

CSE401:

# LL(1) Parsing Example

---

Larry Ruzzo

Spring 2004

Slides by Chambers, Eggers, Notkin, Ruzzo, and others

© W.L.Ruzzo & UW CSE 1994-2004



# LL(1) Parsing Theory

---

**Goal:** Formal, rigorous description of those grammars for which “I can figure out how to do a top-down parse by looking ahead just one token”, plus corresponding algorithms.

**Notation:**

T = Set of **Terminals** (Tokens)

N = Set of **Nonterminals**

\$ = End-of-file character (T-like, but not in  $N \cup T$ )



# Table-driven predictive parser

---

- Automatically compute PREDICT table from grammar
- PREDICT(nonterminal, input-symbol)
  - ➔ action, e.g. which rhs or error



# Example 1

Stmt ::= **1** if expr then Stmt else Stmt |  
          **2** while Expr do Stmt |  
          **3** begin Stmts end  
Stmts ::= **4** Stmt ; Stmts | **5** ε  
Expr ::= **6** id

	if	then	else	while	do	begin	end	id	;	\$
Stmt	<b>1</b>			<b>2</b>		<b>3</b>				
Stmts	<b>4</b>			<b>4</b>		<b>4</b>	<b>5</b>			
Expr								<b>6</b>		

empty = error



# LL(1) Parsing Algorithm

---

```
push S$                                /* S is start symbol */
while Stack not empty
  X := pop(Stack)
  a := peek at next input "token"     /* EOF => $ */
  if X is terminal or $
    if X==a, read token a else abort;
  else look at PREDICT(X, a) /* X is nonterminal*/
    Empty      : abort
    rule X → α : push α
If not at end of input, Abort else Accept
```



# Parser Example

---

- Following slides trace execution of the parser (slide 5) on a token string according to the grammar from slide 4 and the corresponding parse tree
- Snapshots show parser state at the top of the while loop and just before the “if” statement at each iteration, together with a summary of the action taken in the “if”
- Notice how the leaves on the right side of the parse tree correspond to the stack contents

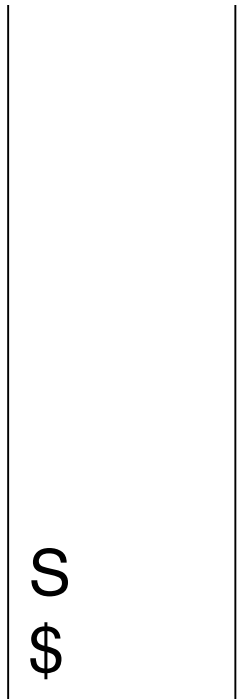
S

At top of loop

X:

a:

Stack:



<b>while</b>	<b>id</b>	<b>do</b>	<b>begin</b>	<b>begin</b>	<b>end</b>	<b>;</b>	<b>end</b>	<b>\$</b>
--------------	-----------	-----------	--------------	--------------	------------	----------	------------	-----------

**Action: 2** S ::= while E do S

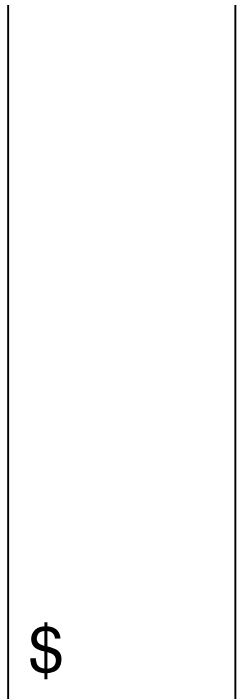
S

Mid loop

X: S

a: while

Stack:



while	id	do	begin	begin	end	;	end	\$
-------	----	----	-------	-------	-----	---	-----	----

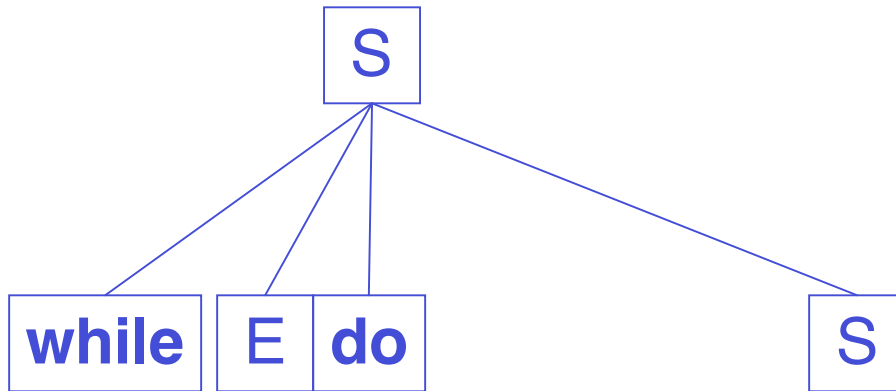
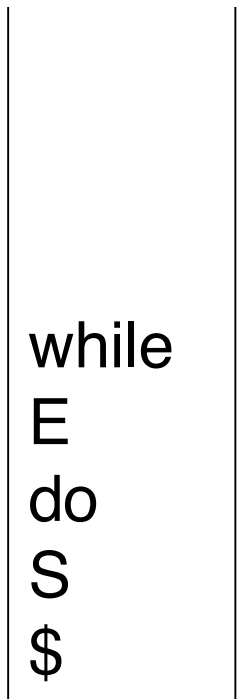


At top of loop

X:

a:

Stack:



while	id	do	begin	begin	end	;	end	\$
-------	----	----	-------	-------	-----	---	-----	----

**Action:** Match

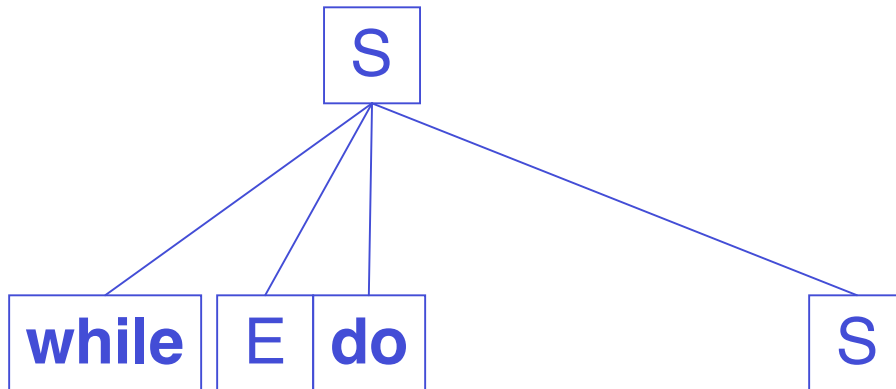
Mid loop

X: while

a: while

Stack:

E  
do  
S  
\$



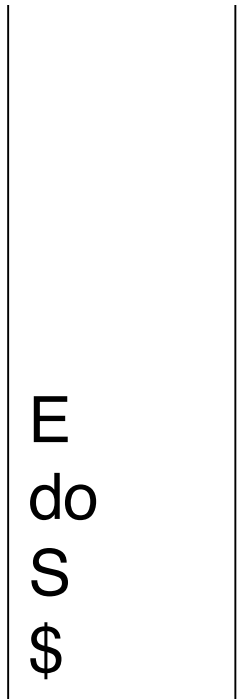
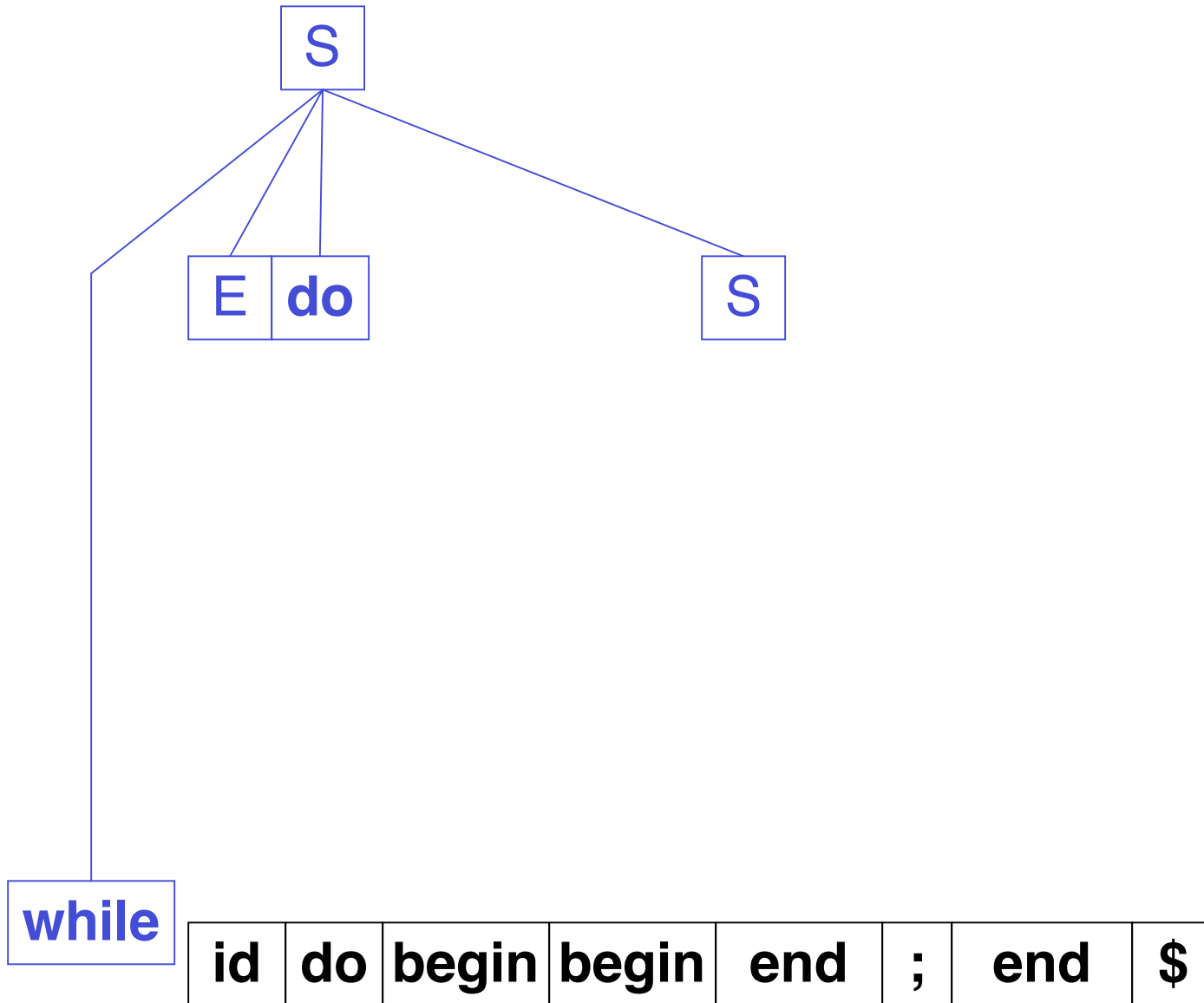
while	id	do	begin	begin	end	;	end	\$
-------	----	----	-------	-------	-----	---	-----	----

At top of loop

X:

a:

Stack:



**Action: 6** E ::= id

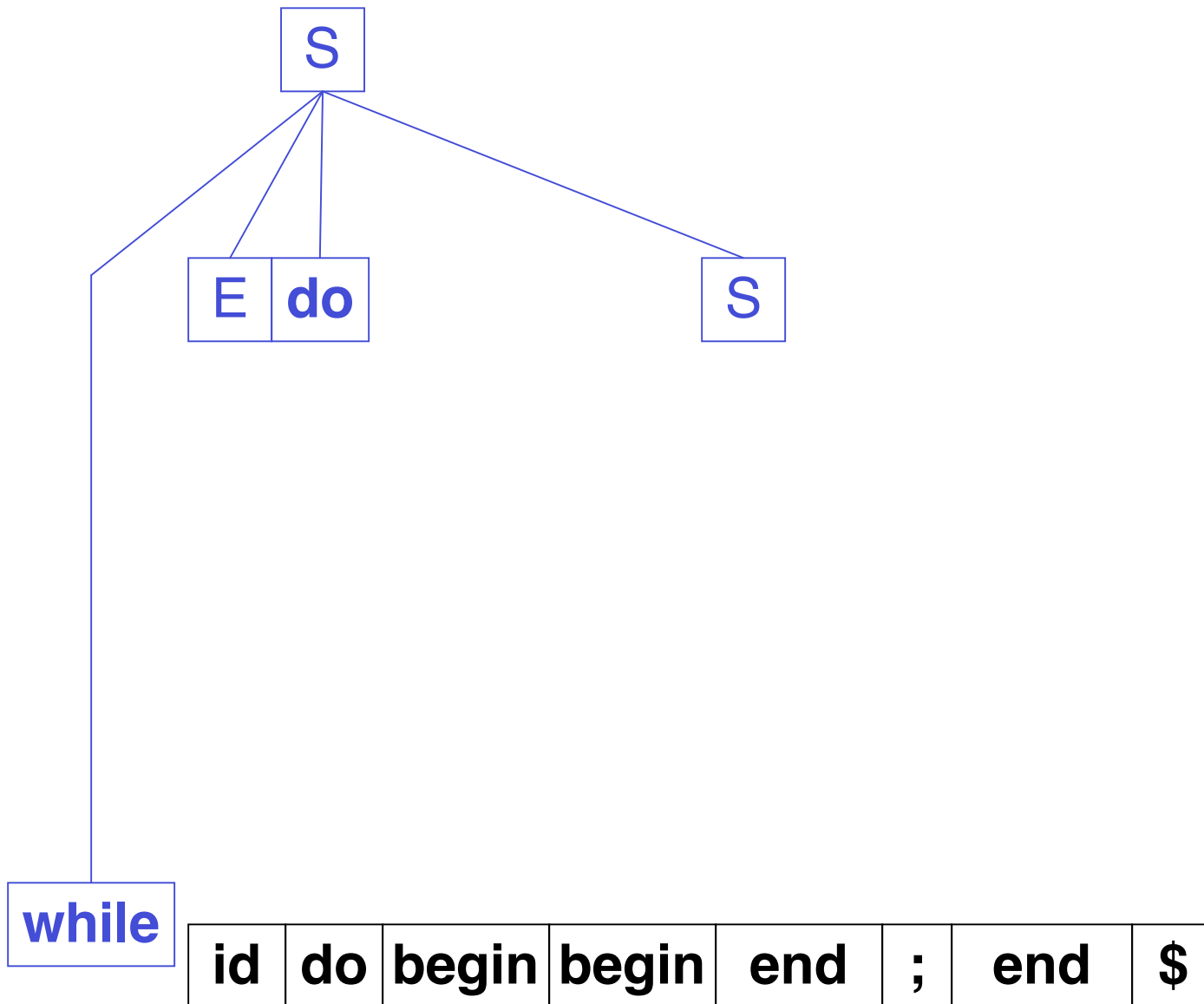
Mid loop

X: E

a: id

Stack:

do  
S  
\$

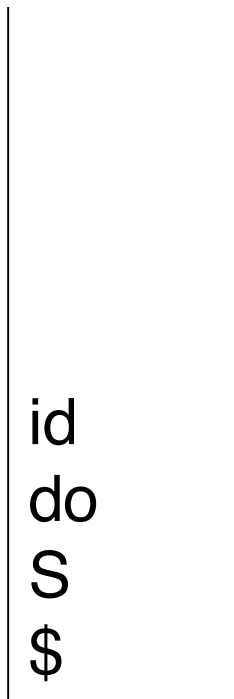
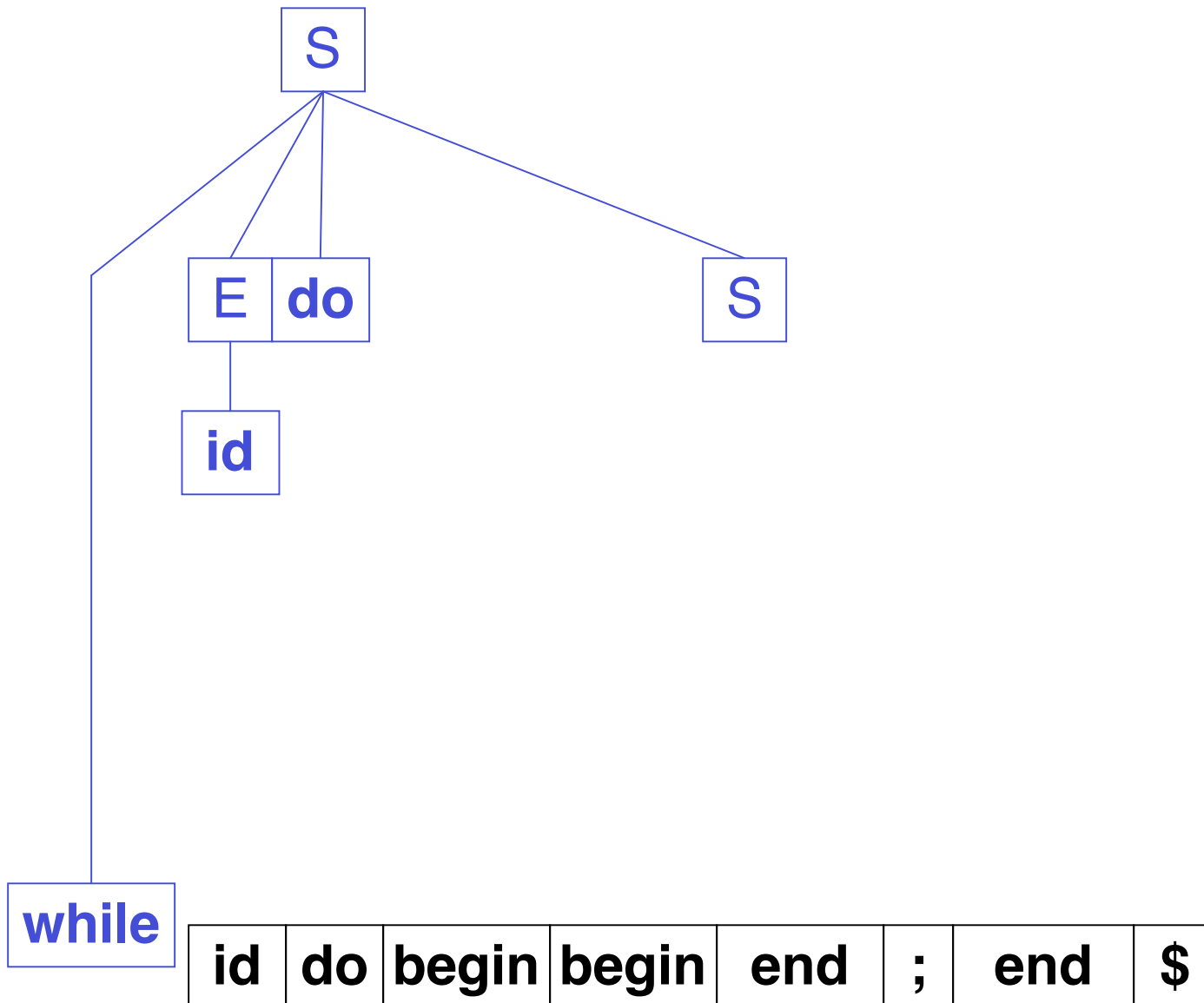


At top of loop

X:

a:

Stack:



**Action:** Match

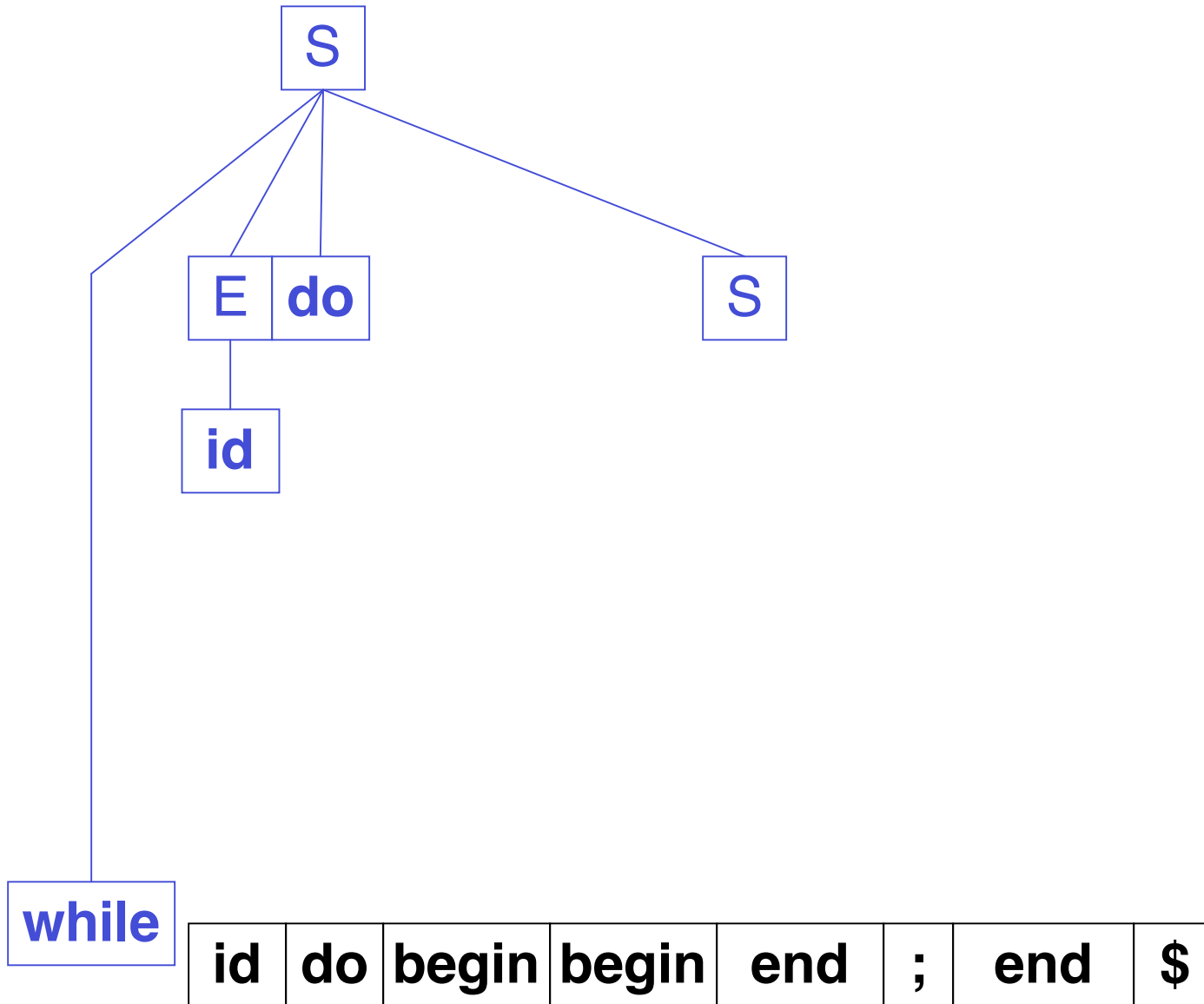
Mid loop

X: id

a: id

Stack:

do  
S  
\$

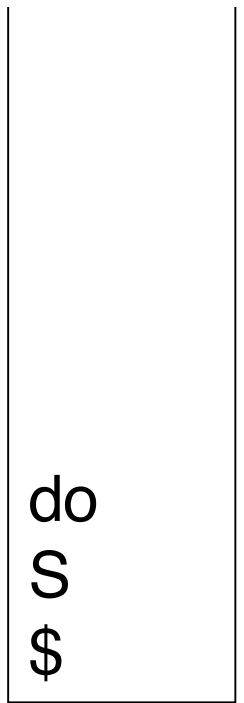
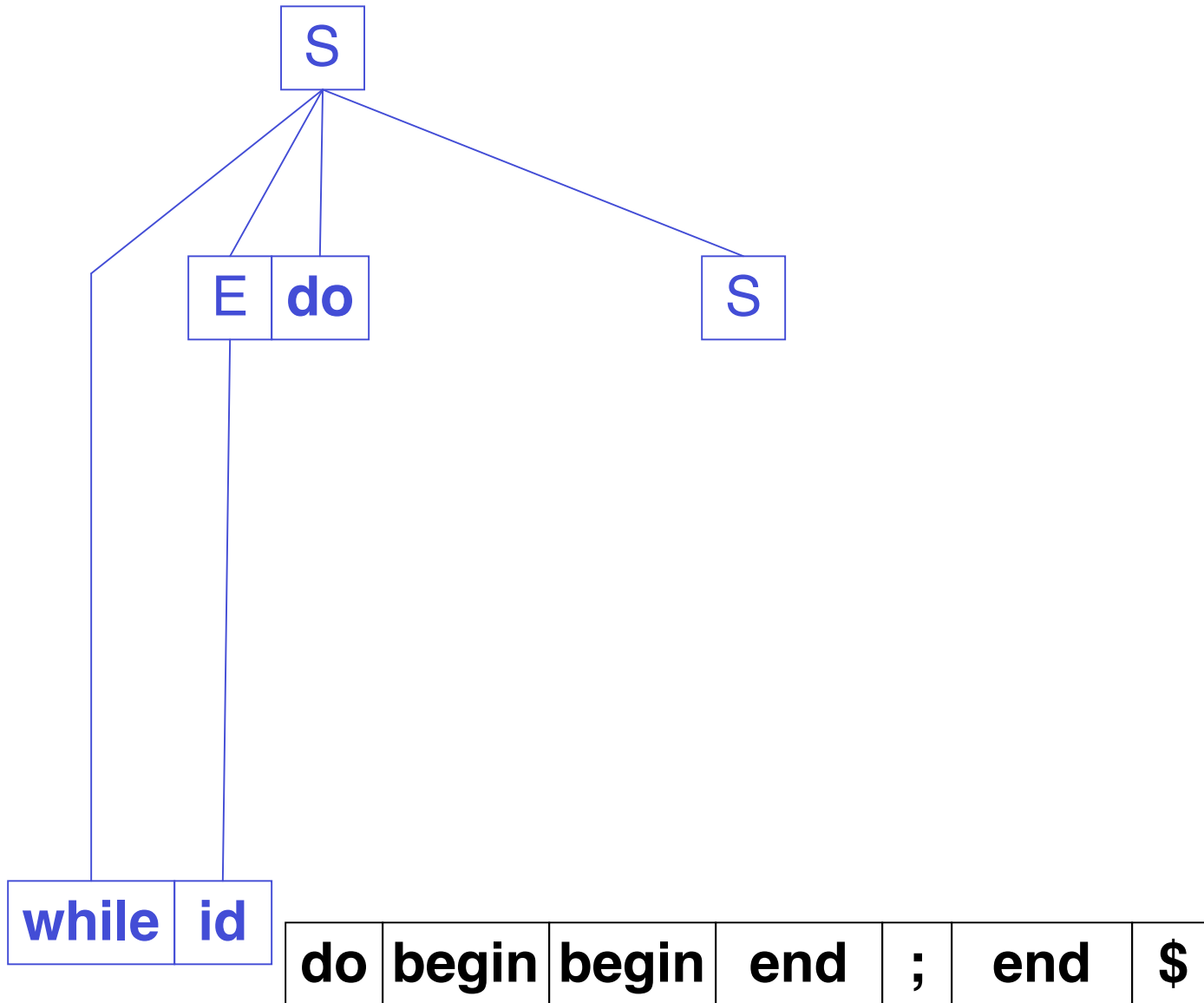


At top of loop

X:

a:

Stack:



**Action:** Match

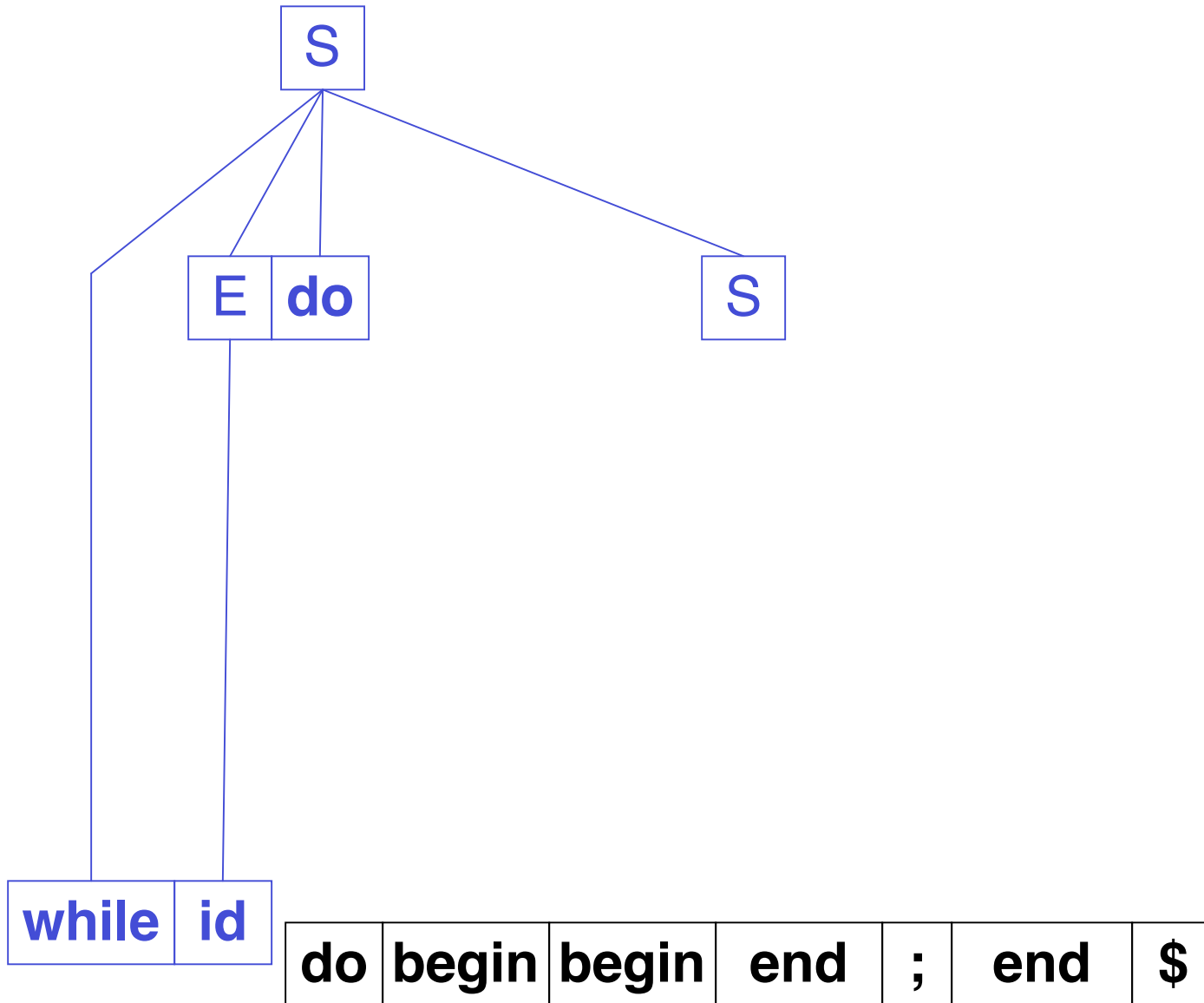
Mid loop

X: do

a: do

Stack:

S  
\$



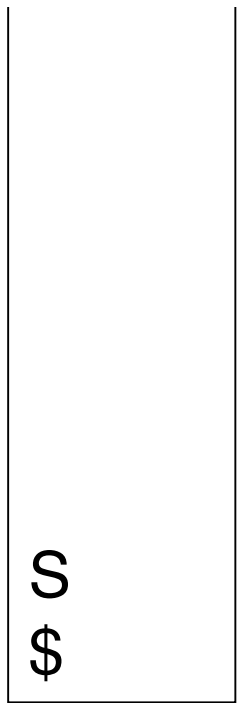
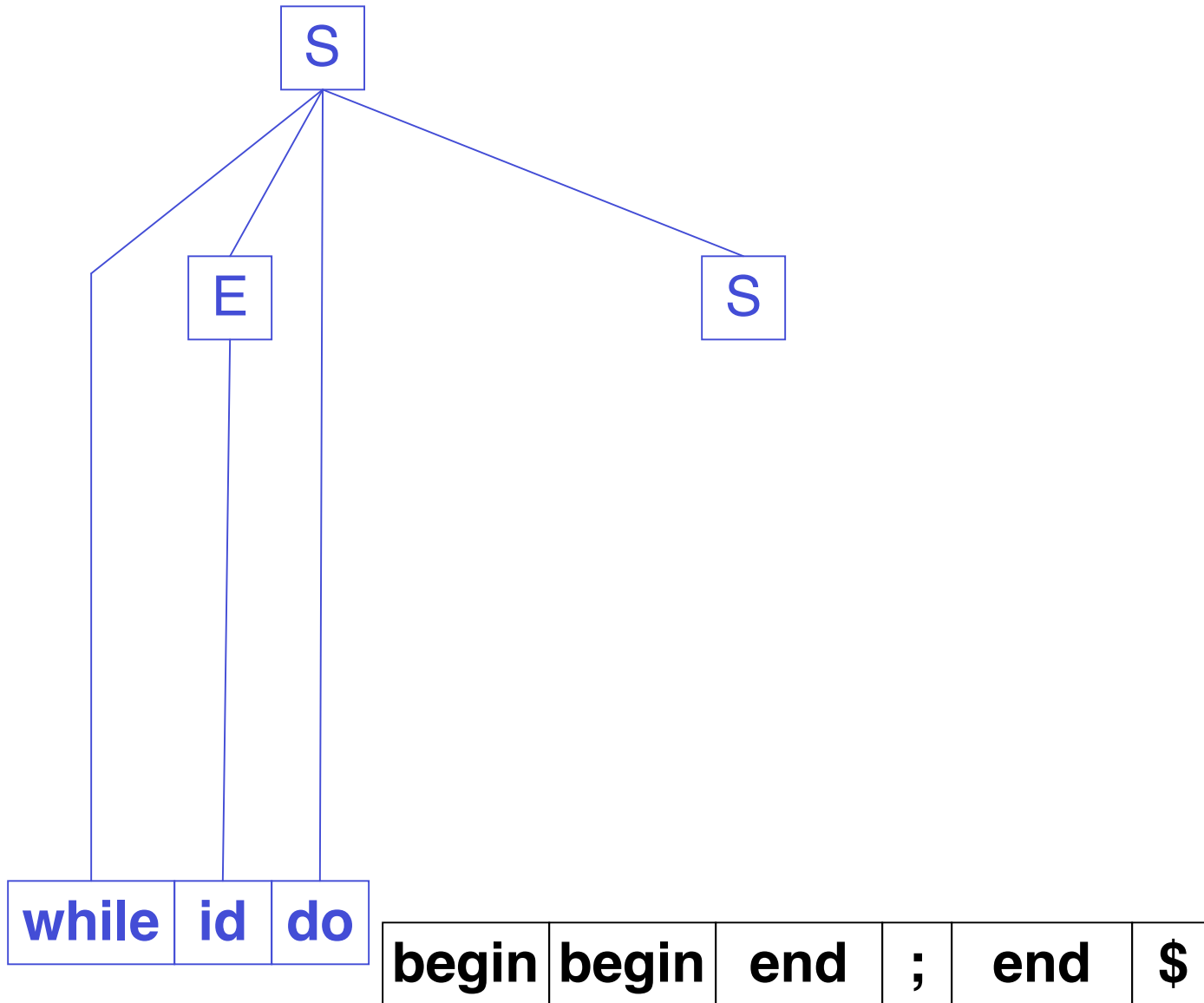


At top of loop

X:

a:

Stack:



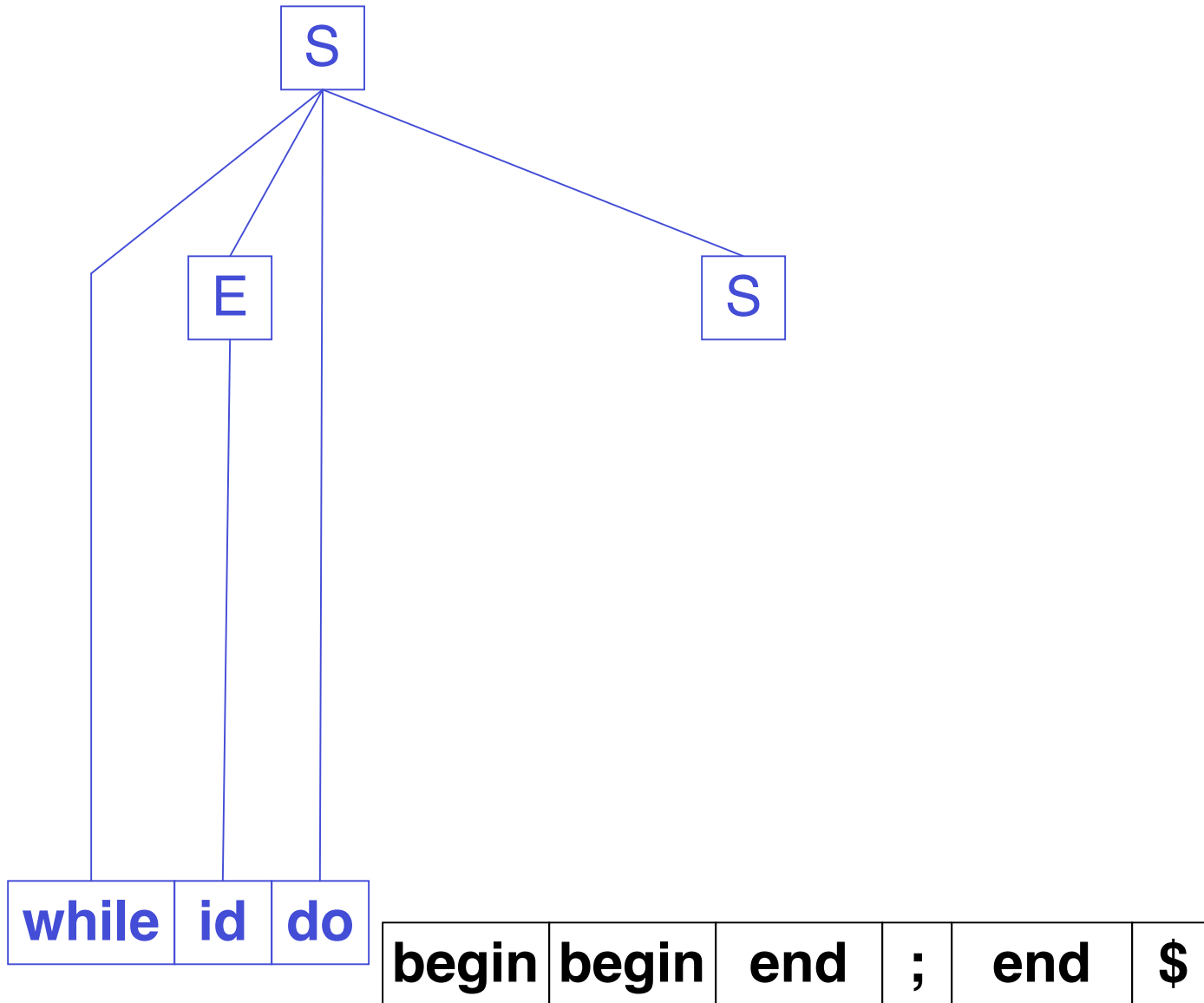
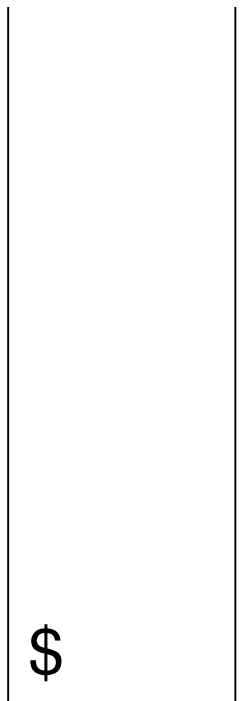
**Action: 3** S ::= begin Ss end

Mid loop

X: S

a: begin

Stack:

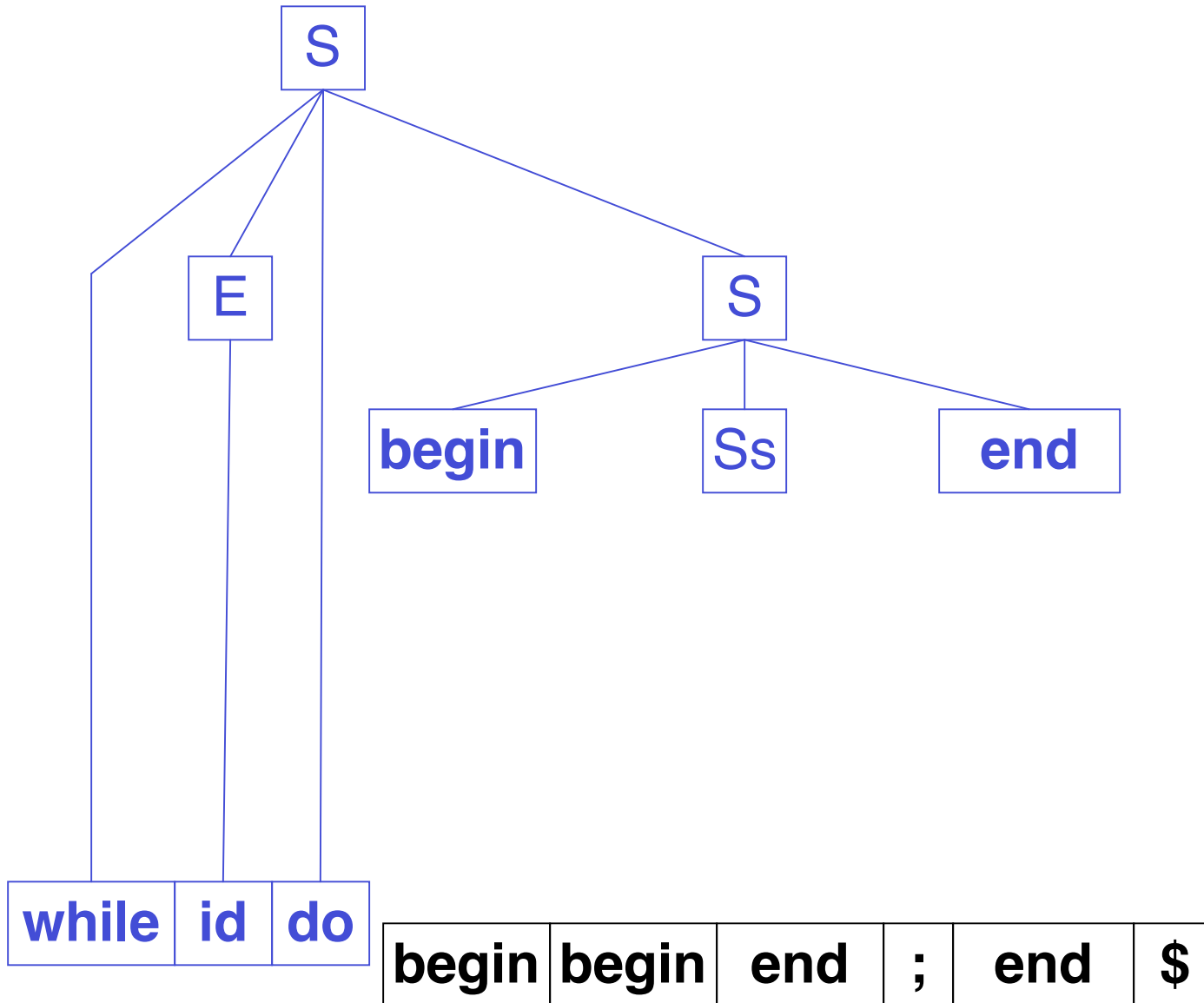


At top of loop

X:

a:

Stack:



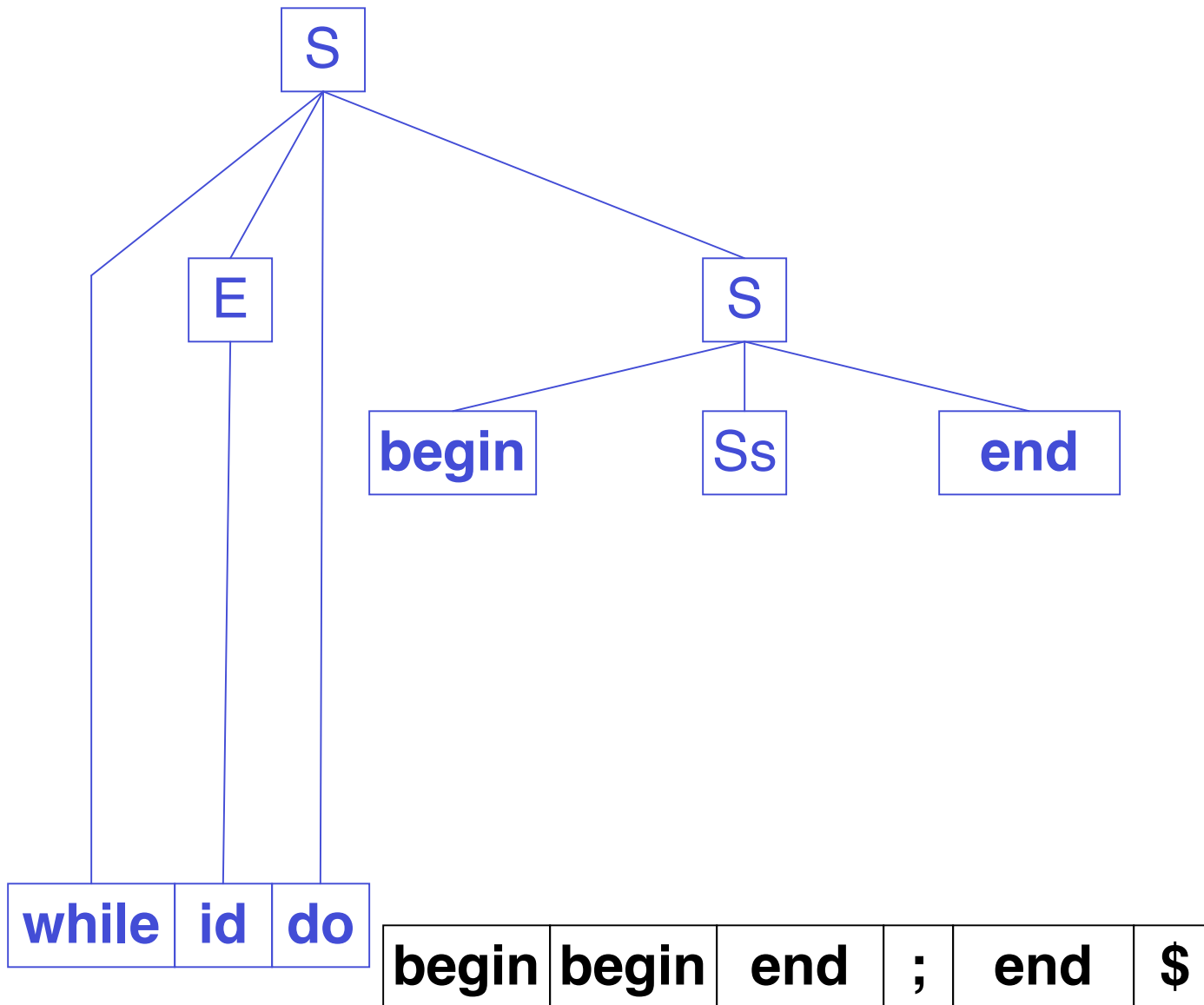
**Action:** Match

Mid loop

X: begin

a: begin

Stack:

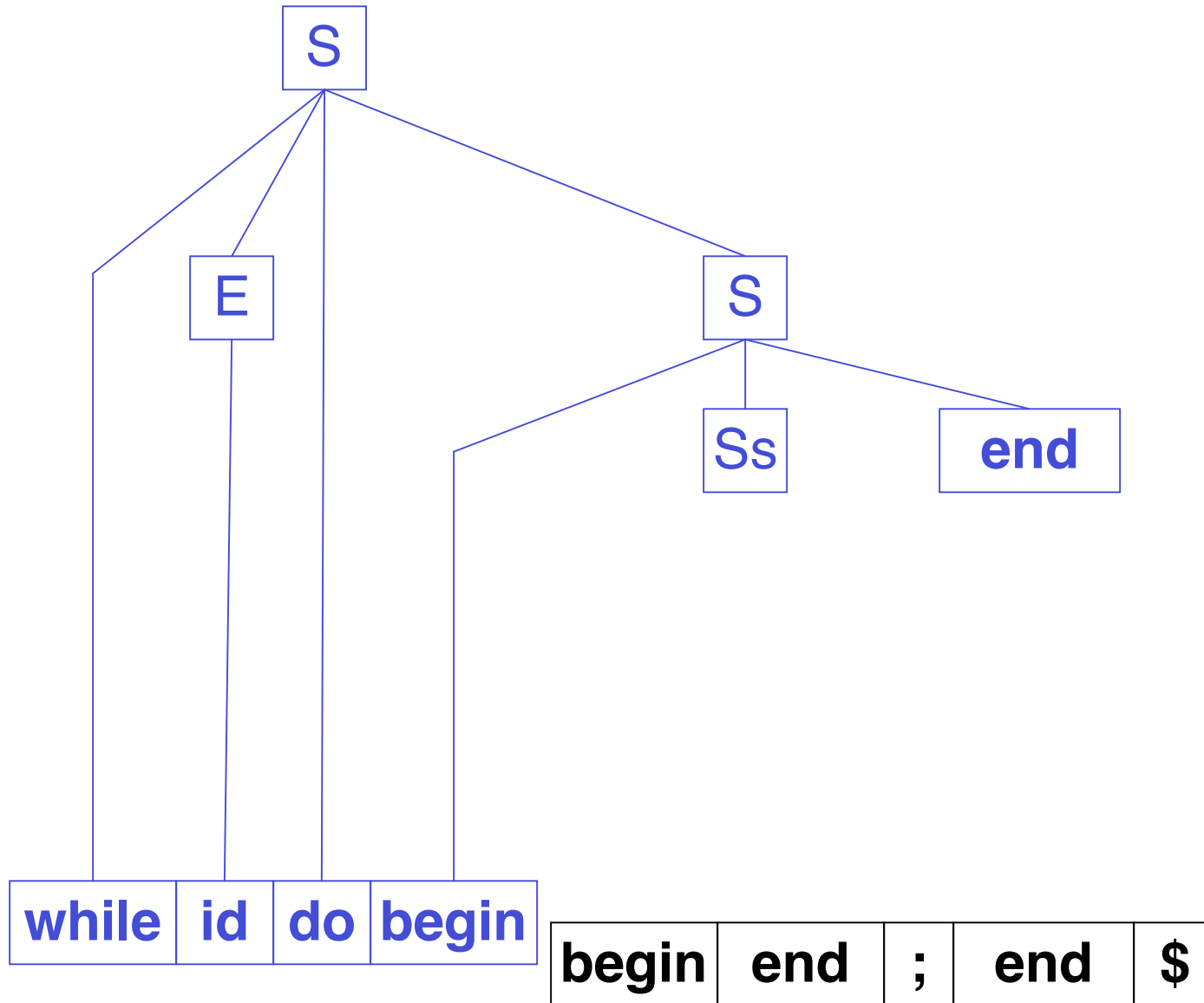


At top of loop

X:

a:

Stack:



**Action: 4**  $Ss ::= S ; Ss$

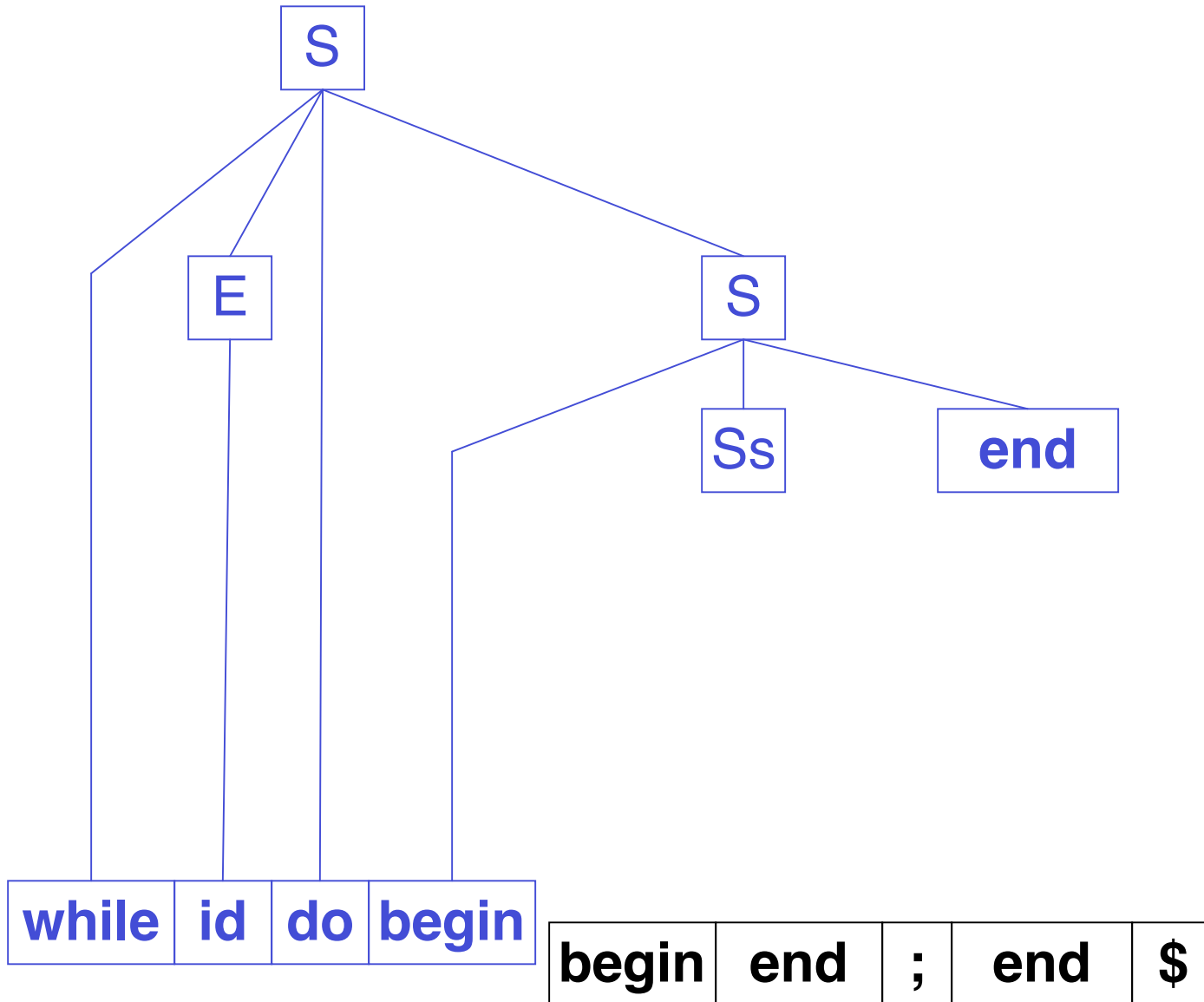
Mid loop

X: Ss

a: begin

Stack:

end  
\$

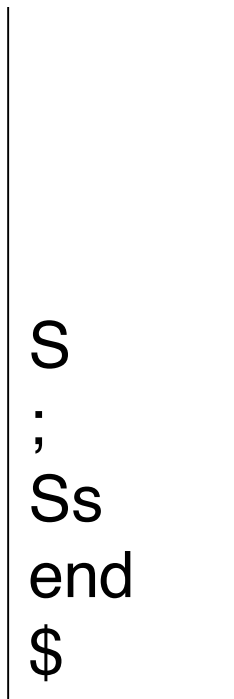
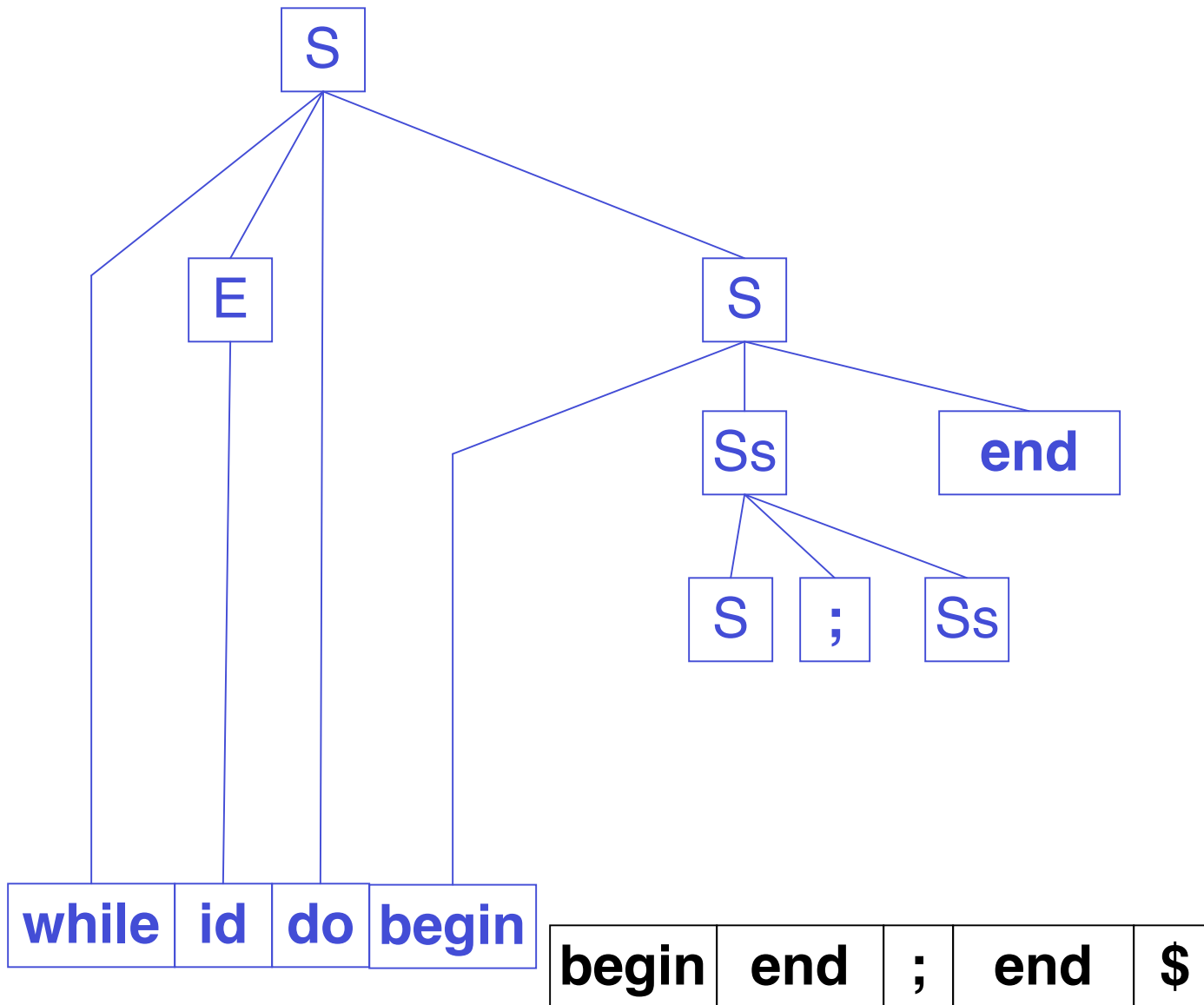


At top of loop

X:

a:

Stack:



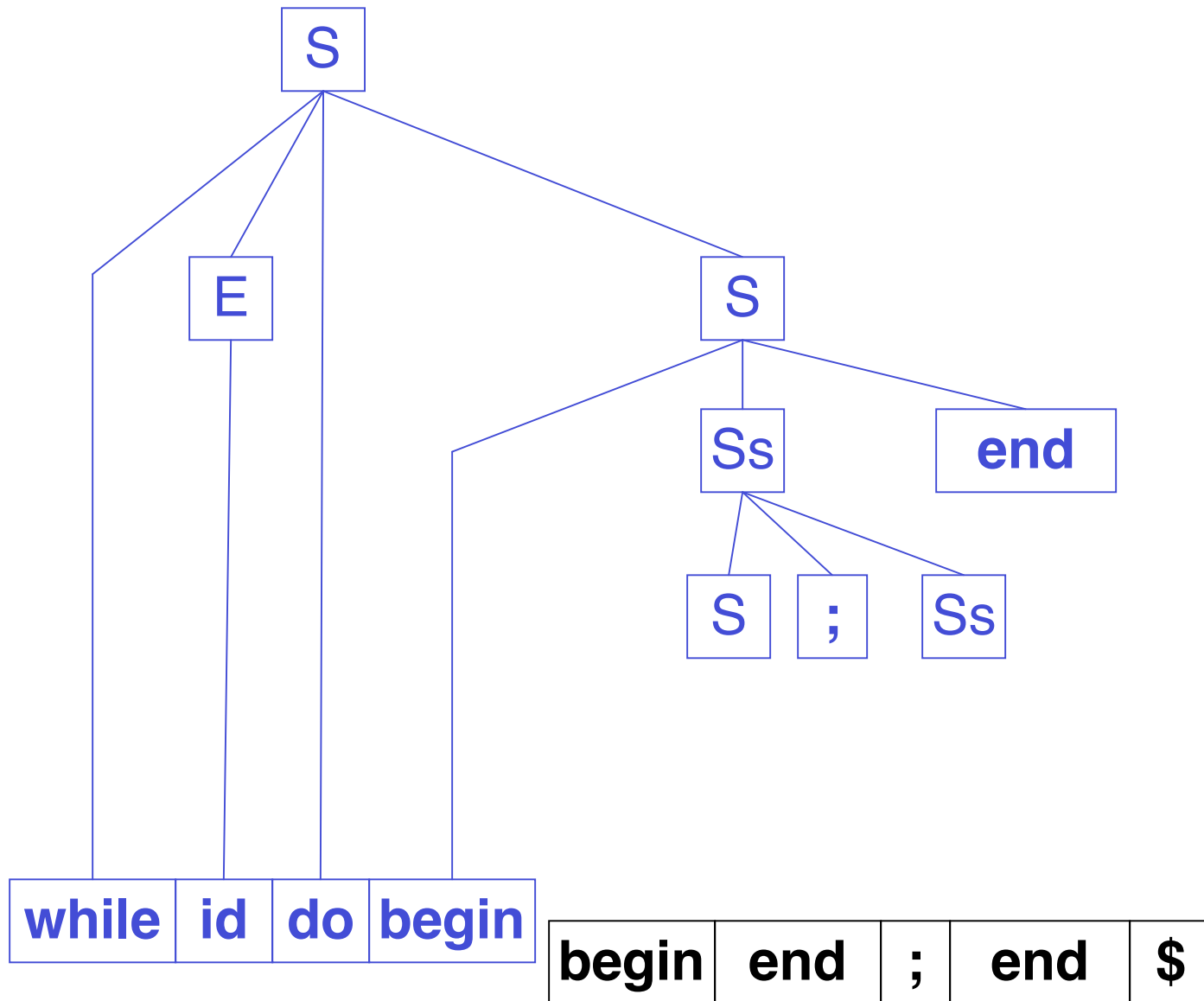
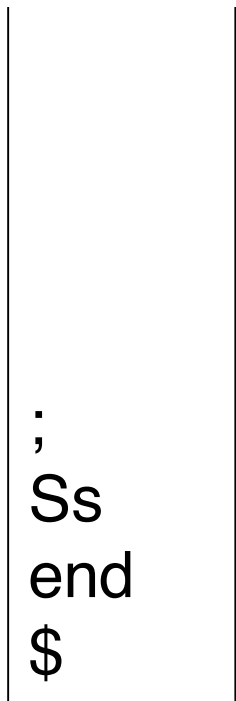
**Action: 3** S ::= begin Ss end

Mid loop

X: S

a: begin

Stack:



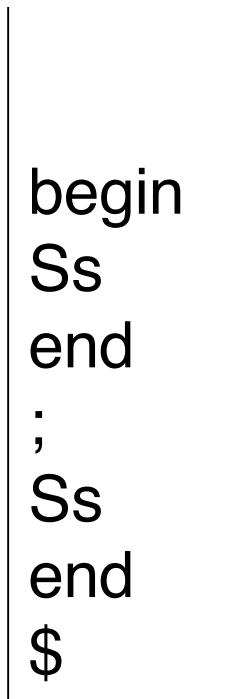
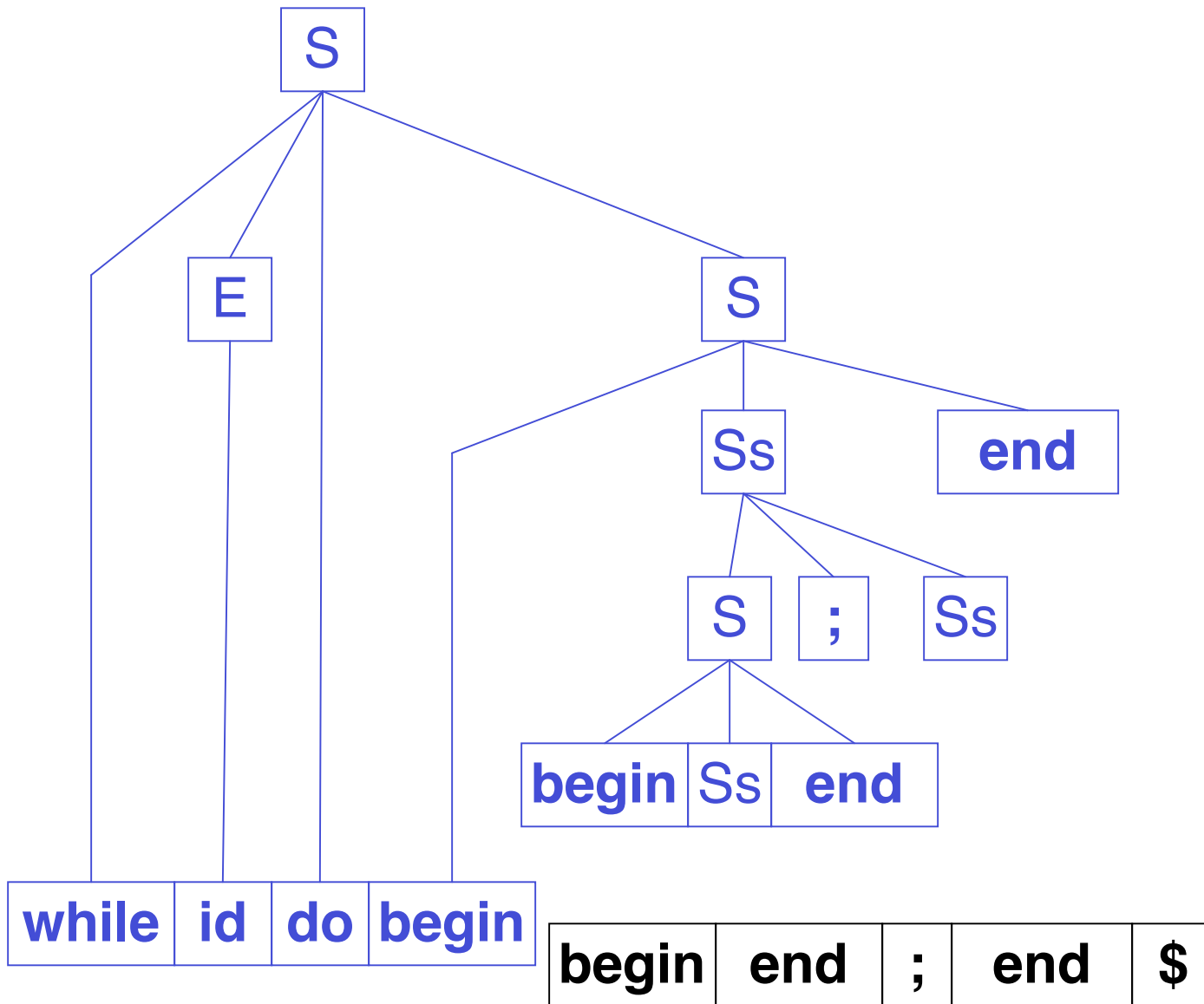


Top of loop

X:

a:

Stack:



**Action:** Match

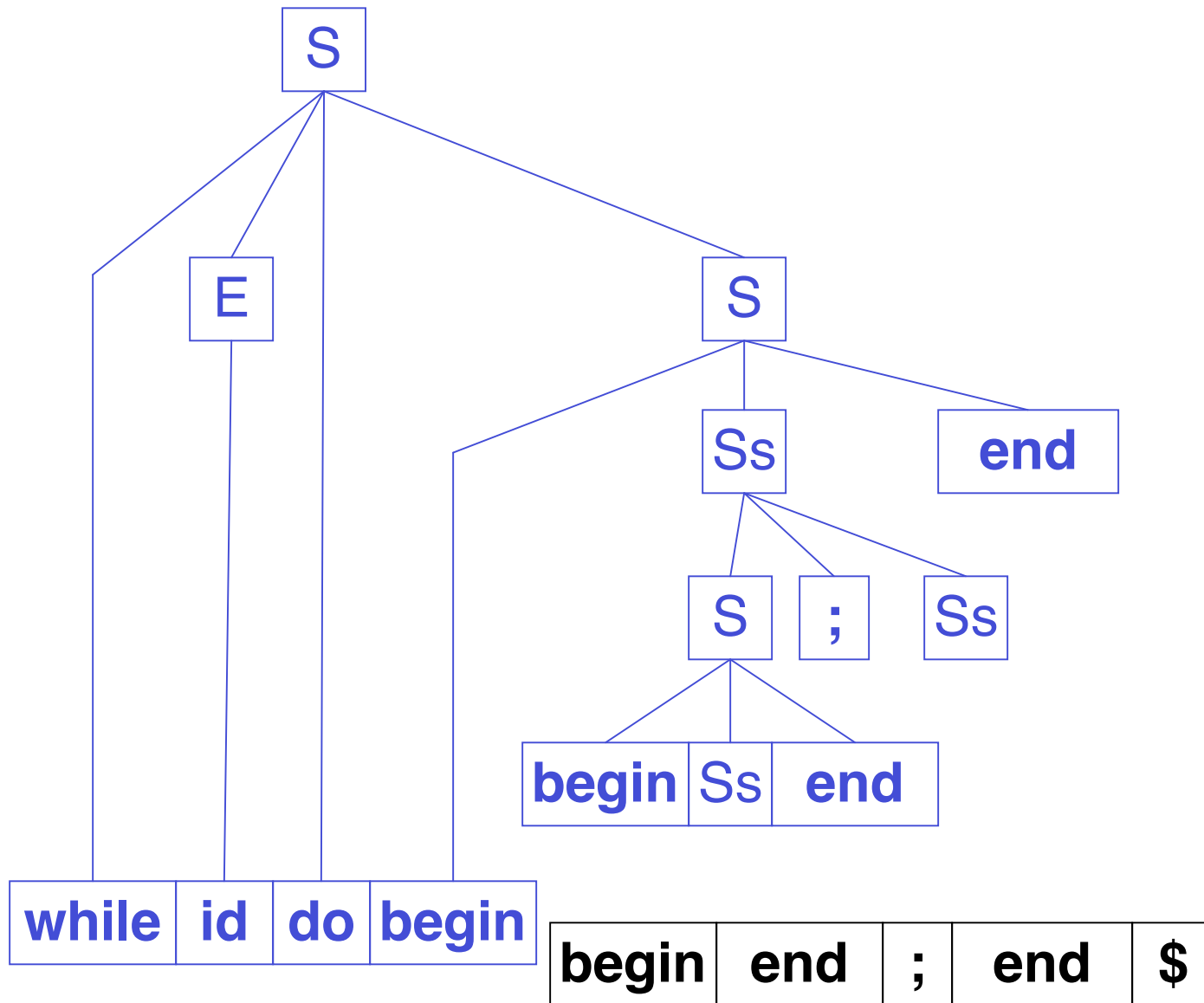
Mid loop

X: begin

a: begin

Stack:

Ss  
end  
;  
Ss  
end  
\$

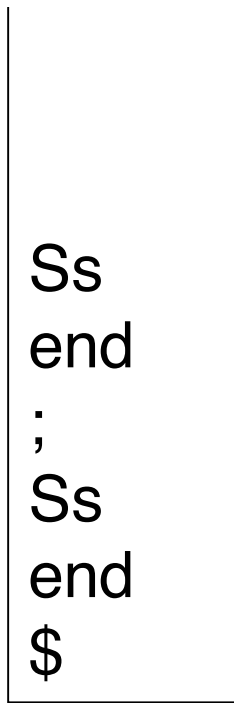
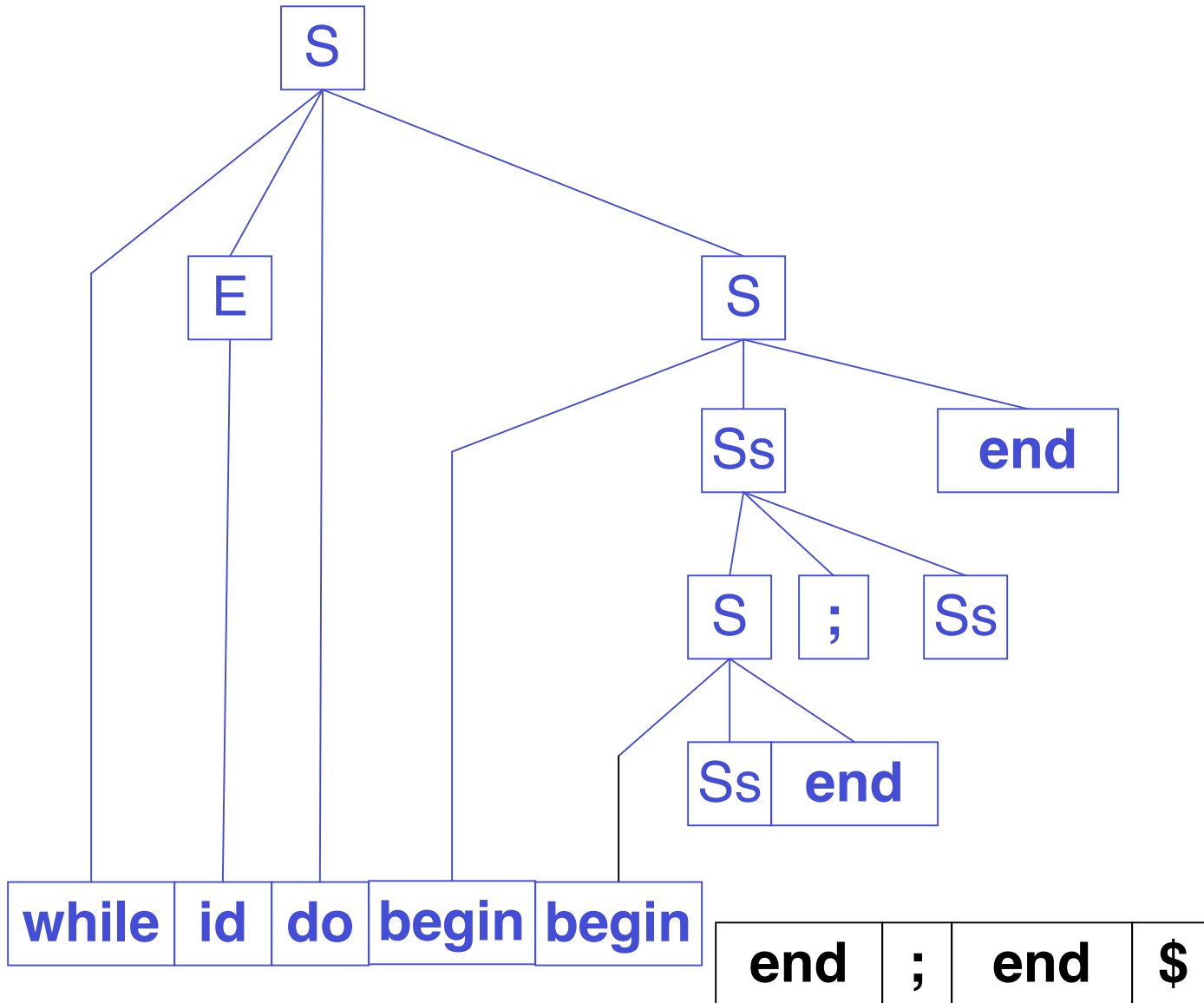


Top of loop

X:

a:

Stack:



**Action: 5**  $Ss ::= \epsilon$

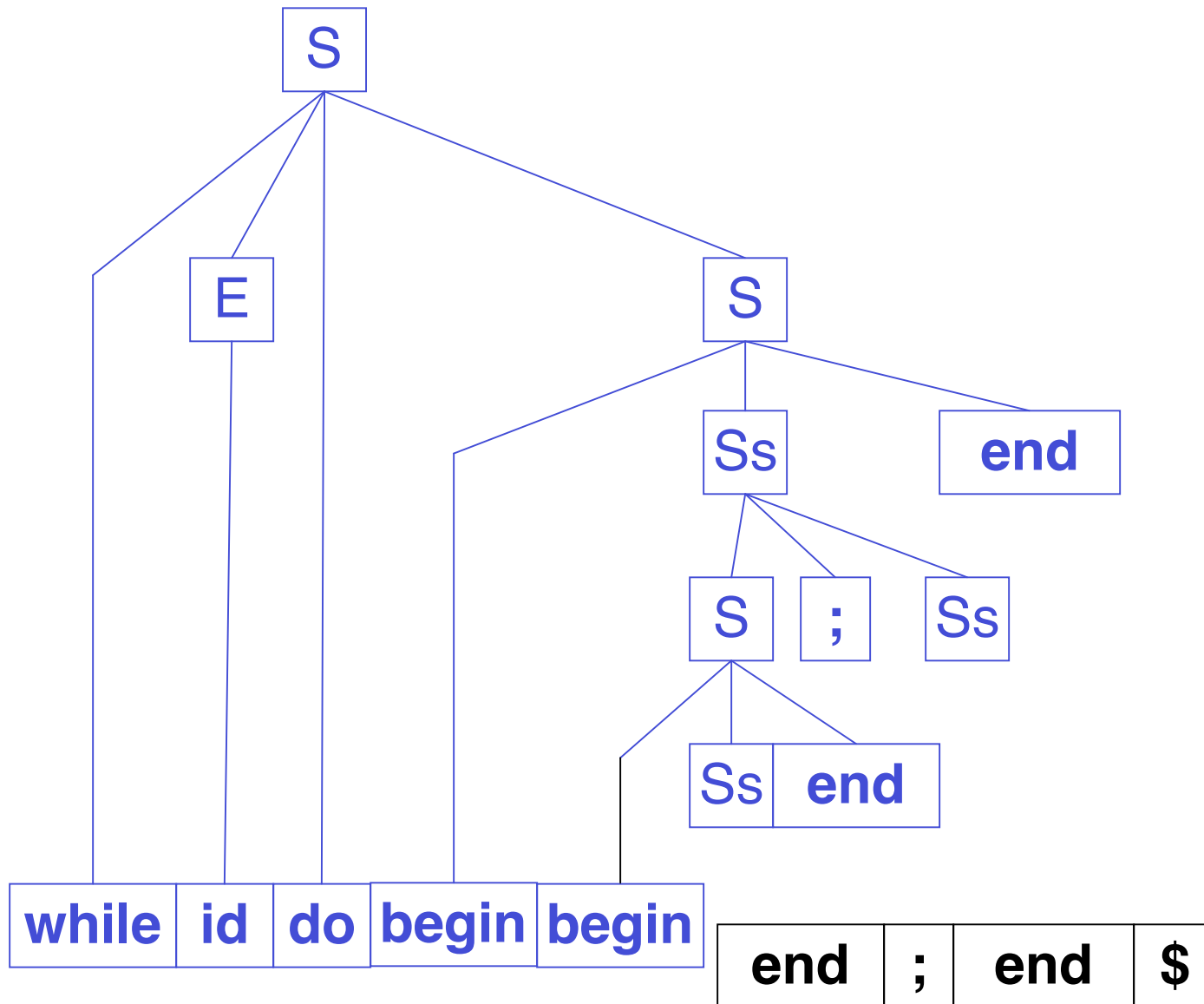
Mid loop

X:  $Ss$

a: end

Stack:

end  
;  
 $Ss$   
end  
\$

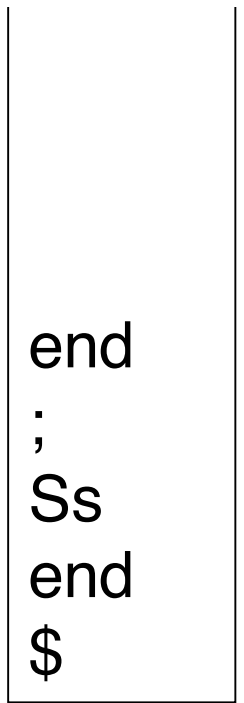
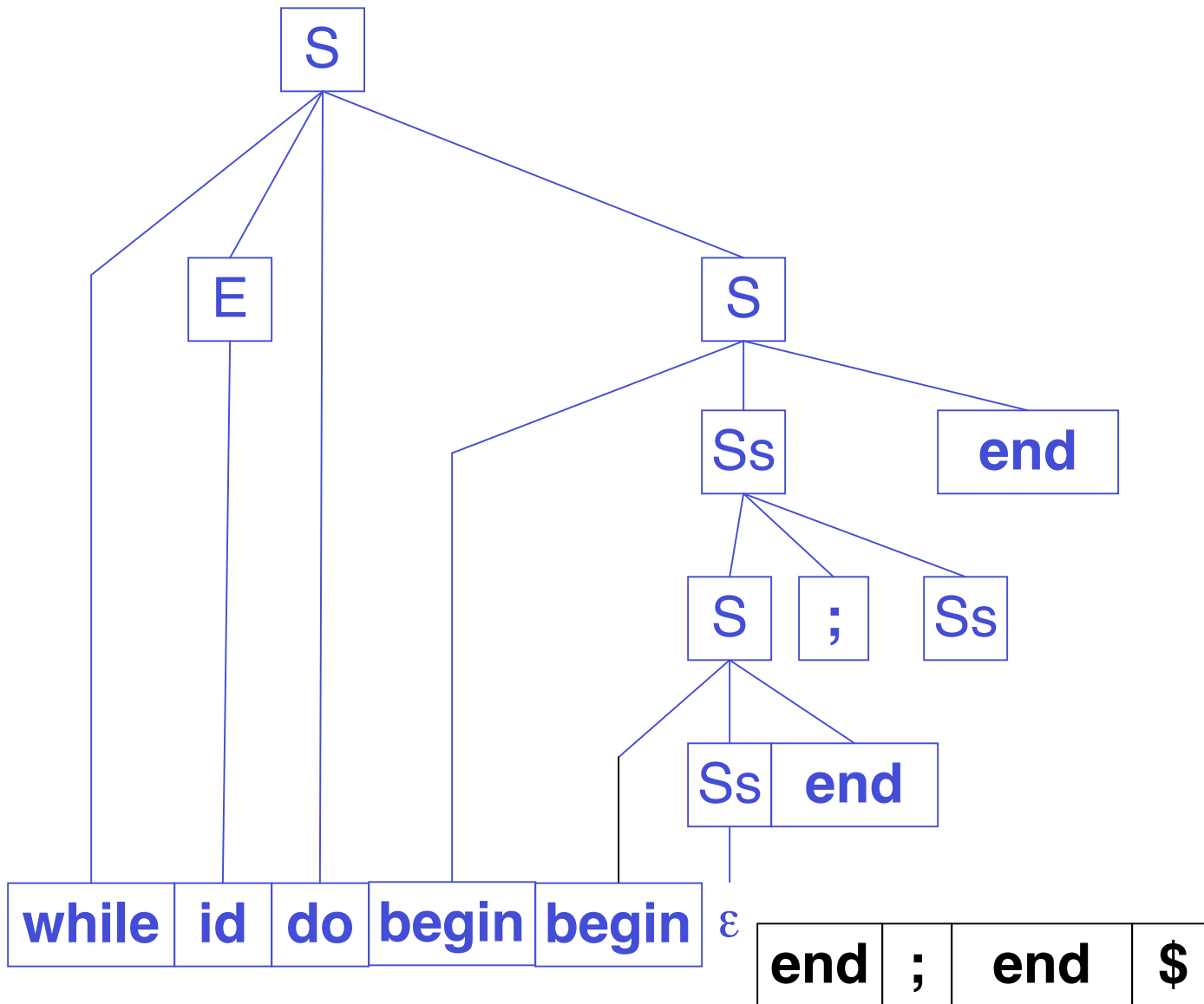


Top of loop

X:

a:

Stack:



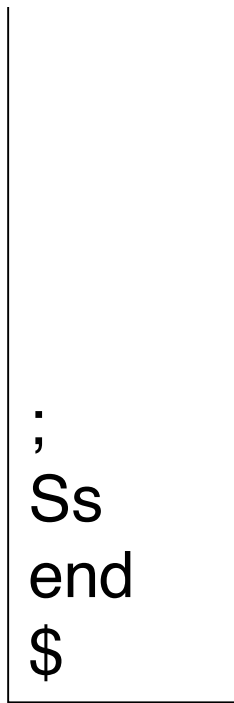
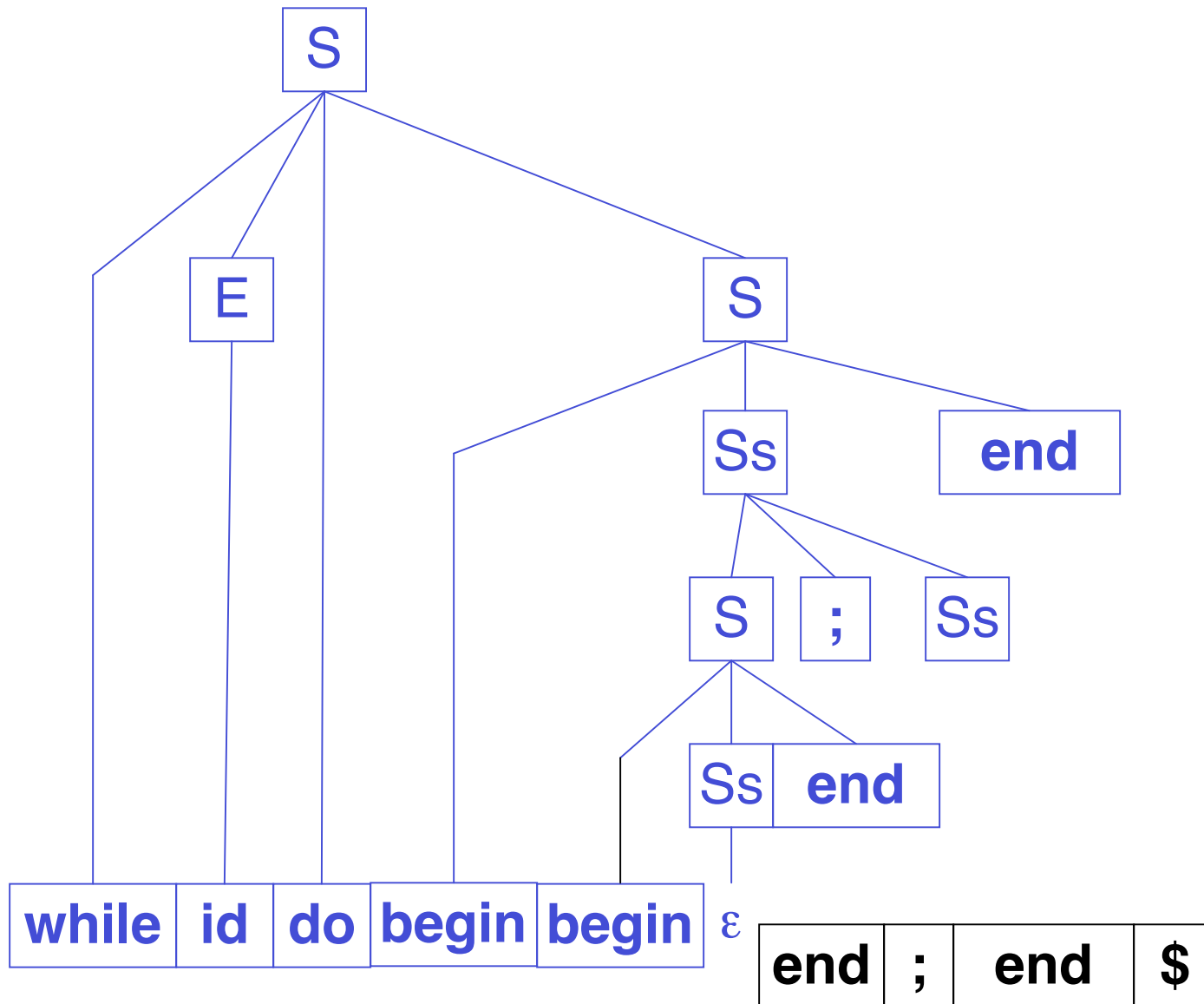
**Action:** Match

Mid loop

X: end

a: end

Stack:

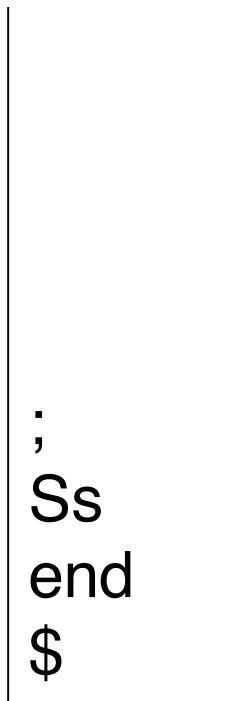
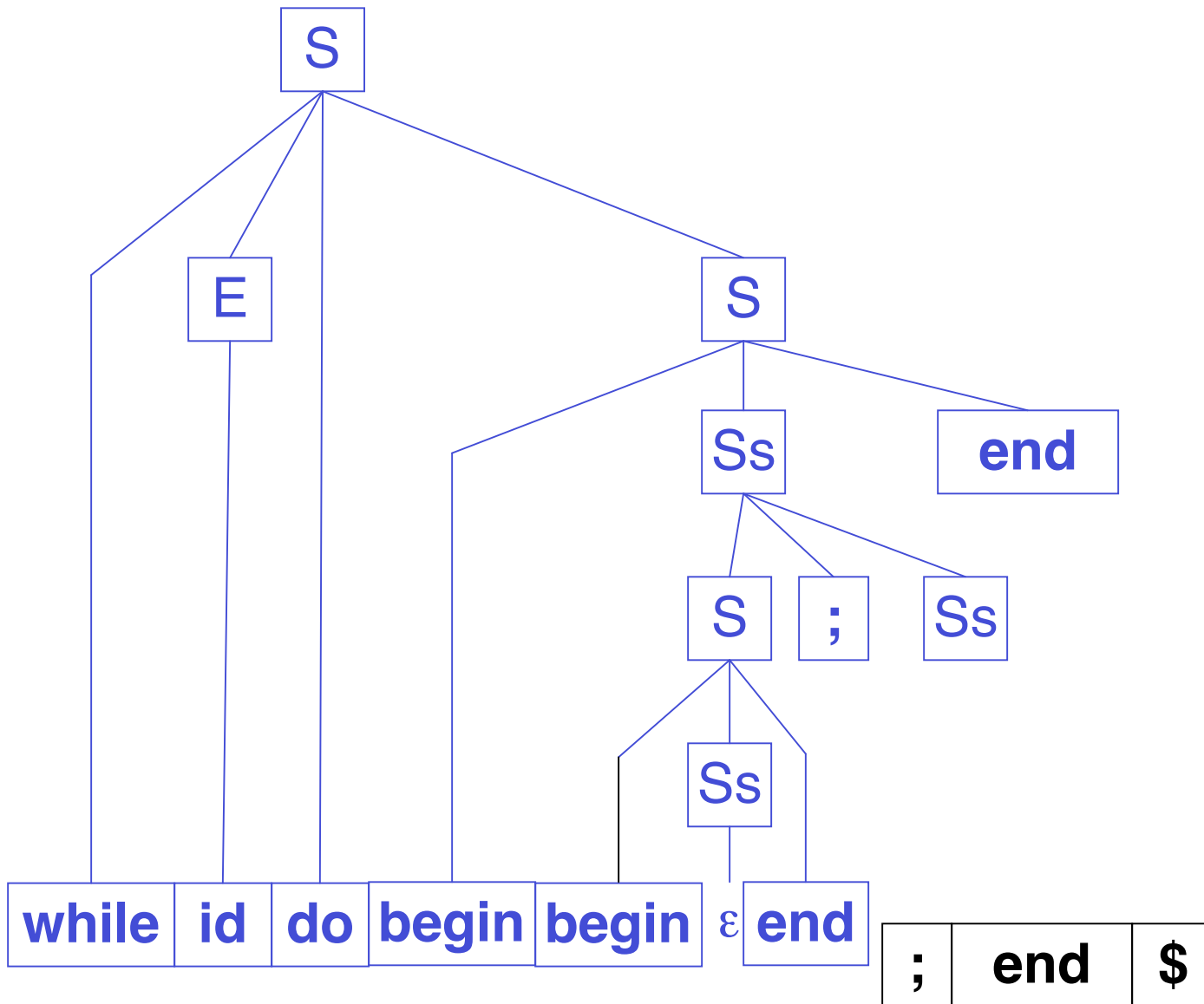


Top of loop

X:

a:

Stack:



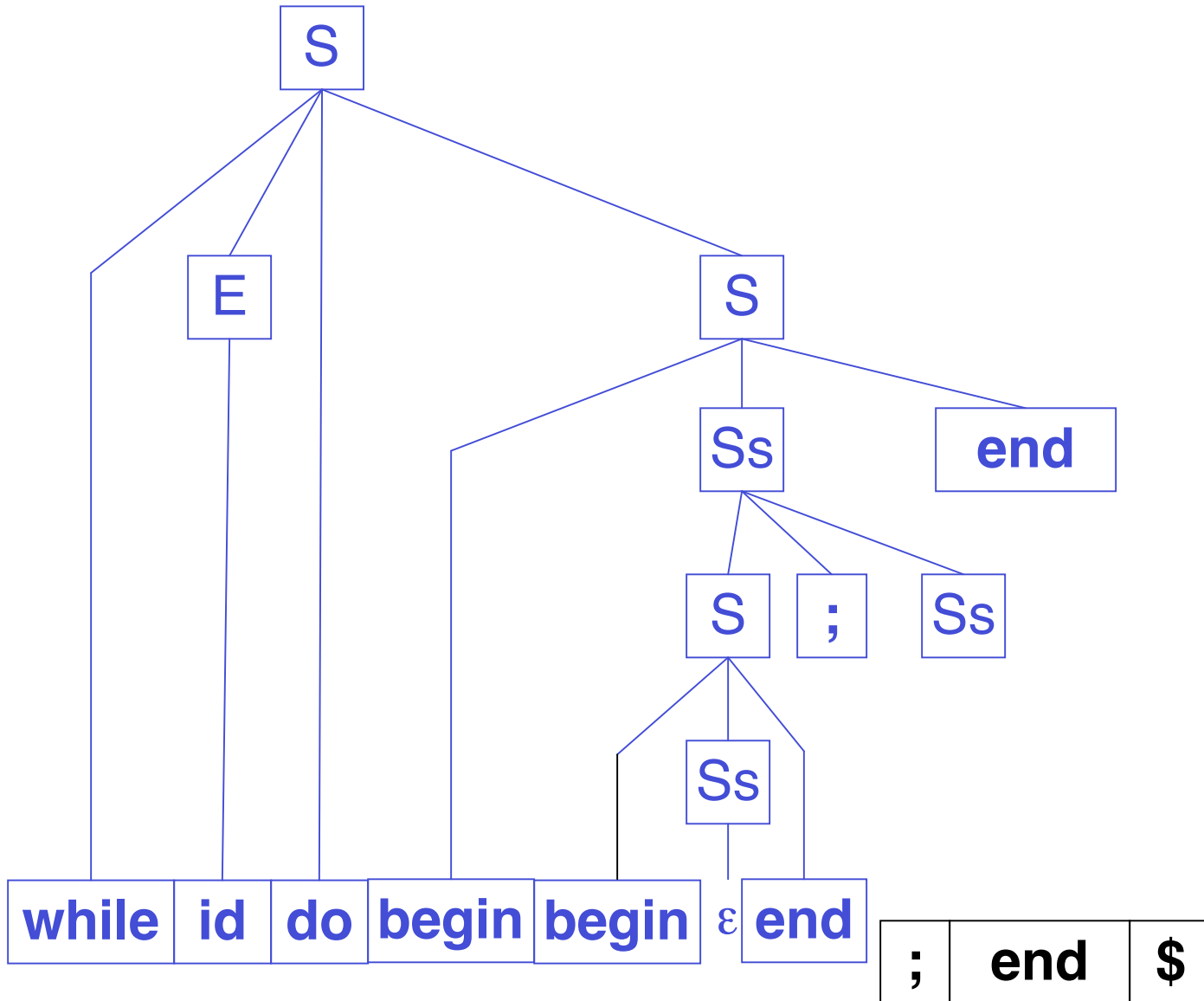
**Action:** Match

Mid loop

X: ;

a: ;

Stack:



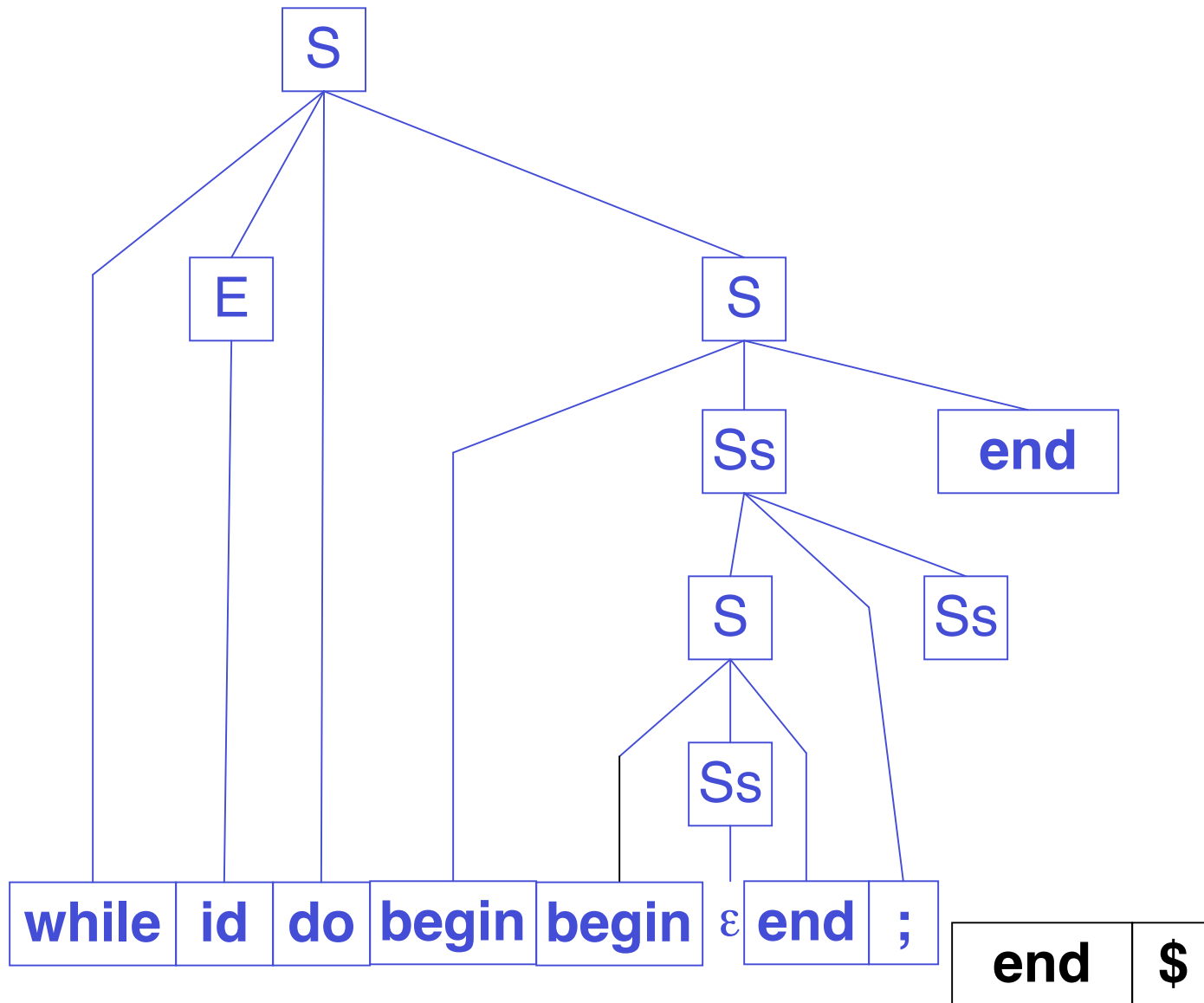


Top of loop

X:

a:

Stack:



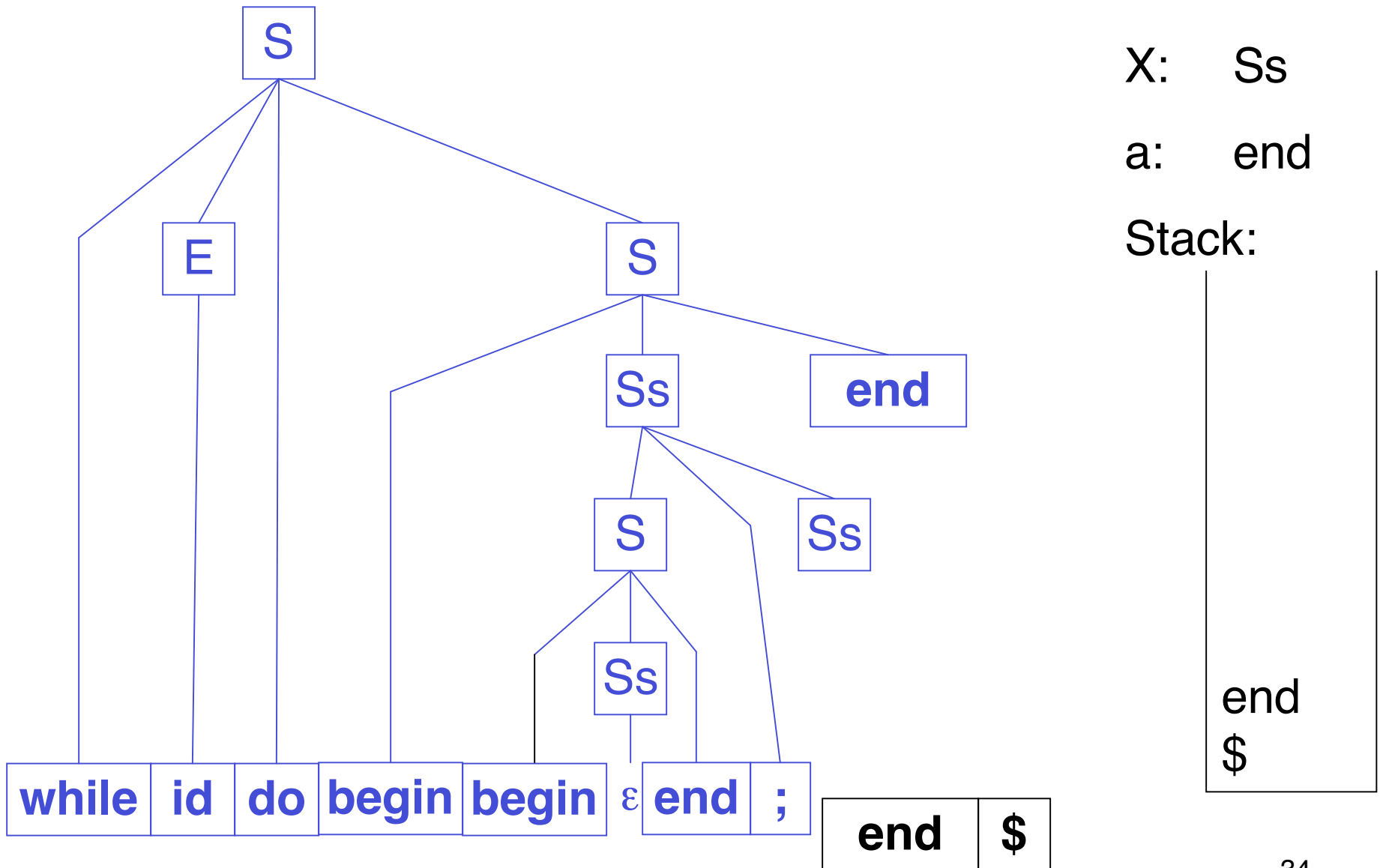
**Action: 5** Ss ::=  $\epsilon$

Mid loop

X: Ss

a: end

Stack:

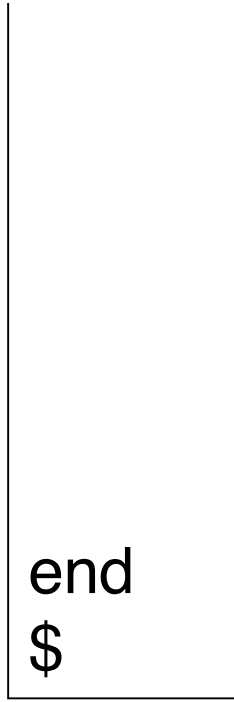
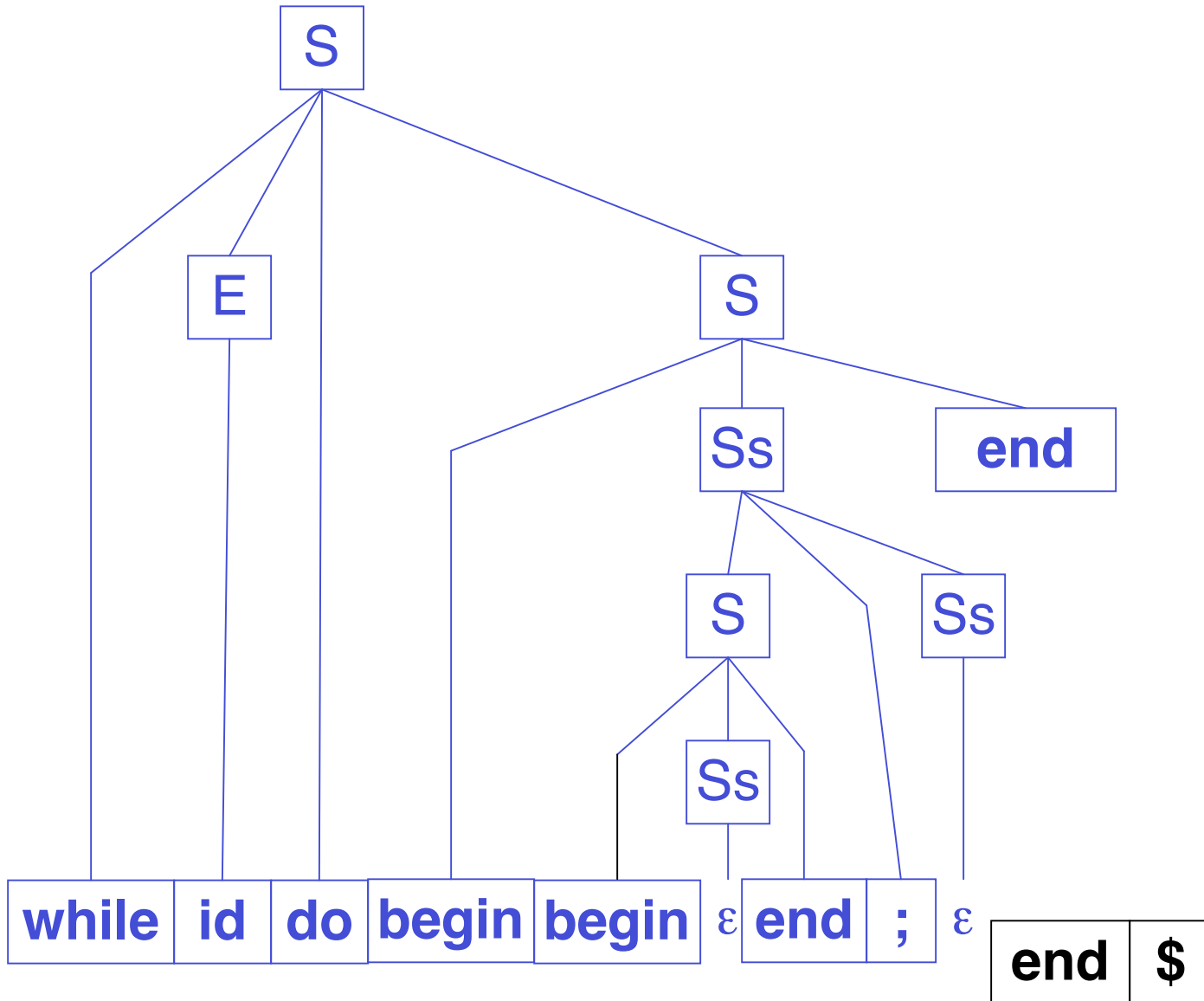


Top of loop

X:

a:

Stack:



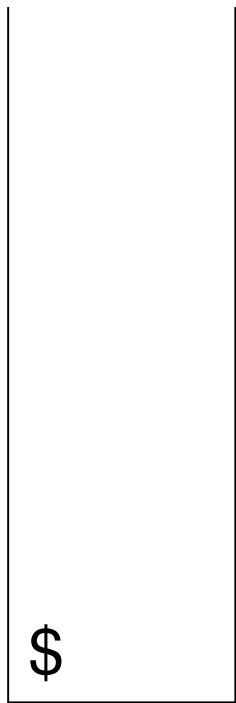
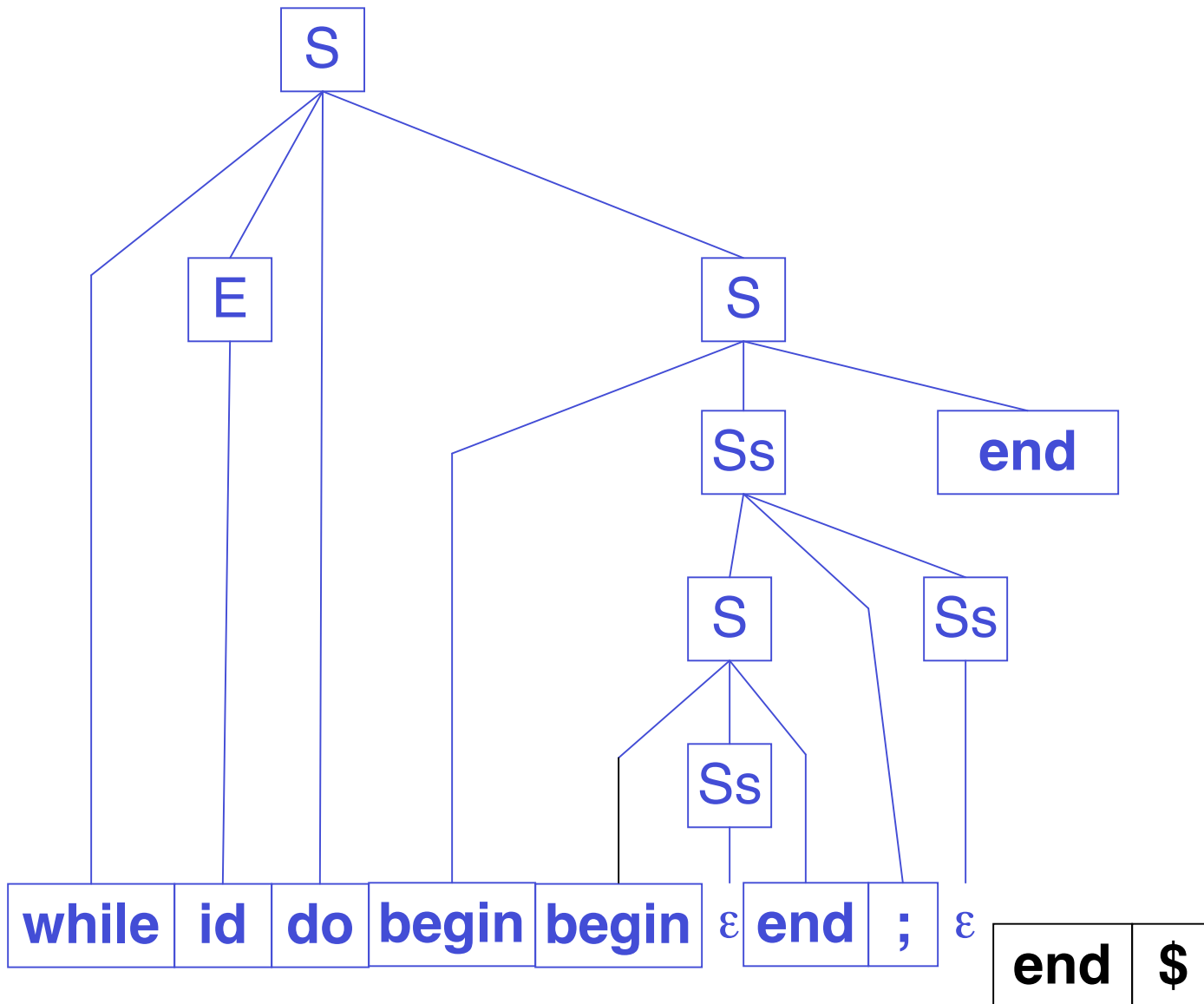
**Action:** Match

Mid loop

X: end

a: end

Stack:

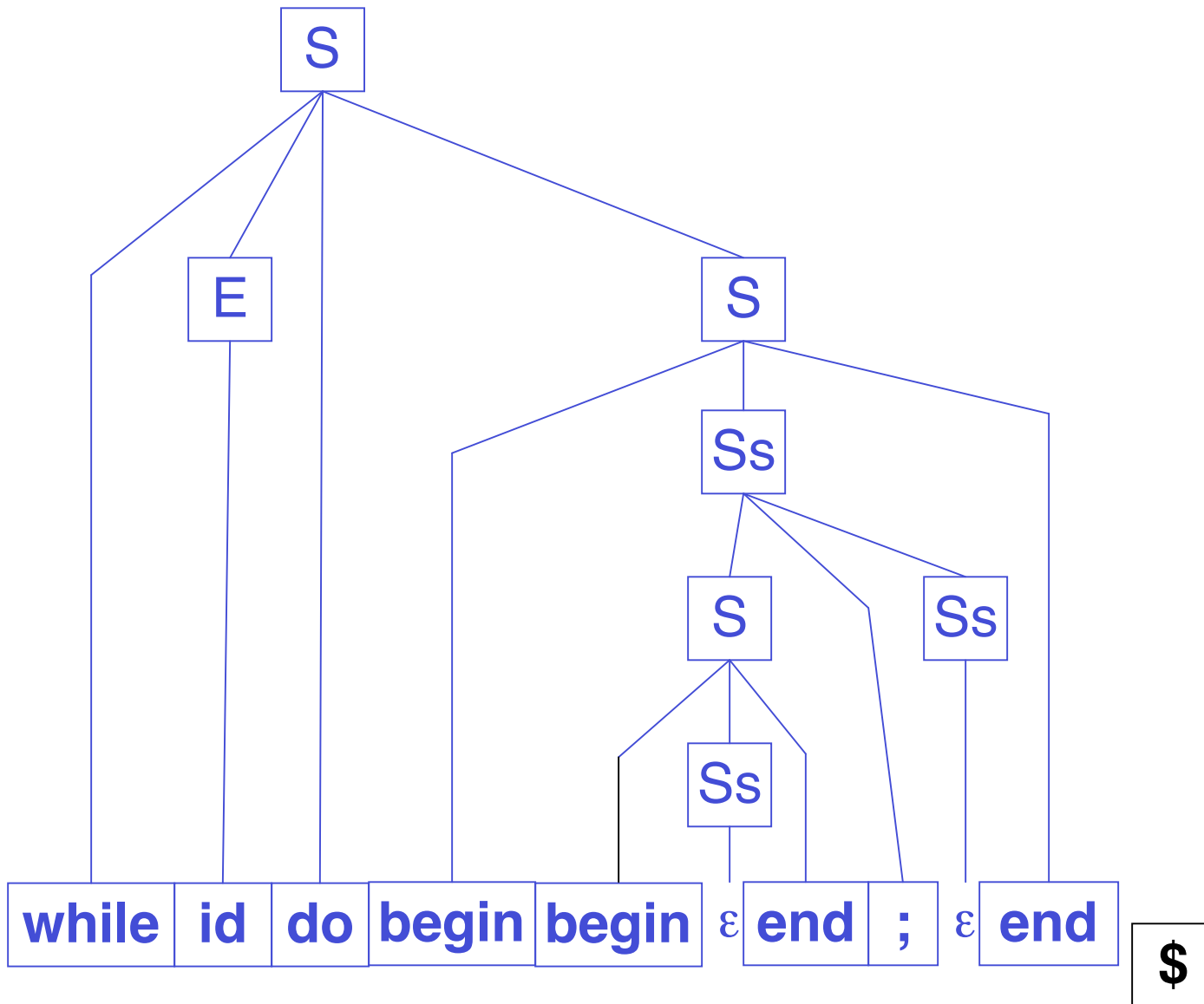


Top of loop

X:

a:

Stack:



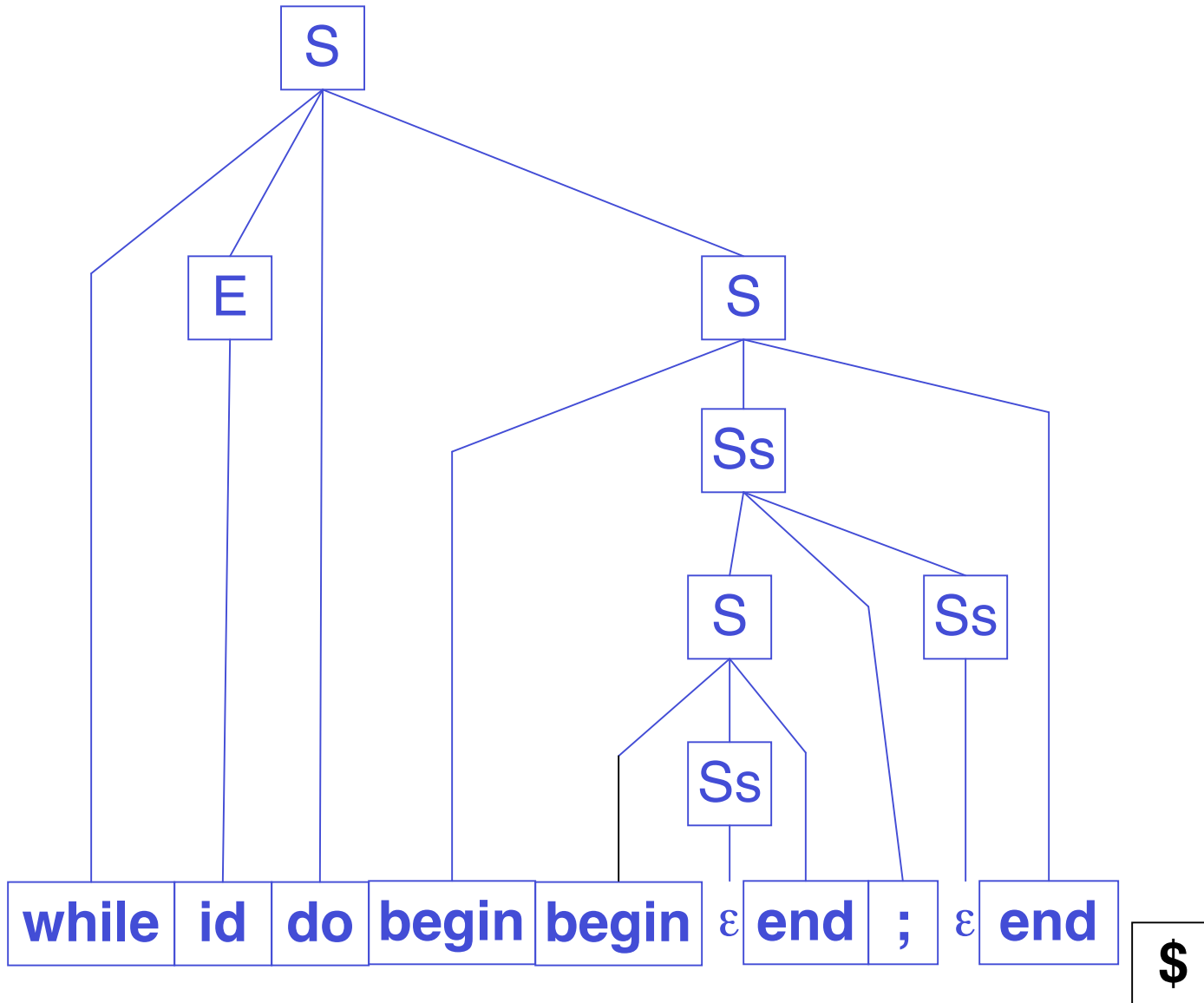
**Action:** Match

Mid loop

X: \$

a: \$

Stack:



Top of loop

