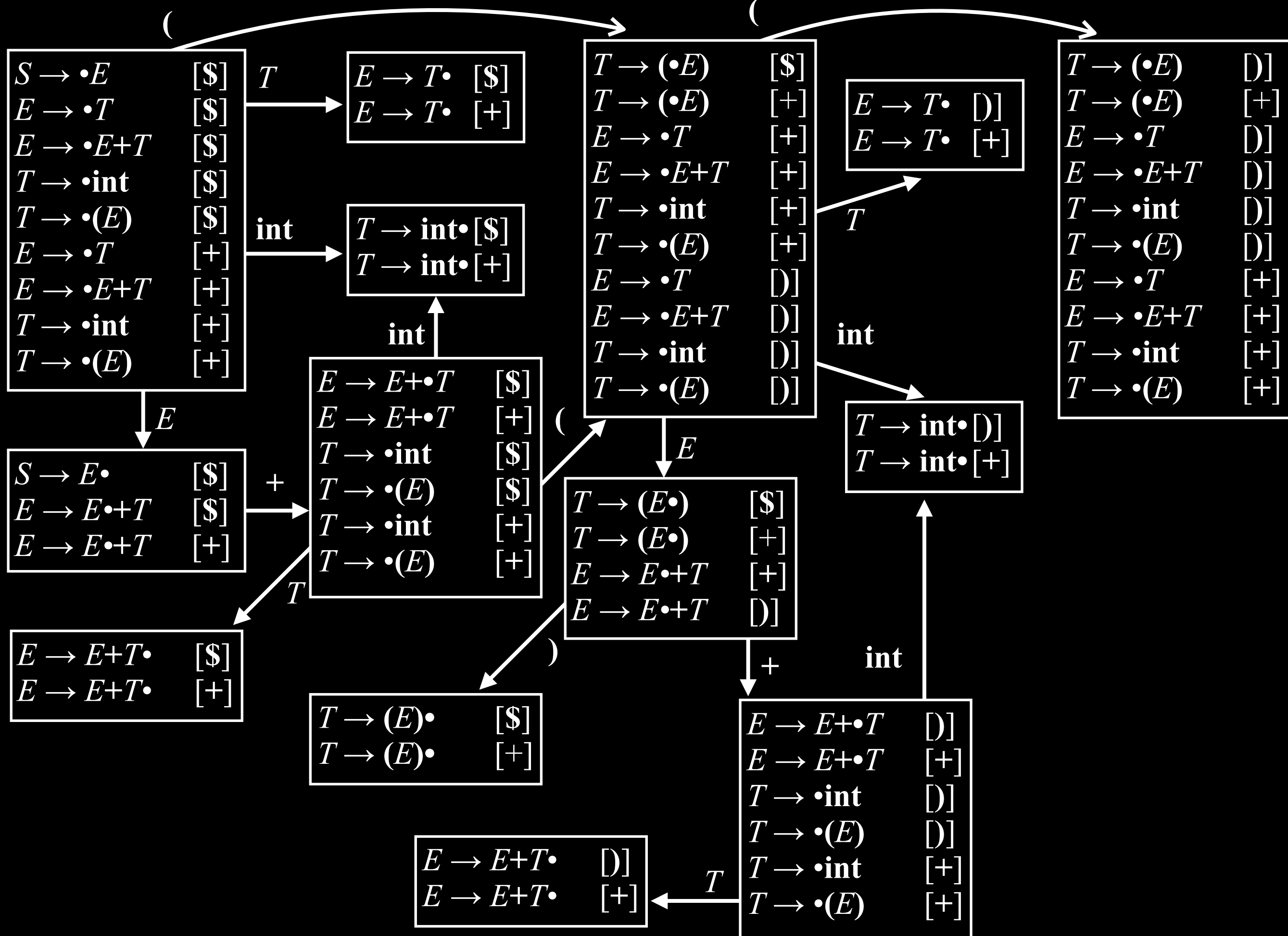


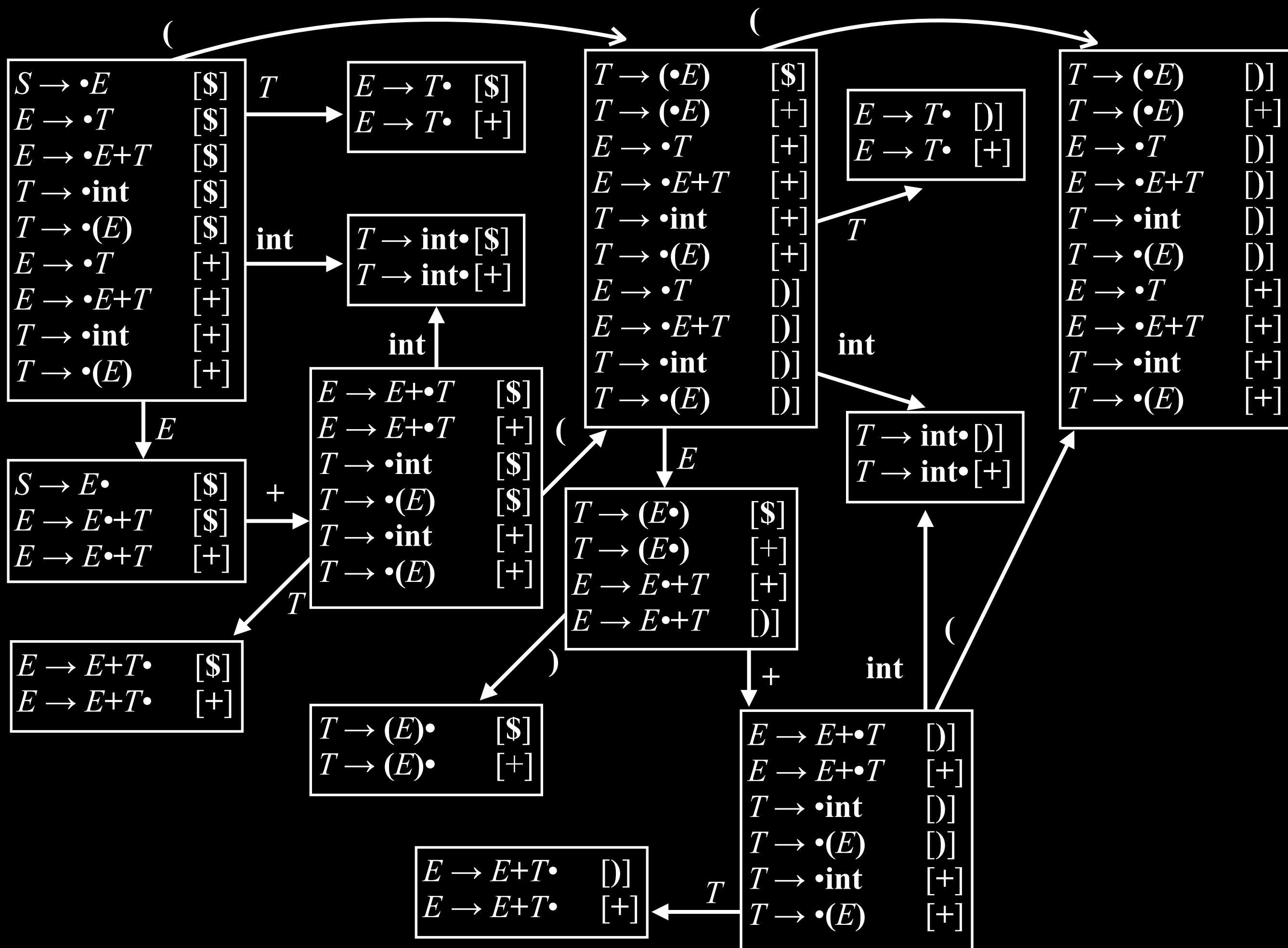


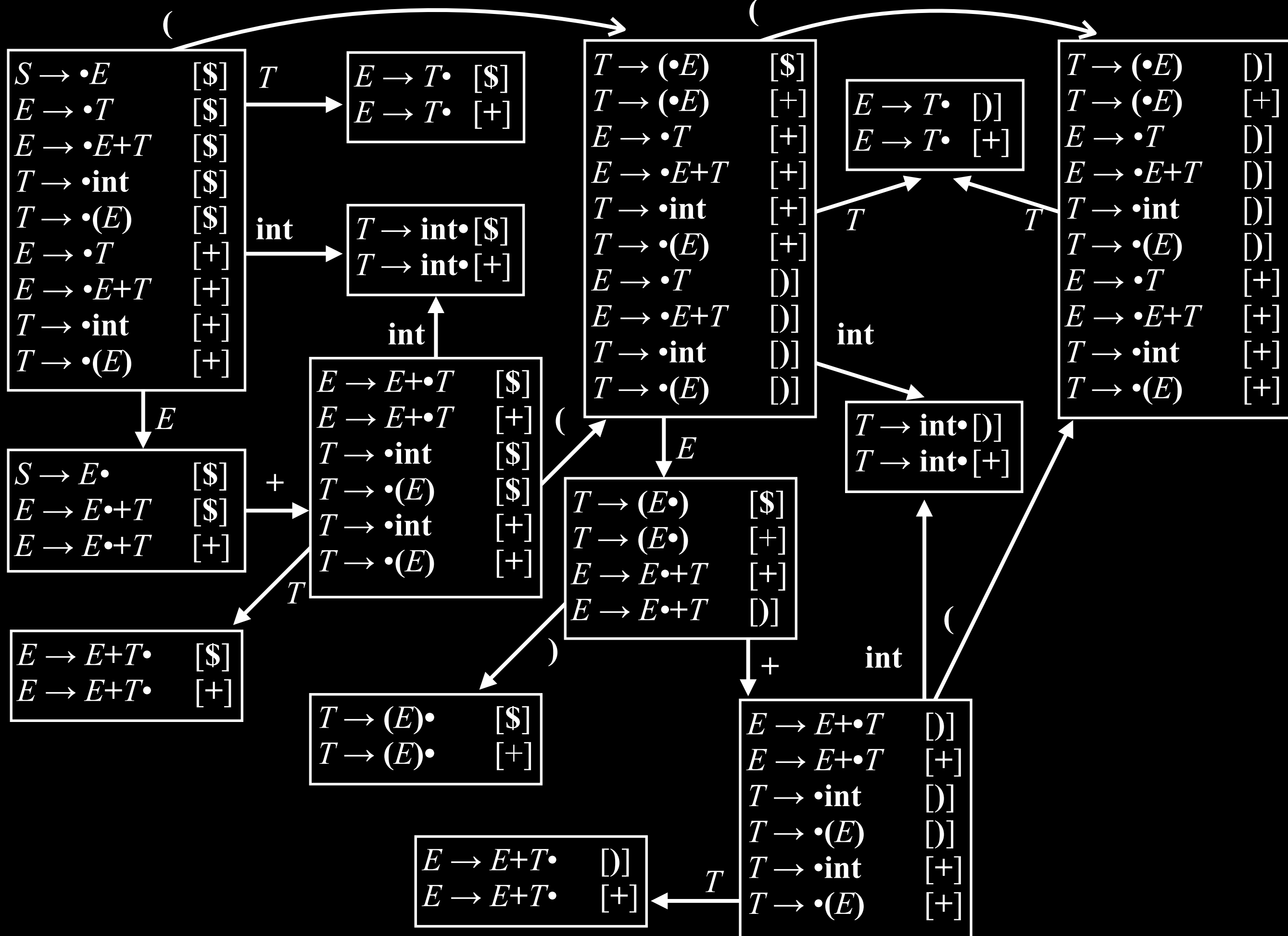


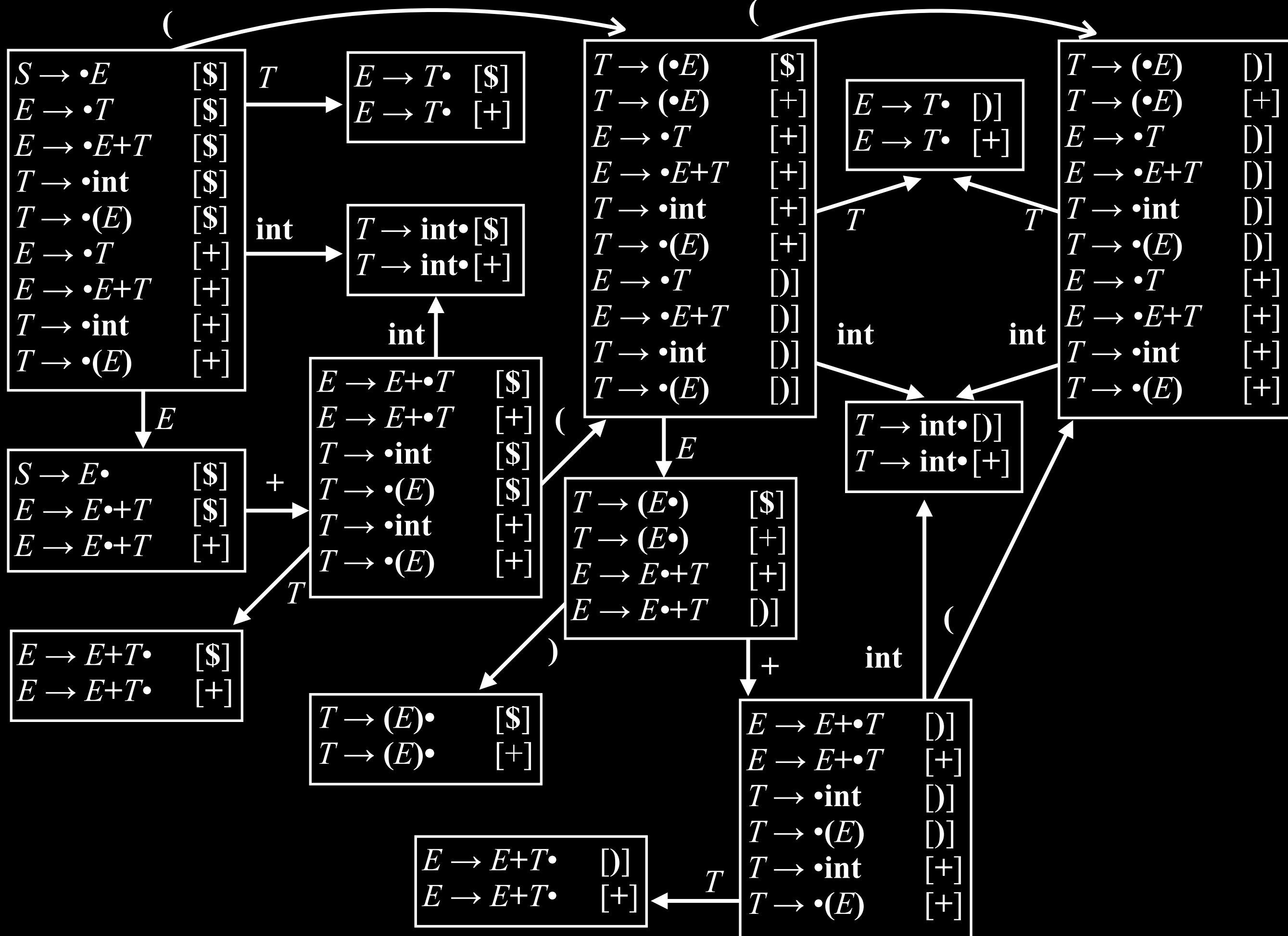


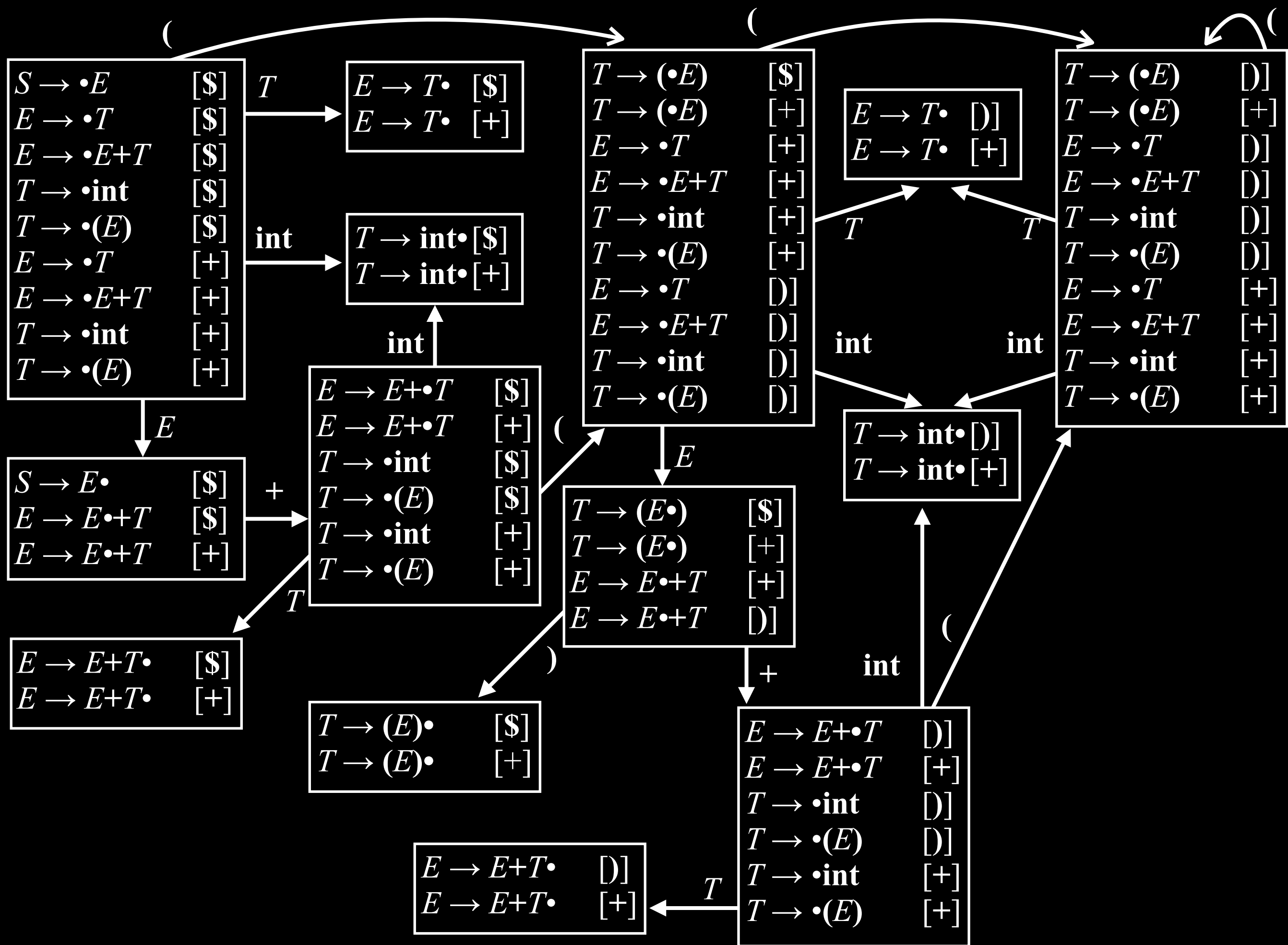




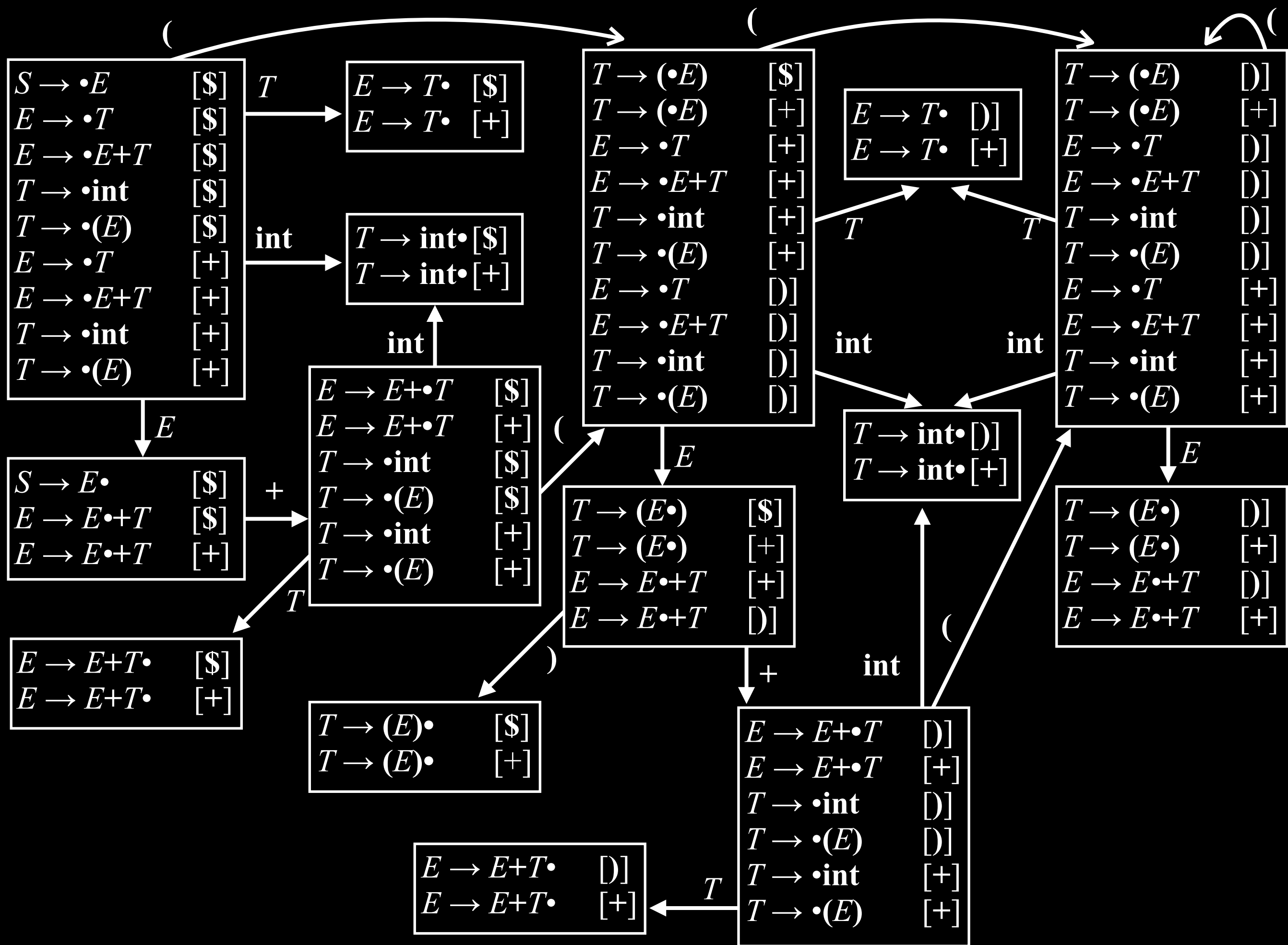


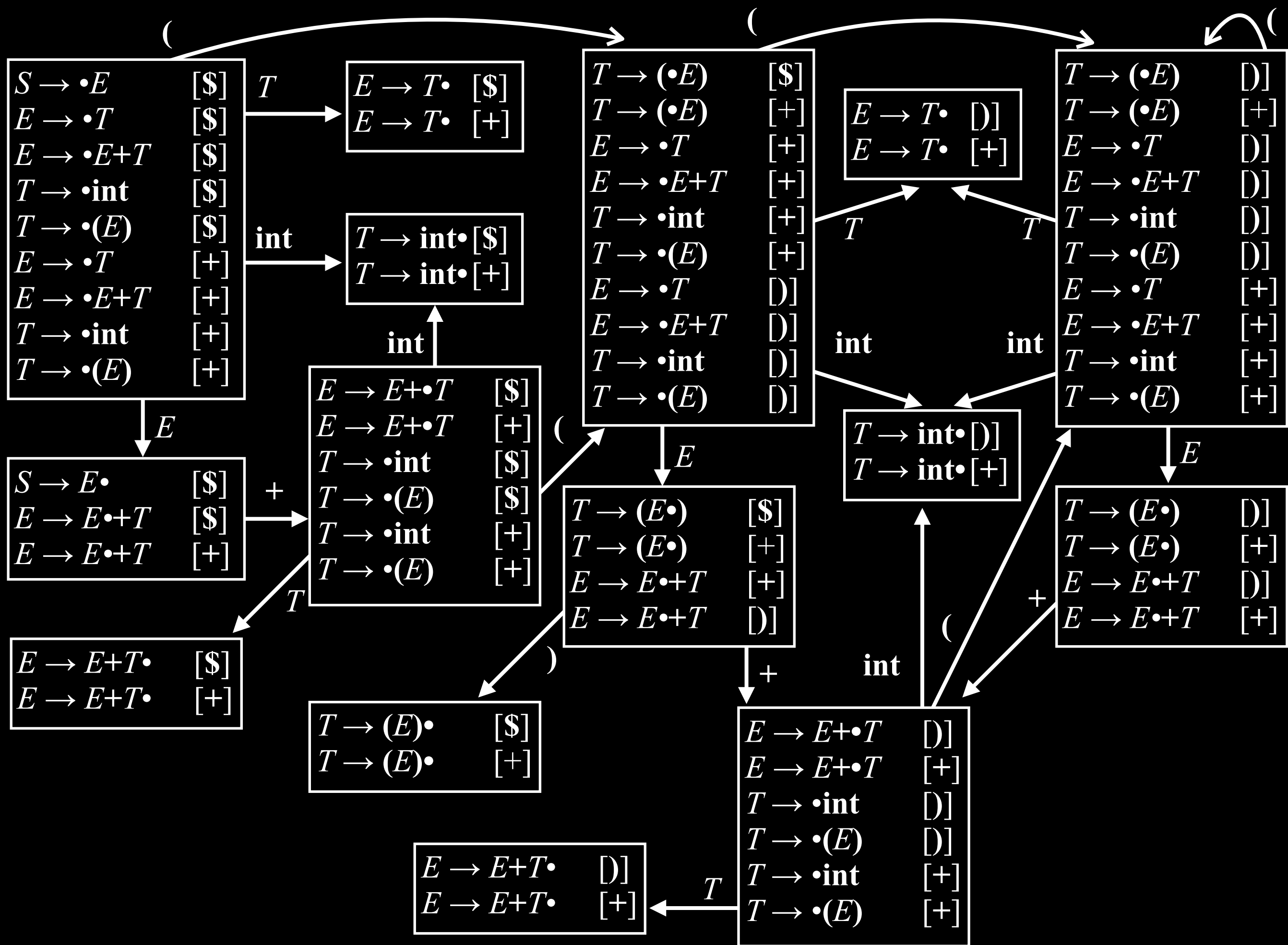


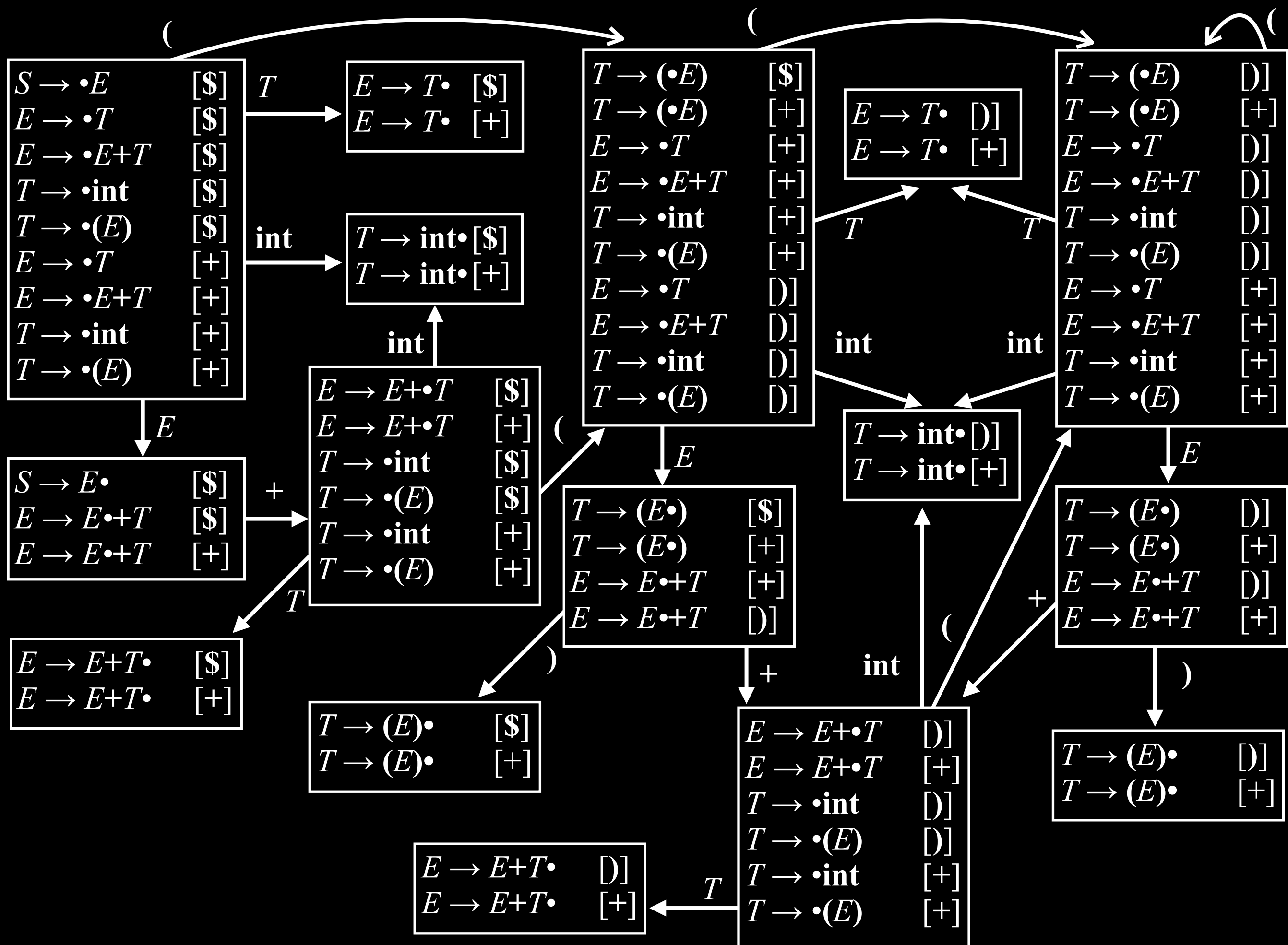


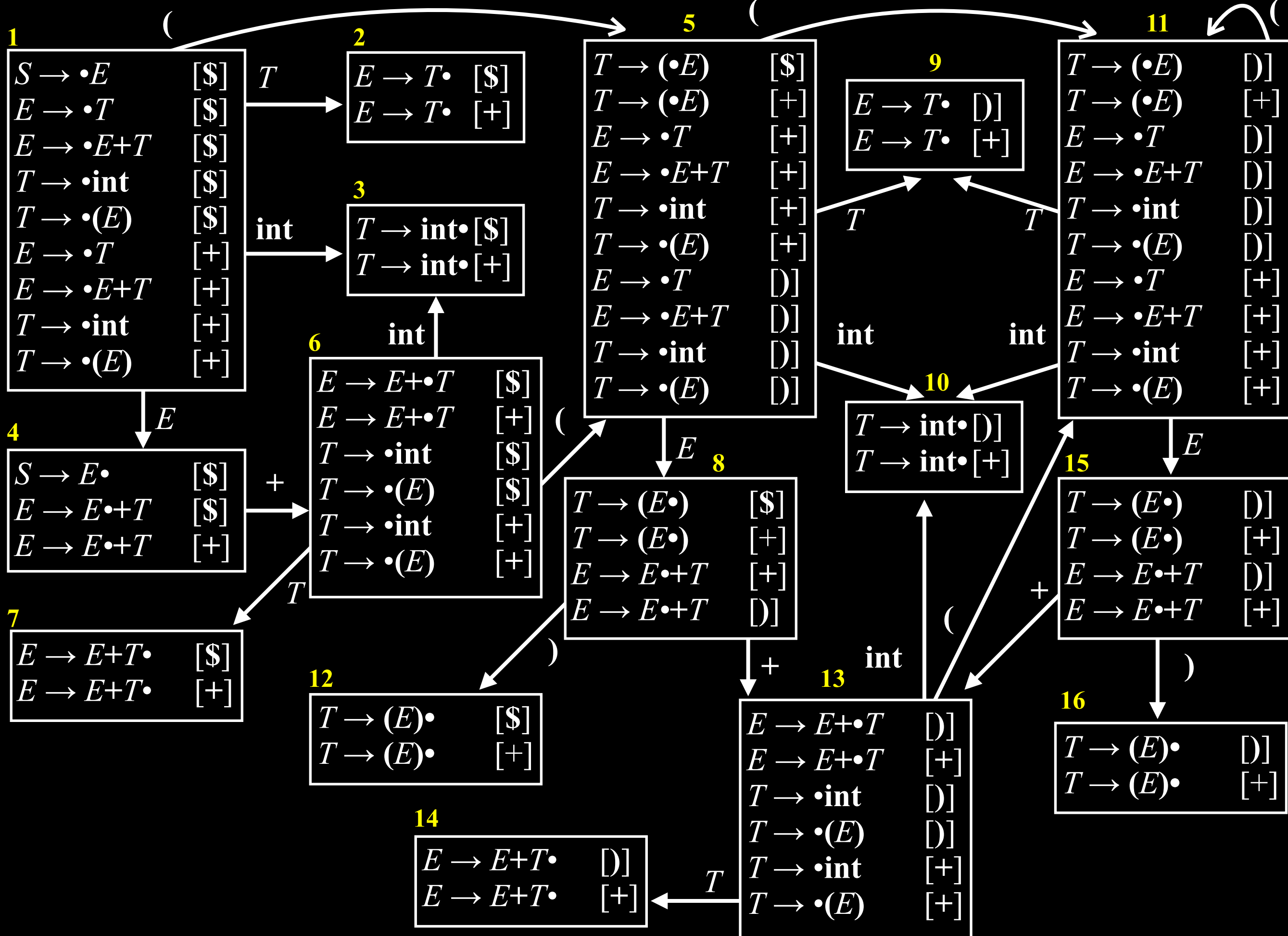












$$_0 S \rightarrow E$$

$$_1 E \rightarrow T$$

$$_3 T \rightarrow \mathbf{int}$$

$$_2 E \rightarrow E + T$$

$$_4 T \rightarrow (E)$$

	int	(	)	+	\$	<i>T</i>	<i>E</i>
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							

$$_0 S \rightarrow E$$

$$_1 E \rightarrow T$$

$$_3 T \rightarrow \mathbf{int}$$

$$_2 E \rightarrow E + T$$

$$_4 T \rightarrow (E)$$

	int	(	)	+	\$	T	E
1	<i>shift 3</i>	<i>shift 5</i>				<i>goto 2</i>	<i>goto 4</i>
2				<i>reduce 1</i>	<i>reduce 1</i>		
3				<i>reduce 3</i>	<i>reduce 3</i>		
4				<i>shift 6</i>	<i>accept</i>		
5	<i>shift 10</i>	<i>shift 11</i>				<i>goto 9</i>	<i>goto 8</i>
6	<i>shift 3</i>	<i>shift 5</i>				<i>goto 7</i>	
7				<i>reduce 2</i>	<i>reduce 2</i>		
8			<i>shift 12</i>	<i>shift 13</i>			
9			<i>reduce 1</i>	<i>reduce 1</i>			
10			<i>reduce 3</i>	<i>reduce 3</i>			
11	<i>shift 10</i>	<i>shift 11</i>				<i>goto 9</i>	<i>goto 15</i>
12				<i>reduce 4</i>	<i>reduce 4</i>		
13	<i>shift 10</i>	<i>shift 11</i>				<i>goto 14</i>	
14			<i>reduce 2</i>	<i>reduce 2</i>			
15			<i>shift 16</i>	<i>shift 13</i>			
16			<i>reduce 4</i>	<i>reduce 4</i>			

# LR(1)

- Extremamente poderoso
  - Toda LL(1) é LR(1)
  - Toda LR(0) é LR(1)
- Qualquer linguagem que pode ser reconhecida por um parser shift-reduce guiado por DFAs tem uma gramática LR(1)

# Autômatos LR(1)

- Em uma gramática com  $n$  terminais, poderia ser  $O(2^n)$  maior que o autômato LR(0)
- replicar cada estados com todos os  $O(2^n)$  *lookaheads* possíveis
- Tabelas LR(1) podem ter centenas de milhares, ou até milhões de estados



# Exercício

$$_0 S' \rightarrow S\$$$

$$_1 S \rightarrow V = E$$

$$_2 S \rightarrow E$$

$$_3 E \rightarrow V$$

$$_4 V \rightarrow \mathbf{x}$$

$$_5 V \rightarrow *E$$

# Exercício

$$_0 S' \rightarrow S\$$$

$$_1 S \rightarrow CC$$

$$_2 C \rightarrow cC$$

$$_3 C \rightarrow d$$