Bottom Up Parsing

TEACHING ASSISTANT: DAVID TRABISH

LR(0) Parsing

- Build the derivation tree from the bottom
- First build the children, then connect to the parent
- Can handle left recursion
    - Which is common in real-world grammars

LR(0) Item

An **LR(0) item** is of the form:

- $N \rightarrow \alpha . \beta$

The **dot** gives us the current location (a local view).

LR(0) Item

An LR(0) item with the dot at the end is called **reduce** item:

- $N \rightarrow \alpha\beta.$

Otherwise, it's a **shift** item:

- $N \rightarrow .\alpha\beta$
- $N \rightarrow \alpha.\beta$

LR(0) Item Closure Set

The **LR(0) closure set** of an LR(0) item i is a set S such that:

- $i \in S$
- If $A \to \alpha . N\beta \in S$ then for each rule $N \to \gamma$:
  - $N \to . \gamma \in S$

LR(0) Item Closure Set

For example, given the following CFG:

- $S \rightarrow E\$$
- $E \rightarrow ID = X$
- $E \rightarrow \{ID\}$
- $X \rightarrow INT$

the closure set of the $S \rightarrow .E\$$ contains:

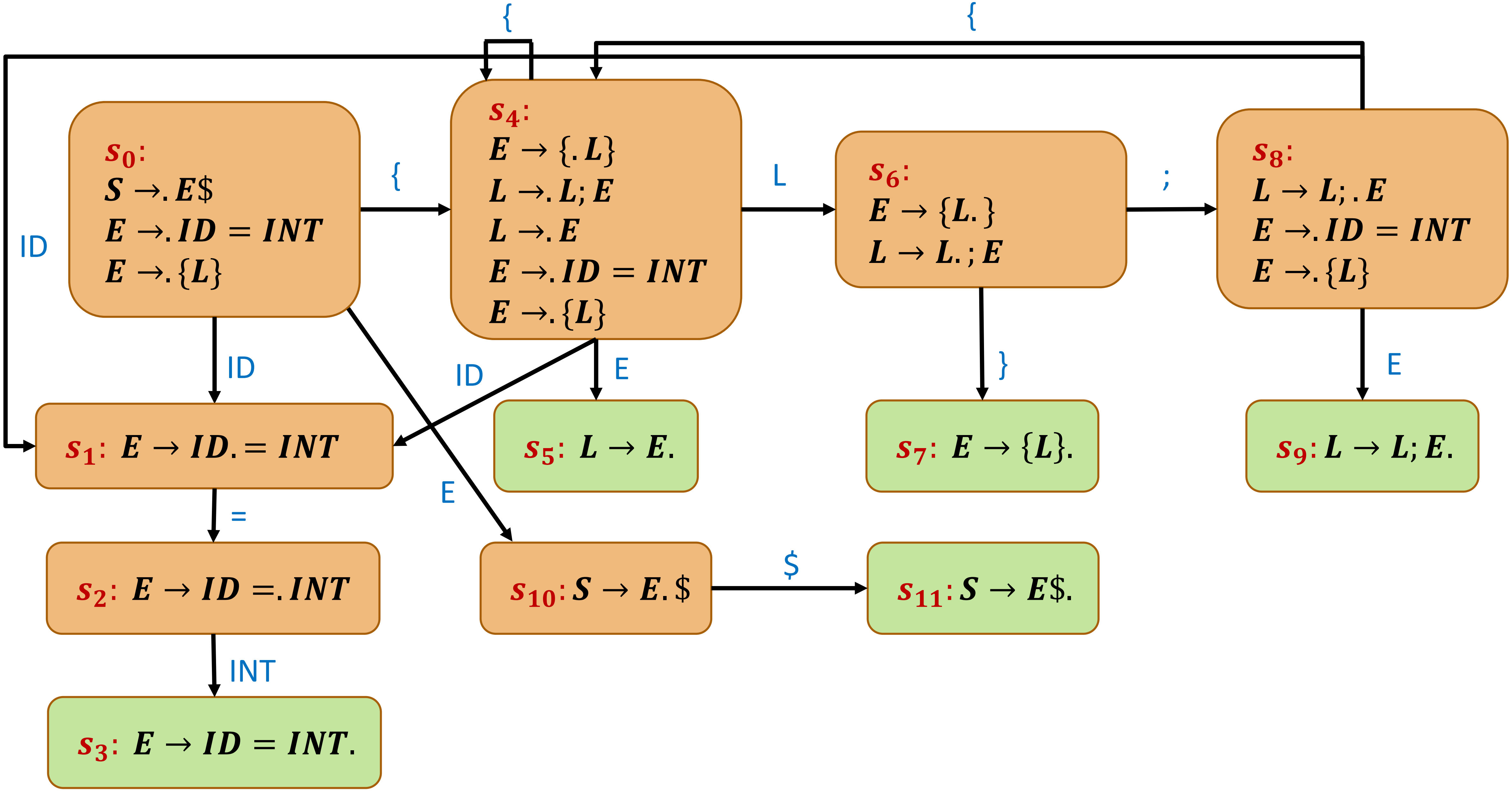
- $S \rightarrow .E\$$
- $E \rightarrow .ID = X$
- $E \rightarrow .\{ID\}$

LR(0) Parsing

Consider the following CFG:

- $S \rightarrow E\$$
- $E \rightarrow ID = INT$
- $E \rightarrow \{L\}$
- $L \rightarrow E$
- $L \rightarrow L; E$

What will be the **transition system** of the LR(0) parser for this CFG?



s_0: $S \to .E\$$ $E \to .ID = INT$ $E \to .\{L\}$
s_1: $E \to ID. = INT$
s_2: $E \to ID =. INT$
s_3: $E \to ID = INT.$
s_4: $E \to \{.L\}$ $L \to .L; E$ $L \to .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_5: $L \to E.$
s_6: $E \to \{L.\}$ $L \to L.; E$
s_7: $E \to \{L\}.$
s_8: $L \to L;.E$ $E \to .ID = INT$ $E \to .\{L\}$
s_9: $L \to L; E.$
s_{10}: $S \to E.\$$
s_{11}: $S \to E\$.$
{
ID
L
;
}
E
=
INT
$

LR(0) Parser

We start with the initial LR(0) item (that comes from the initial rule):

- $S \to .E\$$

The initial state is the ϵ-**closure** of that item, which contains:

$S \to E\$$
$E \to ID = INT$
$E \to \{L\}$
$L \to E$
$L \to L; E$

LR(0) Parser

We start with the initial LR(0) item (that comes from the initial rule):

- $S \rightarrow . E\$$

The initial state is the ϵ-closure of that item, which contains:

- $S \rightarrow . E\$$
- $E \rightarrow . ID = INT$
- $E \rightarrow . \{L\}$

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$

s_0:

$S \rightarrow .E\$$

$E \rightarrow .ID = INT$

$E \rightarrow .\{L\}$

LR(0) Parser

From s_0, if we recognized ID, then the next state will contain:

- $E \rightarrow ID. = INT$

So the next state (the ϵ-closure) contains:

$$S \rightarrow E\$$$
$$E \rightarrow ID = INT$$
$$E \rightarrow \{L\}$$
$$L \rightarrow E$$
$$L \rightarrow L; E$$

LR(0) Parser

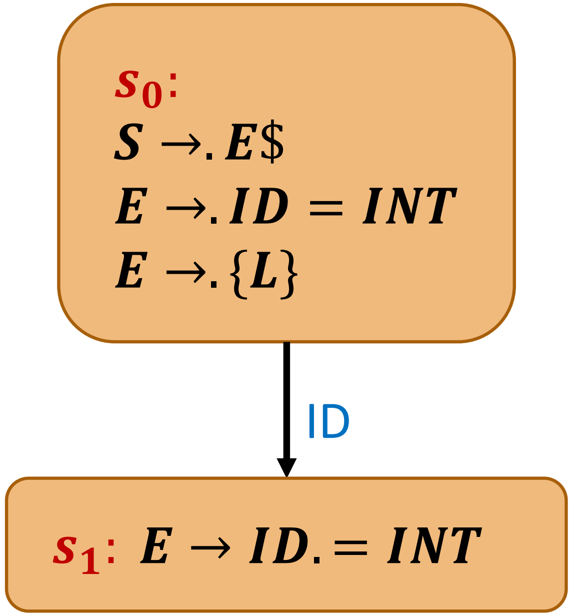
From s_0, if we recognized ID, then the next state will contain:

- $E \rightarrow ID. = INT$

So the next state (the ϵ-closure) contains:

- $E \rightarrow ID. = INT$

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$


s_0:
$S \rightarrow .E\$$
$E \rightarrow .ID = INT$
$E \rightarrow .\{L\}$
ID
s_1: $E \rightarrow ID. = INT$

LR(0) Parser

From s_1, if we recognized $=$, then the next state will contain:

- $E \rightarrow ID =. INT$

So the next state (the ϵ-closure) contains:

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$

LR(0) Parser

From s_1, if we recognized $=$, then the next state will contain:

- $E \rightarrow ID =. INT$

So the next state (the ϵ-closure) contains:

- $E \rightarrow ID =. INT$

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$

s_0:

$S \to .E\$$

$E \to .ID = INT$

$E \to .\{L\}$

$\Big\downarrow \text{ID}$

s_1: $E \to ID. = INT$

$\Big\downarrow =$

s_2: $E \to ID = .INT$

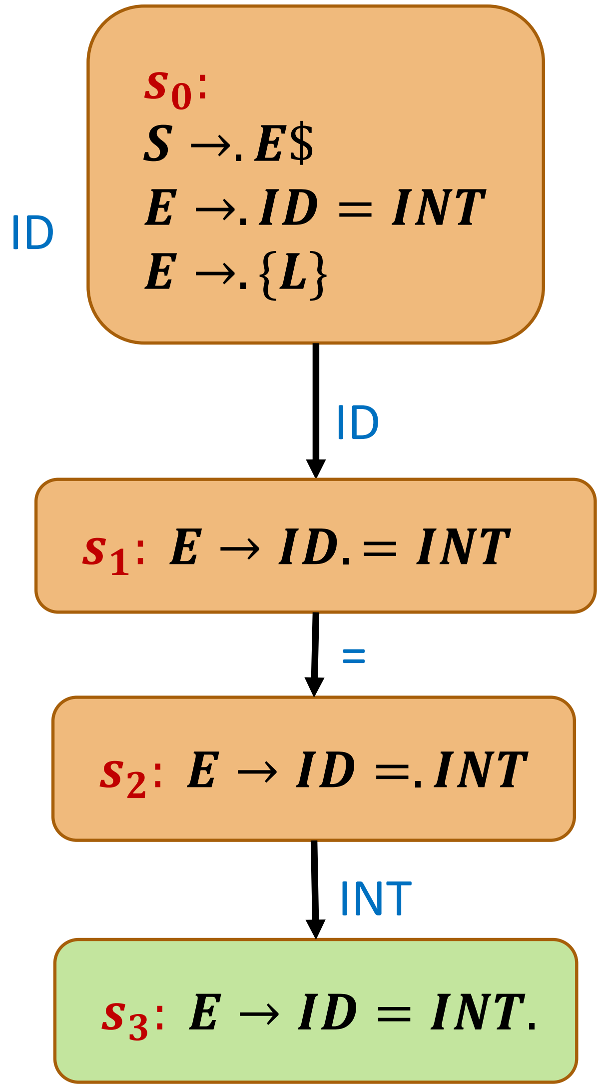
LR(0) Parser

From s_2, if we recognized INT, then the next state will contain:

- $E \rightarrow ID = INT.$

Which is a reduce state.

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$


ID
s_0:
$S \rightarrow .E\$$
$E \rightarrow .ID = INT$
$E \rightarrow .\{L\}$
ID
s_1: $E \rightarrow ID. = INT$
=
s_2: $E \rightarrow ID = .INT$
INT
s_3: $E \rightarrow ID = INT.$

LR(0) Parser

From s_0, if we recognized $\{$, then the next state will contain:

- $E \rightarrow \{.L\}$

So the next state (the ϵ-closure) contains:

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$

LR(0) Parser

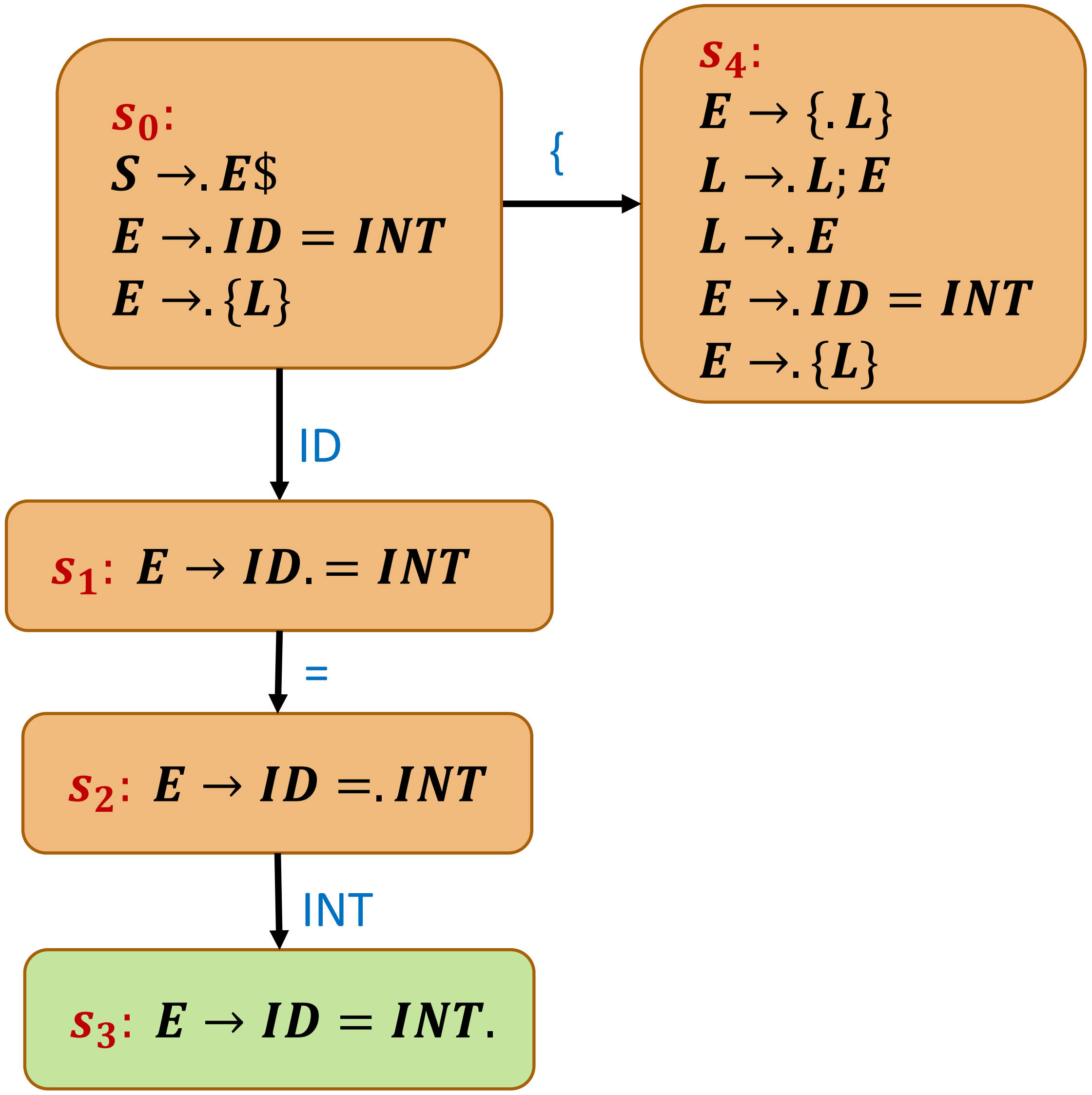
From s_0, if we recognized {, then the next state will contain:

- $E \rightarrow \{.L\}$

So the next state (the ϵ-closure) contains:

- $E \rightarrow \{.L\}$
- $L \rightarrow .L;E$
- $L \rightarrow .E$
- $E \rightarrow .ID = INT$
- $E \rightarrow .\{L\}$

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L;E$


s_0:
$S \to .E\$$
$E \to .ID = INT$
$E \to .\{L\}$
{
s_4:
$E \to \{.L\}$
$L \to .L; E$
$L \to .E$
$E \to .ID = INT$
$E \to .\{L\}$
ID
s_1: $E \to ID. = INT$
=
s_2: $E \to ID =. INT$
INT
s_3: $E \to ID = INT.$

LR(0) Parser

From s_4, if we recognized $\{$, then the next state will contain:

- $E \to \{.L\}$

So the next state (the ϵ-closure) contains:

$S \to E\$$

$E \to ID = INT$

$E \to \{L\}$

$L \to E$

$L \to L; E$

LR(0) Parser

From s_4, if we recognized {, then the next state will contain:

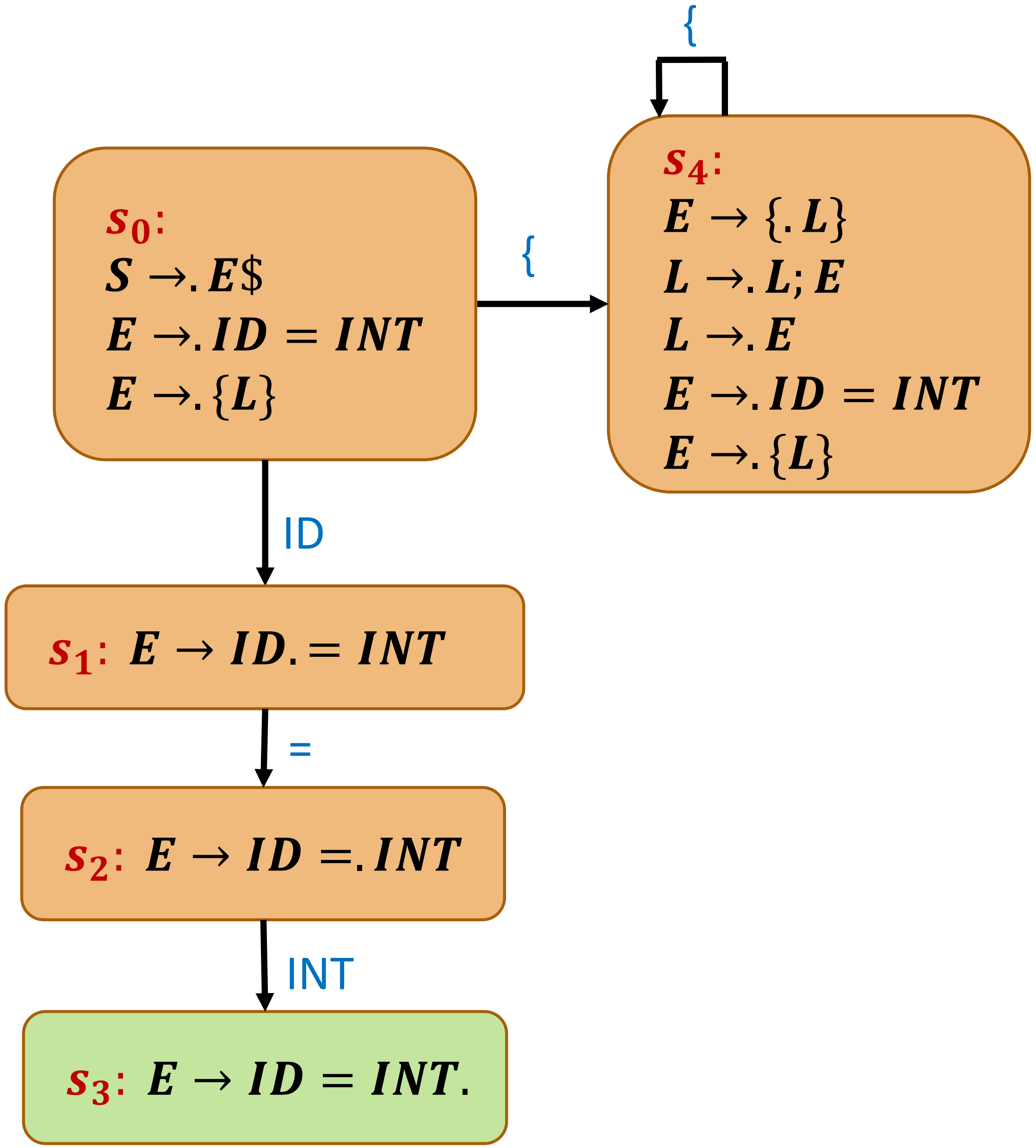
- $E \rightarrow \{.L\}$

So the next state (the ϵ-closure) contains:

- $E \rightarrow \{.L\}$
- $L \rightarrow .L; E$
- $L \rightarrow .E$
- $E \rightarrow .ID = INT$
- $E \rightarrow .\{L\}$

which was already computed: s_4

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$


s_0:
$S \rightarrow .E\$$
$E \rightarrow .ID = INT$
$E \rightarrow .\{L\}$
{
s_4:
$E \rightarrow \{.L\}$
$L \rightarrow .L; E$
$L \rightarrow .E$
$E \rightarrow .ID = INT$
$E \rightarrow .\{L\}$
{
ID
s_1: $E \rightarrow ID. = INT$
=
s_2: $E \rightarrow ID =. INT$
INT
s_3: $E \rightarrow ID = INT.$

LR(0) Parser

From s_4, if we recognized ID, then the next state will contain:

- $E \rightarrow ID. = INT$

So the next state (the ϵ-closure) contains:

$S \rightarrow E\$$

$E \rightarrow ID = INT$

$E \rightarrow \{L\}$

$L \rightarrow E$

$L \rightarrow L; E$

LR(0) Parser

From s_4, if we recognized ID, then the next state will contain:

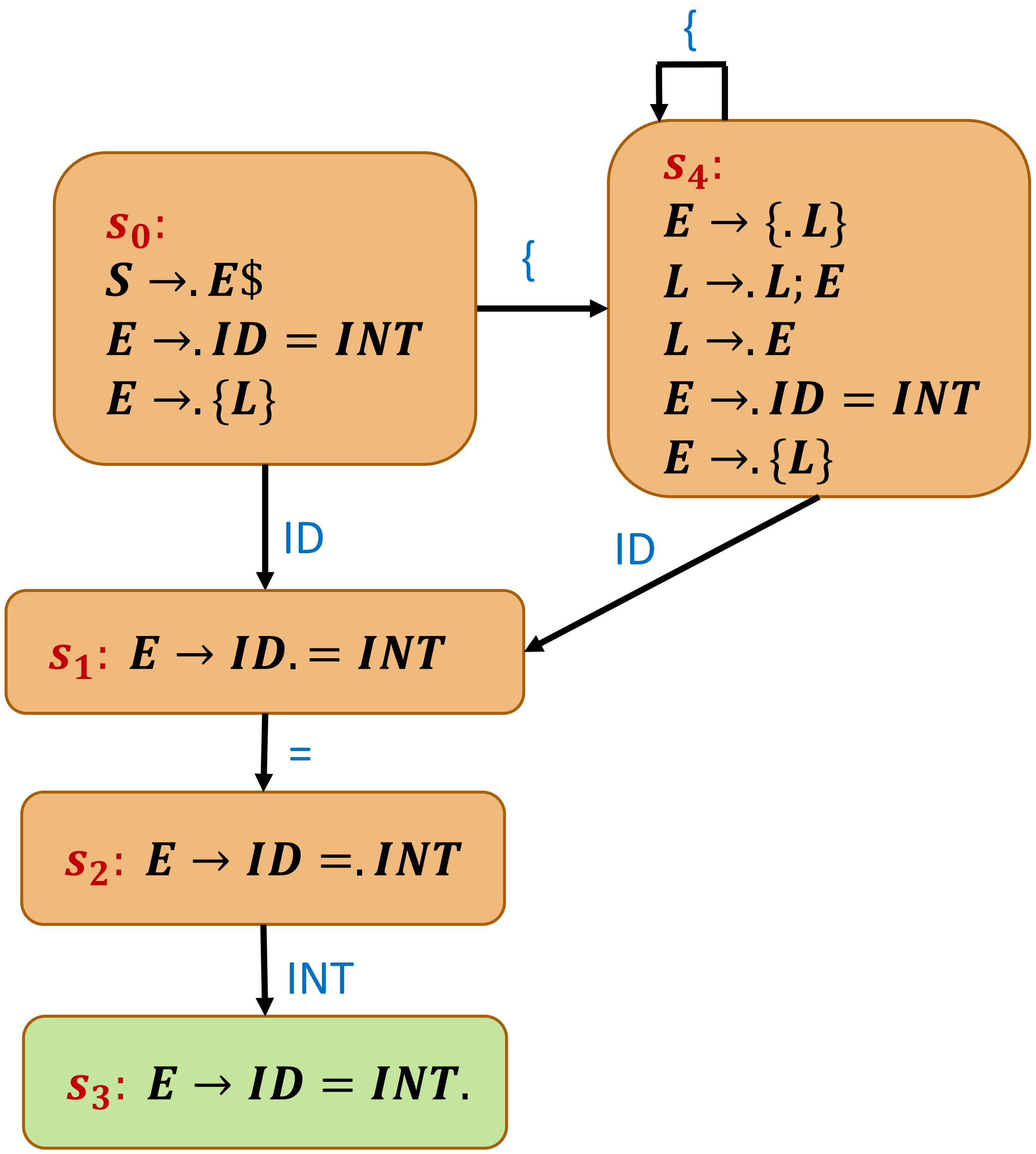
- $E \rightarrow ID. = INT$

So the next state (the ϵ-closure) contains:

- $E \rightarrow ID. = INT$

which was already computed: s_1

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$


s_0:
$S \to .E\$$
$E \to .ID = INT$
$E \to .\{L\}$
s_4:
$E \to \{.L\}$
$L \to .L;E$
$L \to .E$
$E \to .ID = INT$
$E \to .\{L\}$
{
{
ID
ID
s_1: $E \to ID. = INT$
=
s_2: $E \to ID =. INT$
INT
s_3: $E \to ID = INT.$

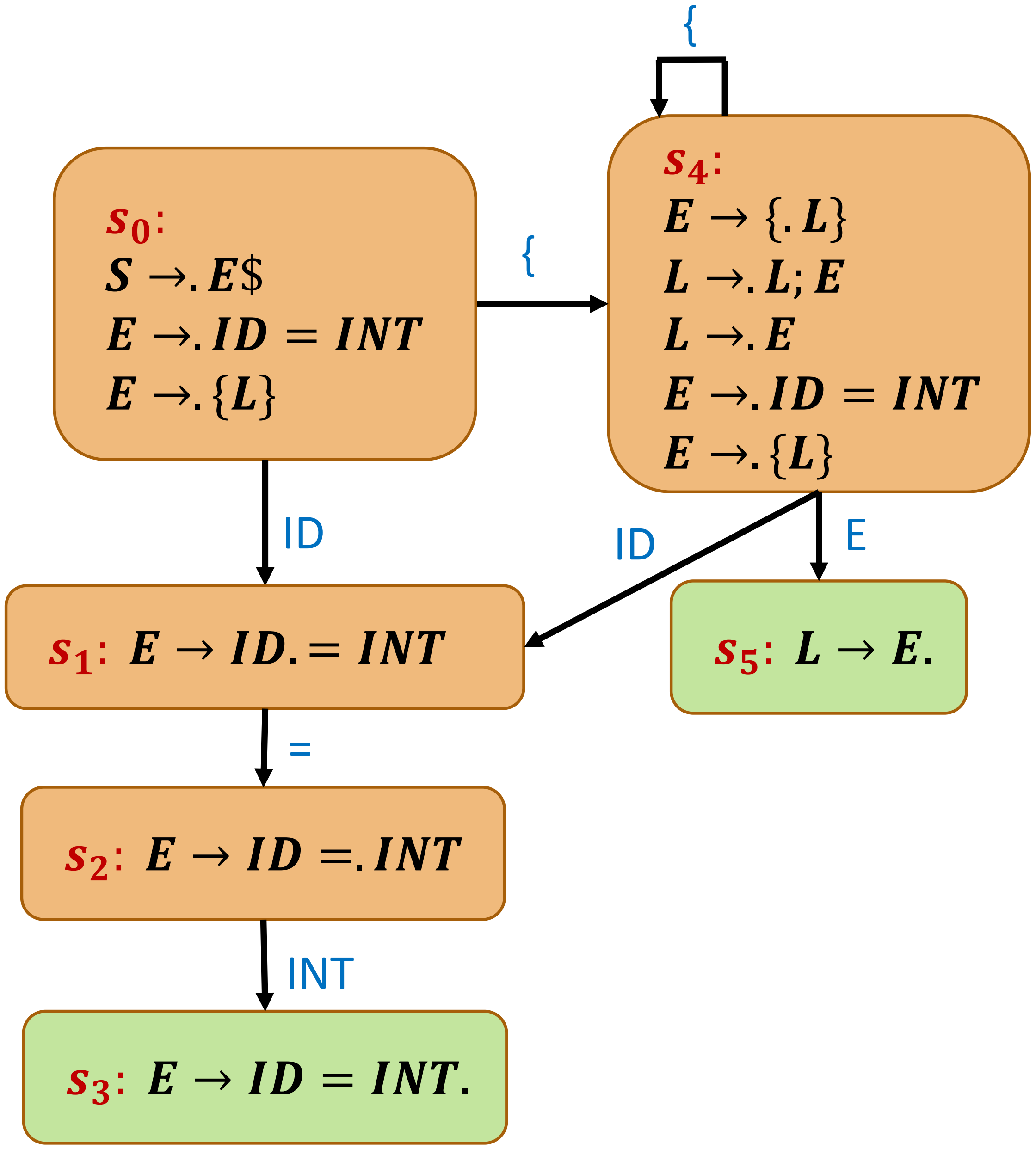
LR(0) Parser

From s_4, if we recognized E, then the next state will contain:

- $L \rightarrow E.$

which is a reduce state.

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$



s_0:
$S \to .E\$$
$E \to .ID = INT$
$E \to .\{L\}$
{
s_4:
$E \to \{.L\}$
$L \to .L;E$
$L \to .E$
$E \to .ID = INT$
$E \to .\{L\}$
{
ID
ID
E
s_1: $E \to ID. = INT$
s_5: $L \to E.$
=
s_2: $E \to ID =.INT$
INT
s_3: $E \to ID = INT.$

LR(0) Parser

From s_4, if we recognized L, then the next state will contain:

- $E \rightarrow \{L.\}$
- $L \rightarrow L.;E$

So the next state (the ϵ-closure) contains:

$$S \rightarrow E\$$$
$$E \rightarrow ID = INT$$
$$E \rightarrow \{L\}$$
$$L \rightarrow E$$
$$L \rightarrow L;E$$

LR(0) Parser

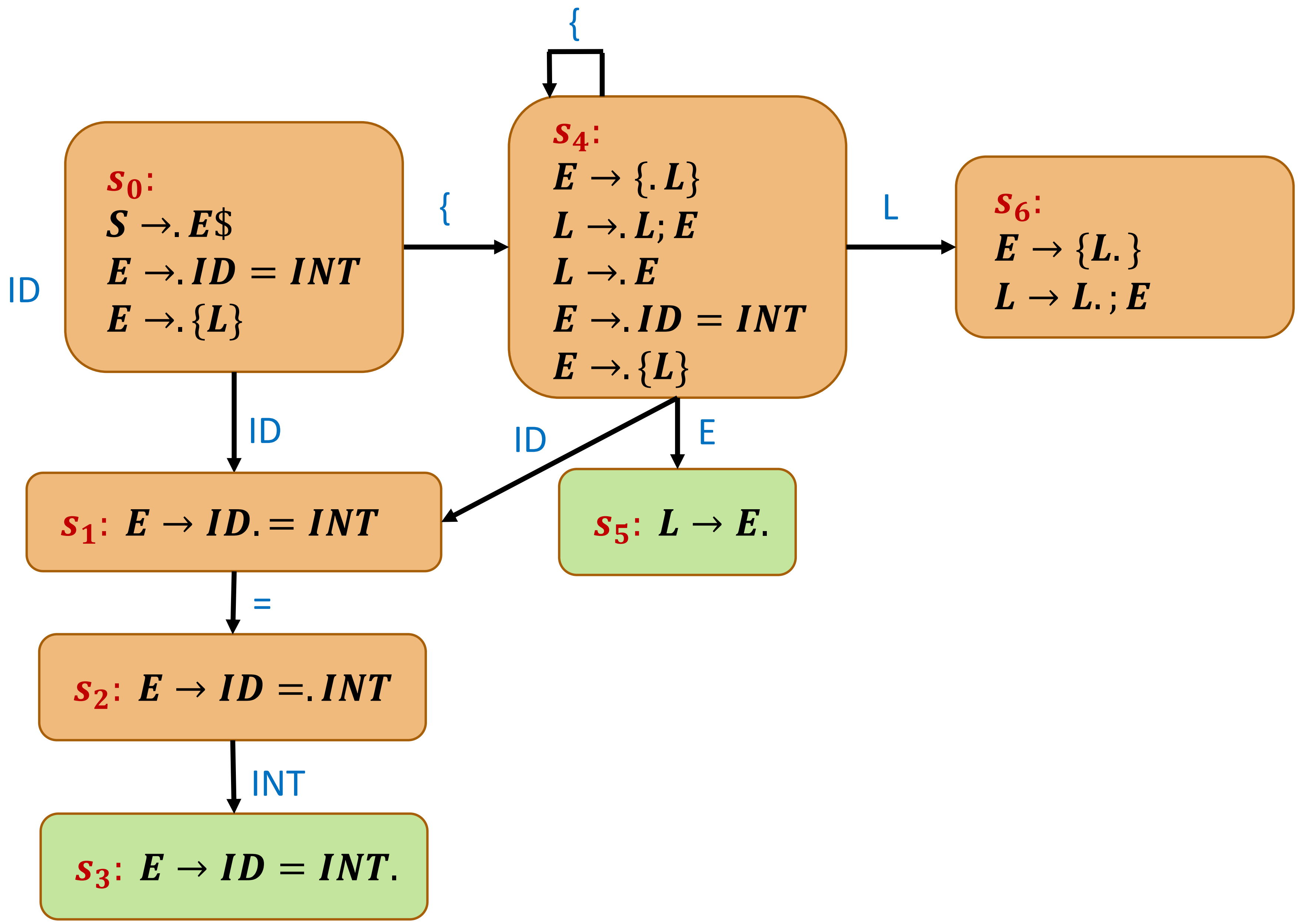
From s_4, if we recognized L, then the next state will contain:

- $E \rightarrow \{L.\}$
- $L \rightarrow L.;E$

So the next state (the ϵ-closure) contains:

- $E \rightarrow \{L.\}$
- $L \rightarrow L.;E$

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L;E$



{
s_0:
$S \rightarrow .E\$$
$E \rightarrow .ID = INT$
$E \rightarrow .\{L\}$
ID
{
s_4:
$E \rightarrow \{.L\}$
$L \rightarrow .L;E$
$L \rightarrow .E$
$E \rightarrow .ID = INT$
$E \rightarrow .\{L\}$
L
s_6:
$E \rightarrow \{L.\}$
$L \rightarrow L.;E$
ID
ID
E
s_1: $E \rightarrow ID. = INT$
s_5: $L \rightarrow E.$
=
s_2: $E \rightarrow ID = .INT$
INT
s_3: $E \rightarrow ID = INT.$

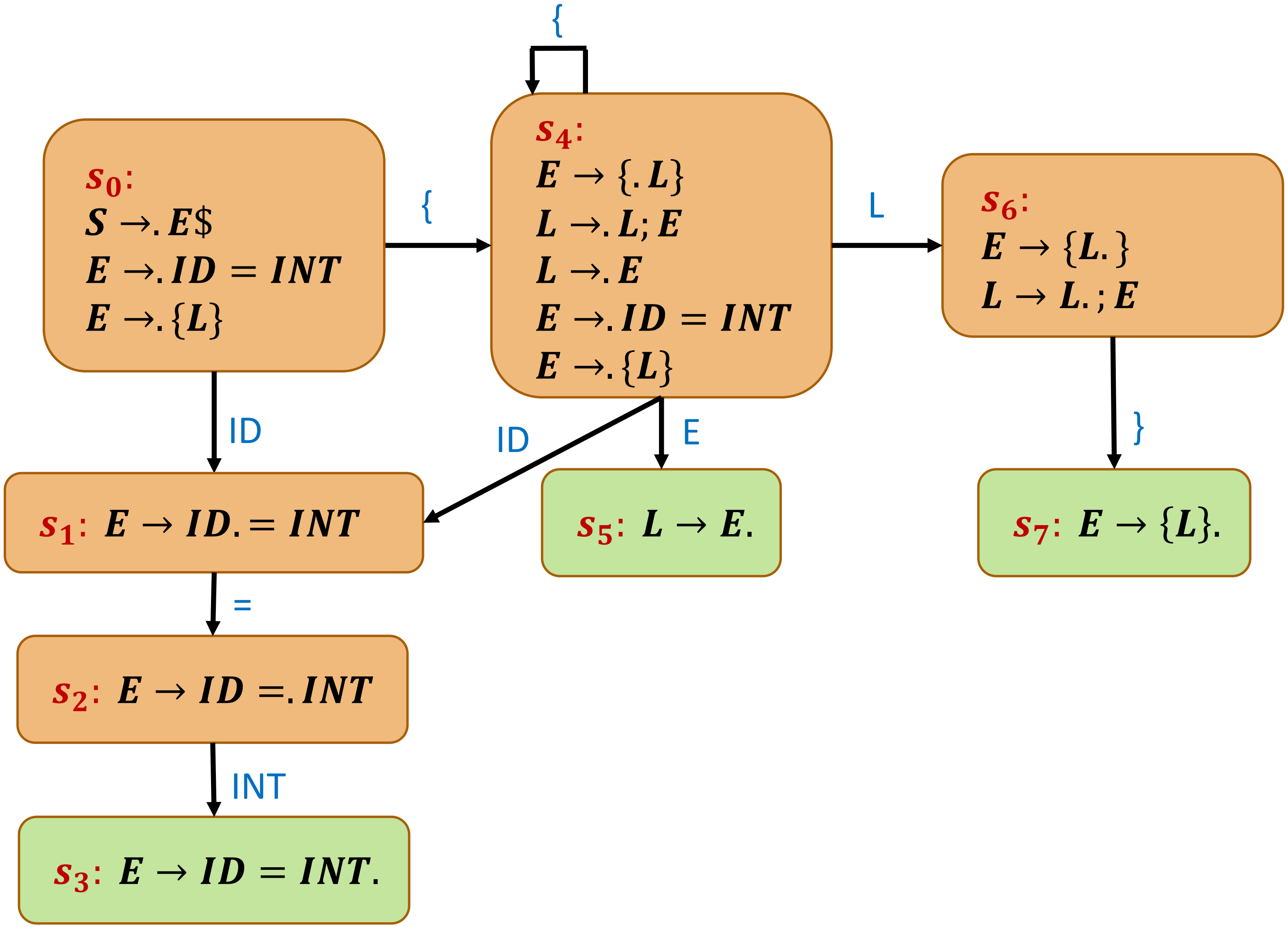
LR(0) Parser

From s_6, if we recognized $\}$, then the next state will contain:

- $E \rightarrow \{L\}.$

Which is a reduce state.

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$


s_0:
$S \rightarrow .E\$$
$E \rightarrow .ID = INT$
$E \rightarrow .\{L\}$
{
s_4:
$E \rightarrow \{.L\}$
$L \rightarrow .L; E$
$L \rightarrow .E$
$E \rightarrow .ID = INT$
$E \rightarrow .\{L\}$
{
L
s_6:
$E \rightarrow \{L.\}$
$L \rightarrow L.; E$
ID
ID
E
}
s_1: $E \rightarrow ID. = INT$
s_5: $L \rightarrow E.$
s_7: $E \rightarrow \{L\}.$
=
s_2: $E \rightarrow ID = .INT$
INT
s_3: $E \rightarrow ID = INT.$

LR(0) Parser

From s_6, if we recognized ;, then the next state will contain:

- $L \rightarrow L;.E$

So the next state (the ϵ-closure) contains:

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$

LR(0) Parser

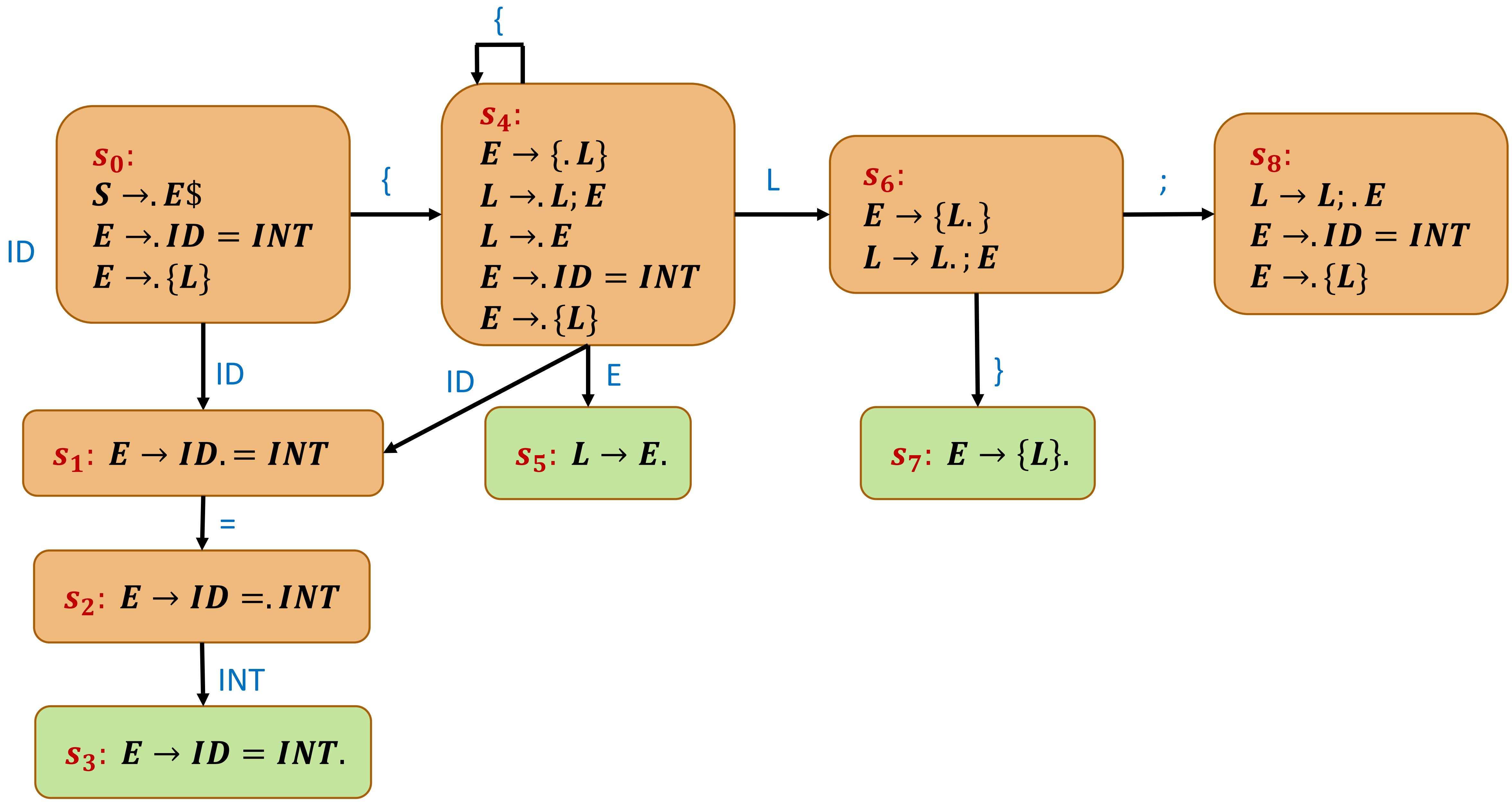
From s_6, if we recognized ;, then the next state will contain:

- $L \rightarrow L; . E$

So the next state (the ϵ-closure) contains:

- $L \rightarrow L; . E$
- $E \rightarrow . ID = INT$
- $E \rightarrow . \{L\}$

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$



s_0: $S \to .E\$$ $E \to .ID = INT$ $E \to .\{L\}$
s_1: $E \to ID. = INT$
s_2: $E \to ID =. INT$
s_3: $E \to ID = INT.$
s_4: $E \to \{.L\}$ $L \to .L; E$ $L \to .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_5: $L \to E.$
s_6: $E \to \{L.\}$ $L \to L.; E$
s_7: $E \to \{L\}.$
s_8: $L \to L; .E$ $E \to .ID = INT$ $E \to .\{L\}$
ID
{
L
;
=
INT
E
}

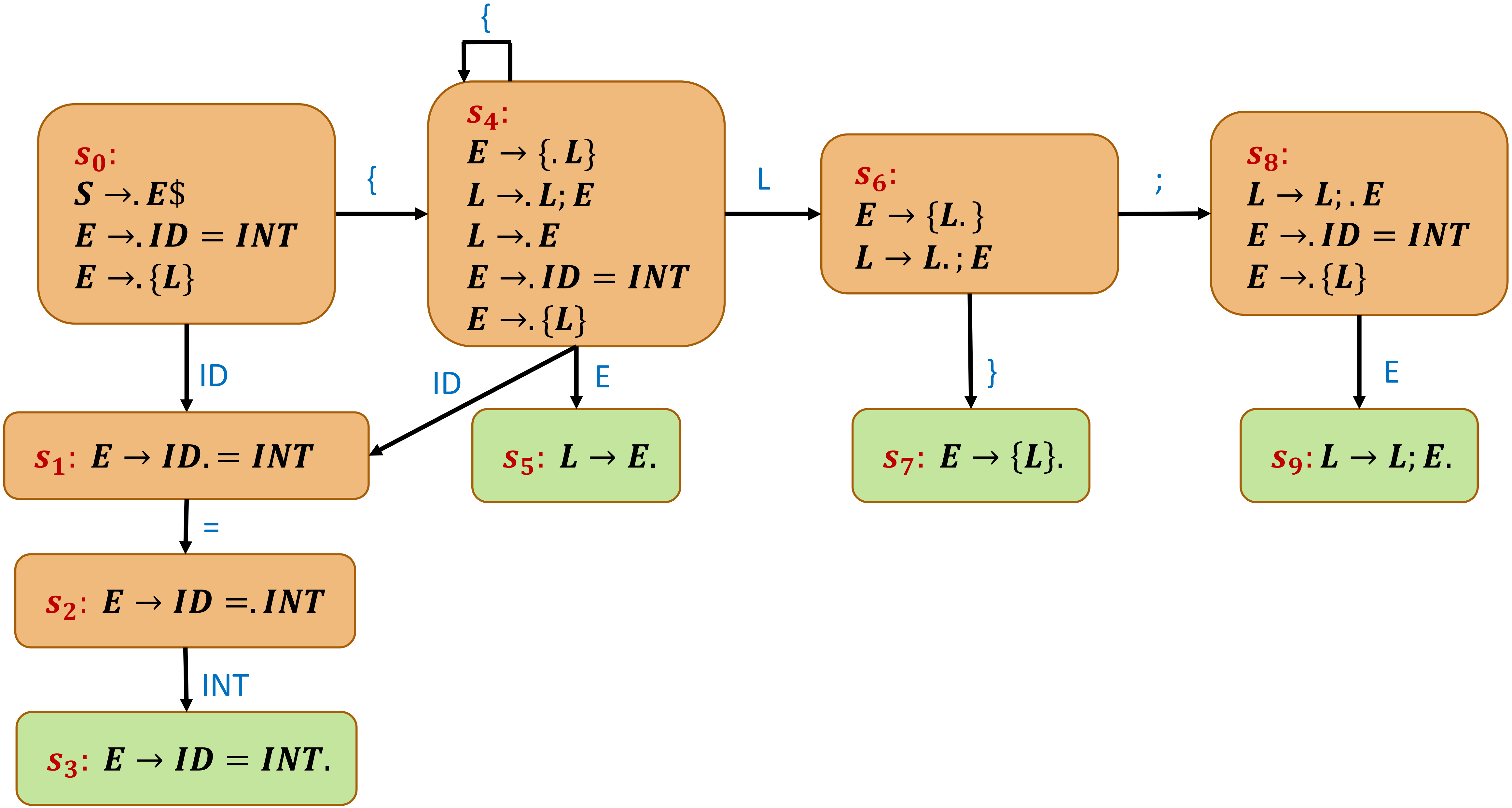
LR(0) Parser

From s_8, if we recognized E, then the next state will contain:

- $E \to L; E.$

which is a reduce state.

$S \to E\$$
$E \to ID = INT$
$E \to \{L\}$
$L \to E$
$L \to L; E$


s_0: $S \to .E\$$ $E \to .ID = INT$ $E \to .\{L\}$
{
s_4: $E \to \{.L\}$ $L \to .L;E$ $L \to .E$ $E \to .ID = INT$ $E \to .\{L\}$
{
L
s_6: $E \to \{L.\}$ $L \to L.;E$
;
s_8: $L \to L;.E$ $E \to .ID = INT$ $E \to .\{L\}$
ID
ID
E
}
E
s_1: $E \to ID. = INT$
s_5: $L \to E.$
s_7: $E \to \{L\}.$
s_9: $L \to L;E.$
=
s_2: $E \to ID =.INT$
INT
s_3: $E \to ID = INT.$

LR(0) Parser

From s_8, if we recognized {, then the next state will contain:

- $E \rightarrow \{.L\}$

So the next state (the ϵ-closure) contains:

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$

LR(0) Parser

From s_8, if we recognized $\{$, then the next state will contain:

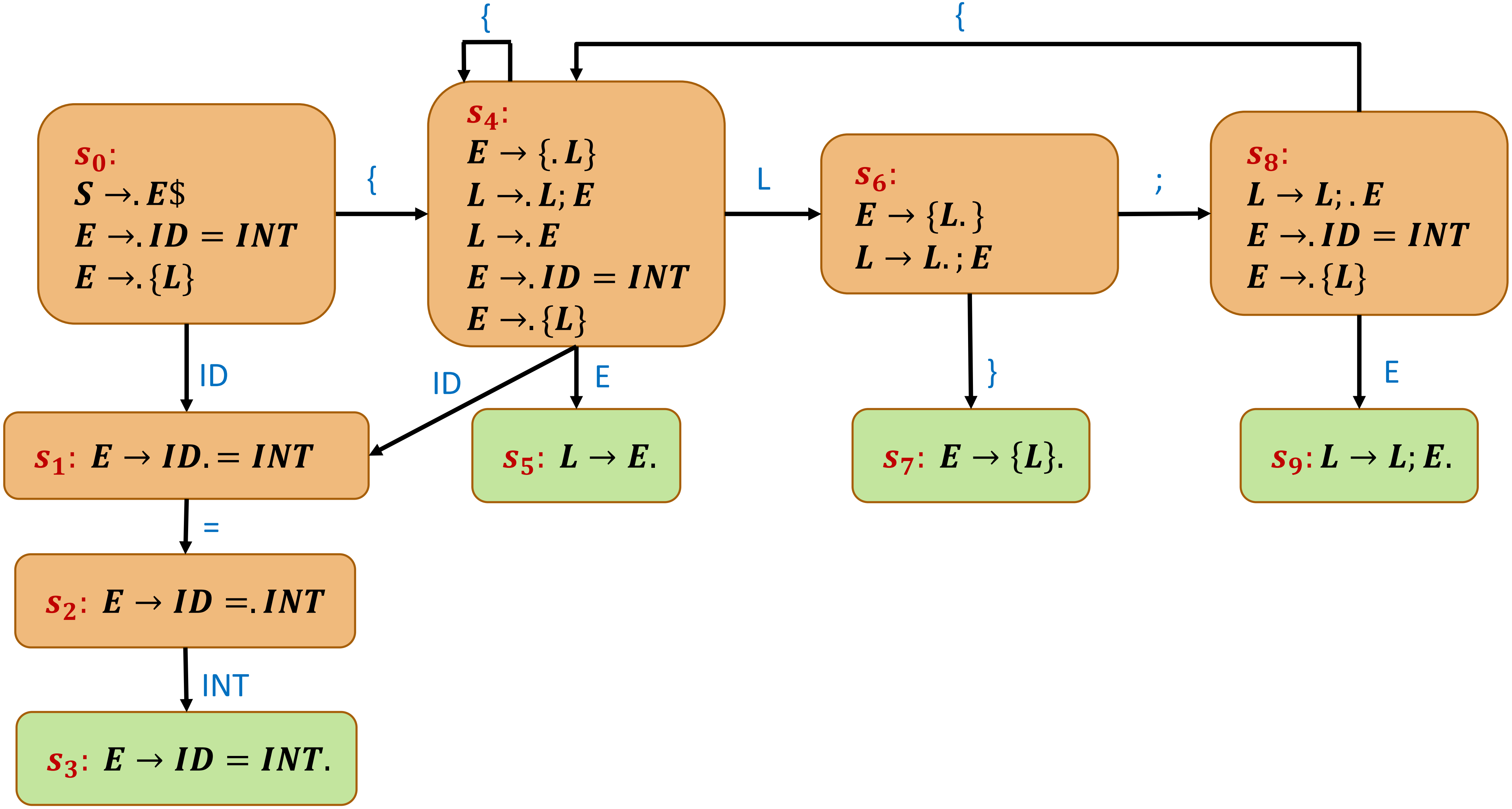
- $E \rightarrow \{.L\}$

So the next state (the ϵ-closure) contains:

- $E \rightarrow \{.L\}$
- $L \rightarrow .L; E$
- $L \rightarrow .E$
- $E \rightarrow .ID = INT$
- $E \rightarrow .\{L\}$

which was already computed: s_4

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$



s_0:
$S \to .E\$$
$E \to .ID = INT$
$E \to .\{L\}$
s_1: $E \to ID. = INT$
s_2: $E \to ID =.INT$
s_3: $E \to ID = INT.$
s_4:
$E \to \{.L\}$
$L \to .L; E$
$L \to .E$
$E \to .ID = INT$
$E \to .\{L\}$
s_5: $L \to E.$
s_6:
$E \to \{L.\}$
$L \to L.; E$
s_7: $E \to \{L\}.$
s_8:
$L \to L; .E$
$E \to .ID = INT$
$E \to .\{L\}$
s_9: $L \to L; E.$
{
{
{
L
;
ID
ID
E
}
E
=
INT

LR(0) Parser

From s_8, if we recognized ID, then the next state will contain:

- $E \rightarrow ID. = INT$

So the next state (the ϵ-closure) contains:

$$S \rightarrow E\$$$
$$E \rightarrow ID = INT$$
$$E \rightarrow \{L\}$$
$$L \rightarrow E$$
$$L \rightarrow L; E$$

LR(0) Parser

From s_8, if we recognized ID, then the next state will contain:

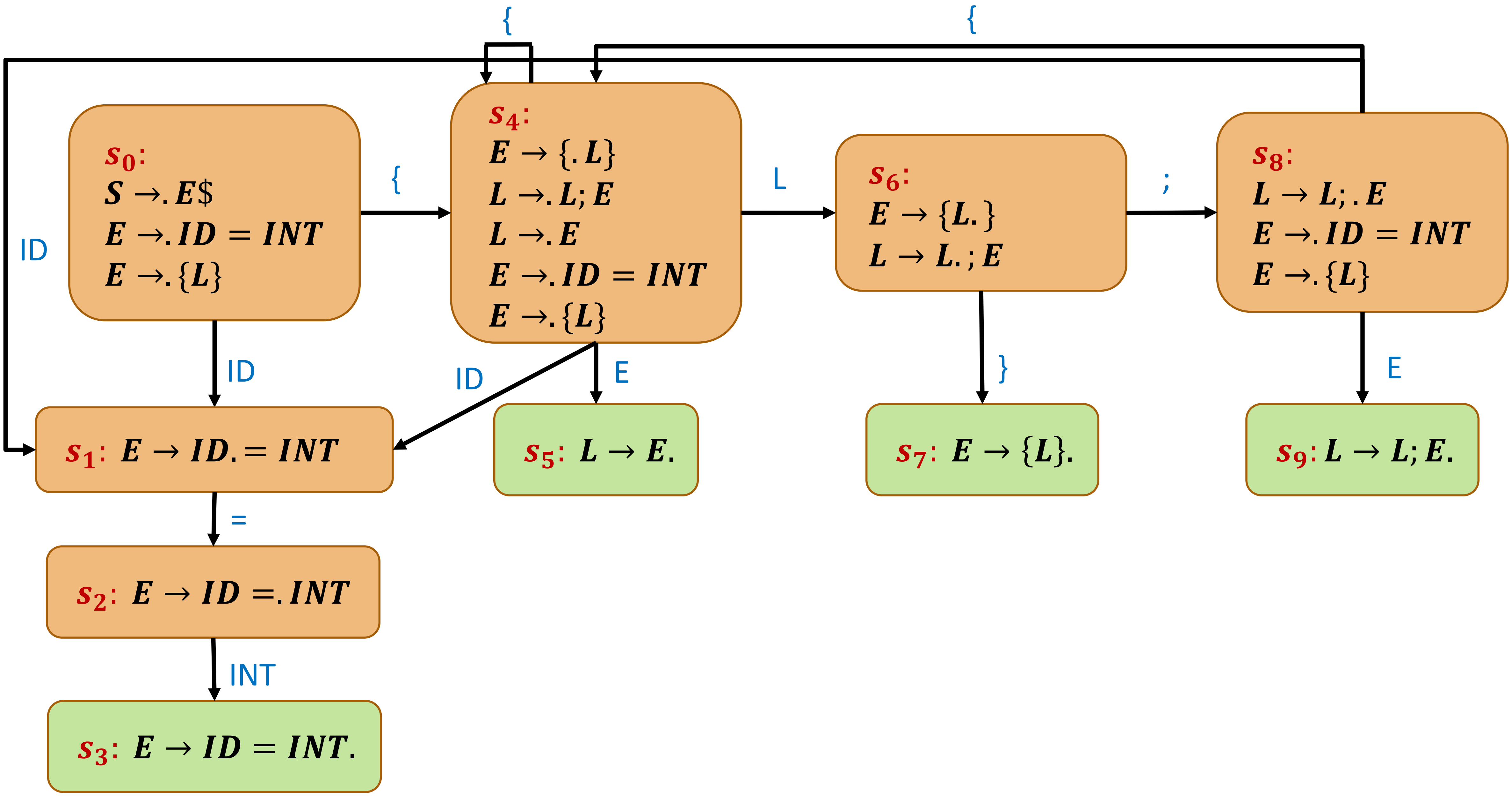
- $E \rightarrow ID. = INT$

So the next state (the ϵ-closure) contains:

- $E \rightarrow ID. = INT$

which was already computed: s_1

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$



s_0: $S \to .E\$$ $E \to .ID = INT$ $E \to .\{L\}$
s_1: $E \to ID. = INT$
s_2: $E \to ID = .INT$
s_3: $E \to ID = INT.$
s_4: $E \to \{.L\}$ $L \to .L; E$ $L \to .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_5: $L \to E.$
s_6: $E \to \{L.\}$ $L \to L.; E$
s_7: $E \to \{L\}.$
s_8: $L \to L; .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_9: $L \to L; E.$
{
{
{
L
;
ID
ID
ID
E
}
E
=
INT

LR(0) Parser

From s_0, if we recognized E, then the next state will contain:

- $S \rightarrow E.\$$

So the next state (the ϵ-closure) contains:

$$S \rightarrow E\$$$
$$E \rightarrow ID = INT$$
$$E \rightarrow \{L\}$$
$$L \rightarrow E$$
$$L \rightarrow L; E$$

LR(0) Parser

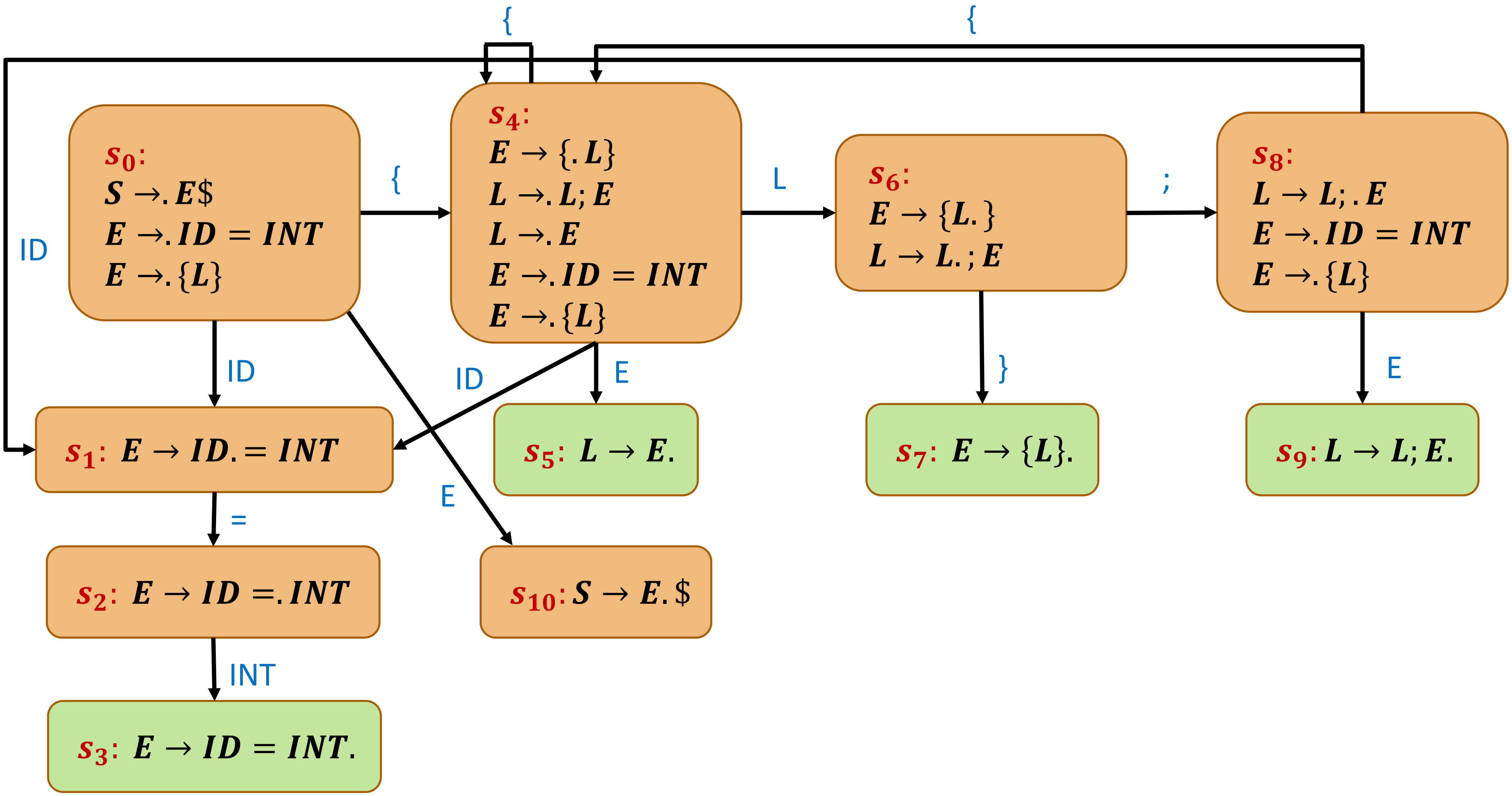
From s_0, if we recognized E, then the next state will contain:

- $S \rightarrow E.\$$

So the next state (the ϵ-closure) contains:

- $S \rightarrow E.\$$

$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$



s_0: $S \to .E\$$
$E \to .ID = INT$
$E \to .\{L\}$
s_1: $E \to ID. = INT$
s_2: $E \to ID = .INT$
s_3: $E \to ID = INT.$
s_4: $E \to \{.L\}$
$L \to .L; E$
$L \to .E$
$E \to .ID = INT$
$E \to .\{L\}$
s_5: $L \to E.$
s_6: $E \to \{L.\}$
$L \to L.; E$
s_7: $E \to \{L\}.$
s_8: $L \to L; .E$
$E \to .ID = INT$
$E \to .\{L\}$
s_9: $L \to L; E.$
s_{10}: $S \to E.\$$
{
{
{
L
;
ID
ID
ID
E
E
E
}
=
INT

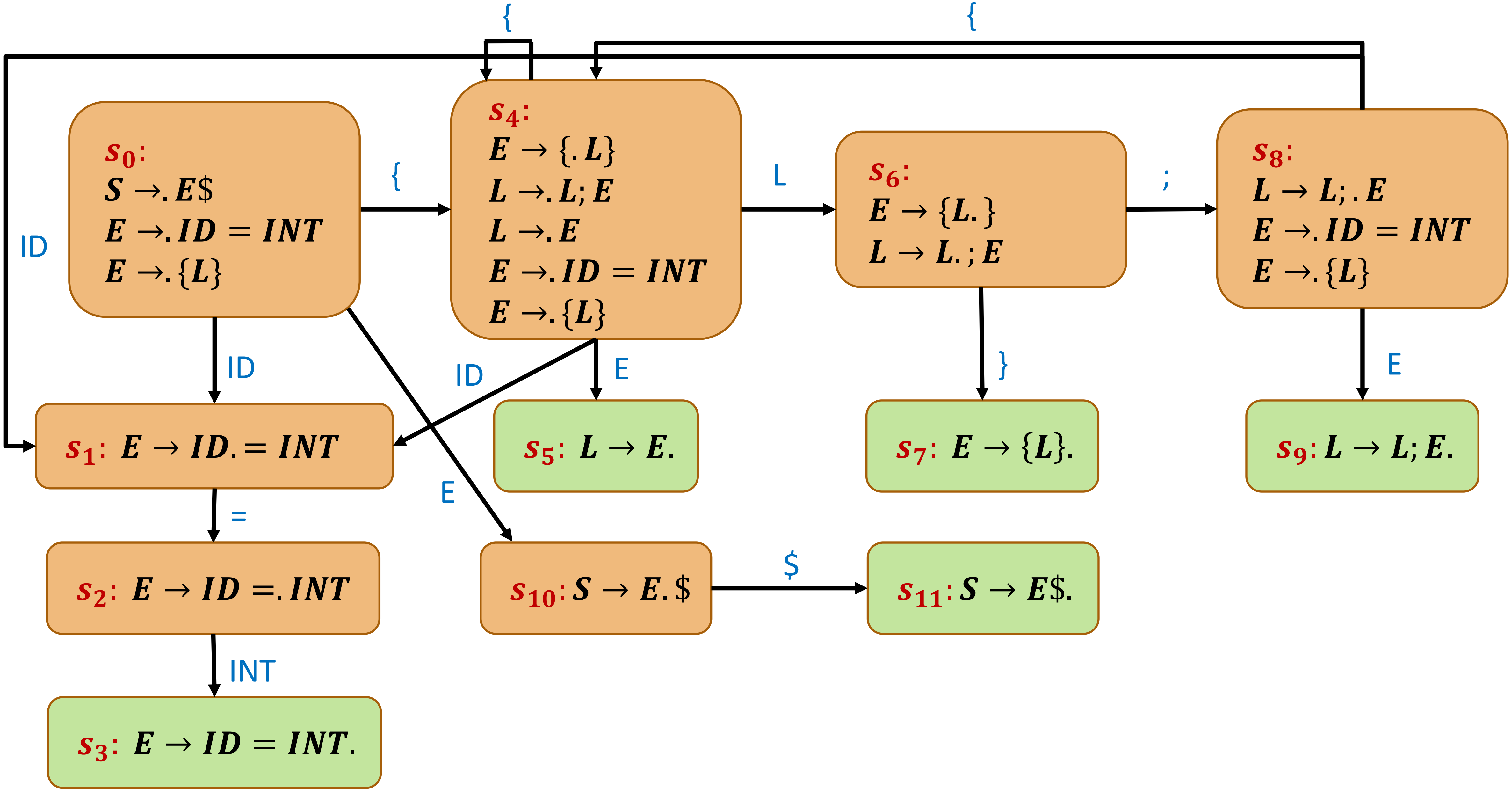
LR(0) Parser

From s_{10}, if we recognized $\$$, then the next state will contain:

- $S \rightarrow E\$.$

which is a reduce state.

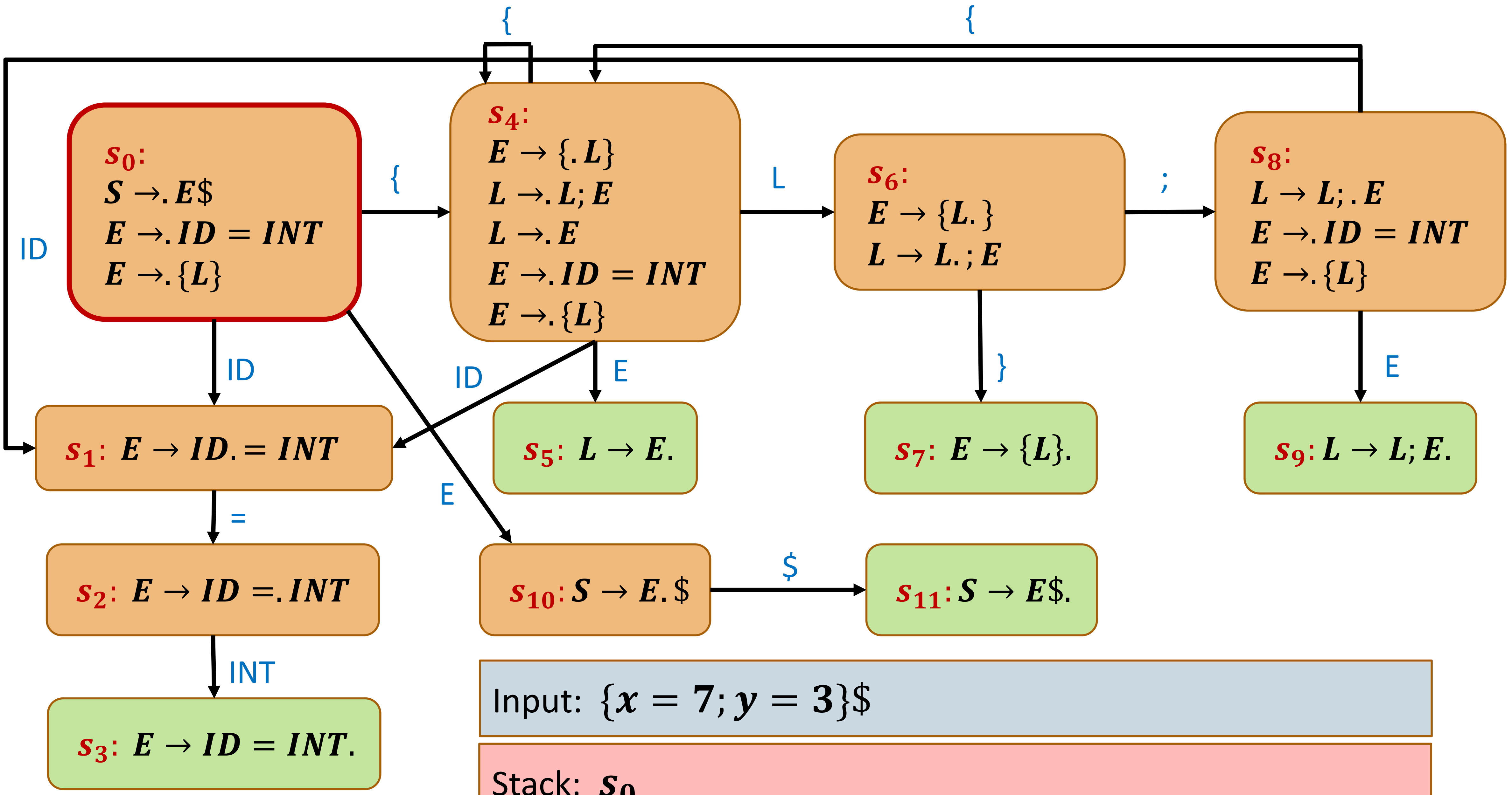
$S \rightarrow E\$$
$E \rightarrow ID = INT$
$E \rightarrow \{L\}$
$L \rightarrow E$
$L \rightarrow L; E$


s0:
S →.E$
E →.ID = INT
E →.{L}
s4:
E → {.L}
L →.L; E
L →.E
E →.ID = INT
E →.{L}
s6:
E → {L.}
L → L.; E
s8:
L → L; .E
E →.ID = INT
E →.{L}
s1: E → ID.= INT
s5: L → E.
s7: E → {L}.
s9: L → L; E.
s2: E → ID =.INT
s10: S → E.$
s11: S → E$.
s3: E → ID = INT.
{
{
{
L
;
ID
ID
ID
E
E
}
E
=
$
INT

LR(0) Parser: Running Example

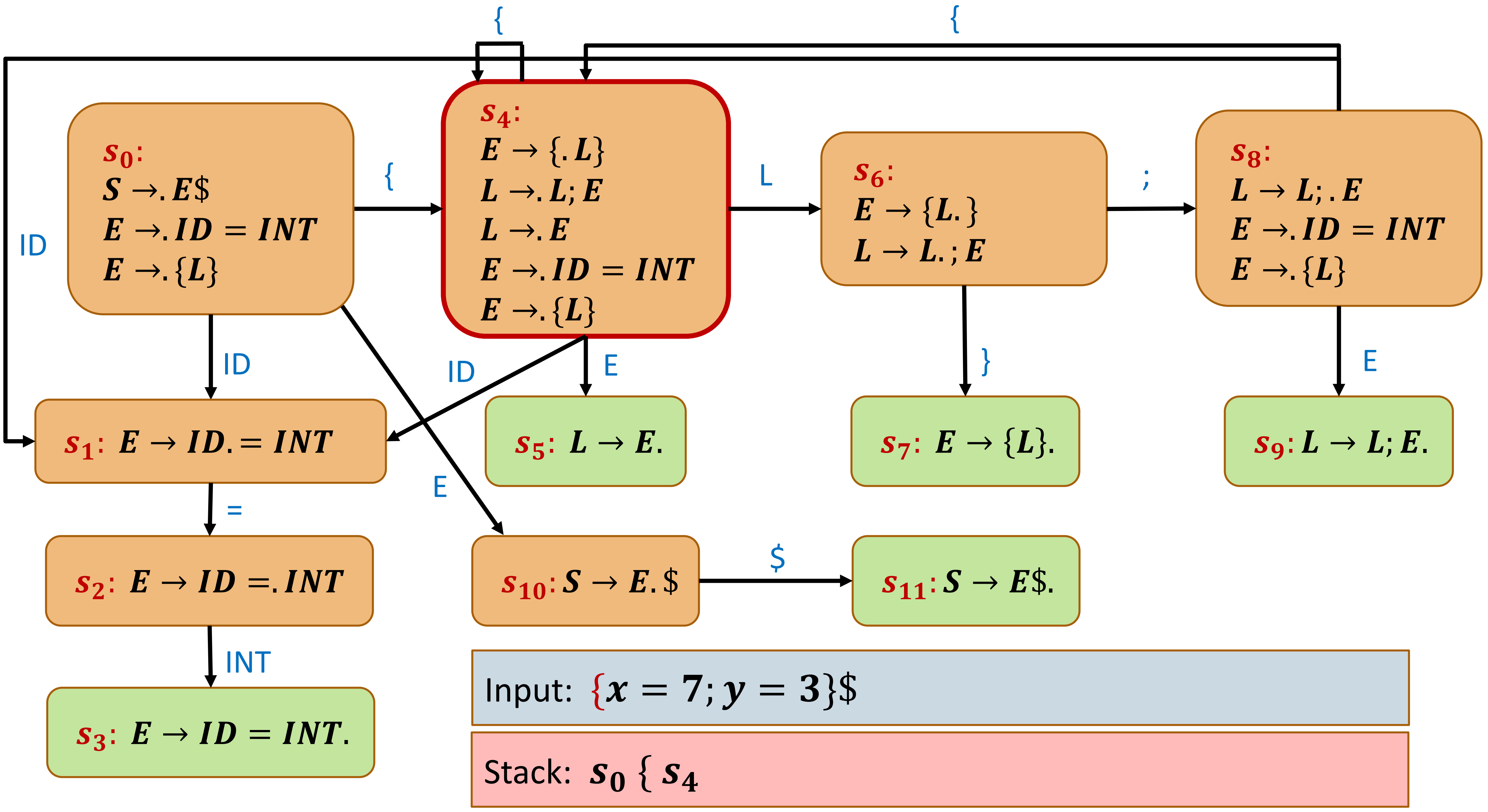
What will happen with the following input:

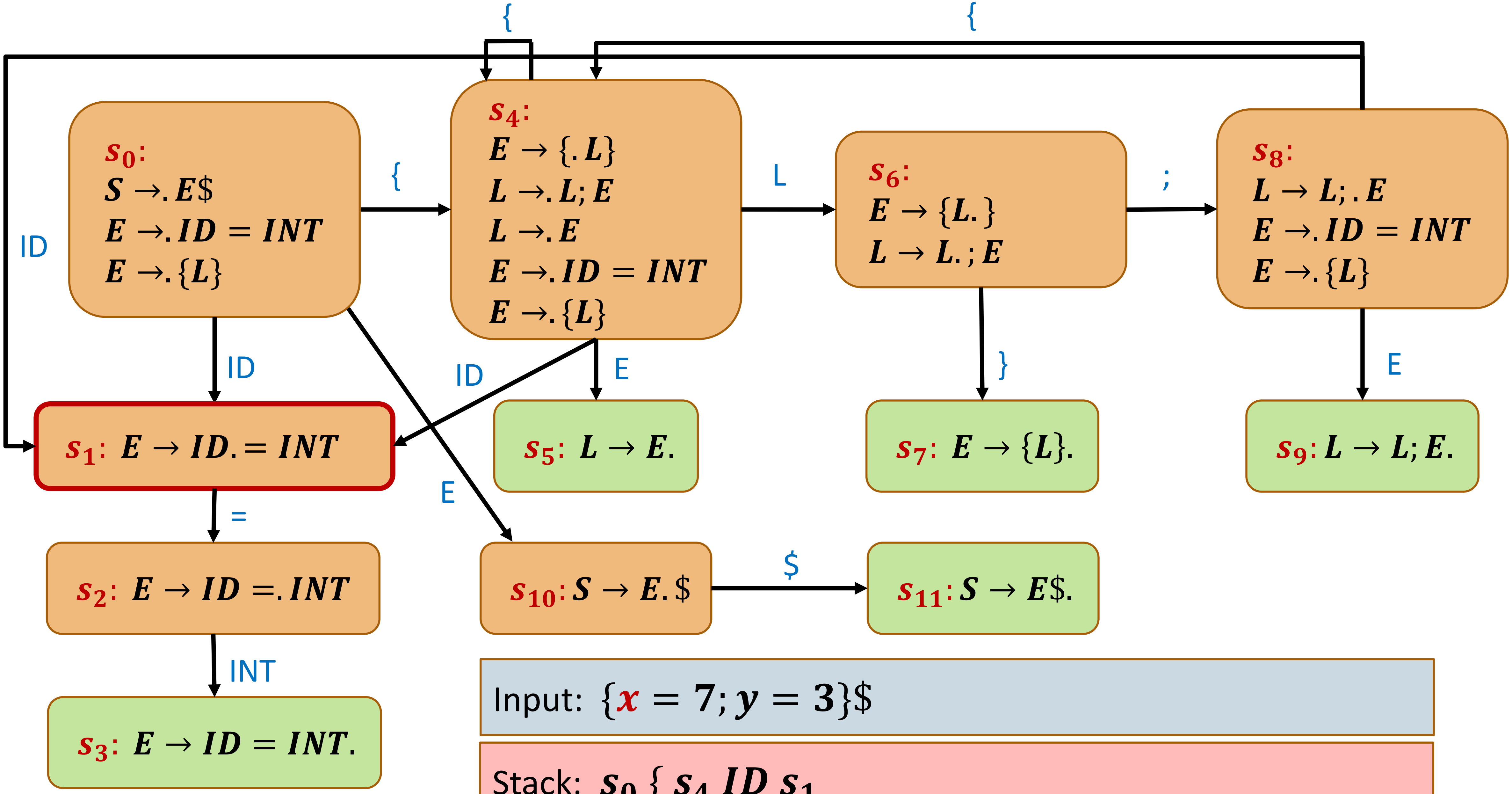
- $\{\mathrm{x} = 7; y = 3\}\$$



Input: $\{x = 7; y = 3\}\$$

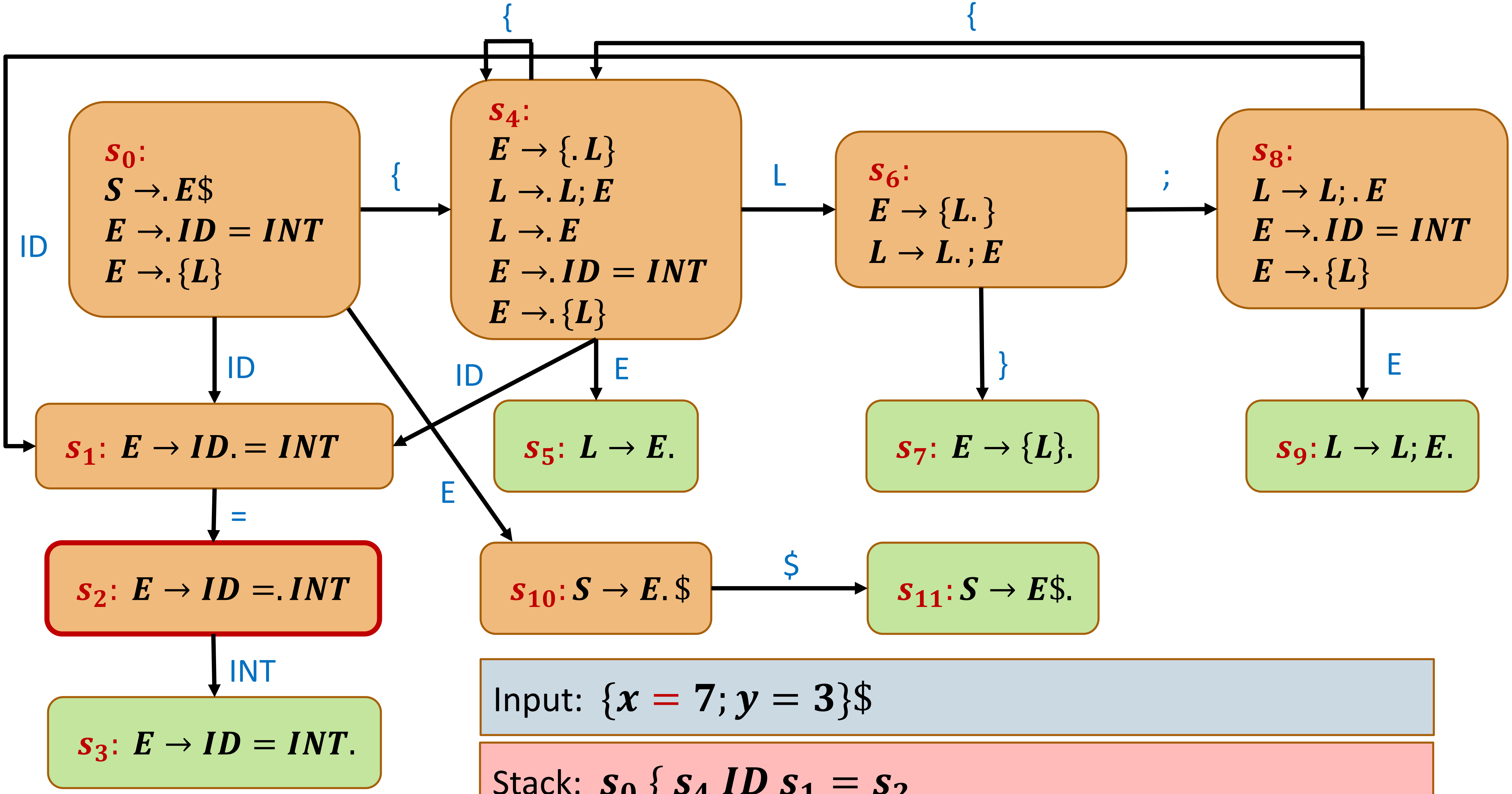
Stack: s_0


s_0: $S \to .E\$$ $E \to .ID = INT$ $E \to .\{L\}$
s_1: $E \to ID. = INT$
s_2: $E \to ID =. INT$
s_3: $E \to ID = INT.$
s_4: $E \to \{.L\}$ $L \to .L; E$ $L \to .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_5: $L \to E.$
s_6: $E \to \{L.\}$ $L \to L.; E$
s_7: $E \to \{L\}.$
s_8: $L \to L; .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_9: $L \to L; E.$
s_{10}: $S \to E.\$$
s_{11}: $S \to E\$.$
{
L
;
ID
E
}
=
INT
$
Input: $\{x = 7; y = 3\}\$$
Stack: $s_0 \{ s_4$



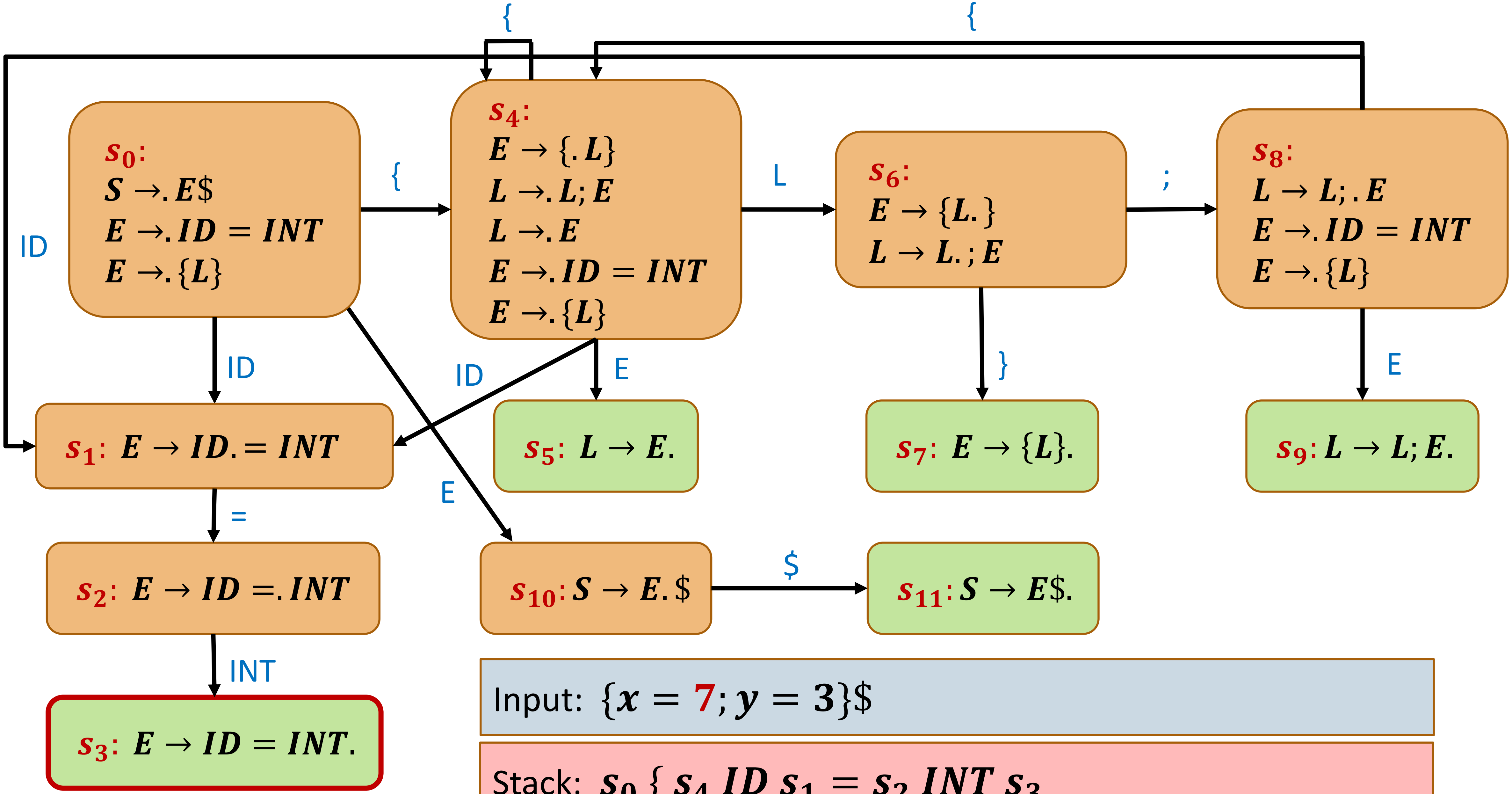
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \{ s_4 \, ID \, s_1$



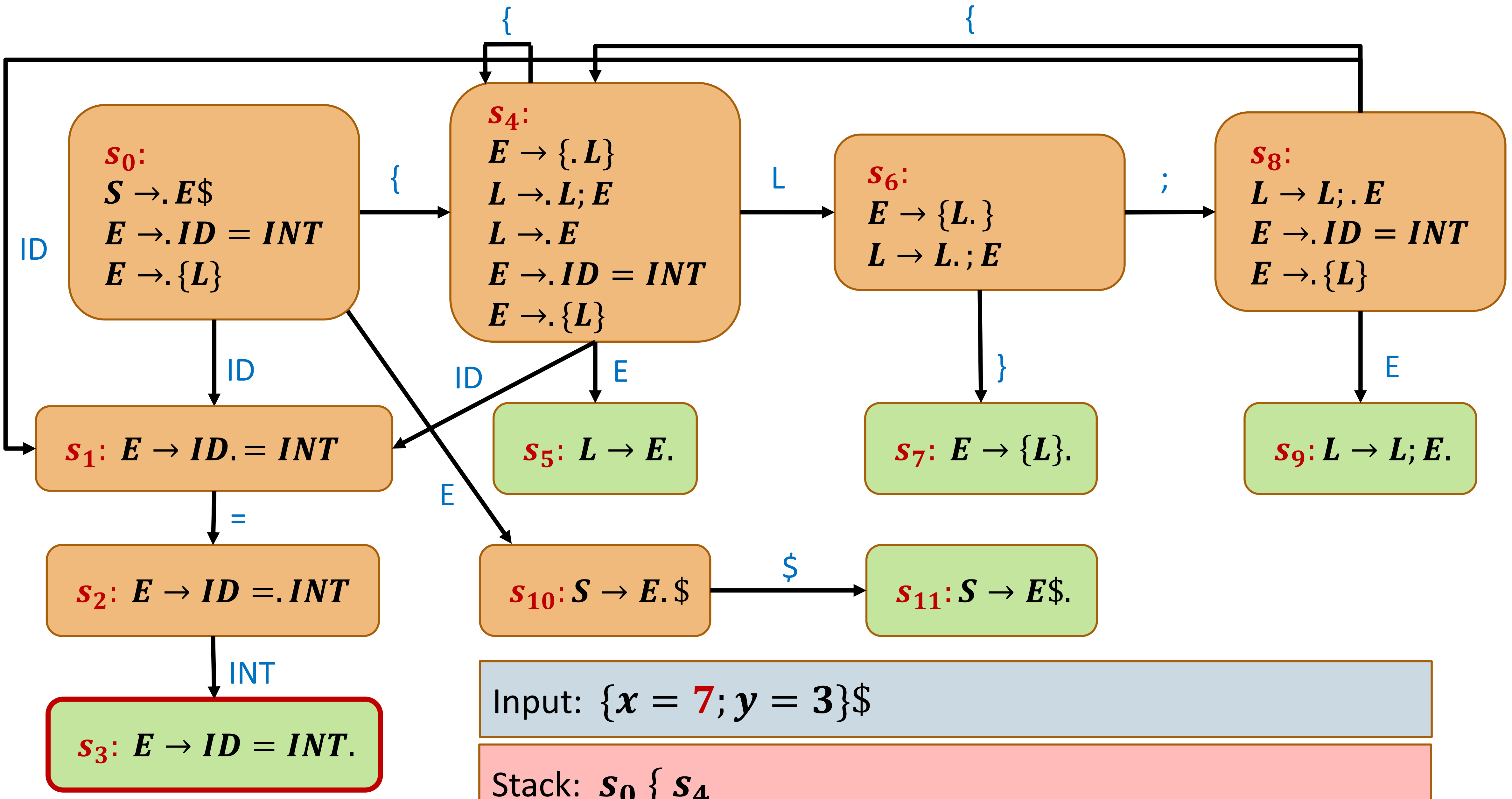
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \{ s_4\ ID\ s_1 = s_2$



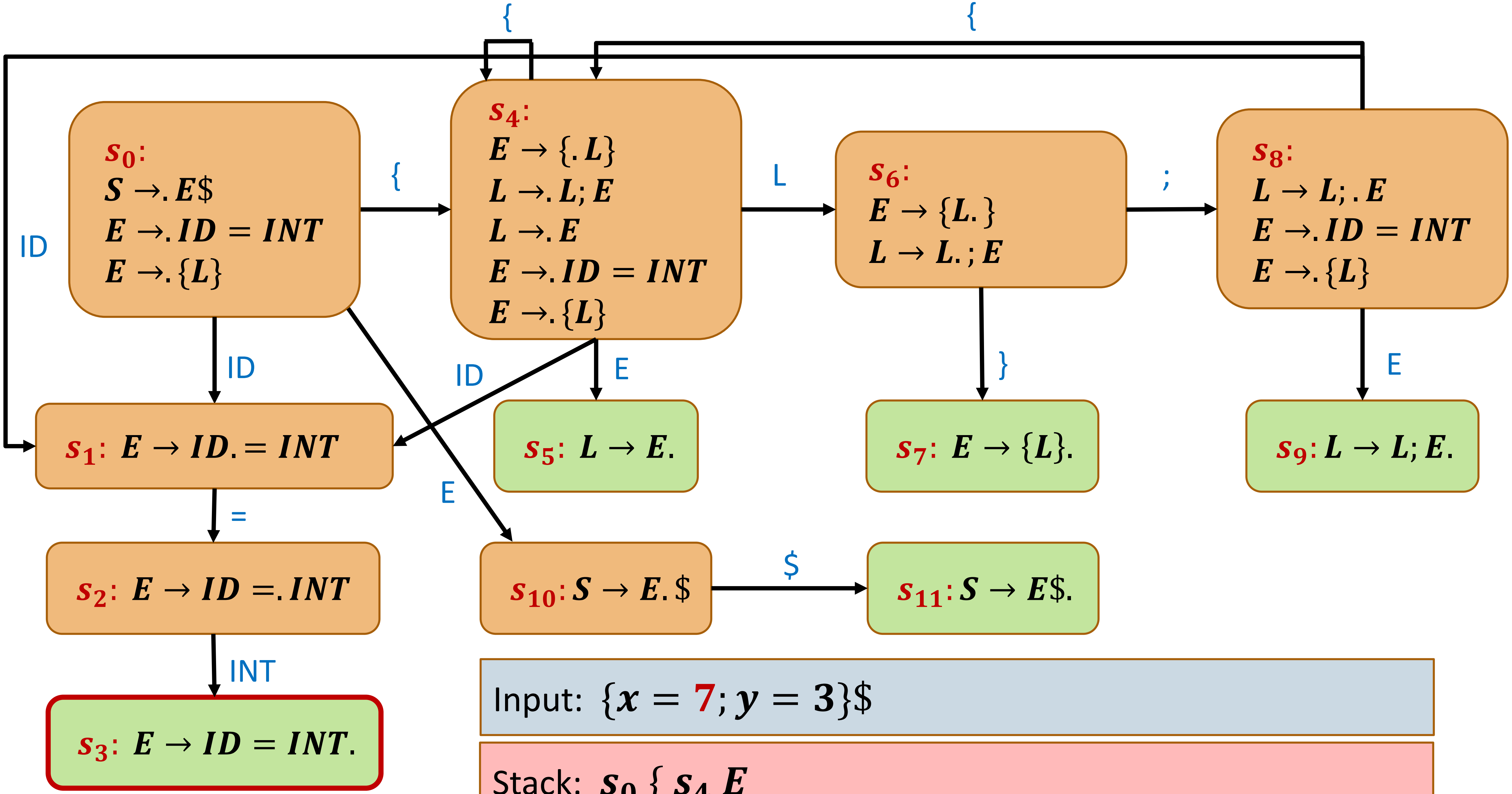
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \, \{ \, s_4 \, ID \, s_1 = s_2 \, INT \, s_3$



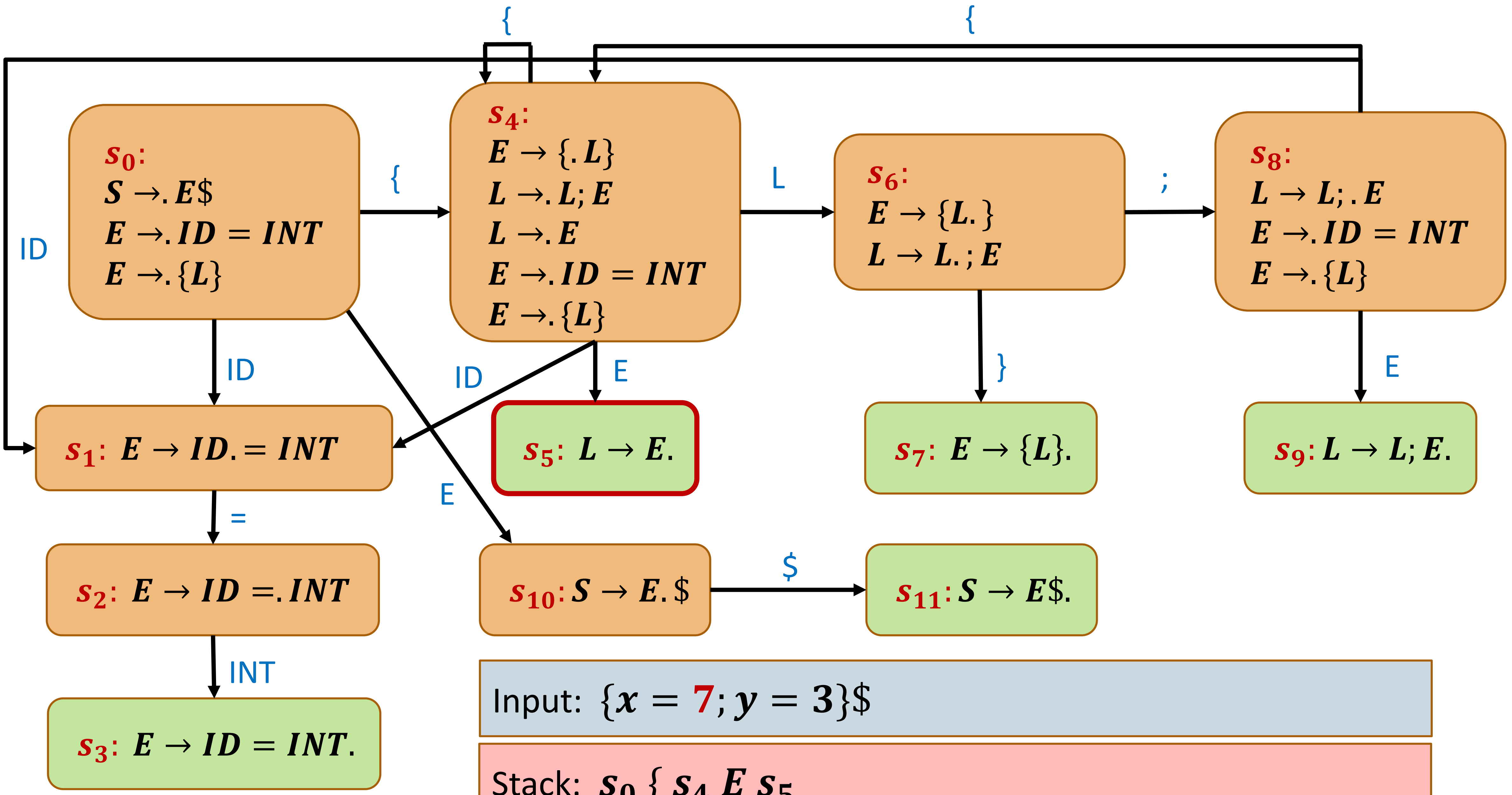
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \; \{ \; s_4$



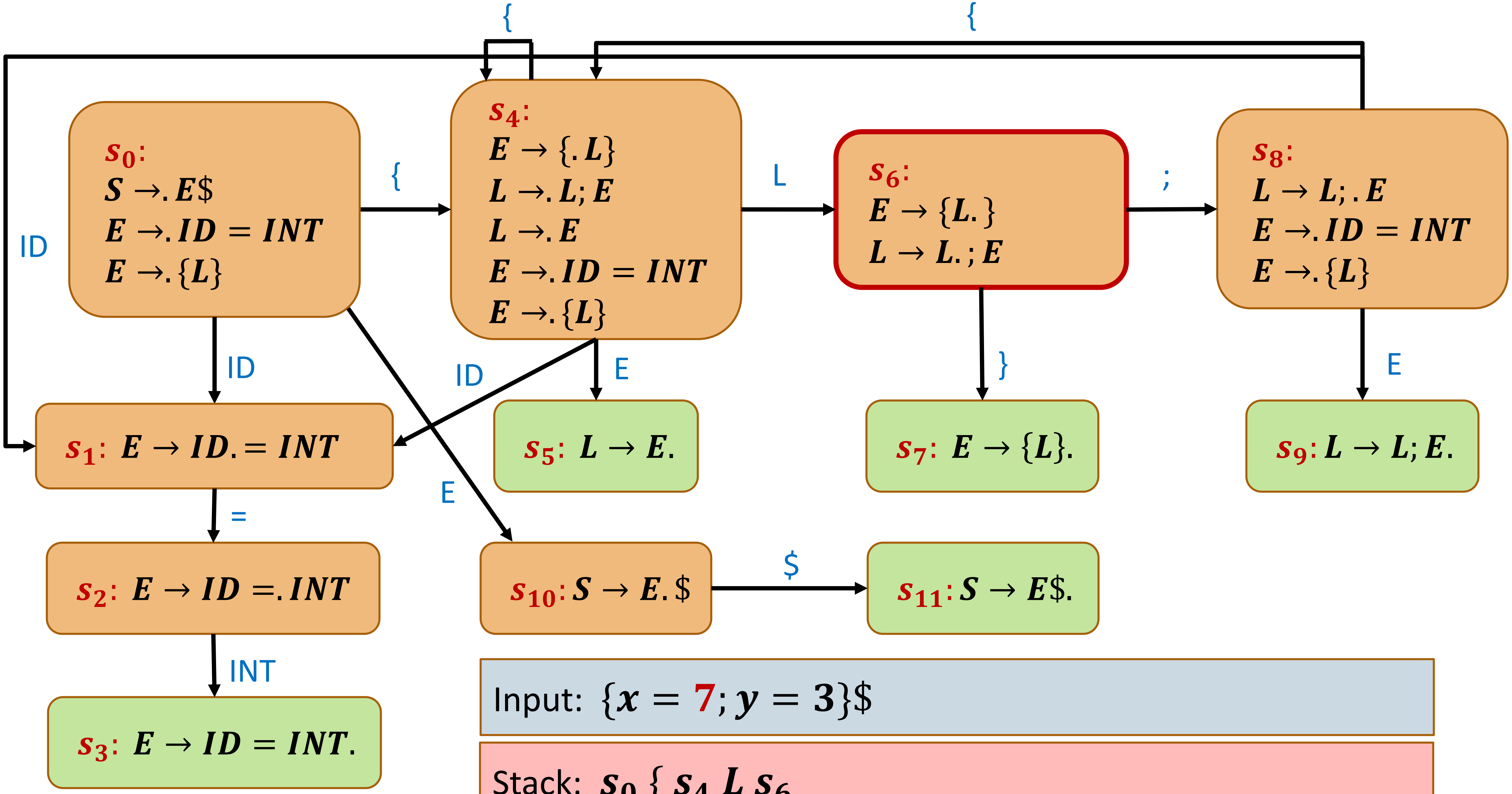
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \; \{ \; s_4 \; E$



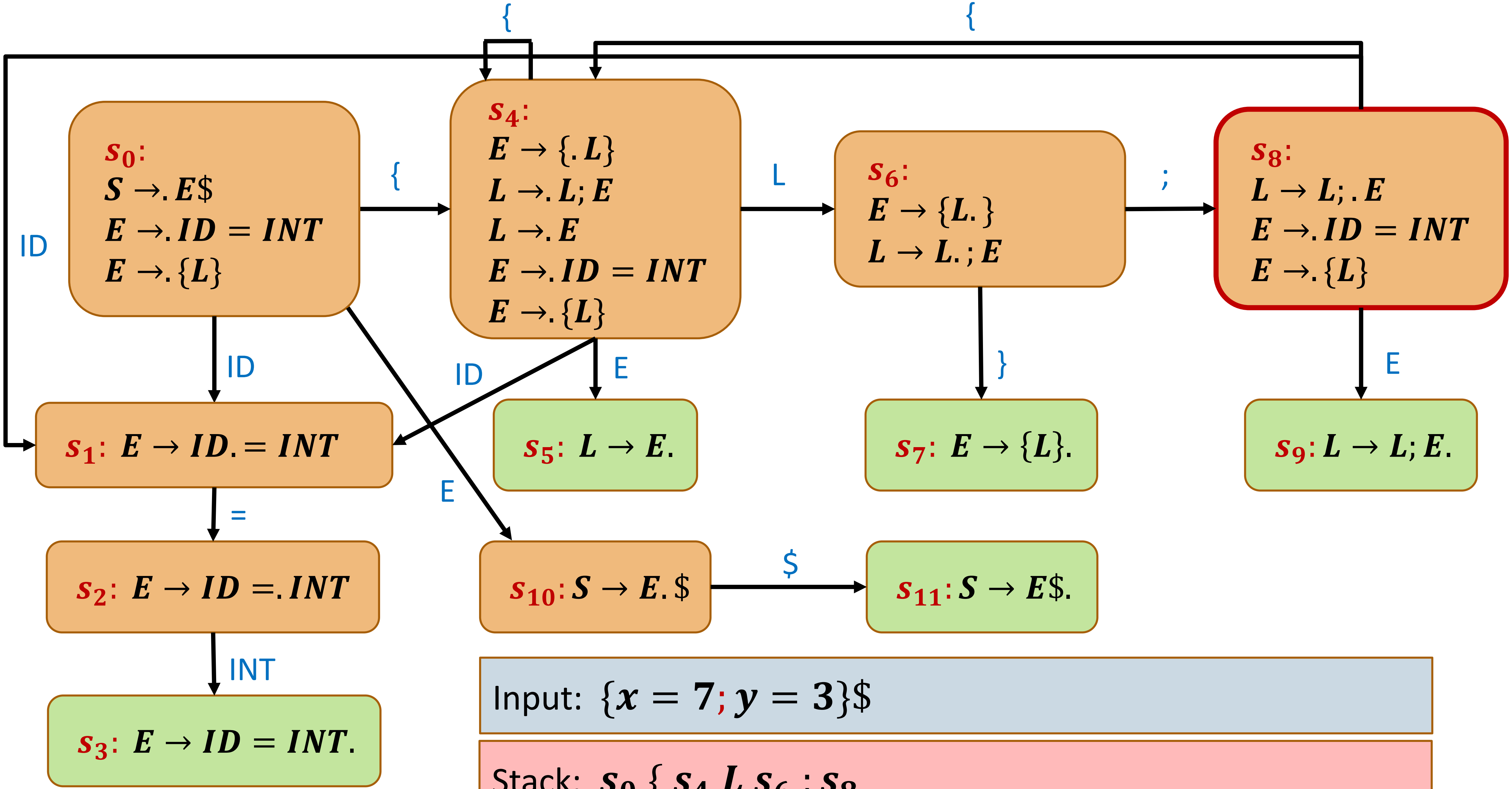
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \{ s_4 E s_5$



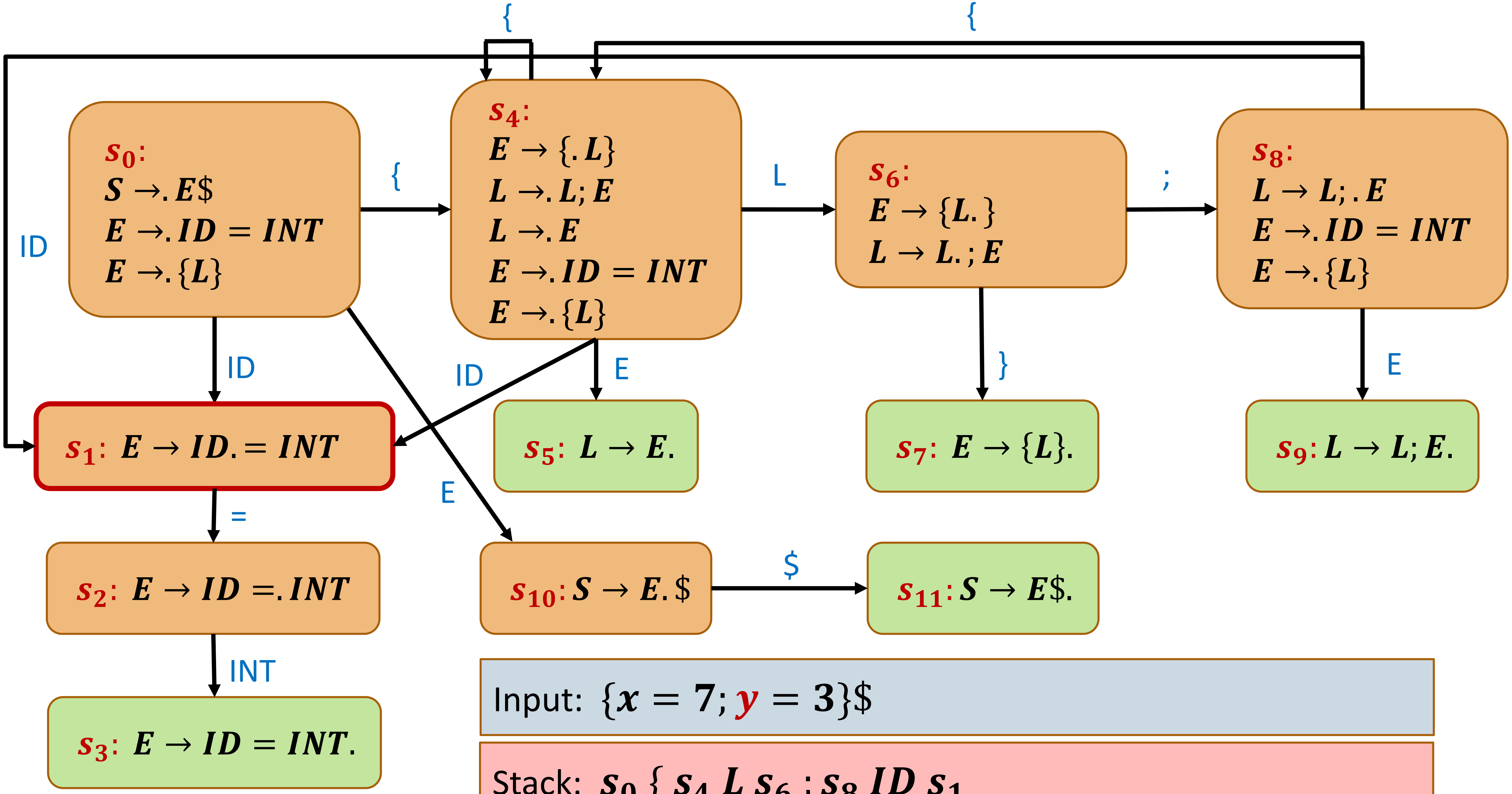
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \; \{ \; s_4 \; L \; s_6$



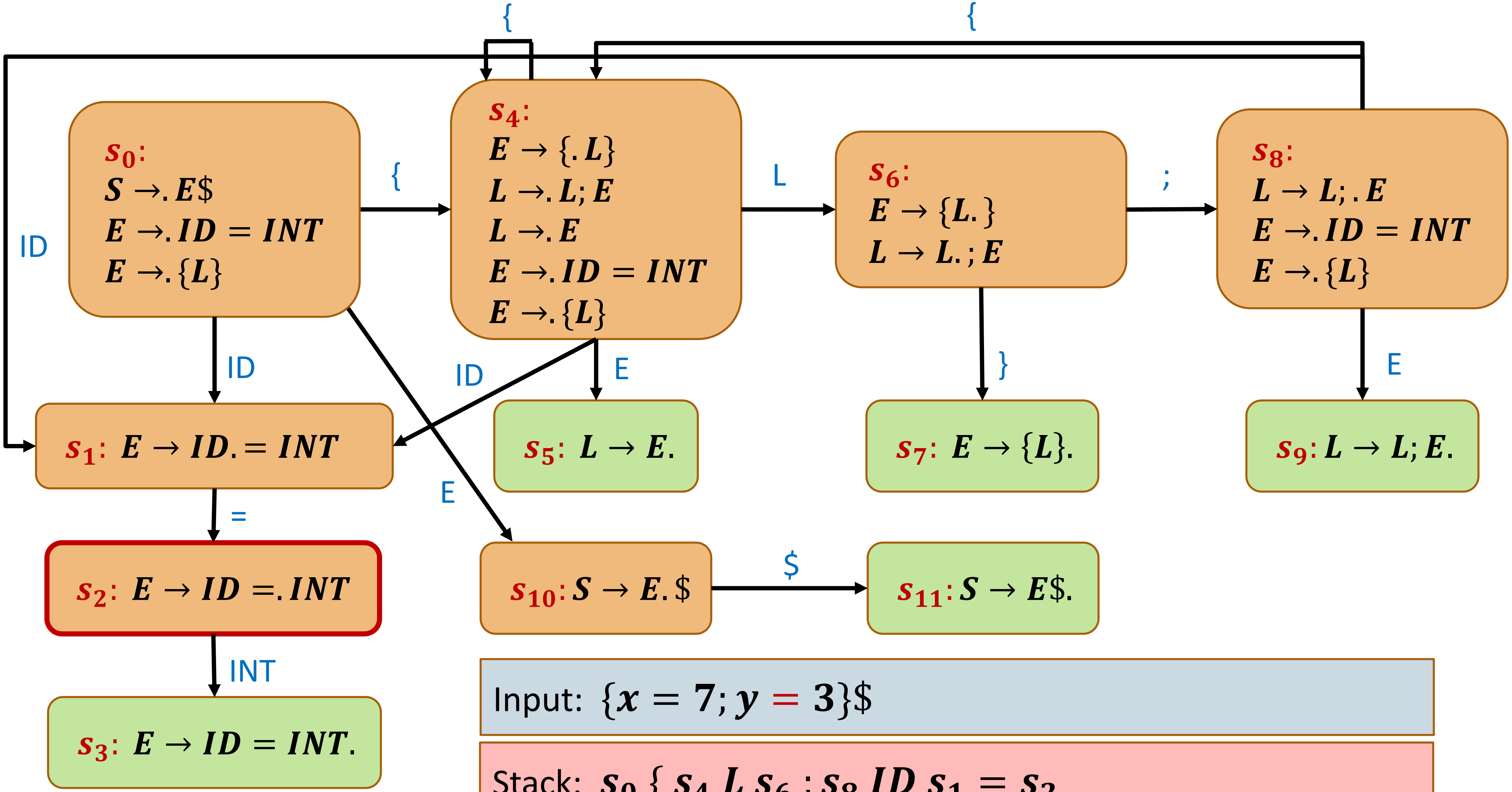
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \, \{ \, s_4 \, L \, s_6 \, ; \, s_8$



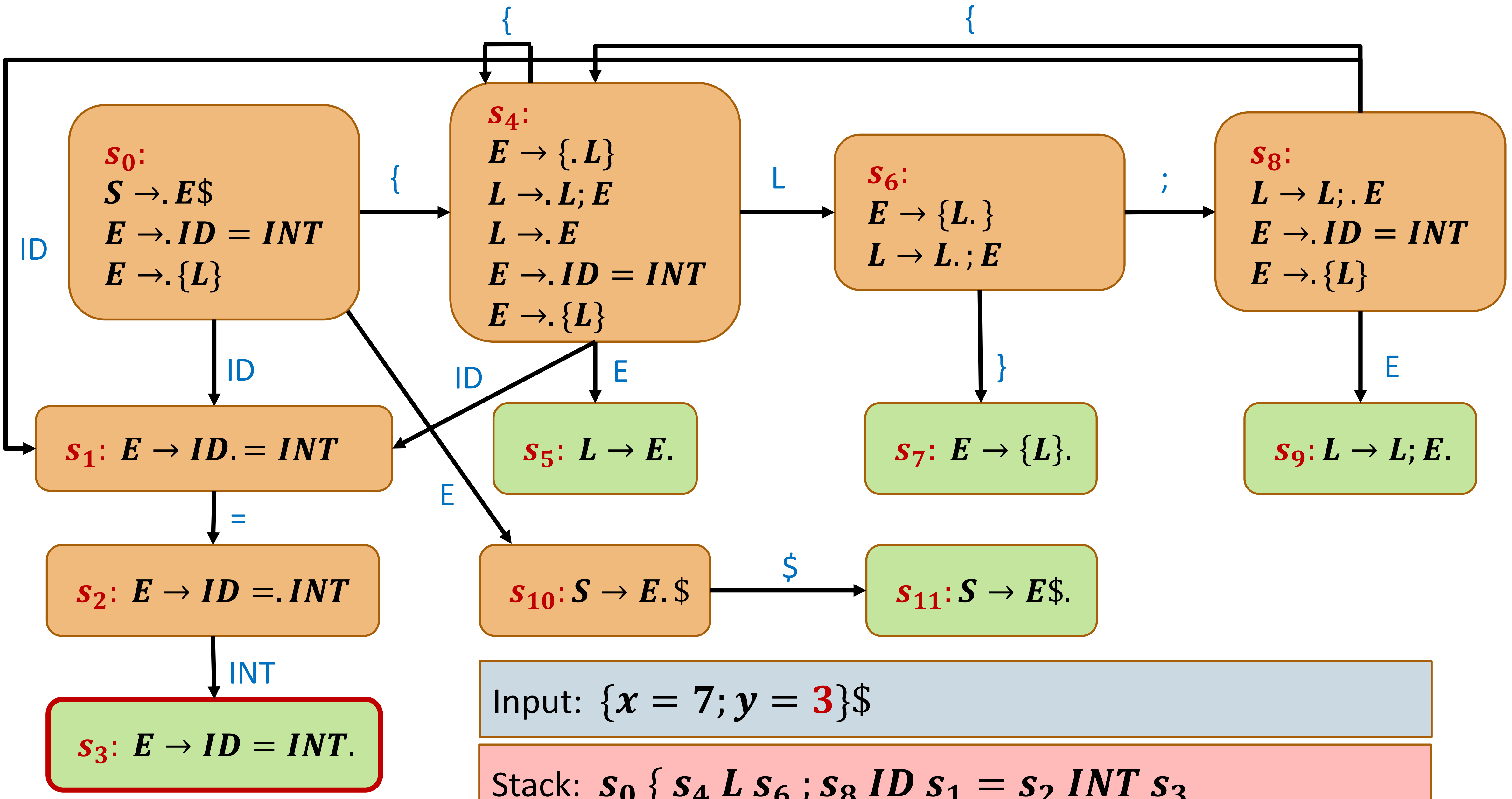
Input: $\{x = 7; {\color{red}y} = 3\}\$$

Stack: $s_0 \,\{\, s_4 \, L \, s_6 \,;\, s_8 \, ID \, s_1$



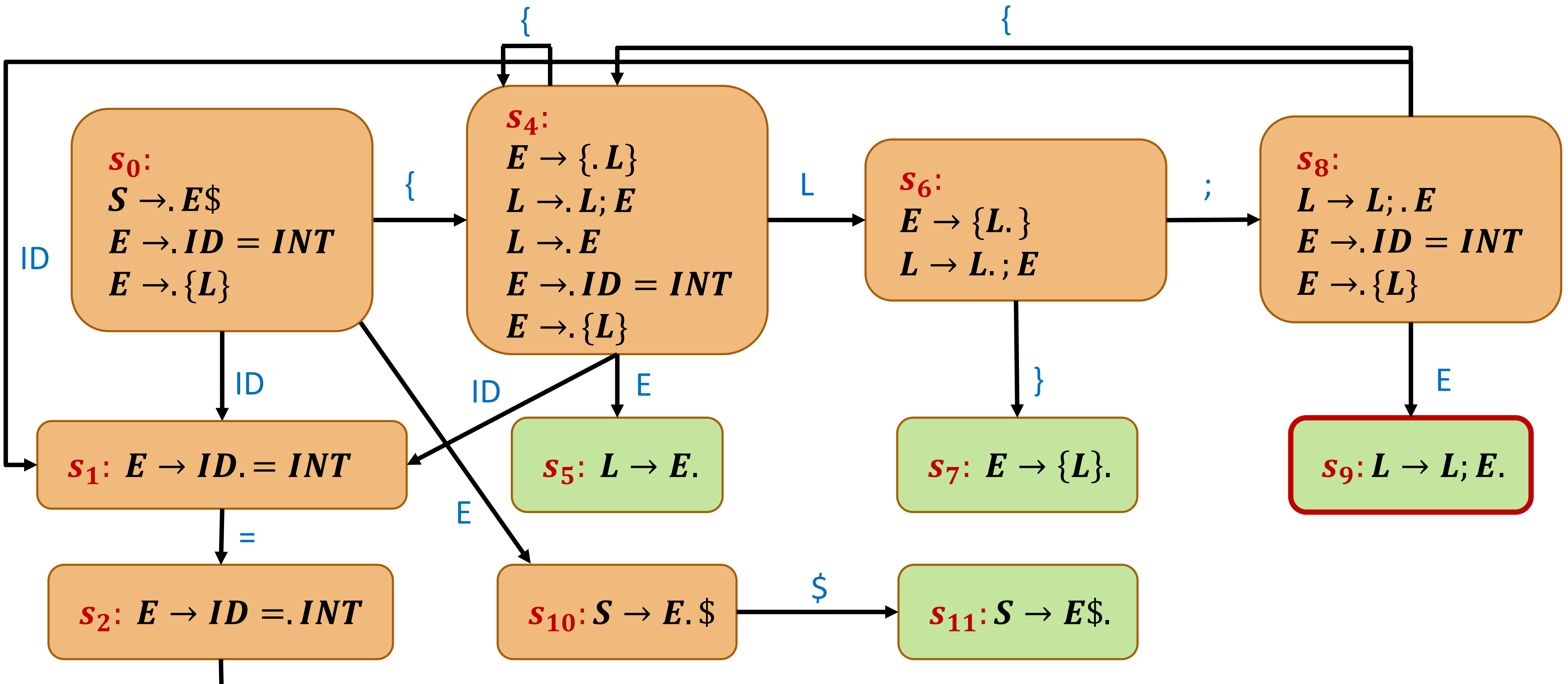
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \, \{ \, s_4 \, L \, s_6 \, ; \, s_8 \, ID \, s_1 = s_2$



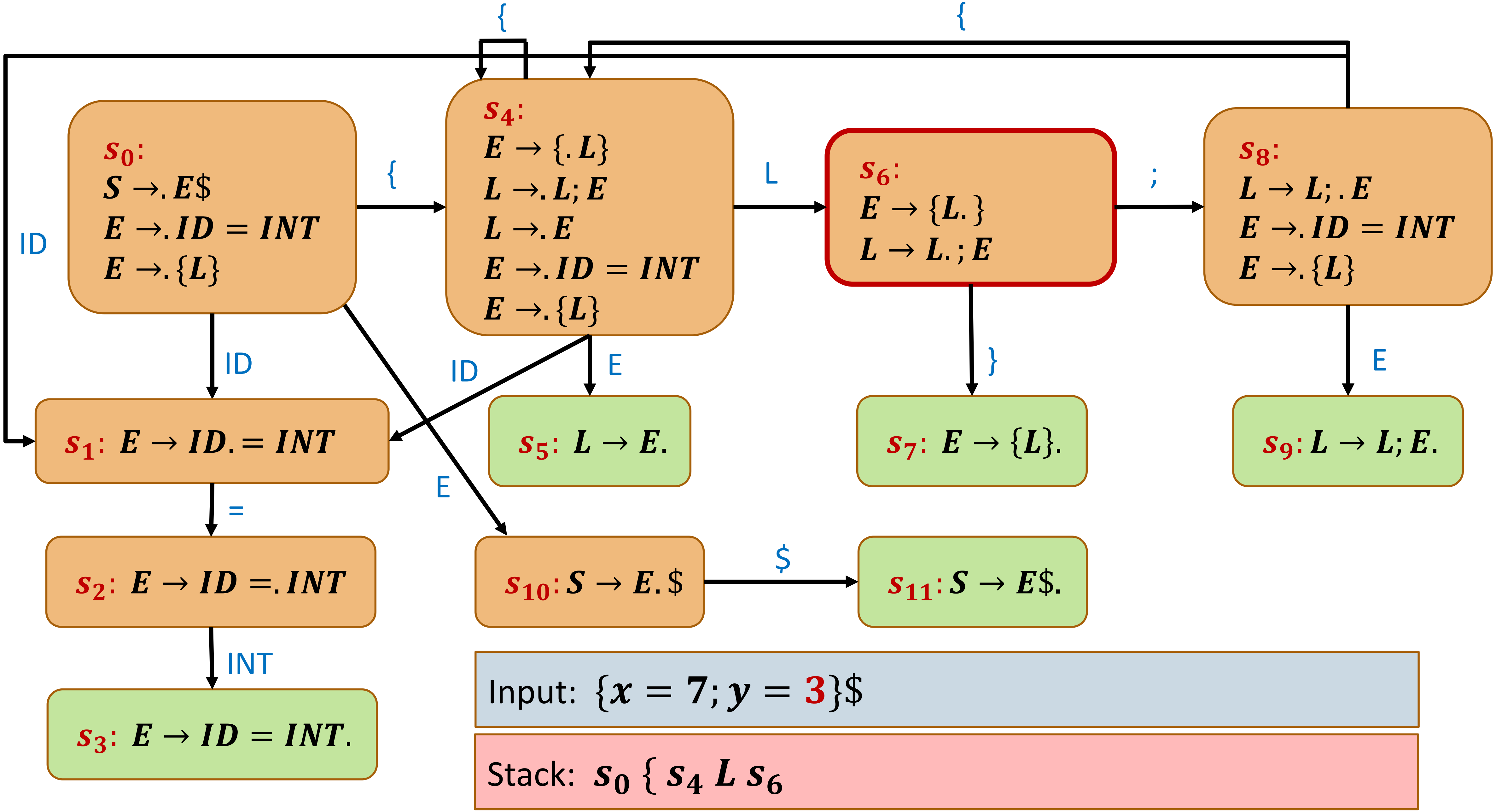
Input: $\{x = 7; y = 3\}\$$

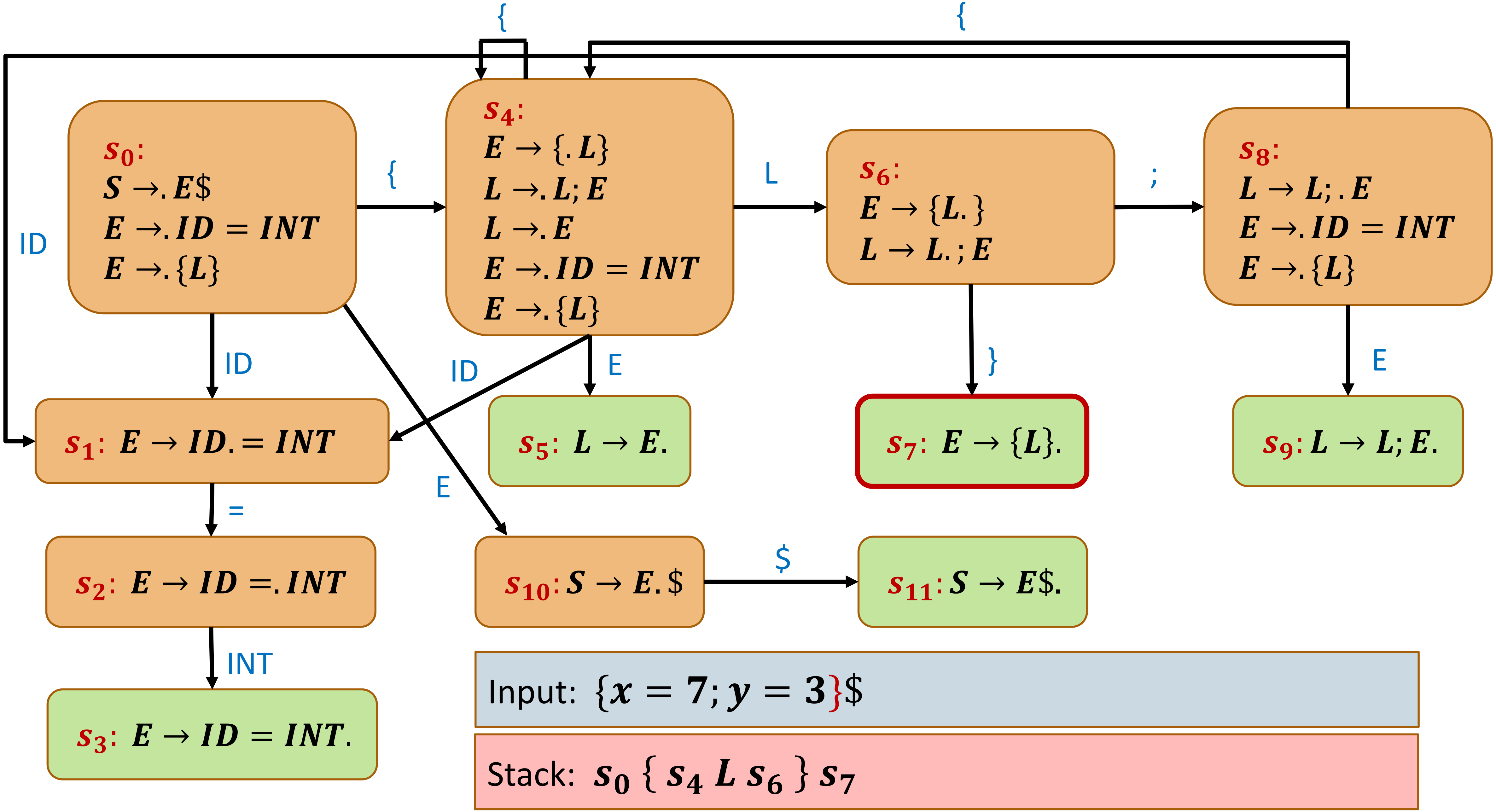
Stack: $s_0 \, \{ \, s_4 \, L \, s_6 \, ; \, s_8 \, ID \, s_1 \, = \, s_2 \, INT \, s_3$



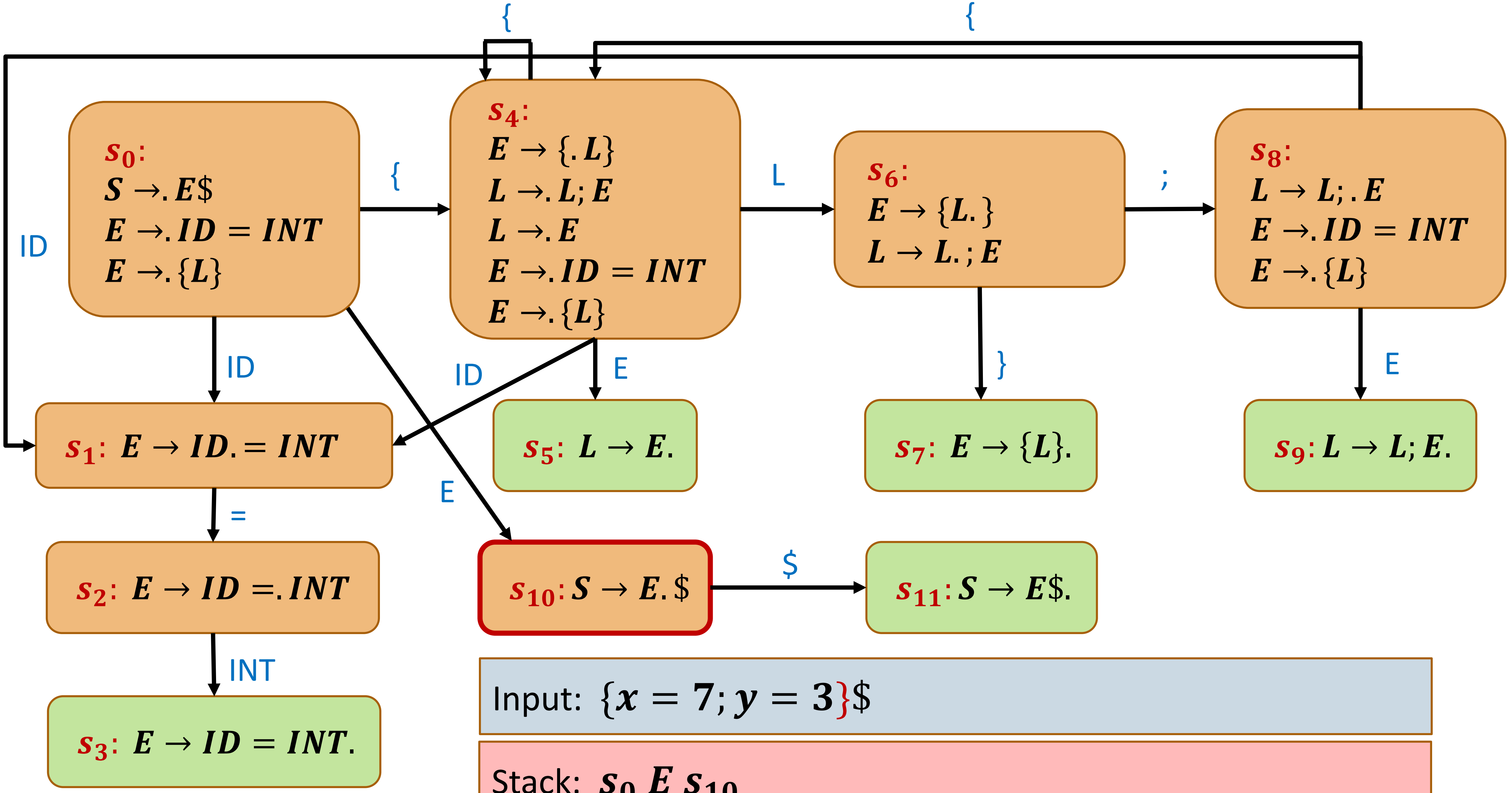
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0 \{ s_4 L s_6 ; s_8 E s_9$


s_0: $S \to .E\$$ $E \to .ID = INT$ $E \to .\{L\}$
s_1: $E \to ID. = INT$
s_2: $E \to ID =. INT$
s_3: $E \to ID = INT.$
s_4: $E \to \{.L\}$ $L \to .L;E$ $L \to .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_5: $L \to E.$
s_6: $E \to \{L.\}$ $L \to L.;E$
s_7: $E \to \{L\}.$
s_8: $L \to L;.E$ $E \to .ID = INT$ $E \to .\{L\}$
s_9: $L \to L;E.$
s_{10}: $S \to E.\$$
s_{11}: $S \to E\$.$
{
ID
E
=
INT
L
;
}
$
Input: $\{x = 7; y = 3\}\$$
Stack: $s_0 \{ s_4 L s_6$

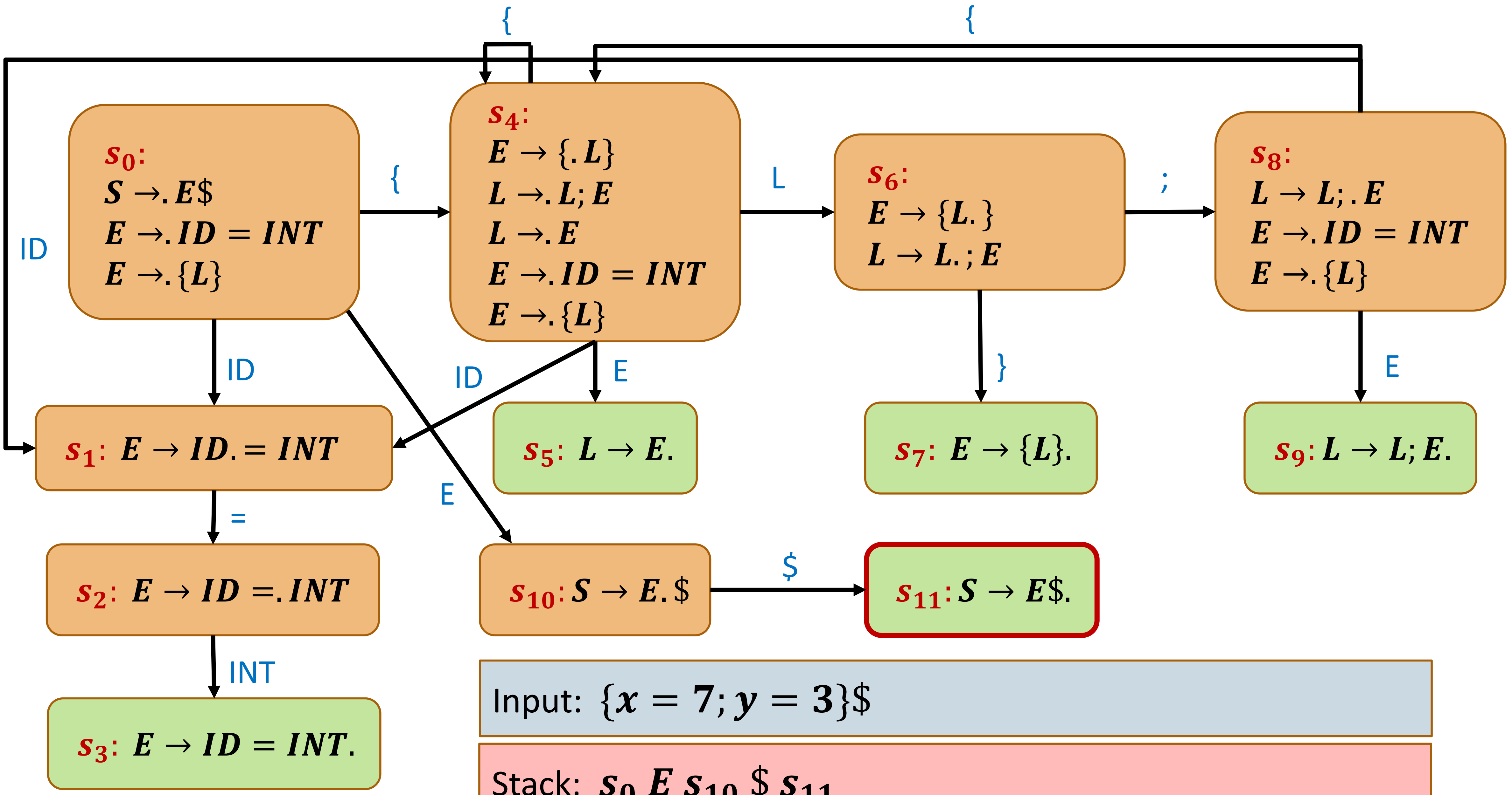


s_0: $S \to .E\$$ $E \to .ID = INT$ $E \to .\{L\}$
s_1: $E \to ID. = INT$
s_2: $E \to ID =. INT$
s_3: $E \to ID = INT.$
s_4: $E \to \{.L\}$ $L \to .L; E$ $L \to .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_5: $L \to E.$
s_6: $E \to \{L.\}$ $L \to L.; E$
s_7: $E \to \{L\}.$
s_8: $L \to L; .E$ $E \to .ID = INT$ $E \to .\{L\}$
s_9: $L \to L; E.$
s_{10}: $S \to E.\$$
s_{11}: $S \to E\$.$
{
{
{
L
;
ID
ID
ID
E
E
E
}
=
INT
$
Input: $\{x = 7; y = 3\}\$$
Stack: $s_0\ \{\ s_4\ L\ s_6\ \}\ s_7$



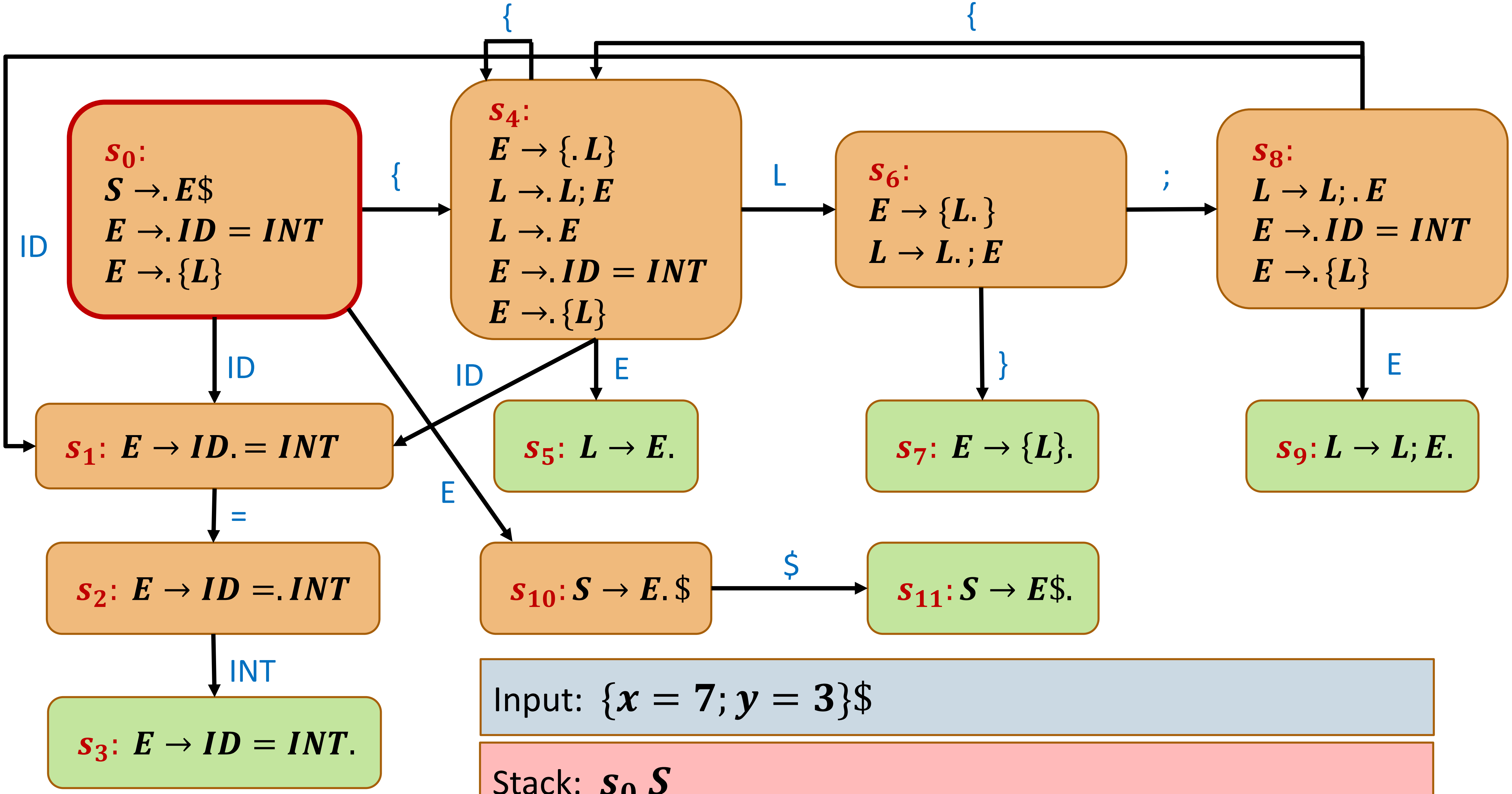
Input: $\{x = 7; y = 3\}\$$

Stack: $s_0\ E\ s_{10}$



Input: $\{x = 7; y = 3\}\$$

Stack: $s_0\ E\ s_{10}\ \$\ s_{11}$



Input: $\{x = 7; y = 3\}\$$

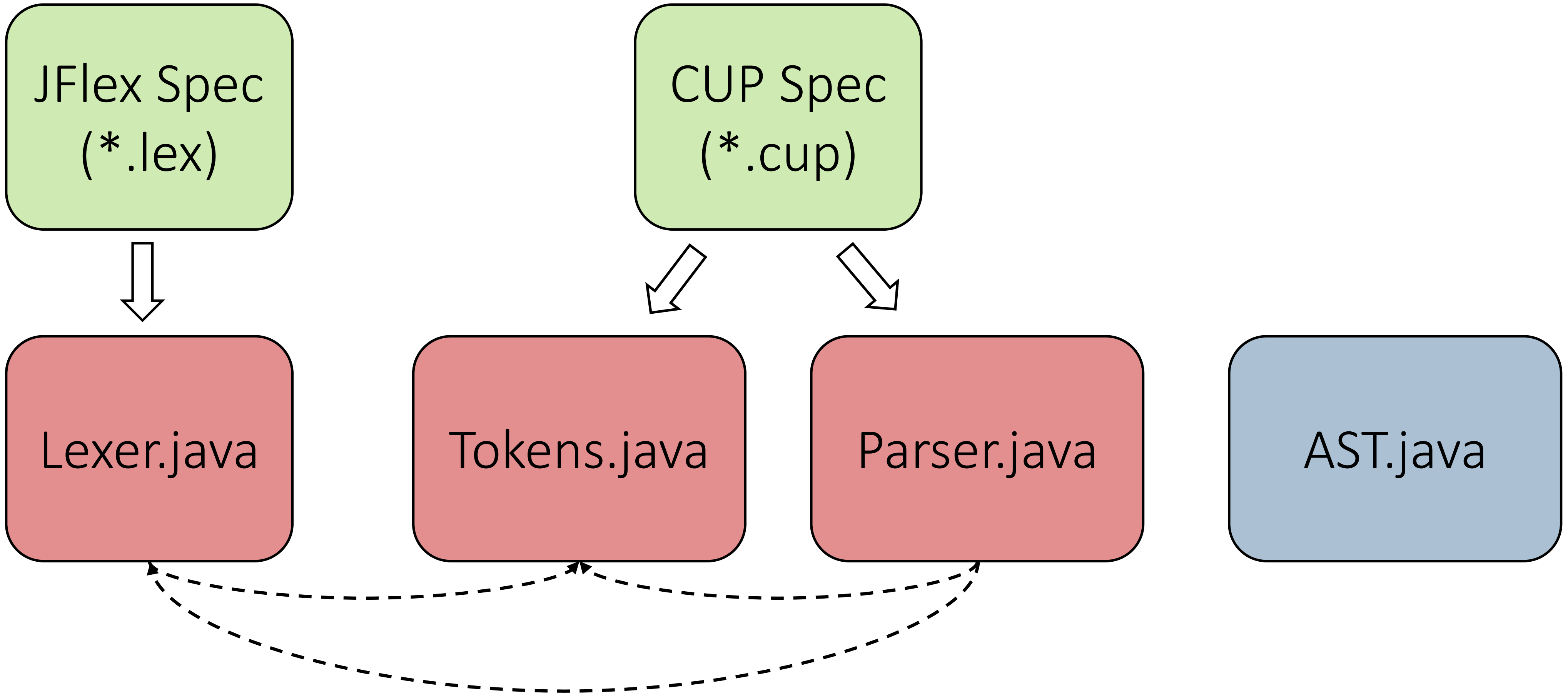
Stack: $s_0\ S$

Parsing with CUP

CUP

- Given a user-specified grammar, generates an LALR parser
- Works with JFlex, which provides the parsed tokens
- Other tools:
  - Bison (for C)

CUP/JFlex Workflow



CUP Format

parser setup
lexer setup
grammar

```
parser code {:
...
:}

scan with {:
...
:}

terminal …
non terminal …
start with …
<derivation rules…>
```

CUP Spec: Parser Setup

```
parser code {:
	public Lexer lexer;
	public Parser(Lexer lexer) {
		super(lexer);
		this.lexer = lexer;
	}
	public void report_error(String message, Object info) {
		System.exit(0);
	}
:}
```

CUP Spec: Lexer Setup

```
scan with {:
	Symbol s;
	s = lexer.next_token();
	// print token...
	return s;
:};
```

CUP Spec: Terminals

terminal T1;
terminal T2;
terminal T3;
terminal T4;
…

CUP Spec: Non-Terminals

non terminal AST_NODE_1 E1;
non terminal AST_NODE_2 E2;
non terminal AST_NODE_3 E3;
…

CUP Spec: Operator Precedence

precedence left OP1;
precedence left OP2;
precedence left OP3;
precedence left OP4;
…

These are token names…

CUP Spec: Grammar

```
start with S;
S ::=
  E1:v1          {: RESULT = new AST_NODE_CLASS_1(v1);  :} ;

S ::=
  E1:v1 E2:v2 {: RESULT = new AST_NODE_CLASS_2(v1, v2);  :}

…
```

CUP Spec: AST Nodes

- We need to **decide** which node types we have in our AST
- We need to **define** the classes for these AST nodes

CUP Example

Consider the following CFG:

- $E \rightarrow INT$
- $E \rightarrow V$
- $E \rightarrow E + E$
- $E \rightarrow E \ - E$
- $V \rightarrow ID$
- $V \rightarrow V\,.ID$

CUP Example: Terminals

terminal Integer INT;
terminal String ID;
terminal PLUS;
terminal MINUS;
terminal DOT;

CUP Example: Non-Terminals

non terminal AST_EXP EXP;
non terminal AST_VAR VAR;

CUP Example: Operator Precedence

precedence left PLUS;
precedence left MINUS;

CUP Example: Grammar

```
start with EXP;
EXP ::=
    INT:i {: RESULT = new AST_EXP_INT(i); :} |
    VAR:v {: RESULT = new AST_EXP_VAR(v); :} |
    EXP:e1 PLUS  EXP:e2 {: RESULT = new AST_EXP_BINOP(e1, e2, 0); :} |
    EXP:e1 MINUS EXP:e2 {: RESULT = new AST_EXP_BINOP(e1, e2, 1); :};
VAR ::=
    ID:name {: RESULT = new AST_VAR_SIMPLE(name); :} |
    VAR:v DOT ID:fieldName {: RESULT = new AST_VAR_FIELD(v, fieldName); :};
```

CUP Example: AST Nodes

For the non-terminal ***VAR***:

```
public abstract class AST_VAR extends AST_Node {

}
```

CUP Example: AST Nodes

For the rule ***VAR ::= ID:name***:

```
public class AST_VAR_SIMPLE extends AST_VAR {
	public String name;
	public AST_VAR_SIMPLE(String name) {
		this.name = name;
	}
}
```

CUP Example: AST Nodes

For the rule ***VAR ::= VAR:v DOT ID:fieldName*** :

```
public class AST_VAR_FIELD extends AST_VAR {
	public AST_VAR var;
	public String fieldName;
	public AST_VAR_FIELD(AST_VAR var, String fieldName) {
		this.var = var;
		this.fieldName = fieldName;
	}
}
```

CUP Example: AST Nodes

For the non-terminal ***EXP***:

public abstract class **AST_EXP** extends AST_Node {

}

CUP Example: AST Nodes

For the rule ***EXP ::= INT:i***:

```
public class AST_EXP_INT extends AST_EXP {
	public int value;
	public AST_EXP_INT(int value)	{
		this.value = value;
	}
}
```

CUP Example: AST Nodes

For the rule ***EXP ::= VAR:v***:

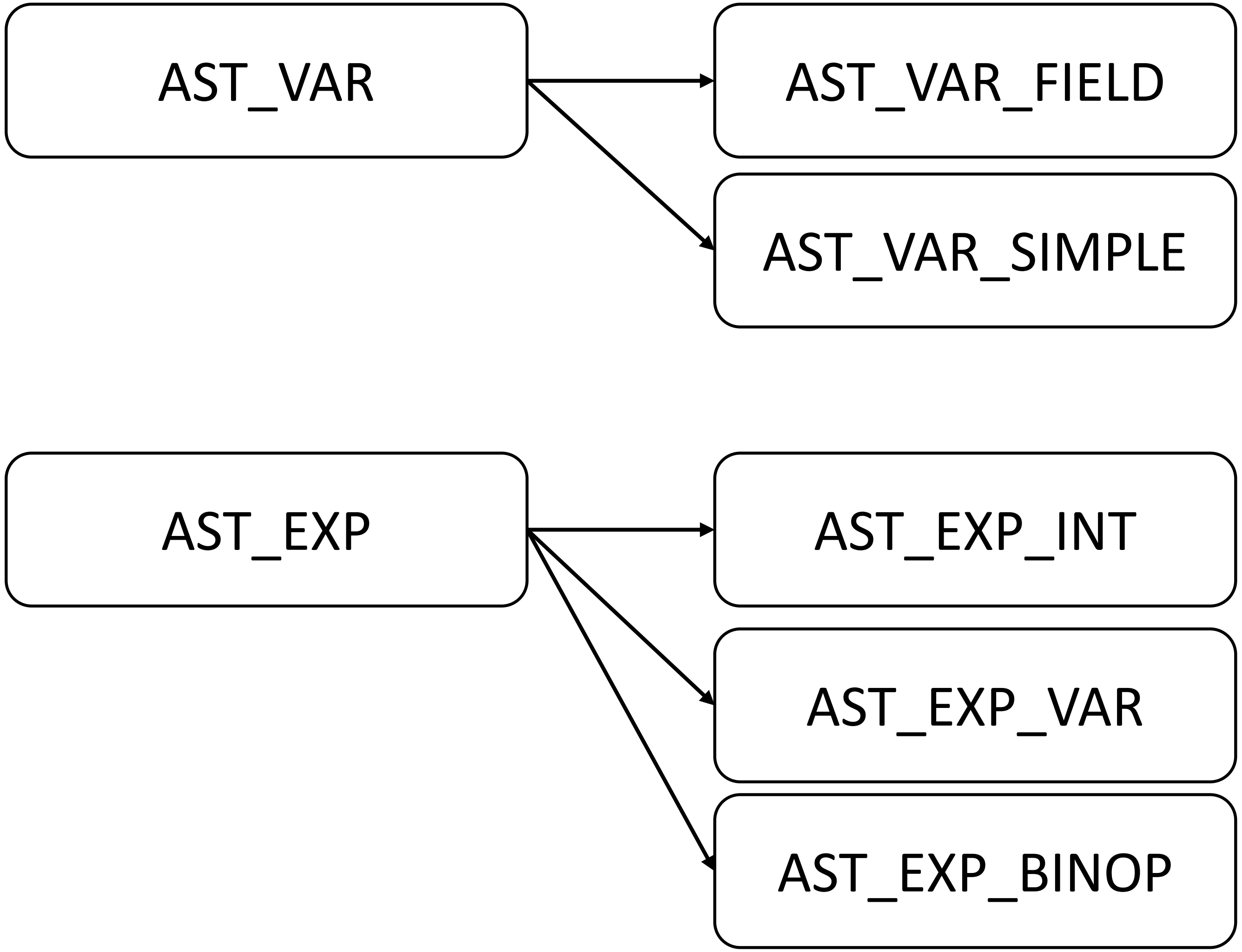
```
public class AST_EXP_VAR extends AST_EXP {
        public AST_VAR var;
        public AST_EXP_VAR(AST_VAR var) {
                this.var = var;
        }
}
```

CUP Example: AST Nodes

For the rule ***EXP ::= EXP:e1 <OP> EXP:e2*** :

```
public class AST_EXP_BINOP extends AST_EXP {
	int OP;
	public AST_EXP left;
	public AST_EXP right;
	public AST_EXP_BINOP(AST_EXP left, AST_EXP right, int OP) {
		this.left = left;
		this.right = right;
		this.OP = OP;
	}
}
```

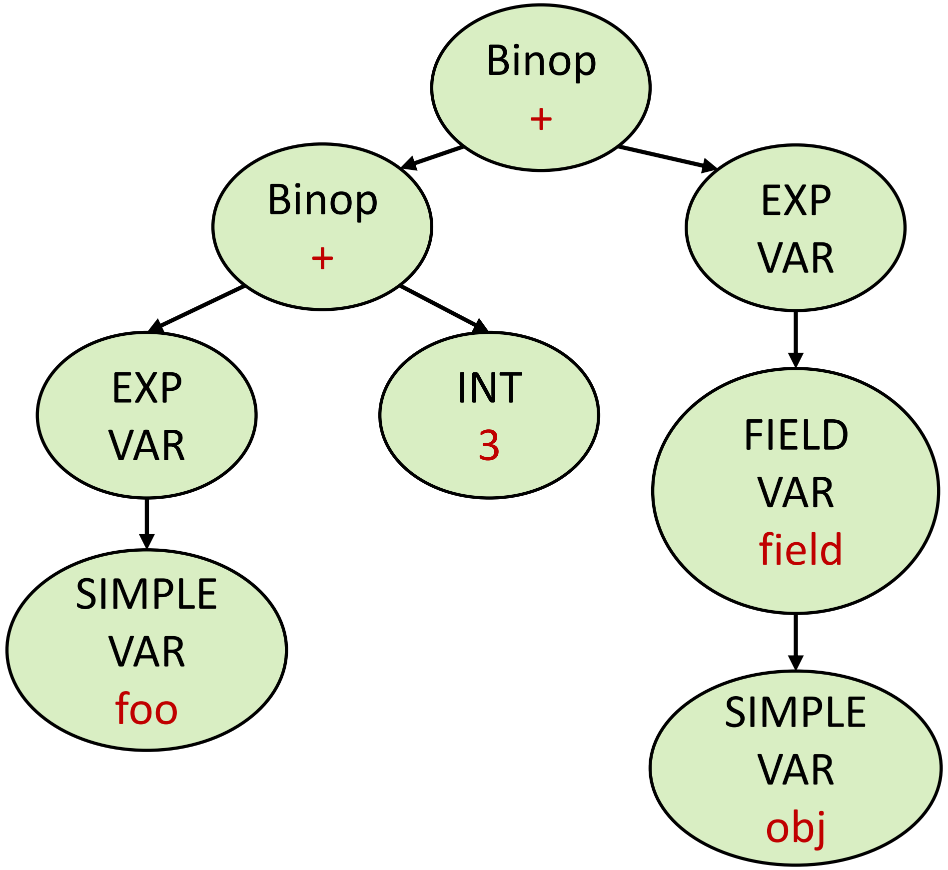
Class Hierarchy (Inheritance)



CUP Example: Debugging

We can generate an image of the AST (using the exercise template)

For the input foo + 3 + obj.field we have:



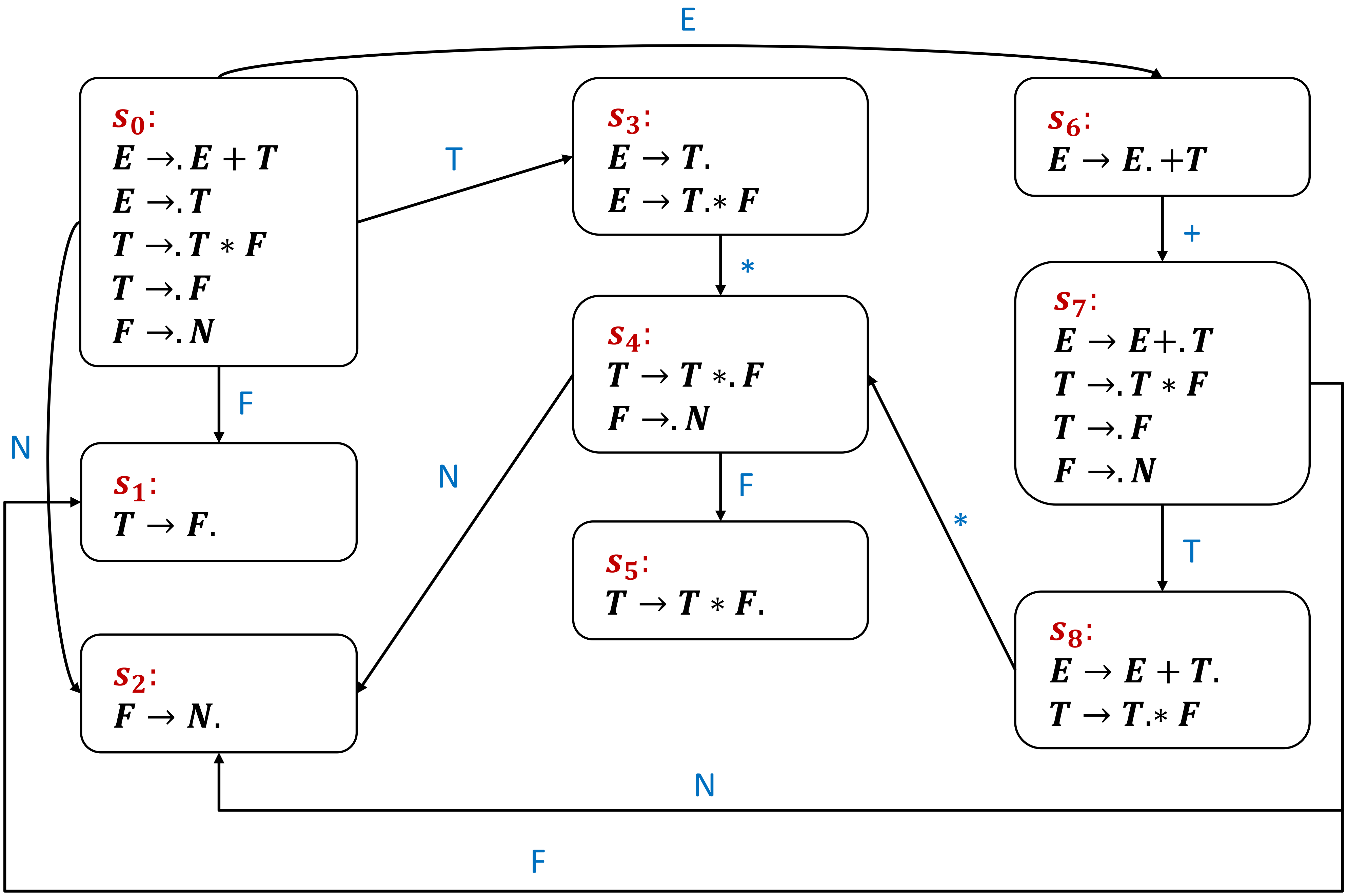
SLR(1), LR(1)

LR(0) Parsing

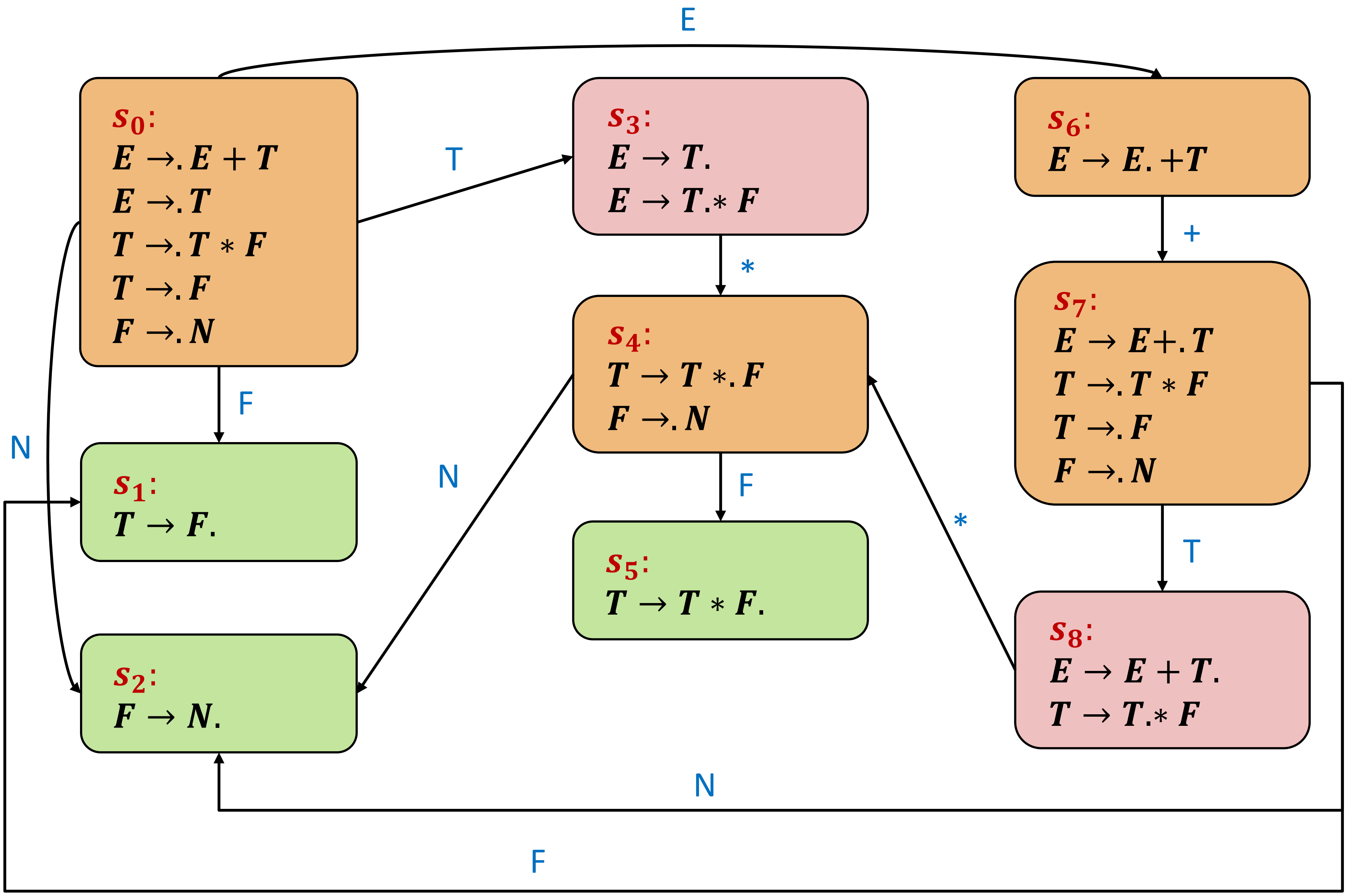
Consider the following CFG:

- $E \to E + T$
- $E \to T$
- $T \to T * F$
- $T \to F$
- $F \to N$

What will be the **transition system** of the LR(0) parser for this CFG?



E
T
F
N
N
*
F
*
+
T
N
F
s_0:
$E \to .E + T$
$E \to .T$
$T \to .T * F$
$T \to .F$
$F \to .N$
s_1:
$T \to F.$
s_2:
$F \to N.$
s_3:
$E \to T.$
$E \to T.* F$
s_4:
$T \to T *.F$
$F \to .N$
s_5:
$T \to T * F.$
s_6:
$E \to E.+T$
s_7:
$E \to E+.T$
$T \to .T * F$
$T \to .F$
$F \to .N$
s_8:
$E \to E + T.$
$T \to T.* F$



E
s_0:
$E \to .E + T$
$E \to .T$
$T \to .T * F$
$T \to .F$
$F \to .N$
T
s_3:
$E \to T.$
$E \to T.* F$
s_6:
$E \to E.+T$
+
*
s_7:
$E \to E+.T$
$T \to .T * F$
$T \to .F$
$F \to .N$
s_4:
$T \to T *.F$
$F \to .N$
F
N
N
s_1:
$T \to F.$
F
s_5:
$T \to T * F.$
*
T
s_2:
$F \to N.$
s_8:
$E \to E + T.$
$T \to T.* F$
N
F

LR(0) Conflict

- The conflict occurs when the next token is: $*$
- But E can be followed only by: $+\$$

- $E \rightarrow E + T$
- $E \rightarrow T$
- $T \rightarrow T * F$
- $T \rightarrow F$
- $F \rightarrow N$

$\boldsymbol{s_3}$:
$\boldsymbol{E \rightarrow T.}$
$\boldsymbol{E \rightarrow T. * F}$

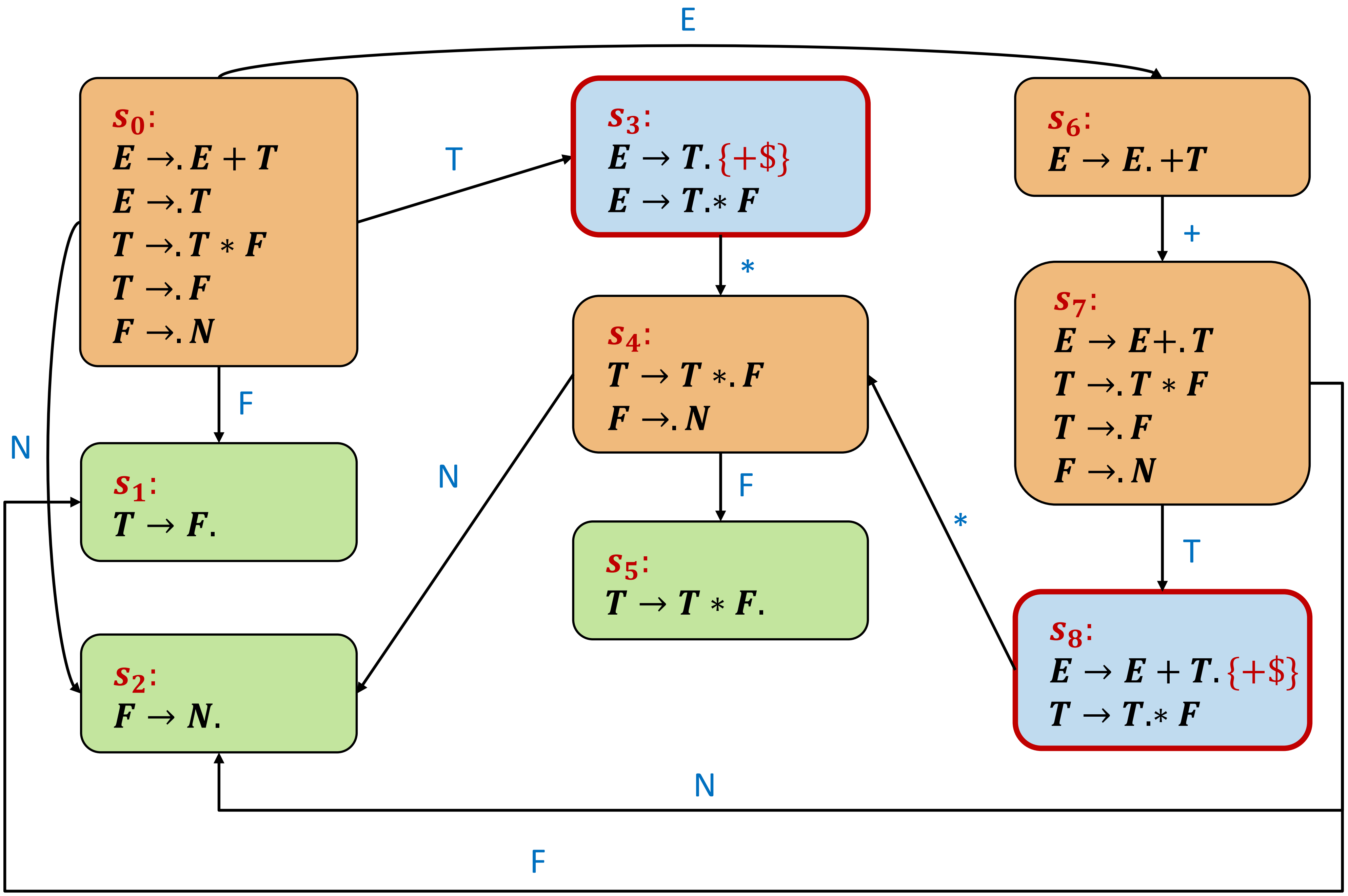
SLR(1)

- Same push-down automaton as in LR(0)
- But reduce items has a look-ahead set
  - $A \rightarrow \alpha . \{t_1, t_2, \ldots\}$
  - where $Follow(A) = \{t_1, t_2, \ldots\}$

SLR(1)

- Solve shift-reduce conflicts using the look-ahead token t
- If $Follow(Y) \cap First(\beta) = \emptyset$
  - If $t \in Follow(Y)$, apply the reduce
  - Otherwise, apply the shift

$$\boldsymbol{Y \rightarrow \gamma.\{\dots\}}$$
$$\boldsymbol{X \rightarrow \alpha.\beta}$$



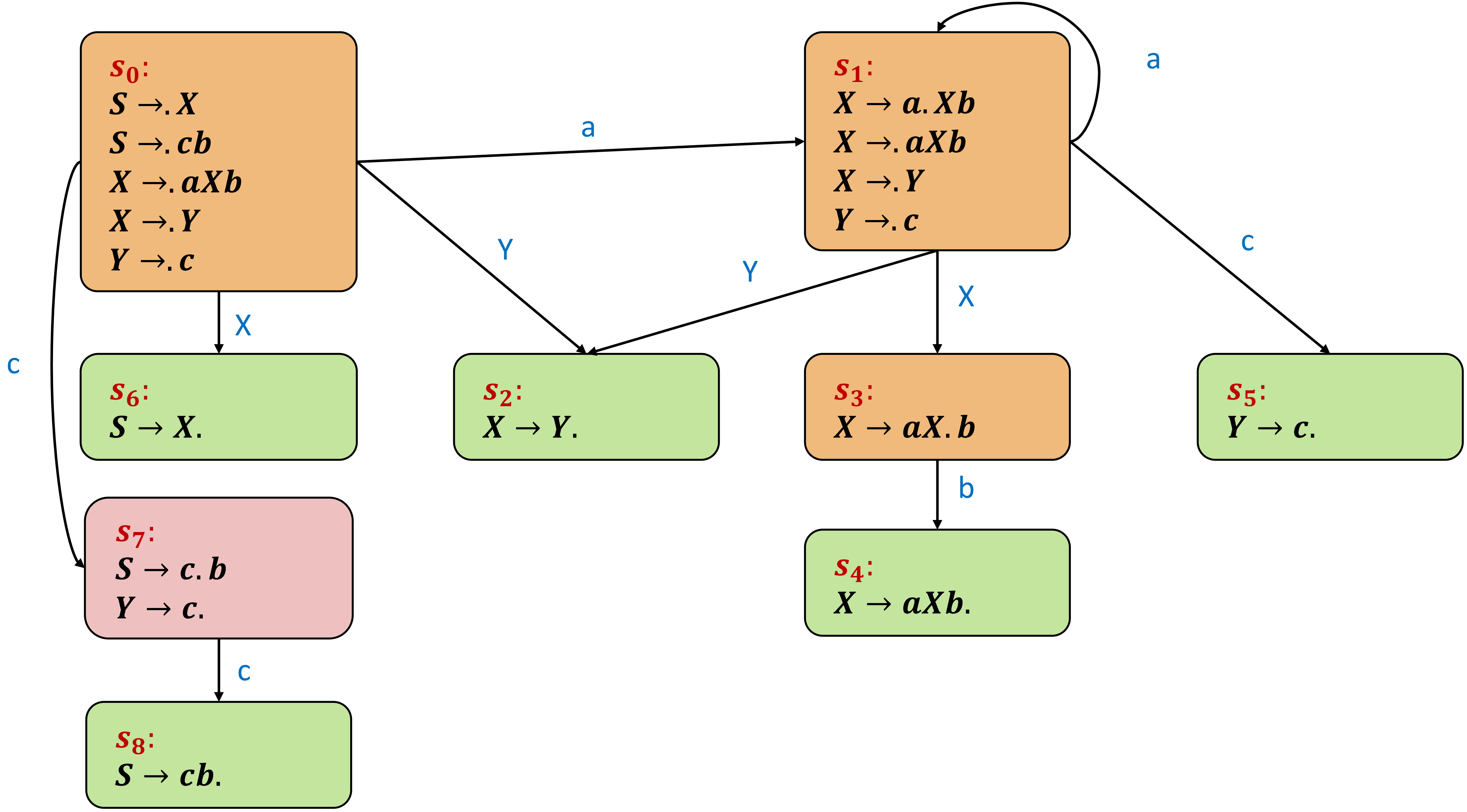
s_0:
$E \to .E + T$
$E \to .T$
$T \to .T * F$
$T \to .F$
$F \to .N$
s_1:
$T \to F.$
s_2:
$F \to N.$
s_3:
$E \to T.\ \{+\$\}$
$E \to T.* F$
s_4:
$T \to T *.F$
$F \to .N$
s_5:
$T \to T * F.$
s_6:
$E \to E.+T$
s_7:
$E \to E+.T$
$T \to .T * F$
$T \to .F$
$F \to .N$
s_8:
$E \to E + T.\ \{+\$\}$
$T \to T.* F$
E
T
F
N
*
+

SLR(1) Parsing

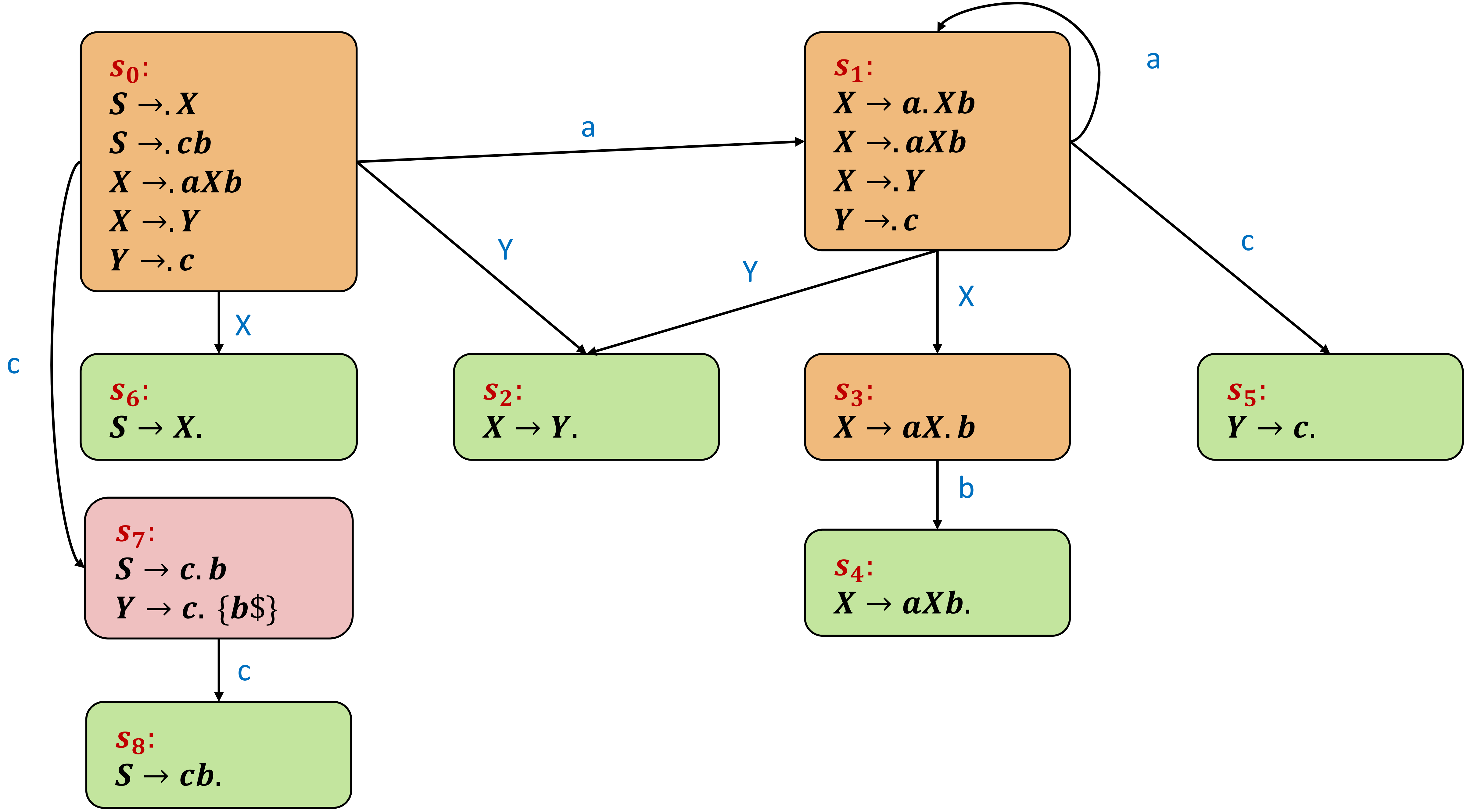
Consider the following CFG:

- $S \rightarrow X$
- $S \rightarrow cb$
- $X \rightarrow aXb$
- $X \rightarrow Y$
- $Y \rightarrow c$

What will be the **transition system** of the SLR(1) parser for this CFG?



s0:
S →. X
S →. cb
X →. aXb
X →. Y
Y →. c
s1:
X → a. Xb
X →. aXb
X →. Y
Y →. c
a
a
Y
Y
X
c
c
X
s6:
S → X.
s2:
X → Y.
s3:
X → aX. b
s5:
Y → c.
b
s7:
S → c. b
Y → c.
c
s4:
X → aXb.
s8:
S → cb.



s_0:
$S \to .X$
$S \to .cb$
$X \to .aXb$
$X \to .Y$
$Y \to .c$
s_1:
$X \to a.Xb$
$X \to .aXb$
$X \to .Y$
$Y \to .c$
a
a
Y
Y
X
X
c
c
c
b
s_6:
$S \to X.$
s_2:
$X \to Y.$
s_3:
$X \to aX.b$
s_5:
$Y \to c.$
s_7:
$S \to c.b$
$Y \to c.\ \{b\$\}$
s_4:
$X \to aXb.$
s_8:
$S \to cb.$

SLR(1) Conflict

- The conflict occurs when the next token is: b
- Relying on $Follow(Y)$
  - Considers all the occurrences of Y in **all the states / grammar**

- $S \rightarrow X$
- $S \rightarrow cb$
- $X \rightarrow aXb$
- $X \rightarrow Y$
- $Y \rightarrow c$

$\boldsymbol{s_7}$:
$\boldsymbol{S \rightarrow c.b}\ \{\$\}$
$\boldsymbol{Y \rightarrow c.}\ \{\boldsymbol{b}\$\}$

LR(1)

Maintain items with **more precise** look-ahead sets

An **LR(1) item** is of the form:

- $N \rightarrow \alpha . \beta \; \{\sigma\}$
- where $\sigma = t_1, t_2, \ldots$ (terminals)

LR(1) Item Closure Set

The **LR(1) closure set** of an LR(1) item i is a set S such that:

- $i \in S$
- If $A \rightarrow \alpha . N\beta\ \{\sigma\} \in S$ then for each rule $N \rightarrow \gamma$:
    - $N \rightarrow . \gamma\{\tau\} \in S$, where $\tau = First(\beta, \{\sigma\})$

Definition for $First(\beta, \{\sigma\})$:

- If β is not nullable:
    - $First(\beta)$
- Otherwise:
    - $(First(\beta) \cup \{\sigma\}) \setminus \{\epsilon\}$

LR(1) Parsing

Consider the following CFG:

- $S \rightarrow X$
- $S \rightarrow cb$
- $X \rightarrow aXb$
- $X \rightarrow Y$
- $Y \rightarrow c$

What will be the **transition system** of the LR(1) parser for this CFG?

s_0:

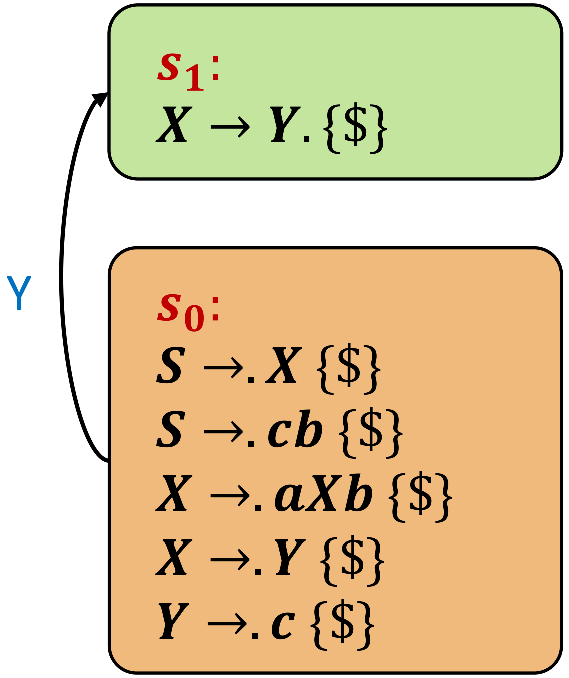
$S \rightarrow .X \ \{\$\}$

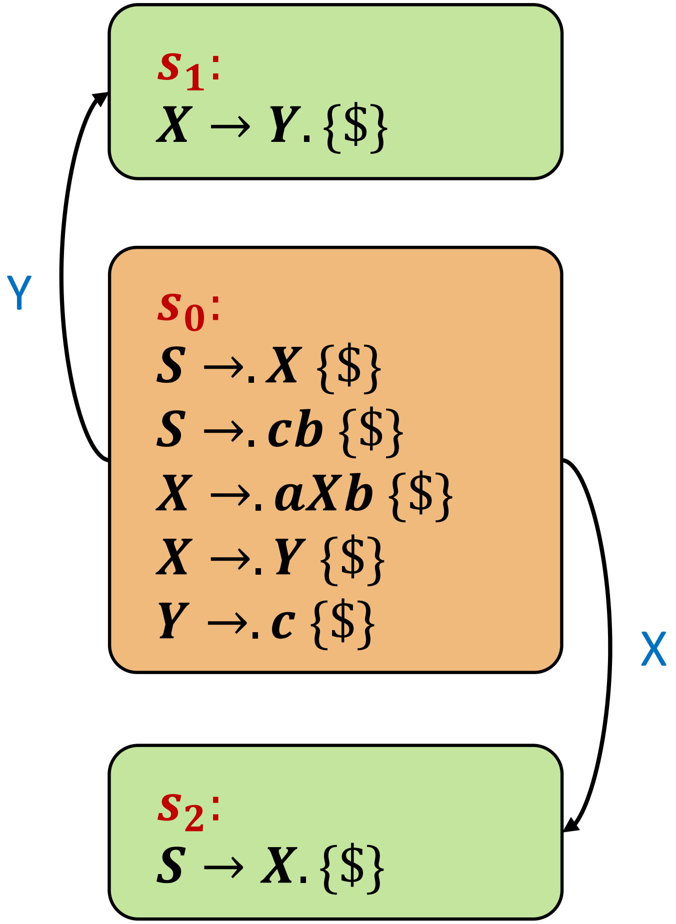
$S \rightarrow .cb \ \{\$\}$

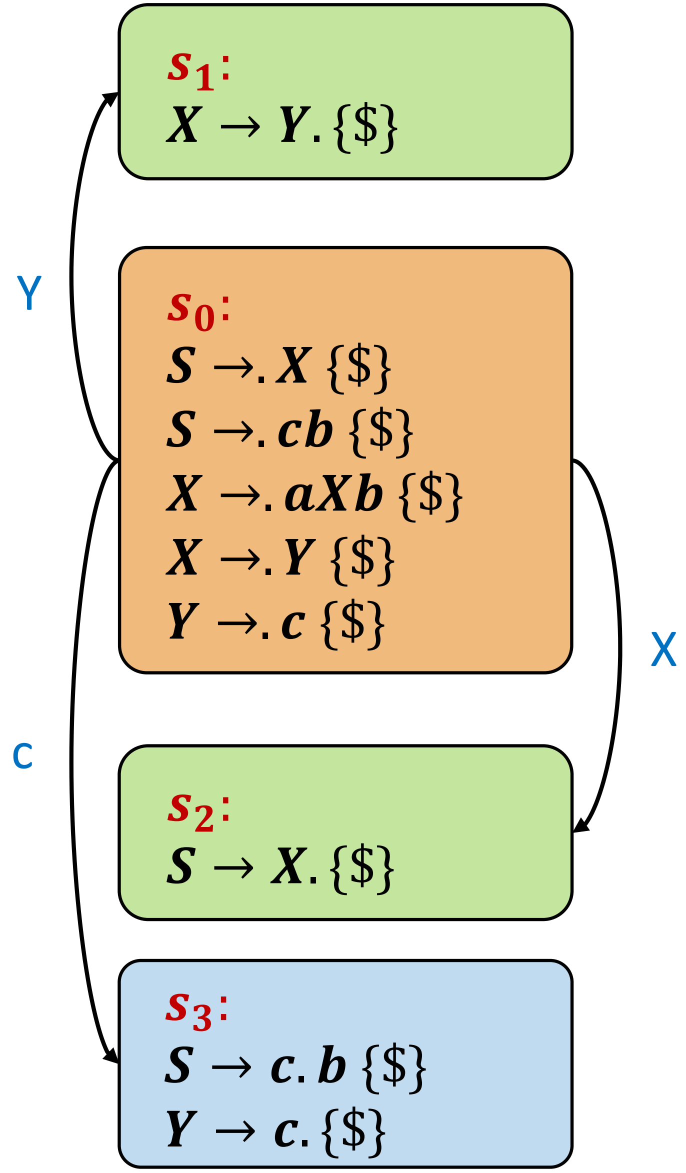
$X \rightarrow .aXb \ \{\$\}$

$X \rightarrow .Y \ \{\$\}$

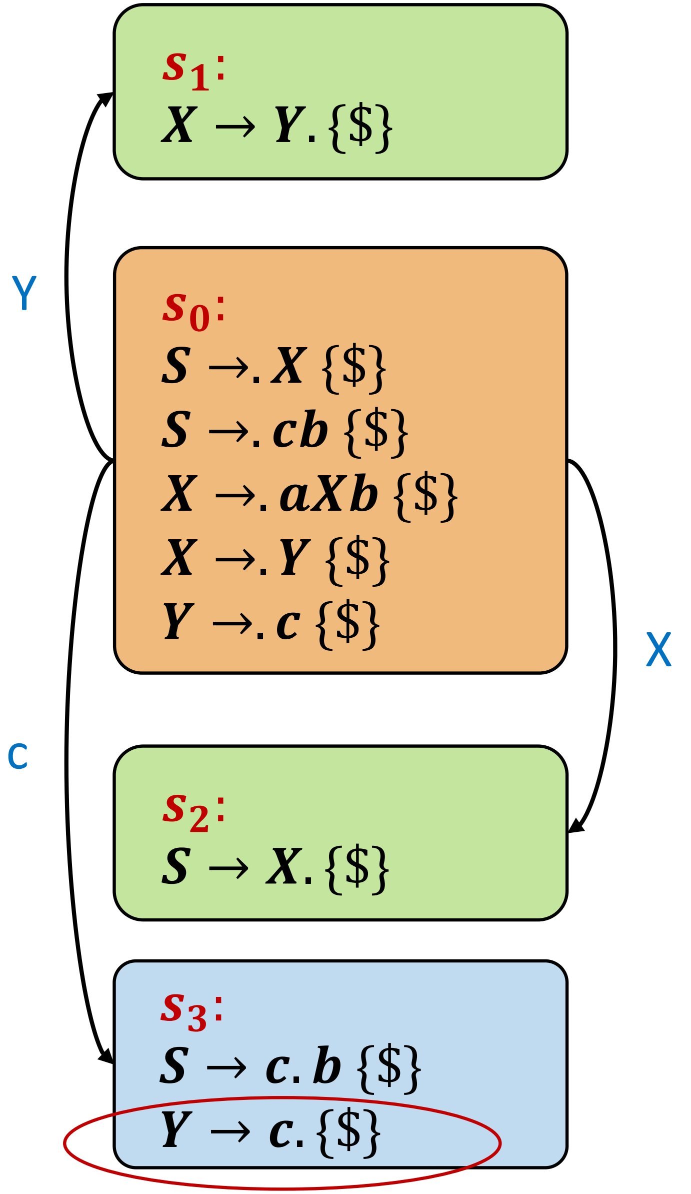
$Y \rightarrow .c \ \{\$\}$


s_1:
$X \rightarrow Y.\ \{\$\}$
Y
s_0:
$S \rightarrow .X\ \{\$\}$
$S \rightarrow .cb\ \{\$\}$
$X \rightarrow .aXb\ \{\$\}$
$X \rightarrow .Y\ \{\$\}$
$Y \rightarrow .c\ \{\$\}$

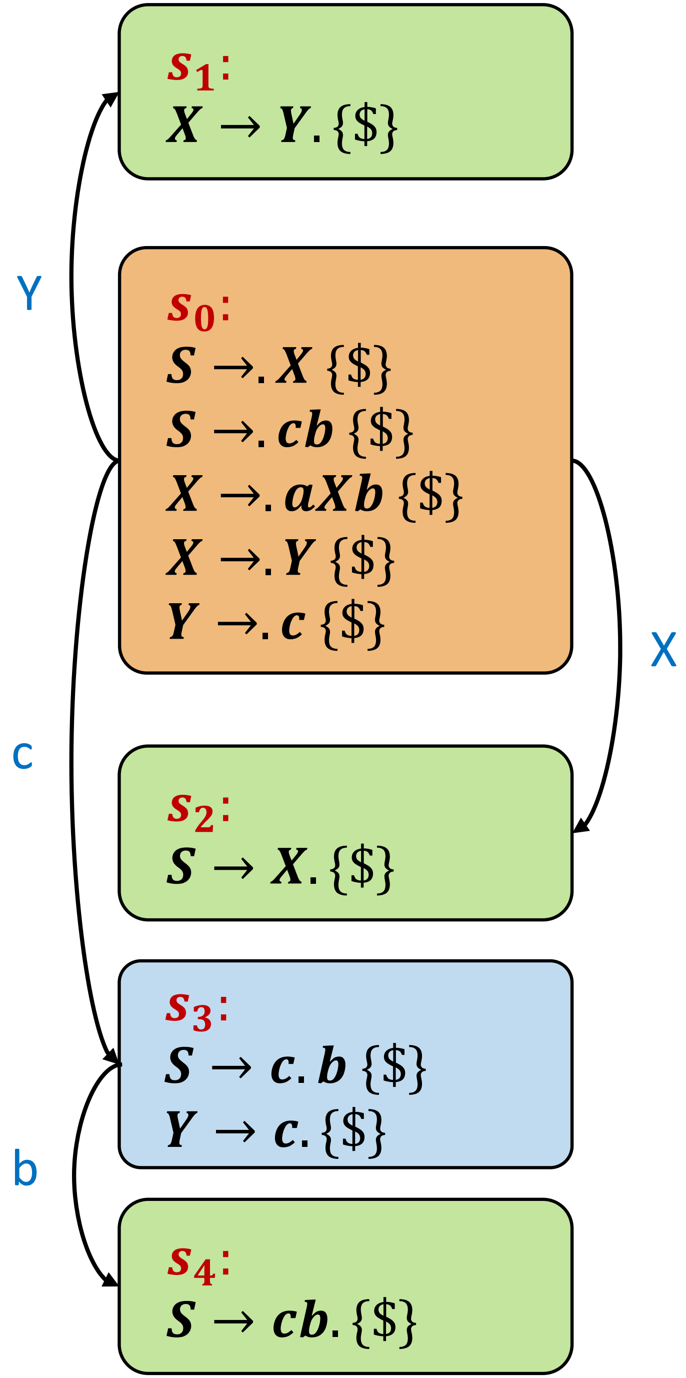

s_1:
$X \to Y.\{\$\}$
Y
s_0:
$S \to .X\ \{\$\}$
$S \to .cb\ \{\$\}$
$X \to .aXb\ \{\$\}$
$X \to .Y\ \{\$\}$
$Y \to .c\ \{\$\}$
X
s_2:
$S \to X.\{\$\}$



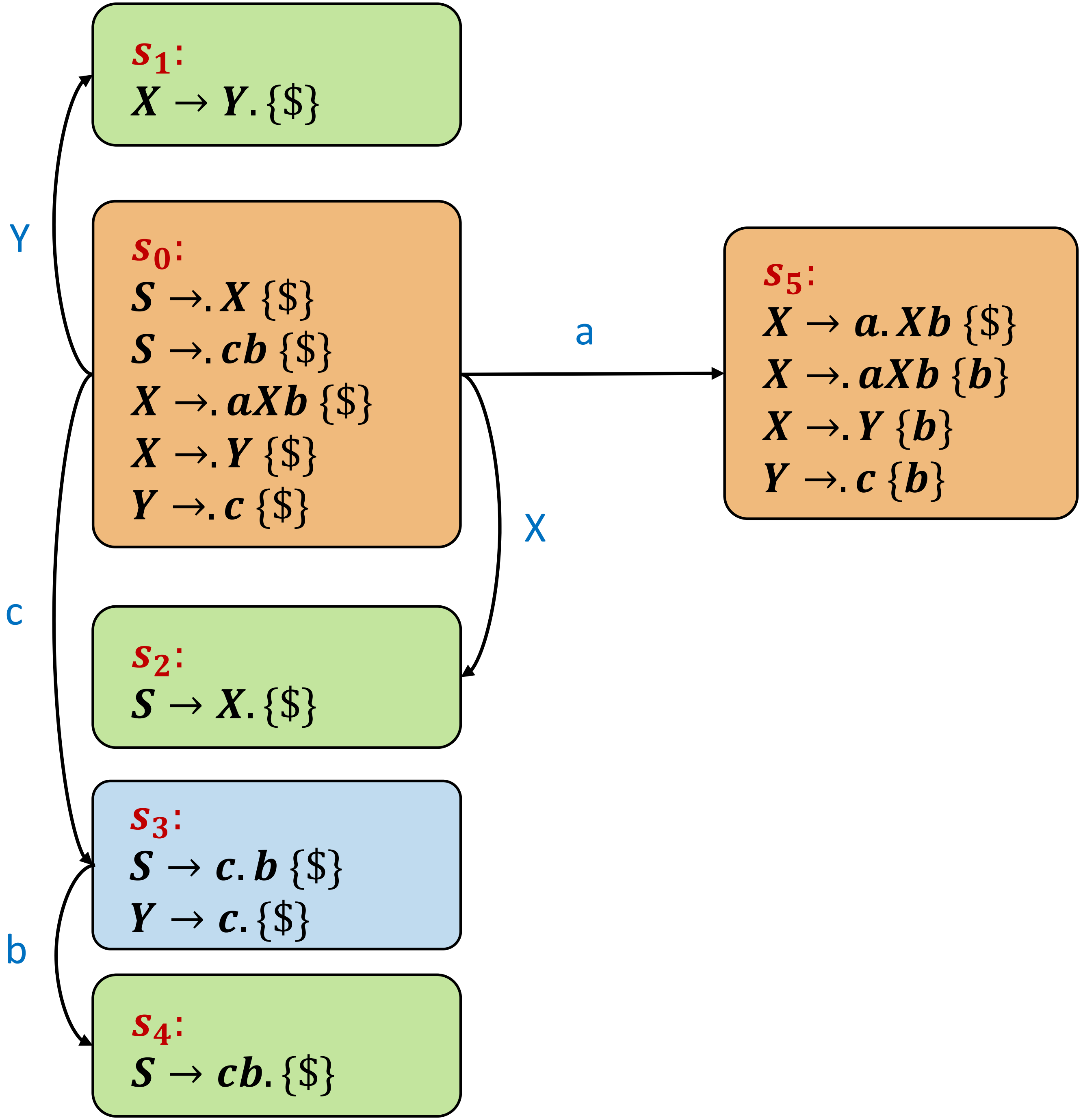
s_1:
$X \to Y.\{\$\}$
Y
s_0:
$S \to .X\ \{\$\}$
$S \to .cb\ \{\$\}$
$X \to .aXb\ \{\$\}$
$X \to .Y\ \{\$\}$
$Y \to .c\ \{\$\}$
X
c
s_2:
$S \to X.\{\$\}$
s_3:
$S \to c.b\ \{\$\}$
$Y \to c.\{\$\}$



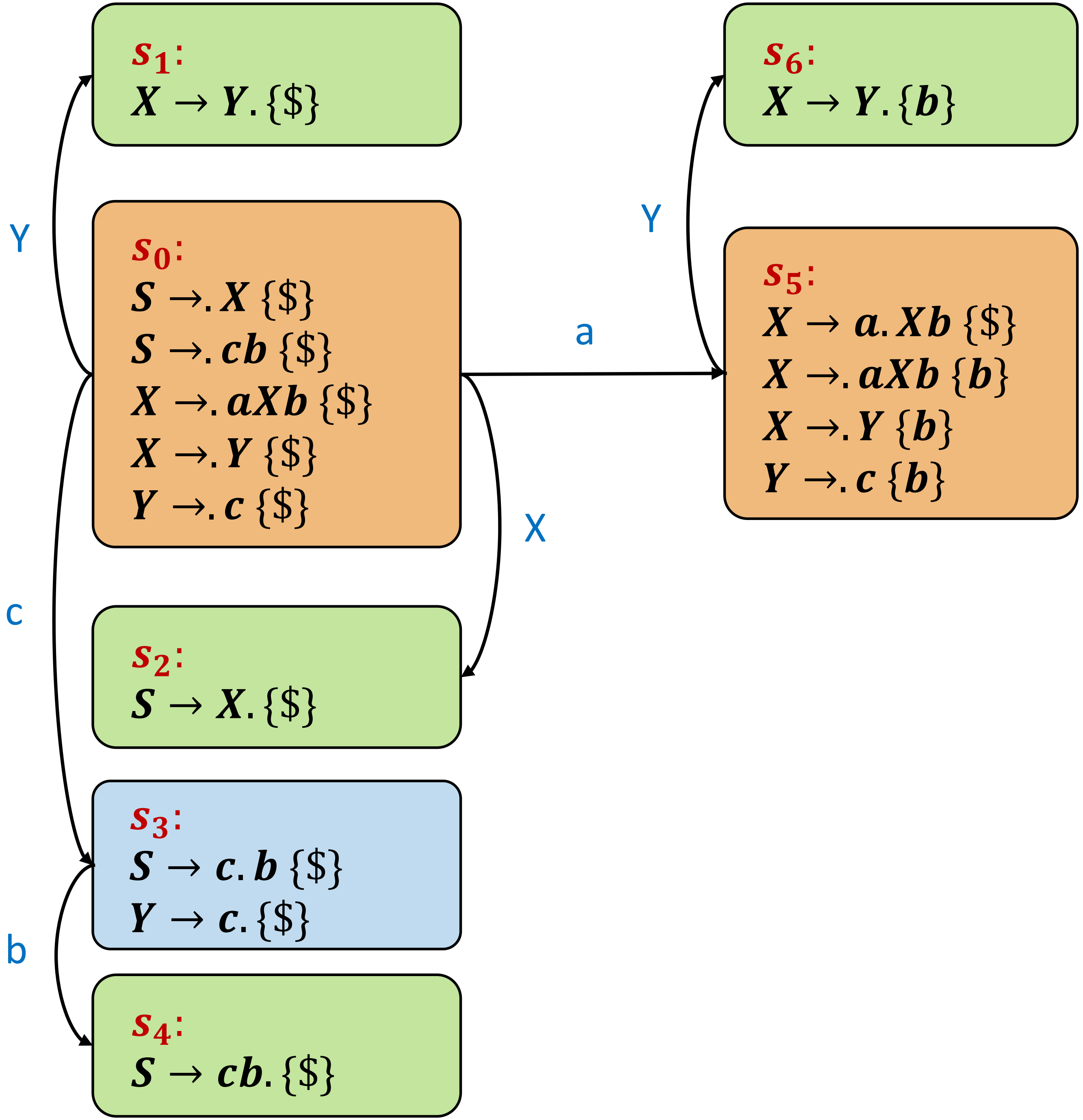
In SLR(1) we had: {b$}

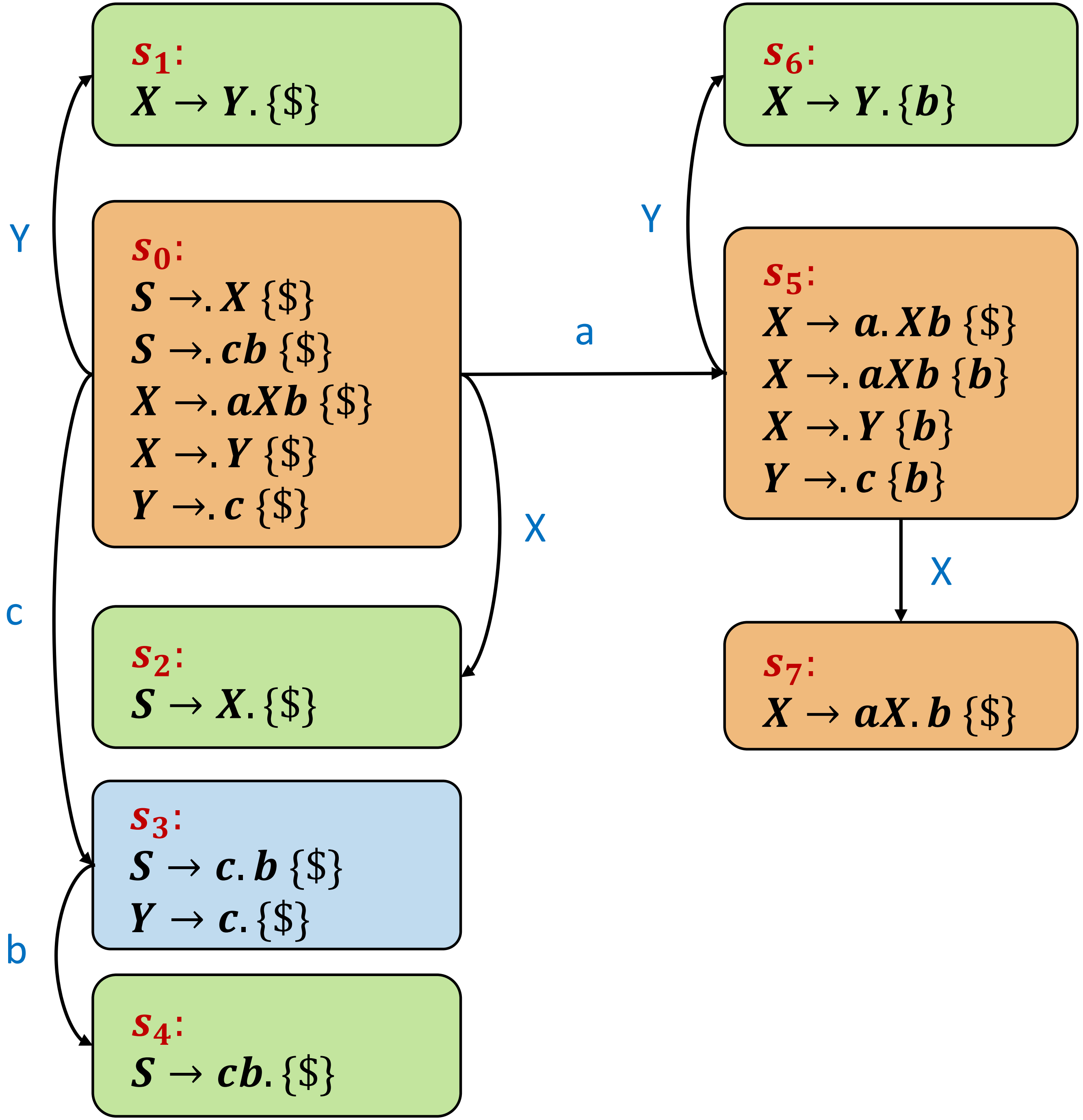


s_1:
$X \rightarrow Y. \{\$\}$
s_0:
$S \rightarrow .X \{\$\}$
$S \rightarrow .cb \{\$\}$
$X \rightarrow .aXb \{\$\}$
$X \rightarrow .Y \{\$\}$
$Y \rightarrow .c \{\$\}$
s_2:
$S \rightarrow X. \{\$\}$
s_3:
$S \rightarrow c.b \{\$\}$
$Y \rightarrow c. \{\$\}$
s_4:
$S \rightarrow cb. \{\$\}$
Y
X
c
b

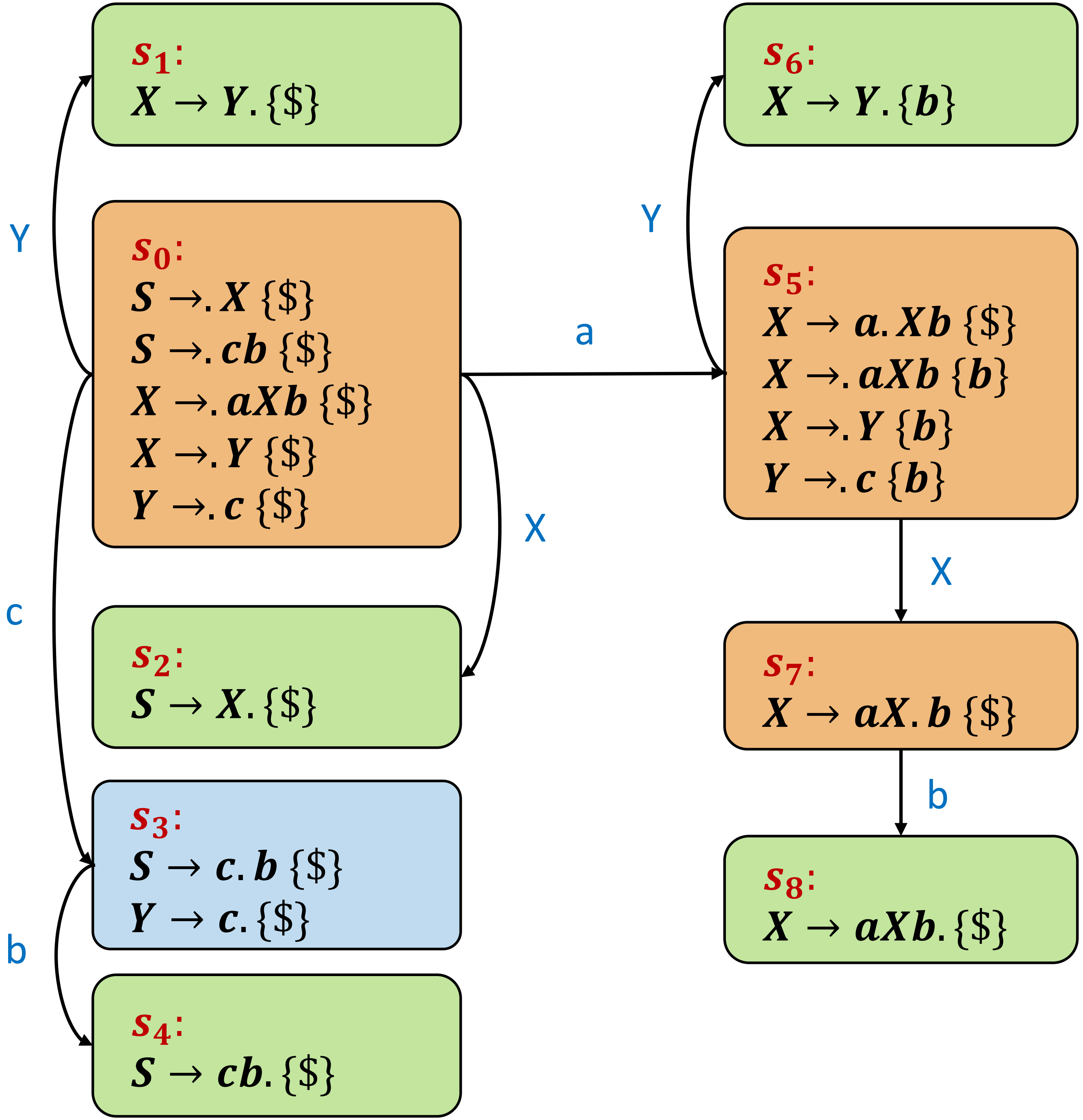


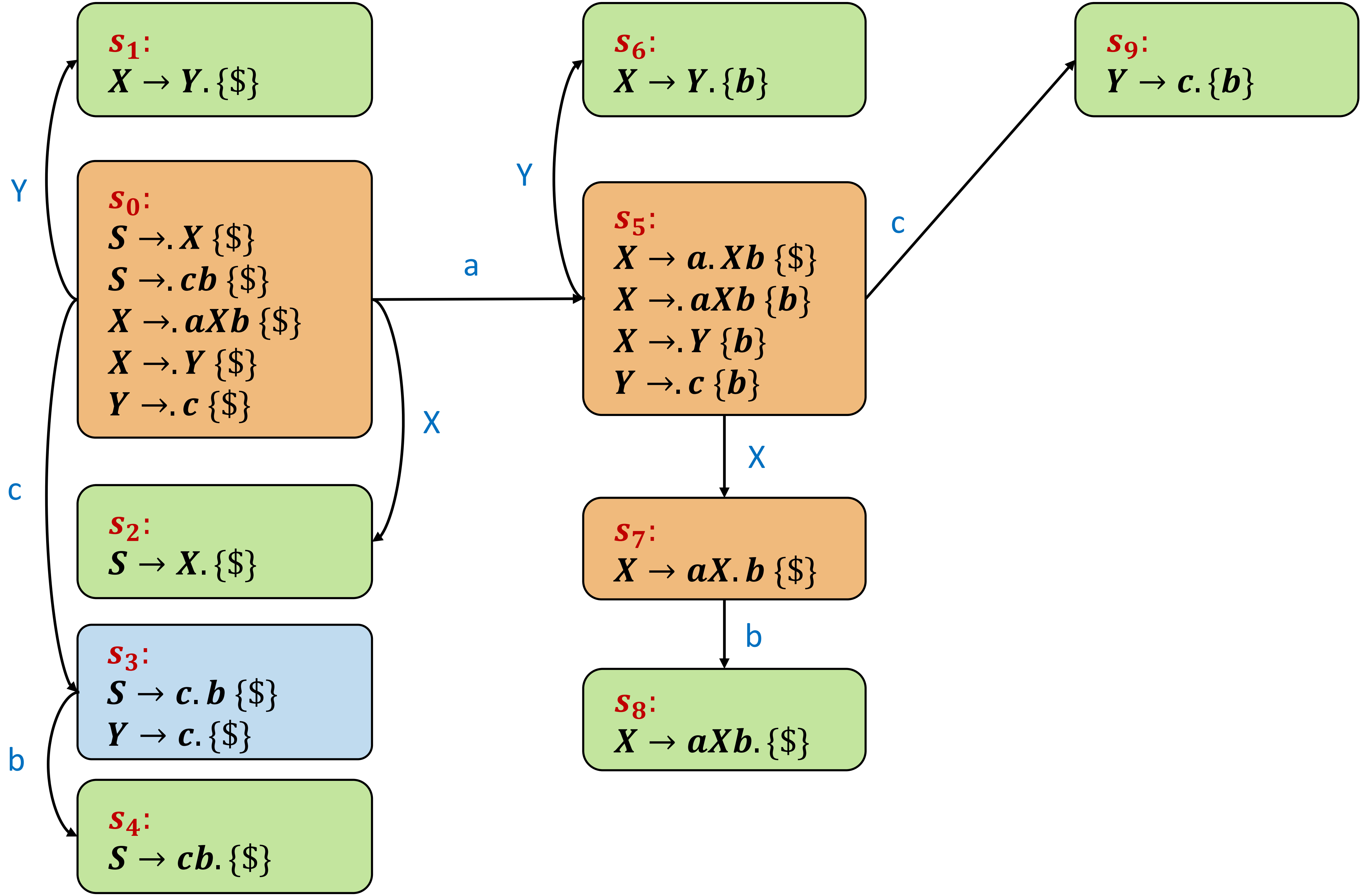
s_1:
$X \rightarrow Y.\ \{\$\}$
s_0:
$S \rightarrow .X\ \{\$\}$
$S \rightarrow .cb\ \{\$\}$
$X \rightarrow .aXb\ \{\$\}$
$X \rightarrow .Y\ \{\$\}$
$Y \rightarrow .c\ \{\$\}$
s_5:
$X \rightarrow a.Xb\ \{\$\}$
$X \rightarrow .aXb\ \{b\}$
$X \rightarrow .Y\ \{b\}$
$Y \rightarrow .c\ \{b\}$
s_2:
$S \rightarrow X.\ \{\$\}$
s_3:
$S \rightarrow c.b\ \{\$\}$
$Y \rightarrow c.\ \{\$\}$
s_4:
$S \rightarrow cb.\ \{\$\}$
Y
a
X
c
b


s_1:
$X \rightarrow Y.\{\$\}$
s_0:
$S \rightarrow .X\ \{\$\}$
$S \rightarrow .cb\ \{\$\}$
$X \rightarrow .aXb\ \{\$\}$
$X \rightarrow .Y\ \{\$\}$
$Y \rightarrow .c\ \{\$\}$
s_2:
$S \rightarrow X.\{\$\}$
s_3:
$S \rightarrow c.b\ \{\$\}$
$Y \rightarrow c.\{\$\}$
s_4:
$S \rightarrow cb.\{\$\}$
s_6:
$X \rightarrow Y.\{b\}$
s_5:
$X \rightarrow a.Xb\ \{\$\}$
$X \rightarrow .aXb\ \{b\}$
$X \rightarrow .Y\ \{b\}$
$Y \rightarrow .c\ \{b\}$
Y
c
b
a
X
Y

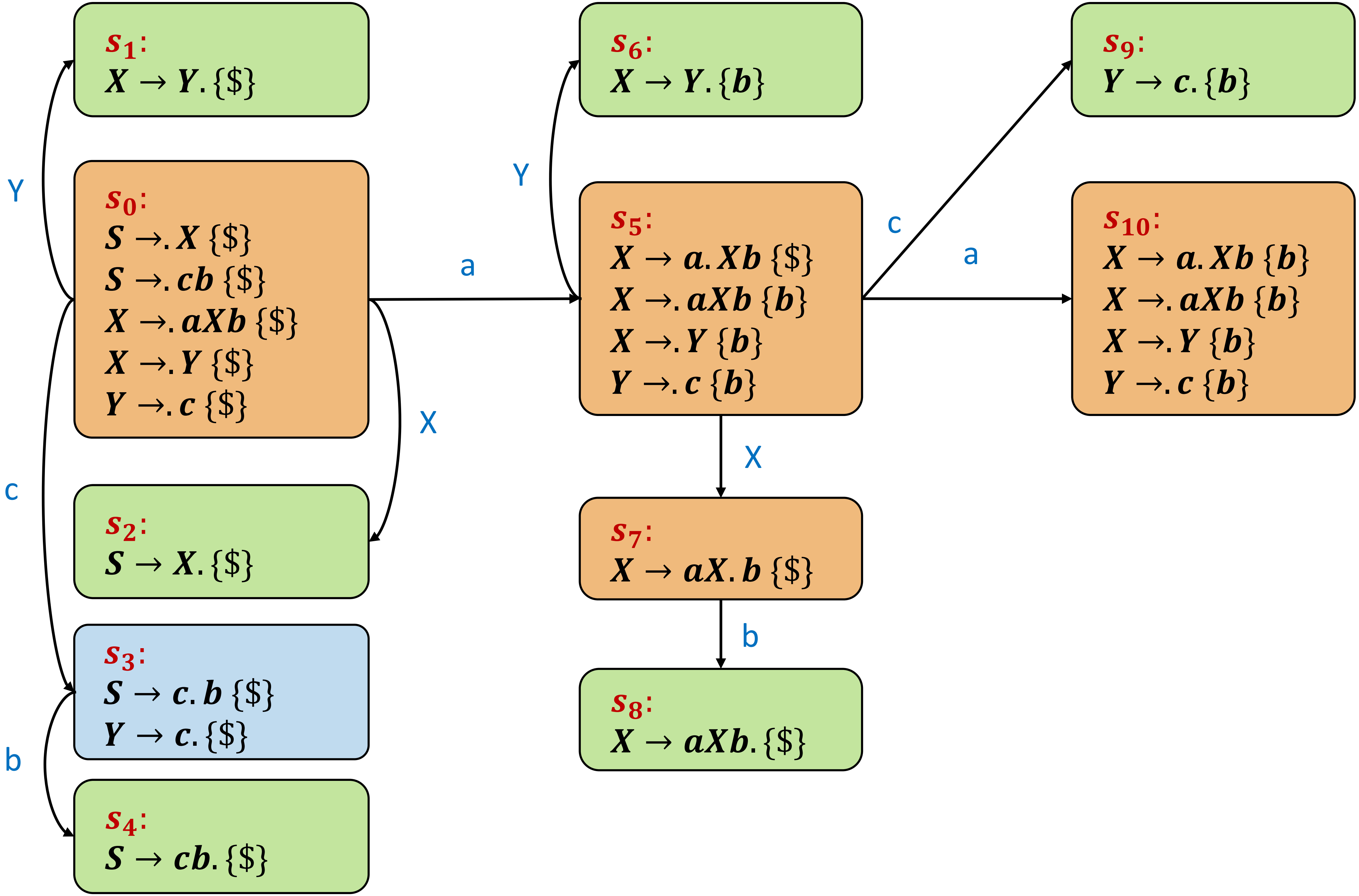


s_1: $X \to Y. \{\$\}$
s_0: $S \to .X \{\$\}$ $S \to .cb \{\$\}$ $X \to .aXb \{\$\}$ $X \to .Y \{\$\}$ $Y \to .c \{\$\}$
s_2: $S \to X. \{\$\}$
s_3: $S \to c.b \{\$\}$ $Y \to c. \{\$\}$
s_4: $S \to cb. \{\$\}$
s_6: $X \to Y. \{b\}$
s_5: $X \to a.Xb \{\$\}$ $X \to .aXb \{b\}$ $X \to .Y \{b\}$ $Y \to .c \{b\}$
s_7: $X \to aX.b \{\$\}$
Y
a
X
c
b
Y
X

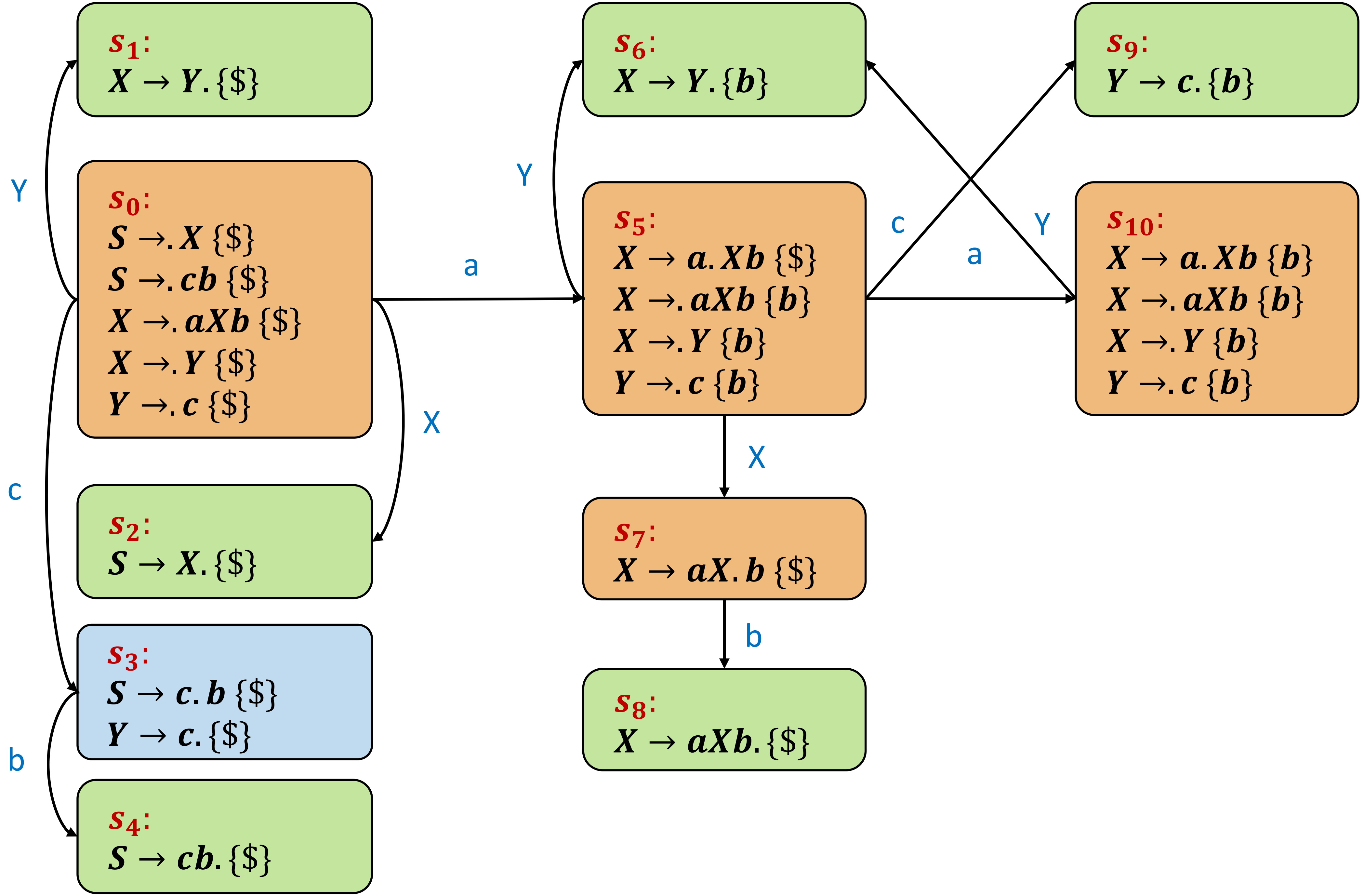

s_1: $X \to Y.\{\$\}$
s_0: $S \to .X\ \{\$\}$ $S \to .cb\ \{\$\}$ $X \to .aXb\ \{\$\}$ $X \to .Y\ \{\$\}$ $Y \to .c\ \{\$\}$
s_2: $S \to X.\{\$\}$
s_3: $S \to c.b\ \{\$\}$ $Y \to c.\{\$\}$
s_4: $S \to cb.\{\$\}$
s_6: $X \to Y.\{b\}$
s_5: $X \to a.Xb\ \{\$\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
s_7: $X \to aX.b\ \{\$\}$
s_8: $X \to aXb.\{\$\}$
Y
c
b
a
X
Y
X
b



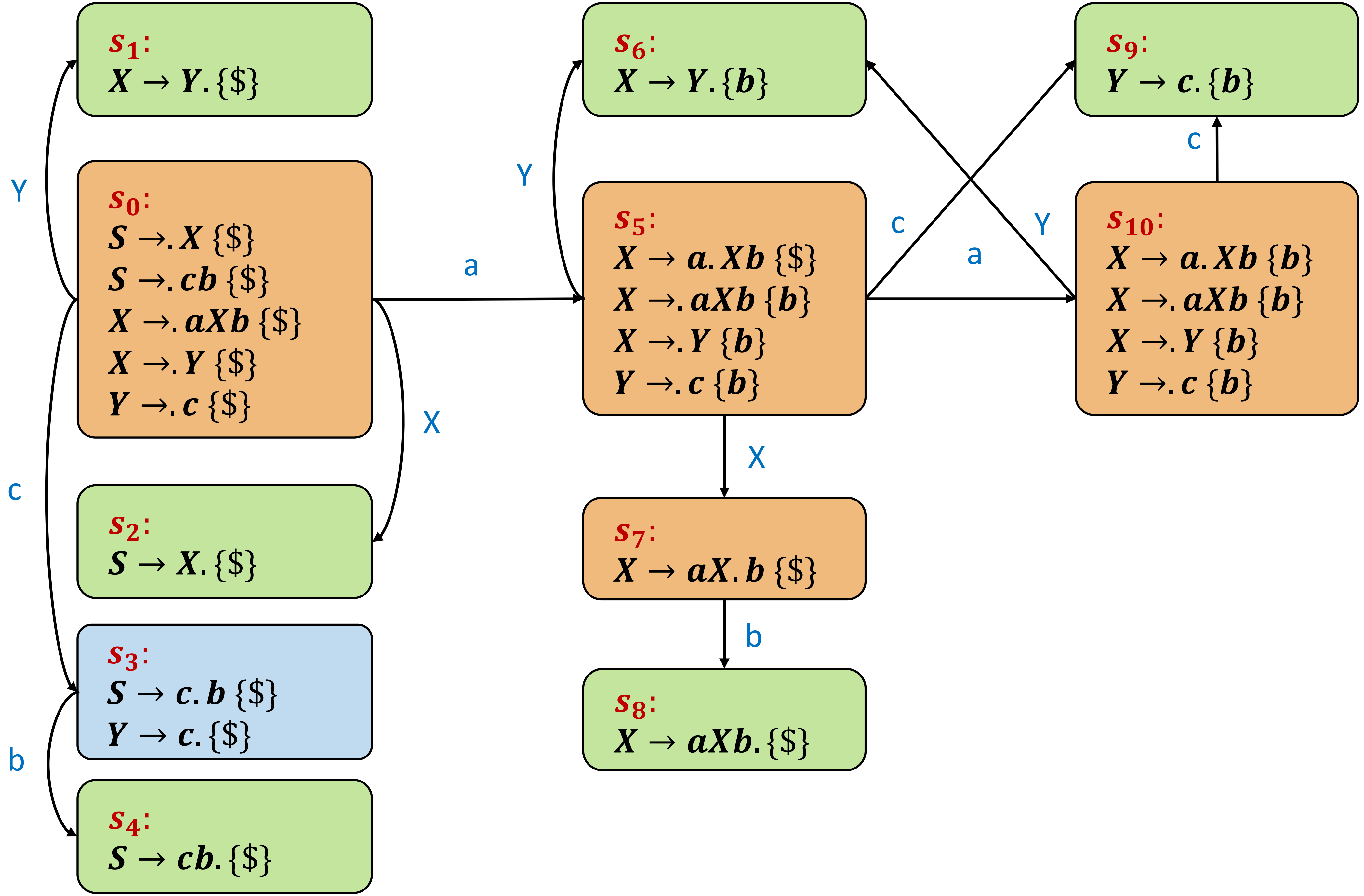
s_0:
$S \to .X\ \{\$\}$
$S \to .cb\ \{\$\}$
$X \to .aXb\ \{\$\}$
$X \to .Y\ \{\$\}$
$Y \to .c\ \{\$\}$
s_1:
$X \to Y.\ \{\$\}$
s_2:
$S \to X.\ \{\$\}$
s_3:
$S \to c.b\ \{\$\}$
$Y \to c.\ \{\$\}$
s_4:
$S \to cb.\ \{\$\}$
s_5:
$X \to a.Xb\ \{\$\}$
$X \to .aXb\ \{b\}$
$X \to .Y\ \{b\}$
$Y \to .c\ \{b\}$
s_6:
$X \to Y.\ \{b\}$
s_7:
$X \to aX.b\ \{\$\}$
s_8:
$X \to aXb.\ \{\$\}$
s_9:
$Y \to c.\ \{b\}$
Y
X
c
b
a
Y
c
X
b



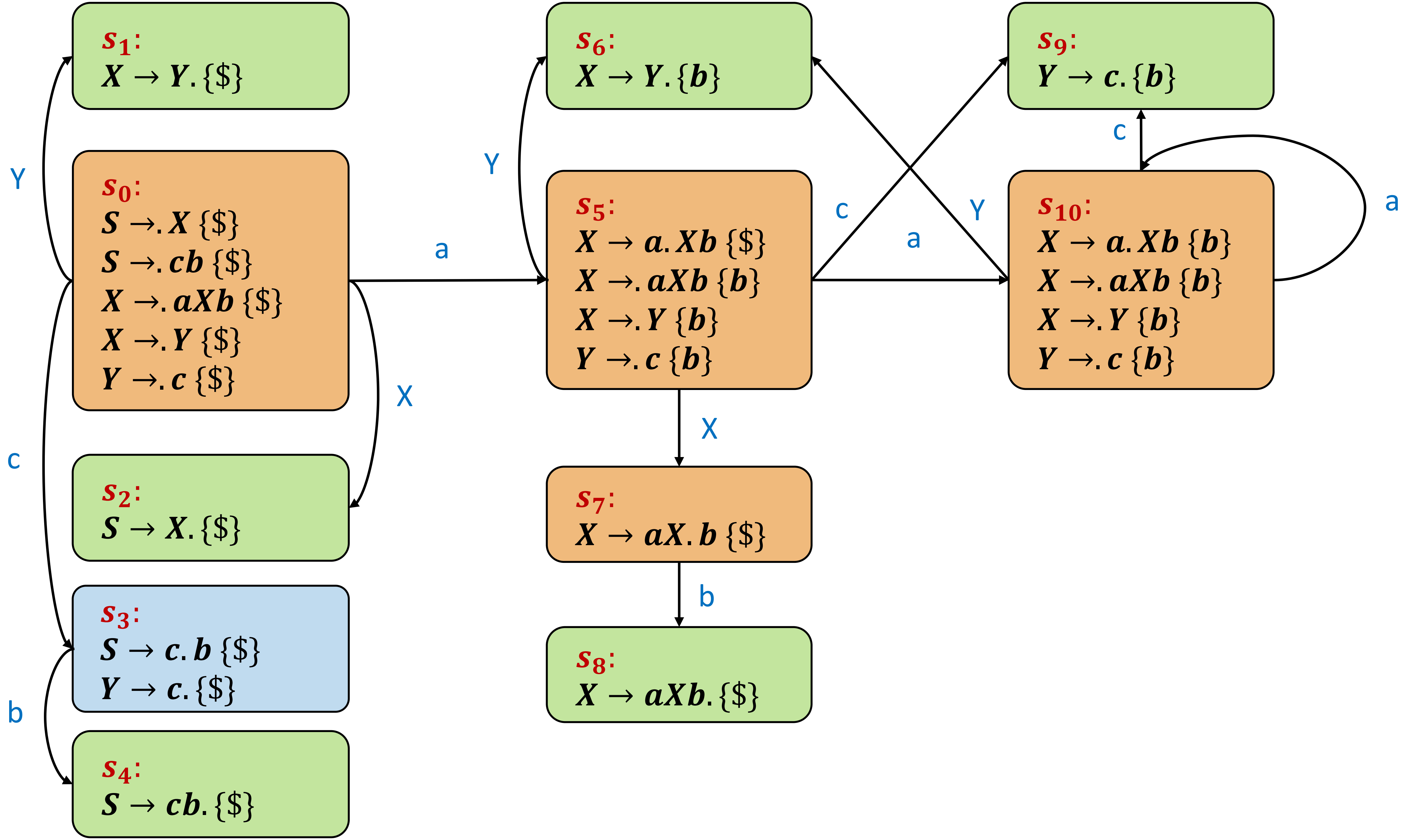
s_1: $X \to Y.\{\$\}$
s_0: $S \to .X\ \{\$\}$ $S \to .cb\ \{\$\}$ $X \to .aXb\ \{\$\}$ $X \to .Y\ \{\$\}$ $Y \to .c\ \{\$\}$
s_2: $S \to X.\{\$\}$
s_3: $S \to c.b\ \{\$\}$ $Y \to c.\{\$\}$
s_4: $S \to cb.\{\$\}$
s_5: $X \to a.Xb\ \{\$\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
s_6: $X \to Y.\{b\}$
s_7: $X \to aX.b\ \{\$\}$
s_8: $X \to aXb.\{\$\}$
s_9: $Y \to c.\{b\}$
s_{10}: $X \to a.Xb\ \{b\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
Y
a
X
c
b
Y
c
a
X
b



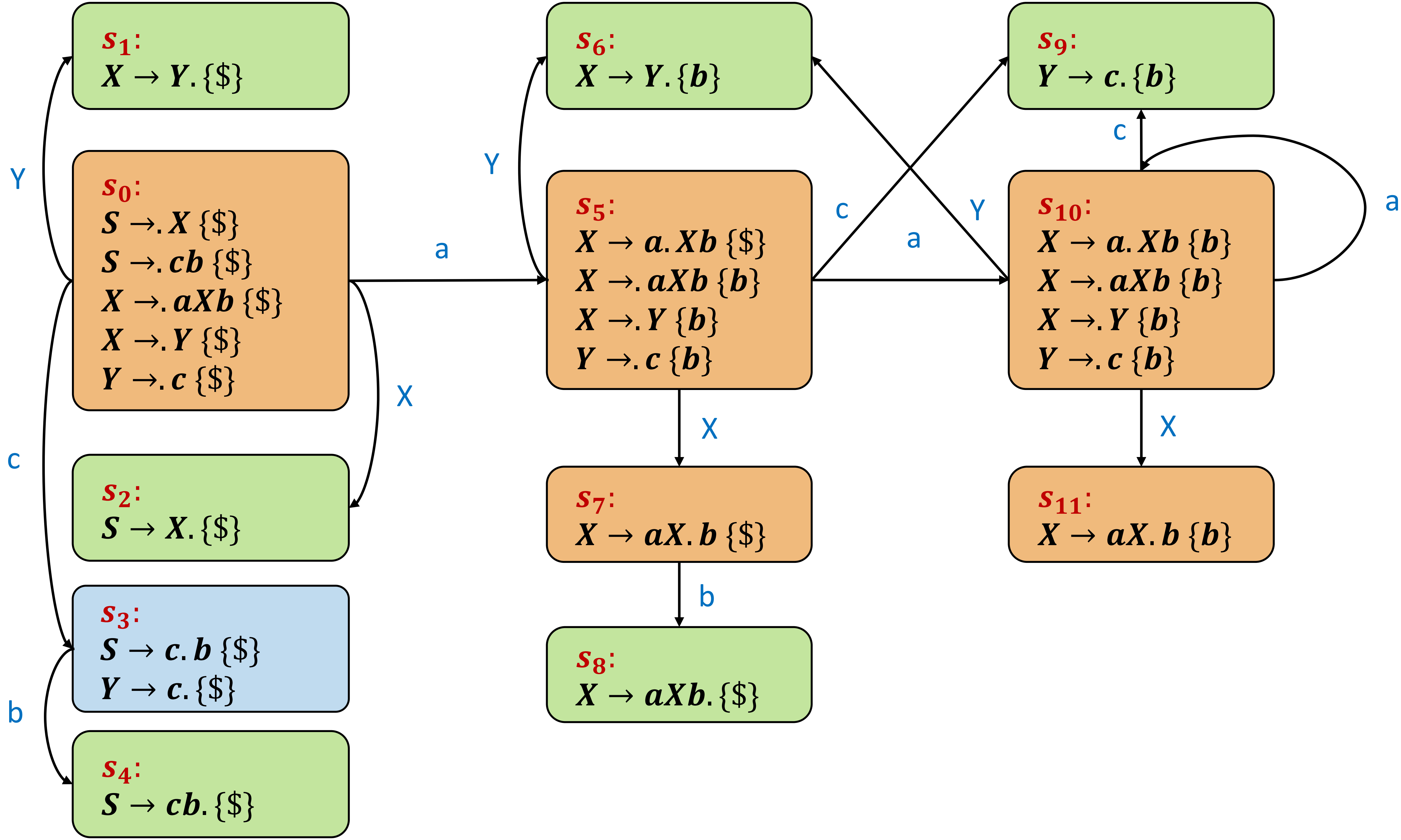
s_0:
$S \to .X\ \{\$\}$
$S \to .cb\ \{\$\}$
$X \to .aXb\ \{\$\}$
$X \to .Y\ \{\$\}$
$Y \to .c\ \{\$\}$
s_1:
$X \to Y.\{\$\}$
s_2:
$S \to X.\{\$\}$
s_3:
$S \to c.b\ \{\$\}$
$Y \to c.\{\$\}$
s_4:
$S \to cb.\{\$\}$
s_5:
$X \to a.Xb\ \{\$\}$
$X \to .aXb\ \{b\}$
$X \to .Y\ \{b\}$
$Y \to .c\ \{b\}$
s_6:
$X \to Y.\{b\}$
s_7:
$X \to aX.b\ \{\$\}$
s_8:
$X \to aXb.\{\$\}$
s_9:
$Y \to c.\{b\}$
s_{10}:
$X \to a.Xb\ \{b\}$
$X \to .aXb\ \{b\}$
$X \to .Y\ \{b\}$
$Y \to .c\ \{b\}$
Y
c
b
X
a
Y
X
b
c
a
Y

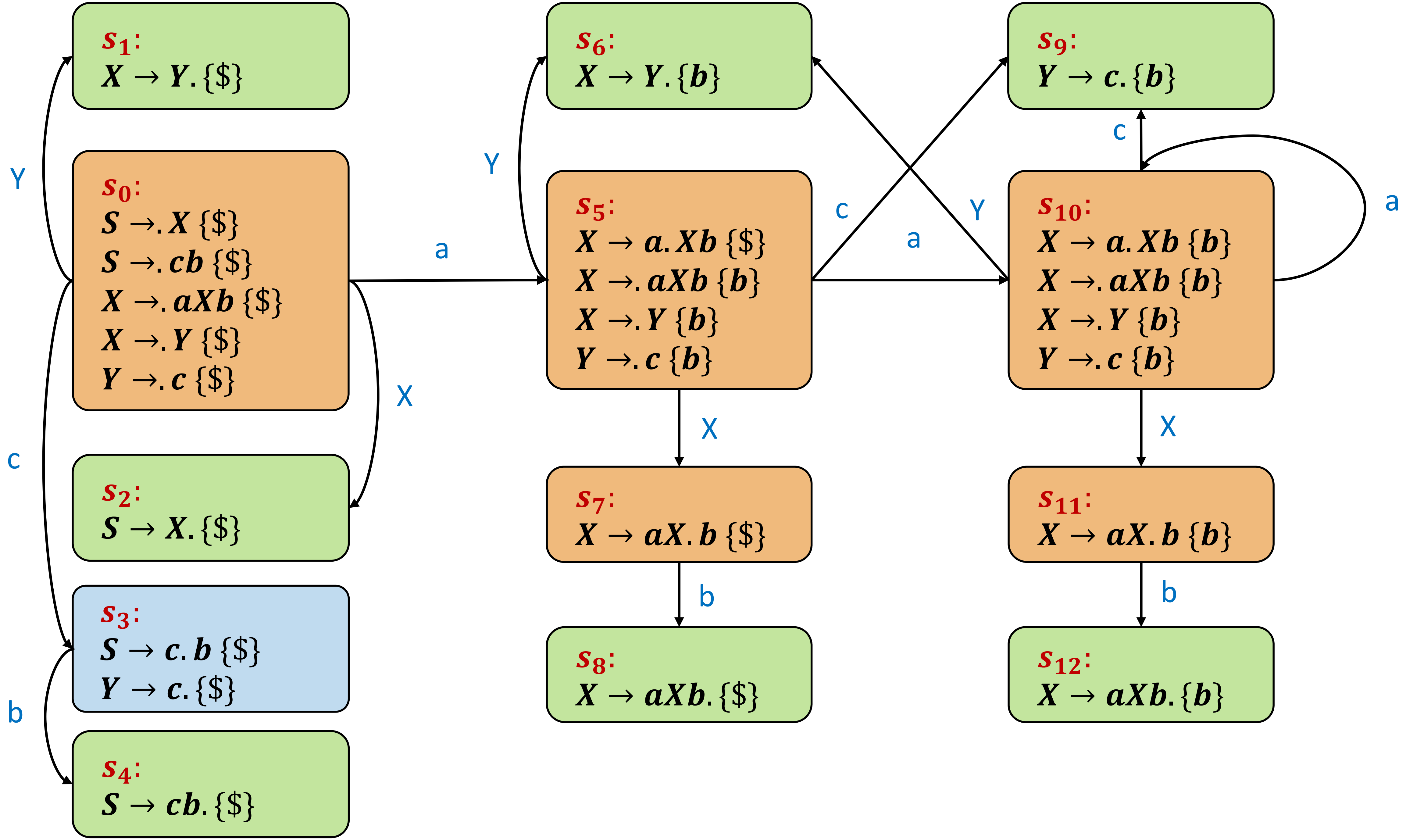


s_1: $X \to Y.\{\$\}$
s_0: $S \to .X\ \{\$\}$ $S \to .cb\ \{\$\}$ $X \to .aXb\ \{\$\}$ $X \to .Y\ \{\$\}$ $Y \to .c\ \{\$\}$
s_2: $S \to X.\{\$\}$
s_3: $S \to c.b\ \{\$\}$ $Y \to c.\{\$\}$
s_4: $S \to cb.\{\$\}$
s_5: $X \to a.Xb\ \{\$\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
s_6: $X \to Y.\{b\}$
s_7: $X \to aX.b\ \{\$\}$
s_8: $X \to aXb.\{\$\}$
s_9: $Y \to c.\{b\}$
s_{10}: $X \to a.Xb\ \{b\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
Y
c
b
a
X
Y
X
b
c
a
Y
c



s_1: $X \to Y.\{\$\}$
s_0: $S \to .X\ \{\$\}$ $S \to .cb\ \{\$\}$ $X \to .aXb\ \{\$\}$ $X \to .Y\ \{\$\}$ $Y \to .c\ \{\$\}$
s_2: $S \to X.\{\$\}$
s_3: $S \to c.b\ \{\$\}$ $Y \to c.\{\$\}$
s_4: $S \to cb.\{\$\}$
s_6: $X \to Y.\{b\}$
s_5: $X \to a.Xb\ \{\$\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
s_7: $X \to aX.b\ \{\$\}$
s_8: $X \to aXb.\{\$\}$
s_9: $Y \to c.\{b\}$
s_{10}: $X \to a.Xb\ \{b\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
Y
c
b
a
X
Y
c
a
Y
X
b
c
a


s_1: $X \to Y.\{\$\}$
s_0: $S \to .X\ \{\$\}$ $S \to .cb\ \{\$\}$ $X \to .aXb\ \{\$\}$ $X \to .Y\ \{\$\}$ $Y \to .c\ \{\$\}$
s_2: $S \to X.\{\$\}$
s_3: $S \to c.b\ \{\$\}$ $Y \to c.\{\$\}$
s_4: $S \to cb.\{\$\}$
s_5: $X \to a.Xb\ \{\$\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
s_6: $X \to Y.\{b\}$
s_7: $X \to aX.b\ \{\$\}$
s_8: $X \to aXb.\{\$\}$
s_9: $Y \to c.\{b\}$
s_{10}: $X \to a.Xb\ \{b\}$ $X \to .aXb\ \{b\}$ $X \to .Y\ \{b\}$ $Y \to .c\ \{b\}$
s_{11}: $X \to aX.b\ \{b\}$
Y
c
b
a
X
Y
X
b
c
a
Y
c
a
X

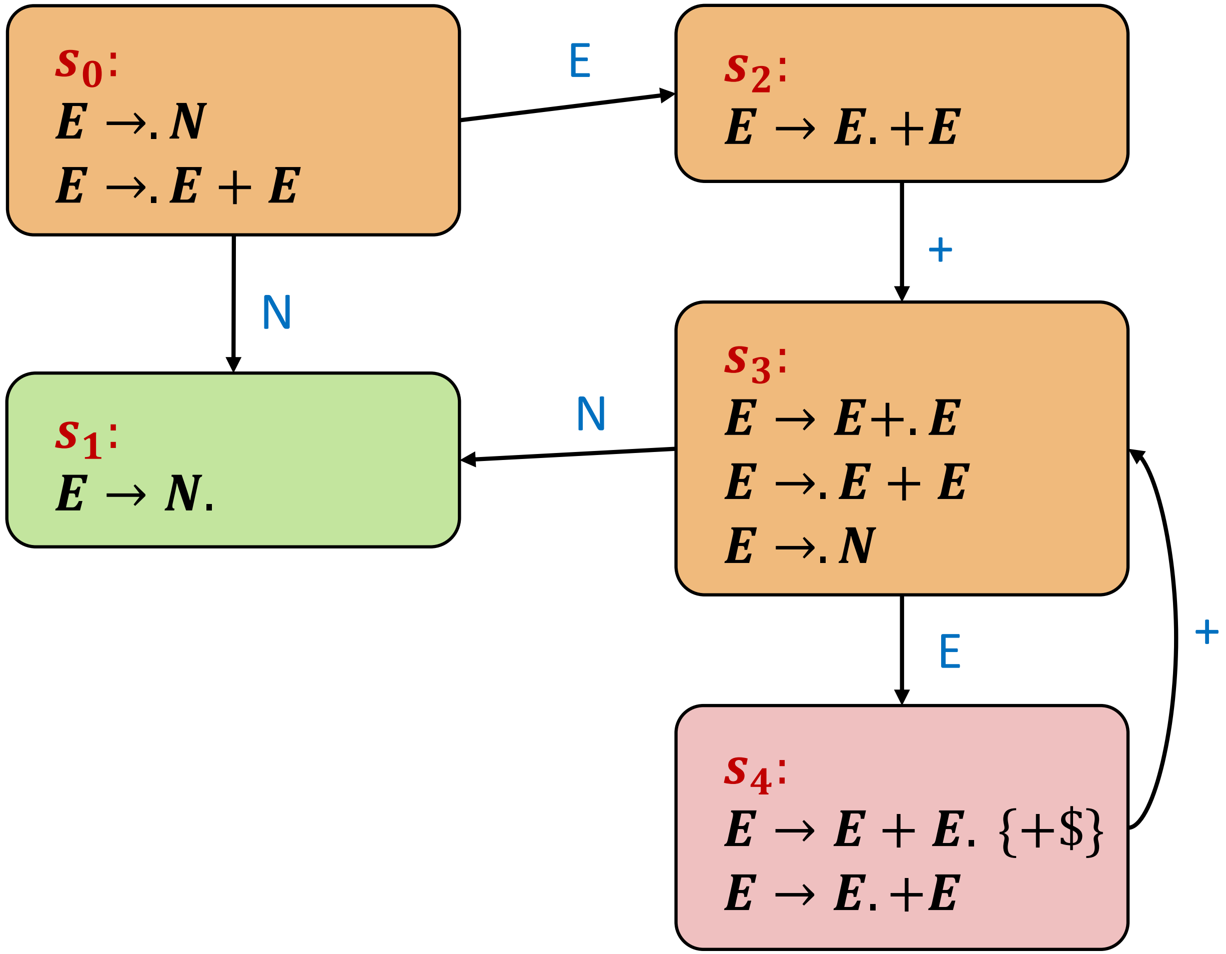

s_1:
$X \to Y.\{\$\}$
s_0:
$S \to .X\ \{\$\}$
$S \to .cb\ \{\$\}$
$X \to .aXb\ \{\$\}$
$X \to .Y\ \{\$\}$
$Y \to .c\ \{\$\}$
s_2:
$S \to X.\{\$\}$
s_3:
$S \to c.b\ \{\$\}$
$Y \to c.\{\$\}$
s_4:
$S \to cb.\{\$\}$
s_5:
$X \to a.Xb\ \{\$\}$
$X \to .aXb\ \{b\}$
$X \to .Y\ \{b\}$
$Y \to .c\ \{b\}$
s_6:
$X \to Y.\{b\}$
s_7:
$X \to aX.b\ \{\$\}$
s_8:
$X \to aXb.\{\$\}$
s_9:
$Y \to c.\{b\}$
s_{10}:
$X \to a.Xb\ \{b\}$
$X \to .aXb\ \{b\}$
$X \to .Y\ \{b\}$
$Y \to .c\ \{b\}$
s_{11}:
$X \to aX.b\ \{b\}$
s_{12}:
$X \to aXb.\{b\}$
Y
c
b
a
X
Y
c
a
Y
X
b
c
a
X
b

Resolving Conflicts

Consider the following CFG:

- $E \rightarrow N$
- $E \rightarrow E + E$

What will be the **transition system** of the SLR(1) parser for this CFG?



s_0:
$E \to .N$
$E \to .E + E$
E
s_2:
$E \to E.+E$
+
N
s_1:
$E \to N.$
N
s_3:
$E \to E+.E$
$E \to .E + E$
$E \to .N$
E
+
s_4:
$E \to E + E.\ \{+\$\}$
$E \to E.+E$

Resolving Conflicts

How will affect associativity?

$\boldsymbol{s_4}$:

$\boldsymbol{E} \rightarrow \boldsymbol{E} + \boldsymbol{E}.\ \{+\$\}$

$\boldsymbol{E} \rightarrow \boldsymbol{E}. + \boldsymbol{E}$

Resolving Conflicts

How will affect associativity?

Shift:

- Right associative

Reduce:

- Left associative

s_4:

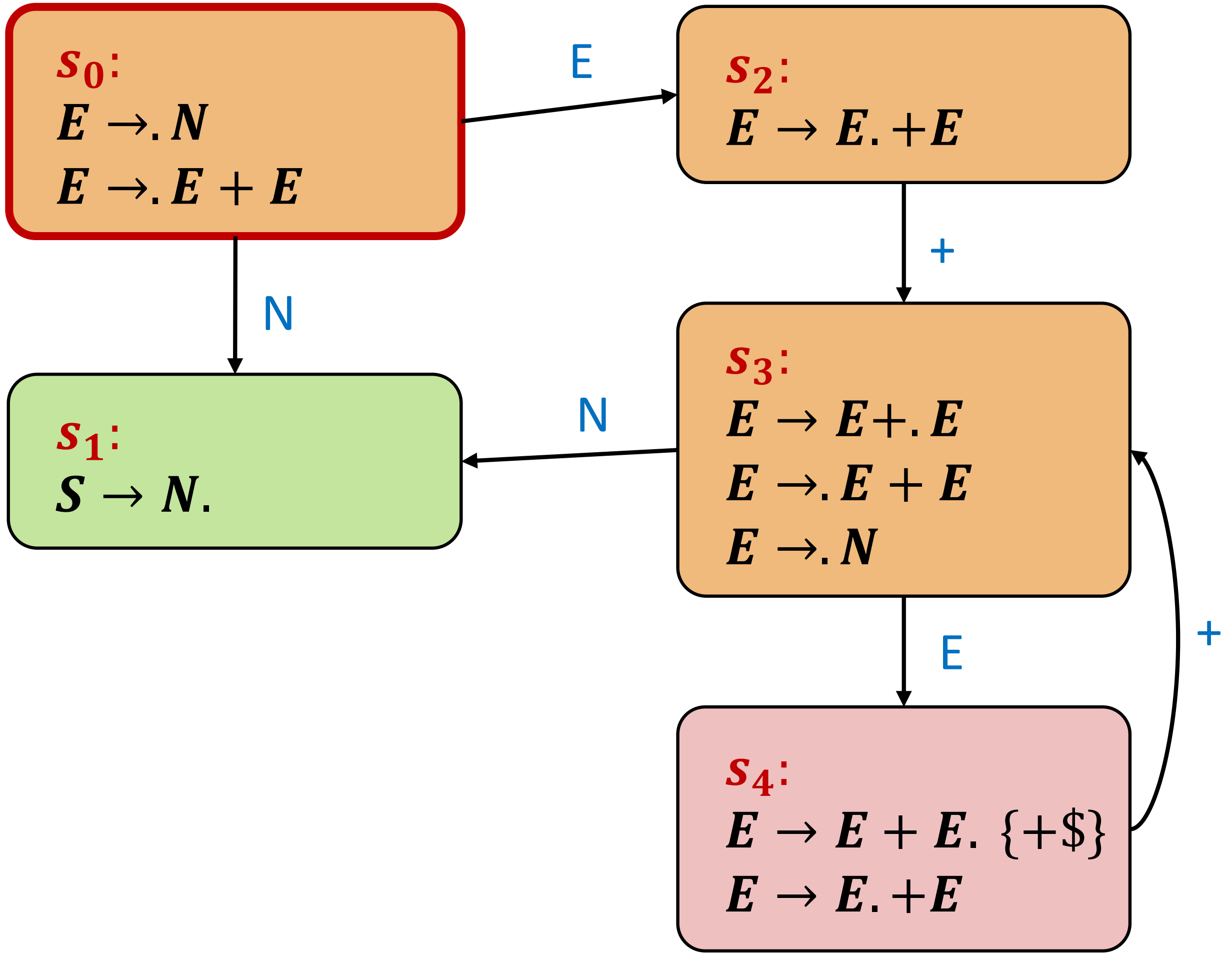
$E \rightarrow E + E.\ \{+\$\}$

$E \rightarrow E. + E$

Resolving Conflicts

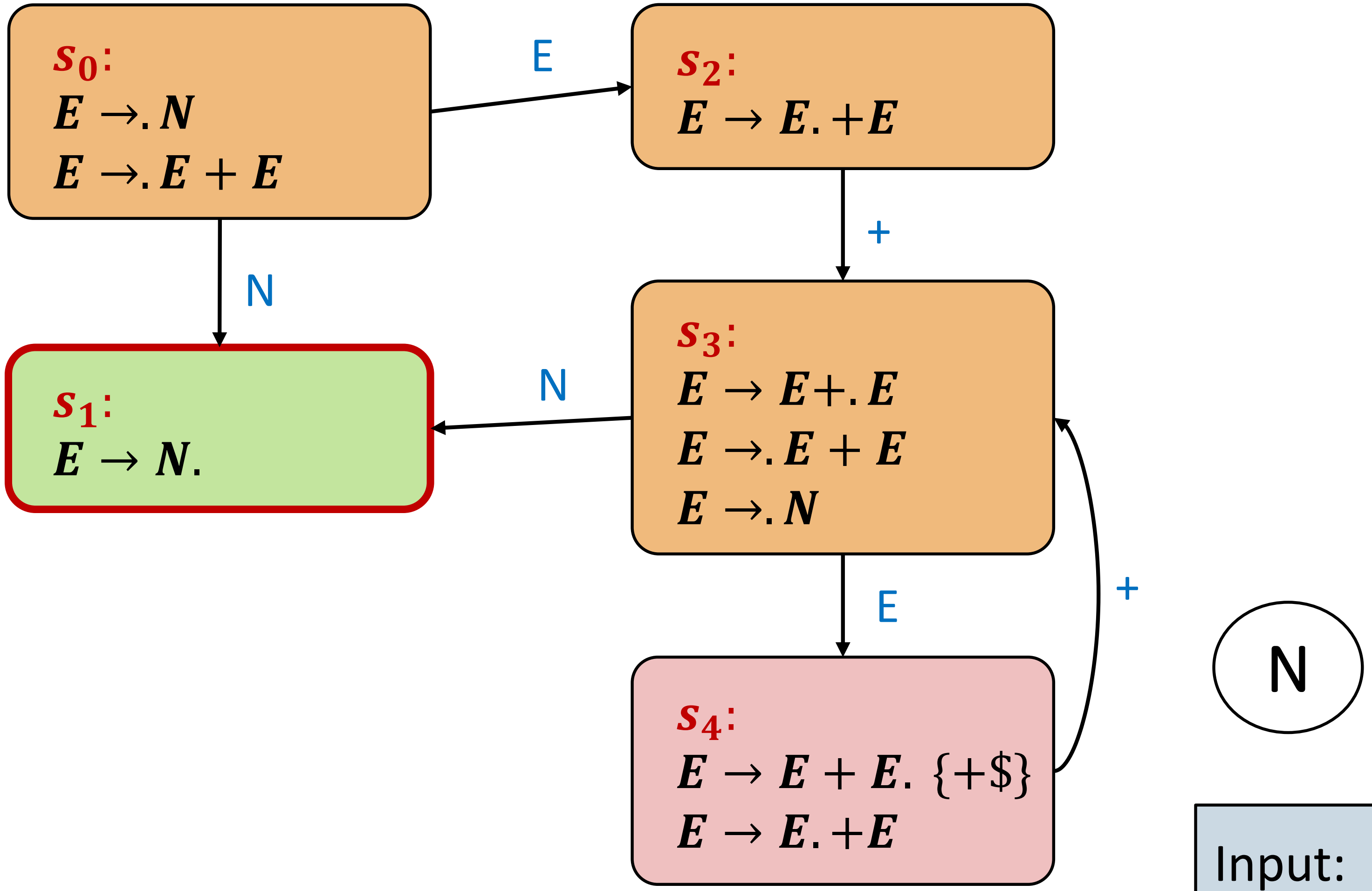
When resolving using the **reduce** item:

- Left associative



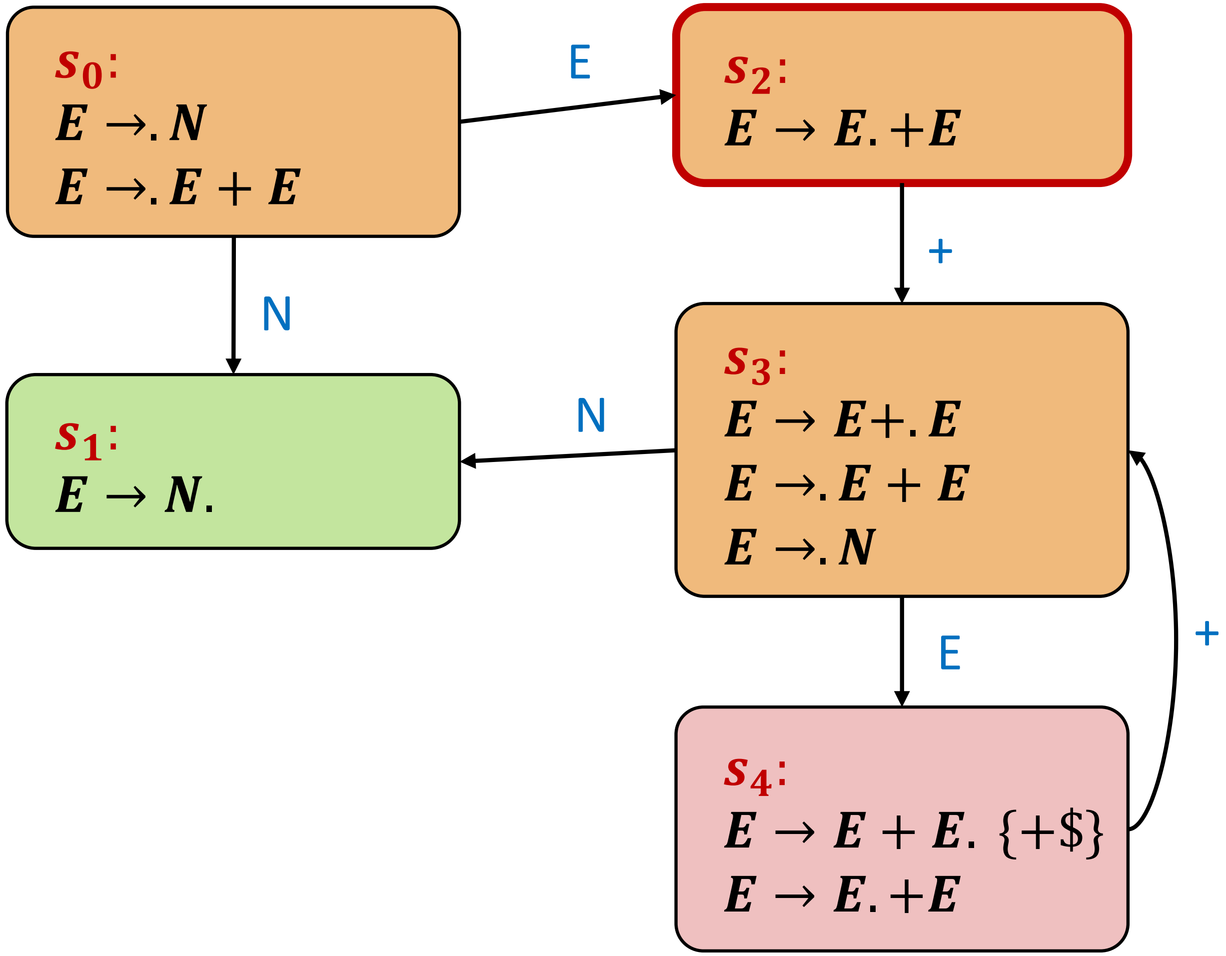
Input: $1 + 2 + 3\$$

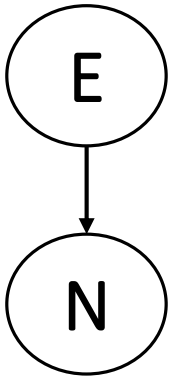
Stack: s_0



Input: $\mathbf{1} + \mathbf{2} + \mathbf{3}\$$

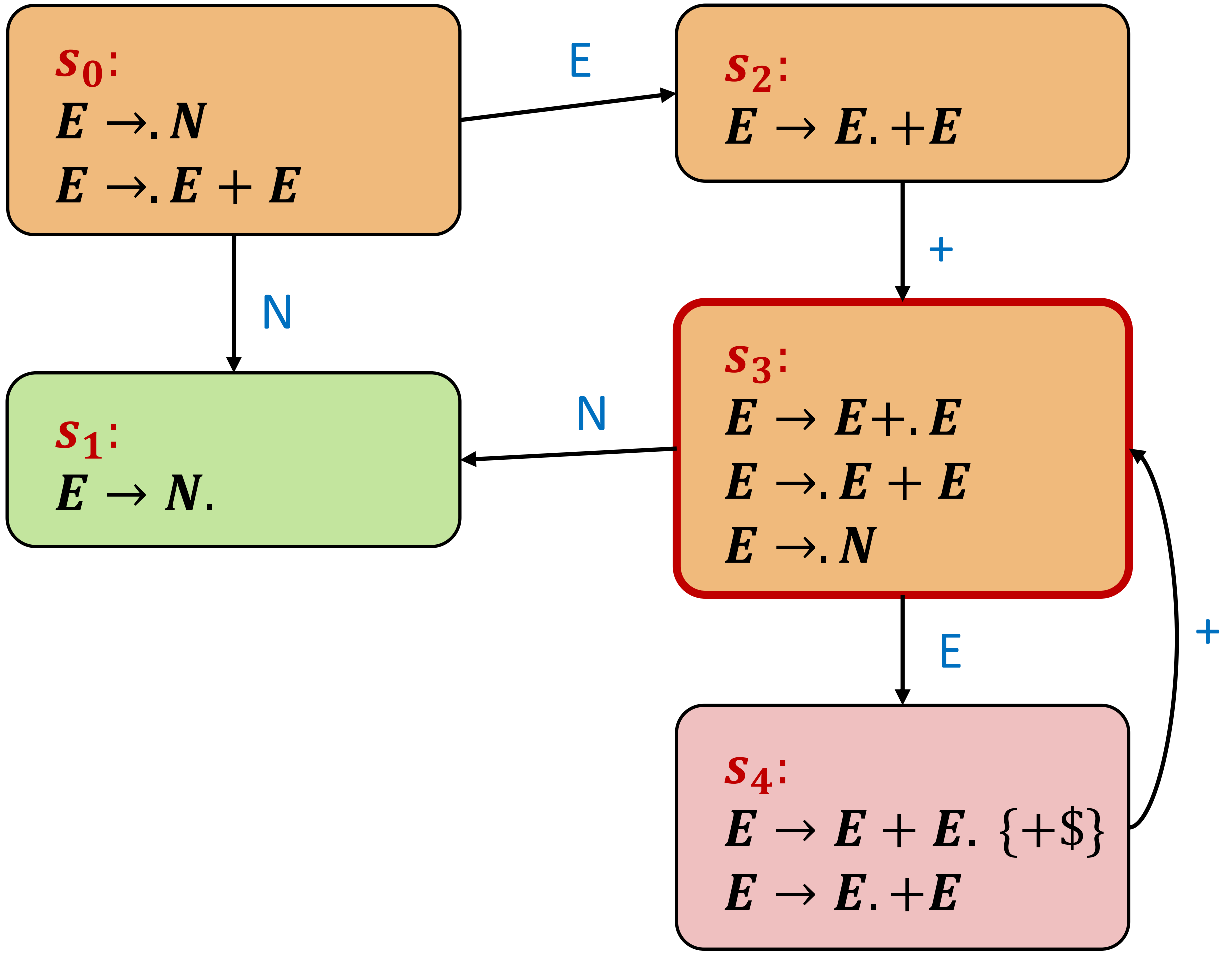
Stack: $s_0 N s_1$

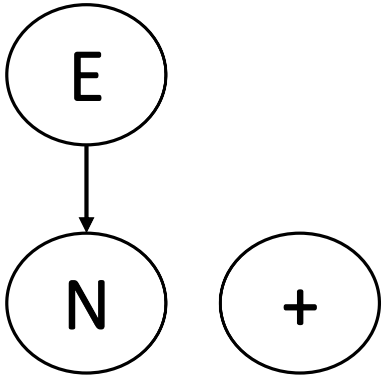




Input: $\mathbf{1} + 2 + 3\$$

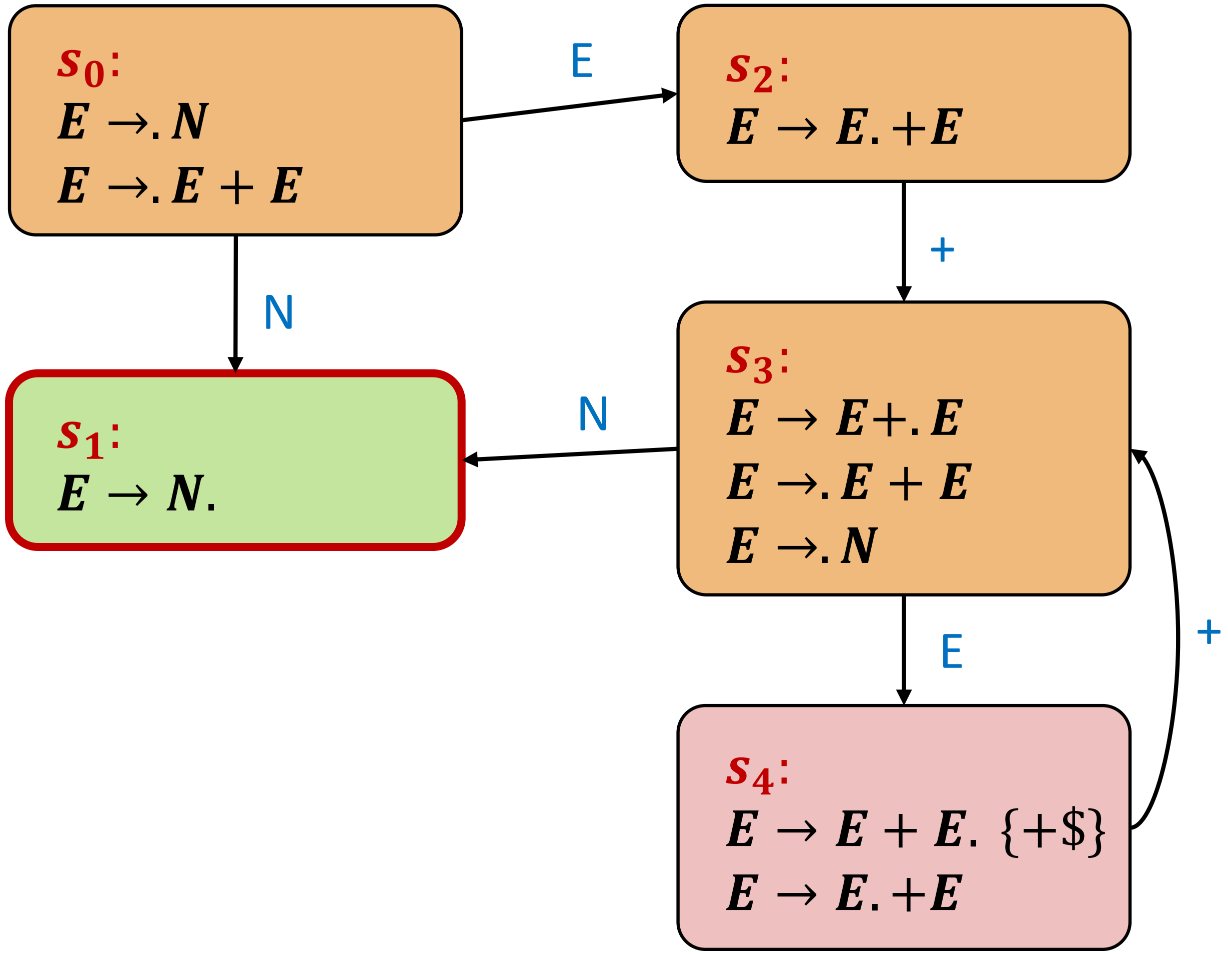
Stack: $s_0 E s_2$

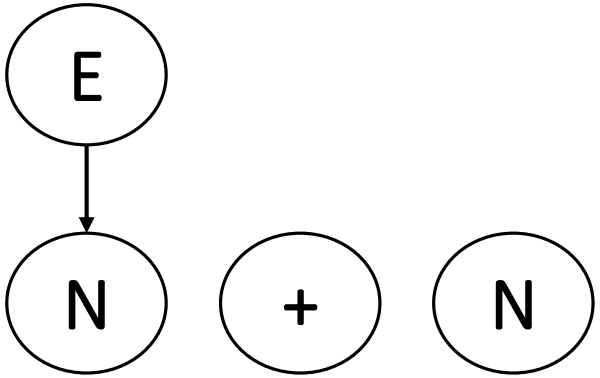




Input: $\mathbf{1} + \mathbf{2} + \mathbf{3\$}$

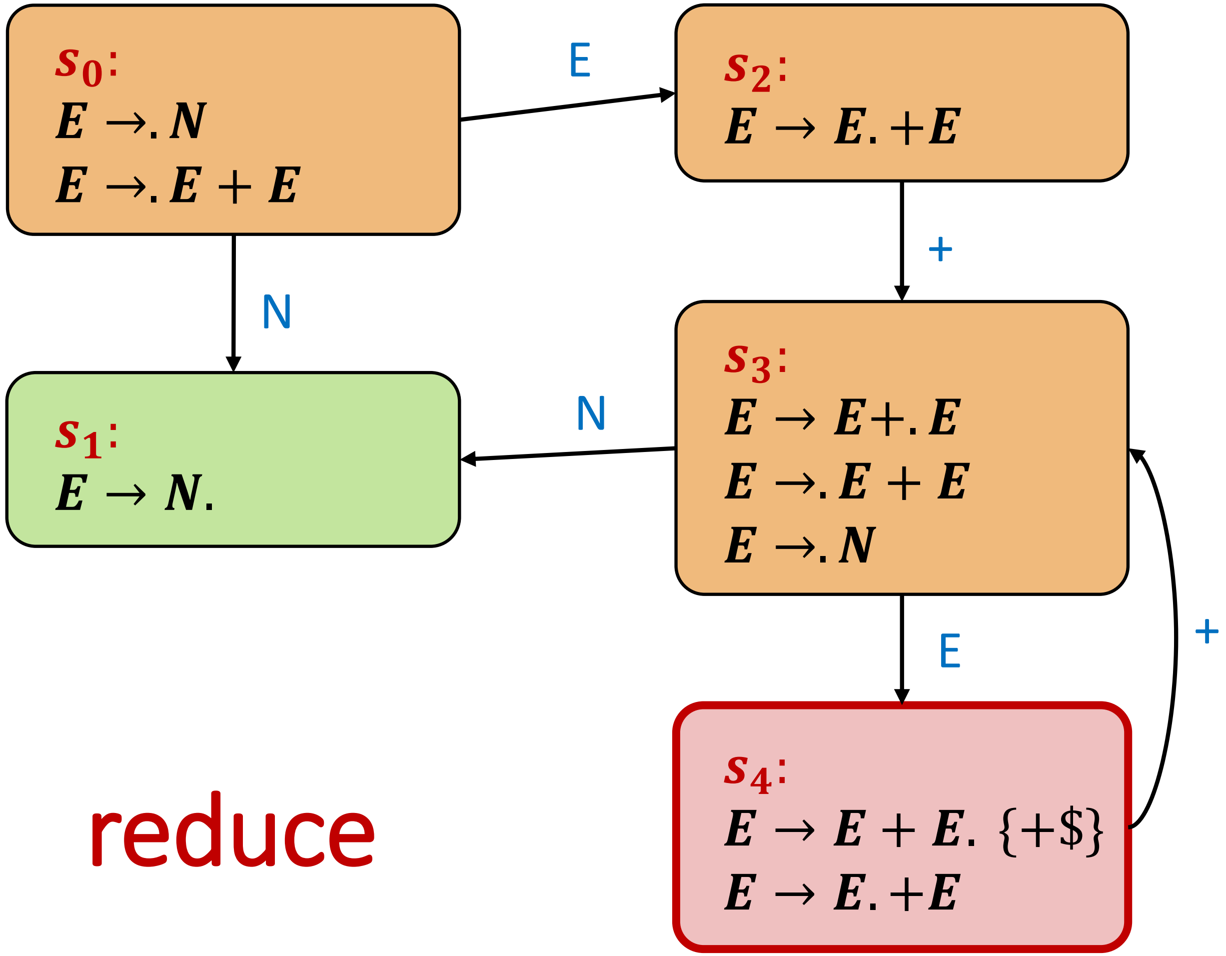
Stack: $s_0 E s_2 + s_3$



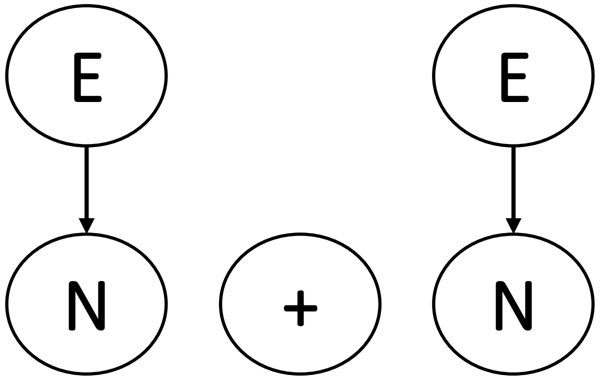


Input: $1 + 2 + 3\$$

Stack: $s_0 E s_2 + s_3 N s_1$

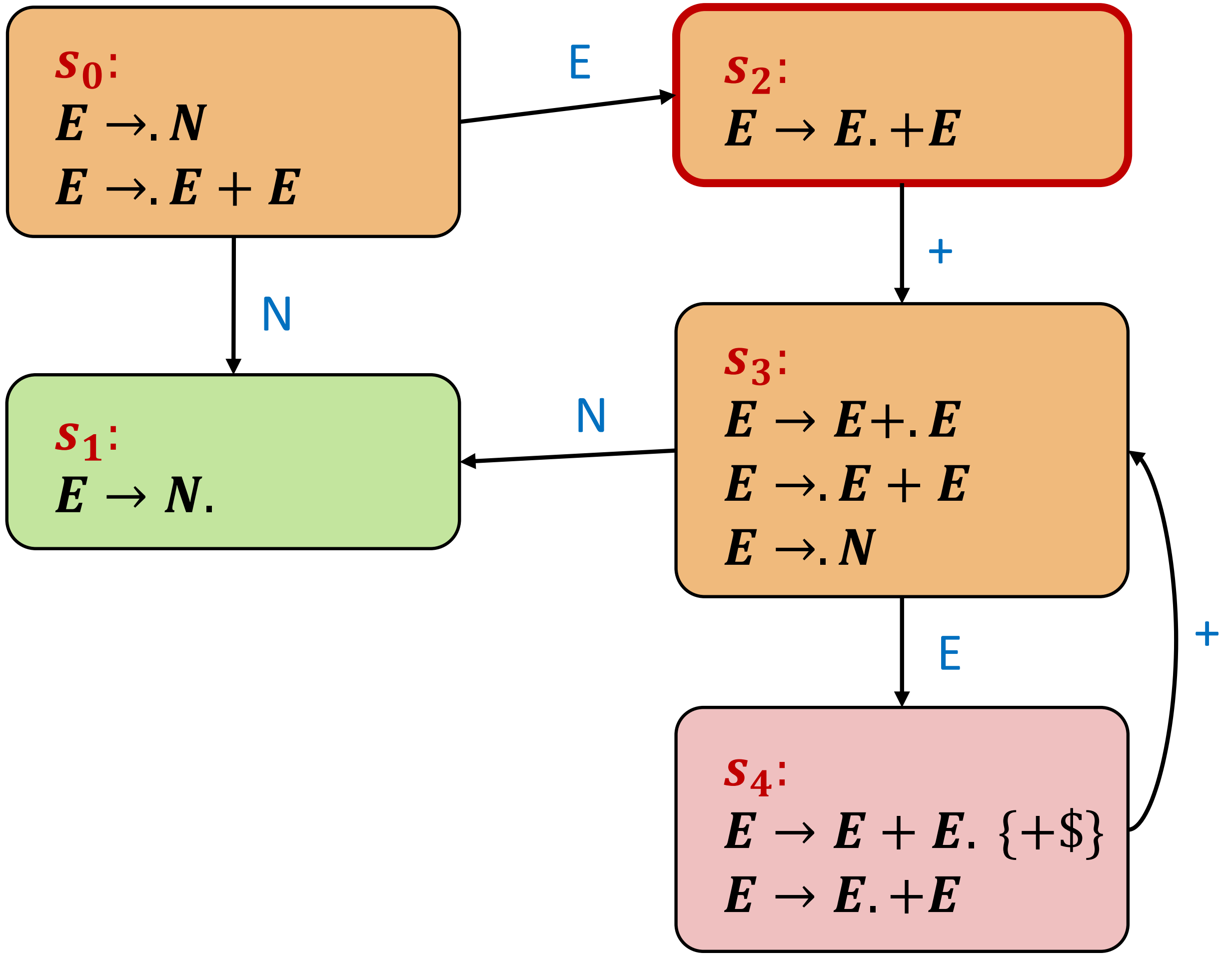


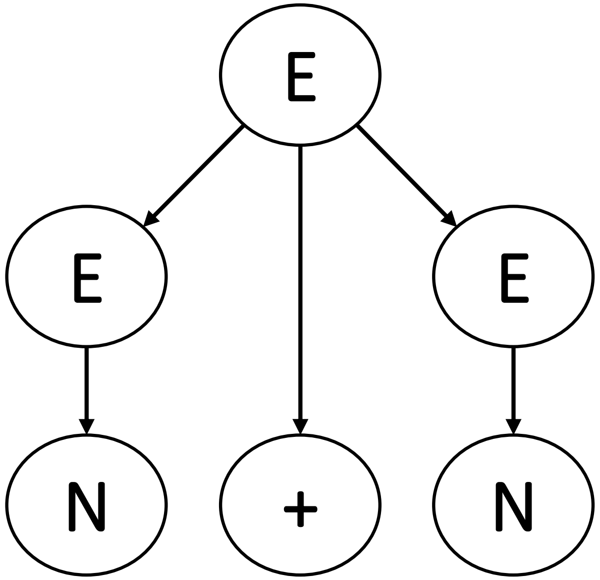
reduce



Input: $1 + 2 + 3\$$

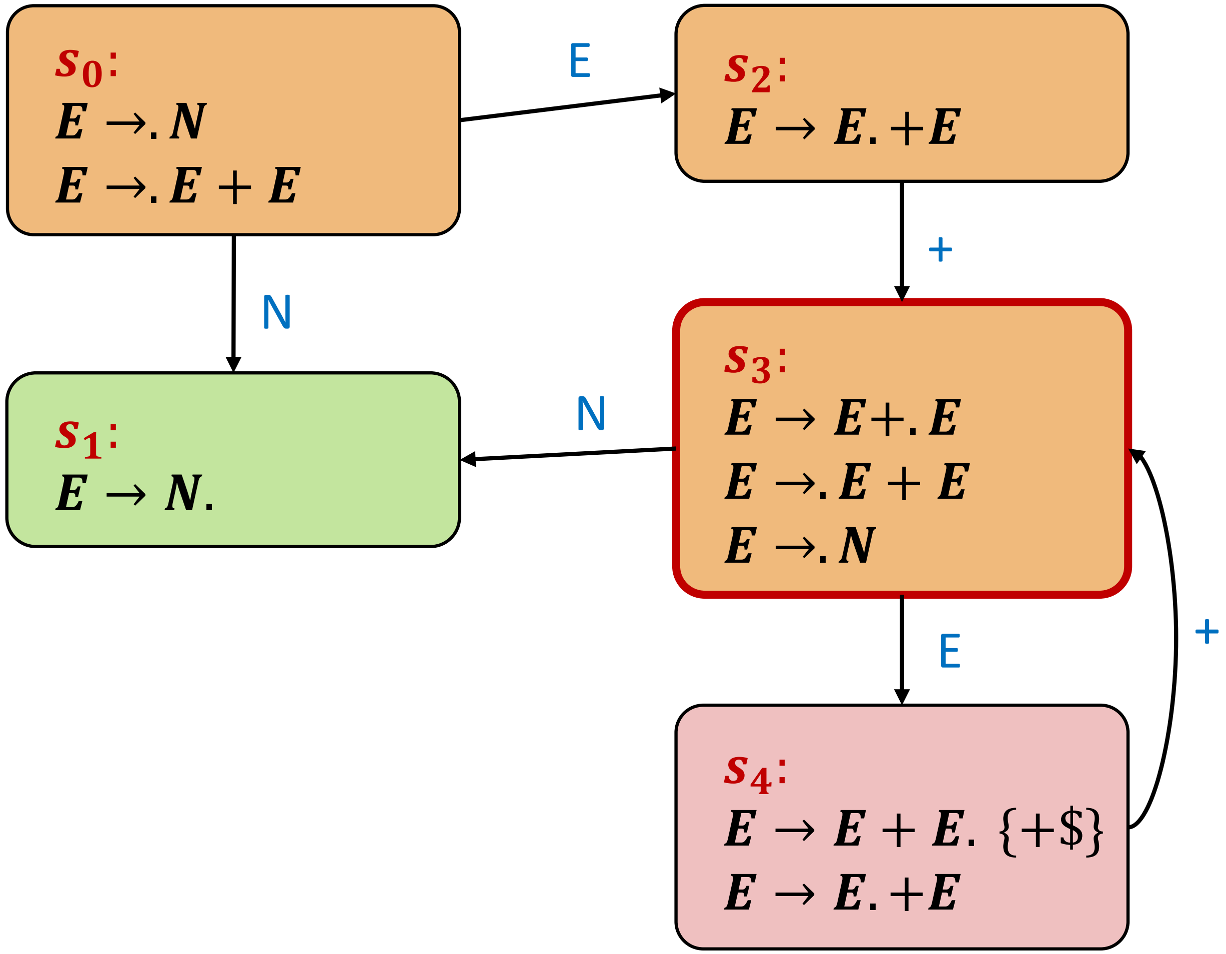
Stack: $s_0 E s_2 + s_3 E s_4$

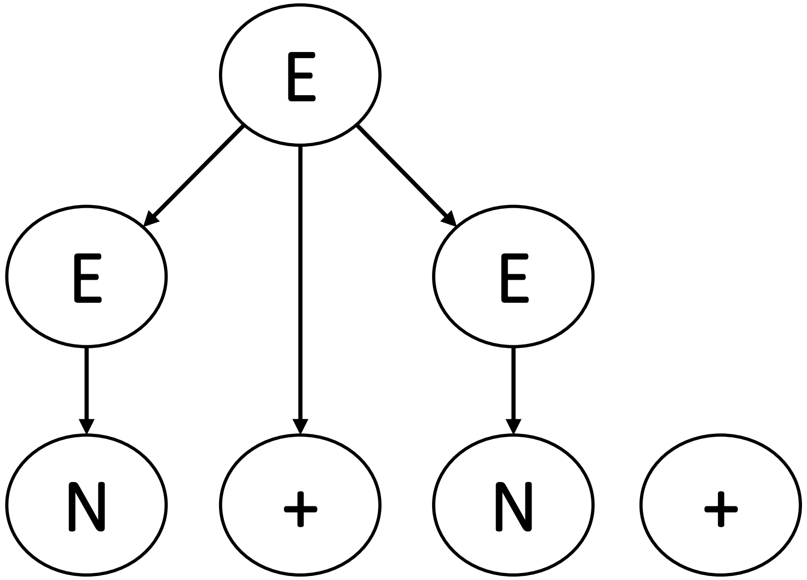




Input: $1 + 2 + 3\$$

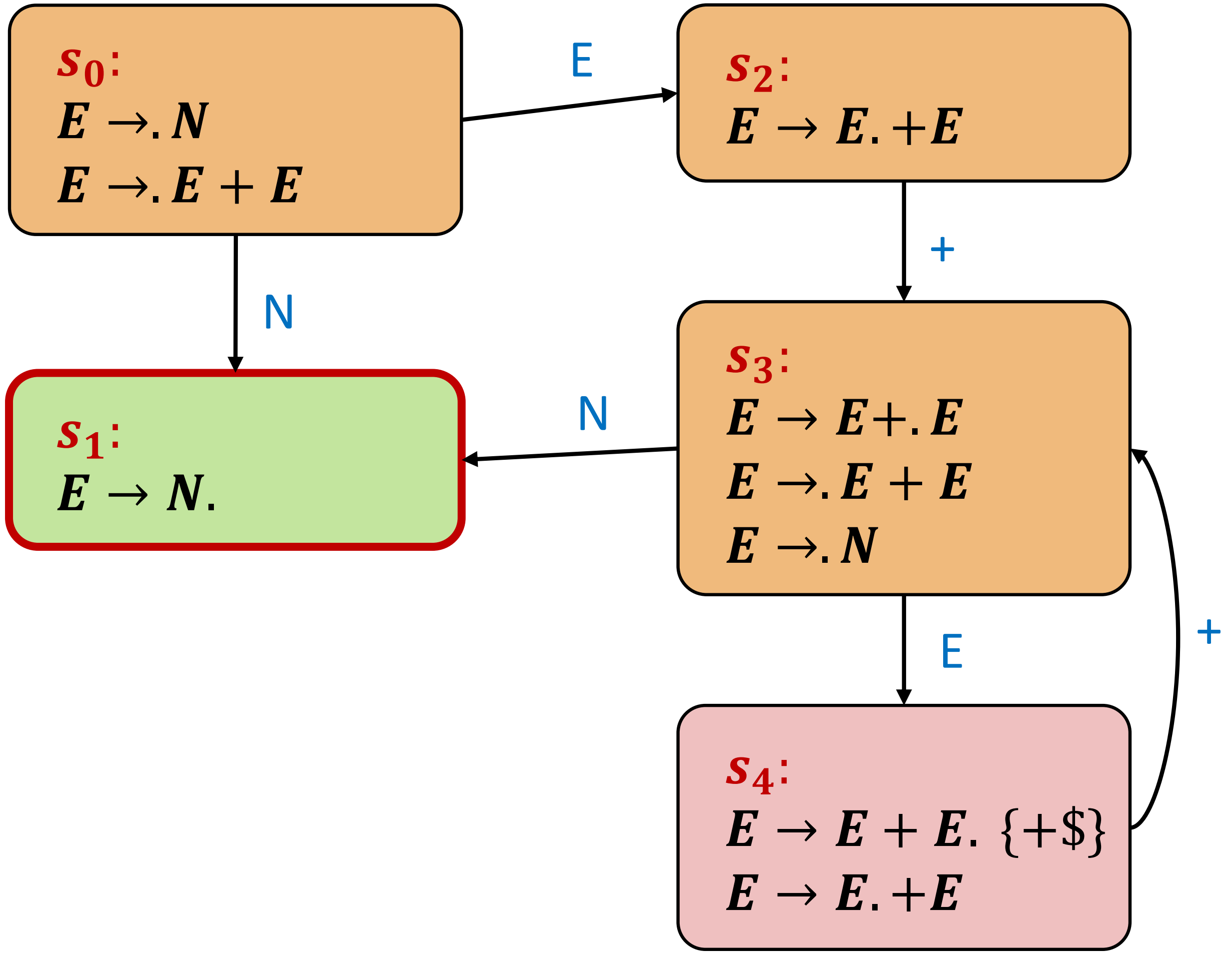
Stack: $s_0 E s_2$

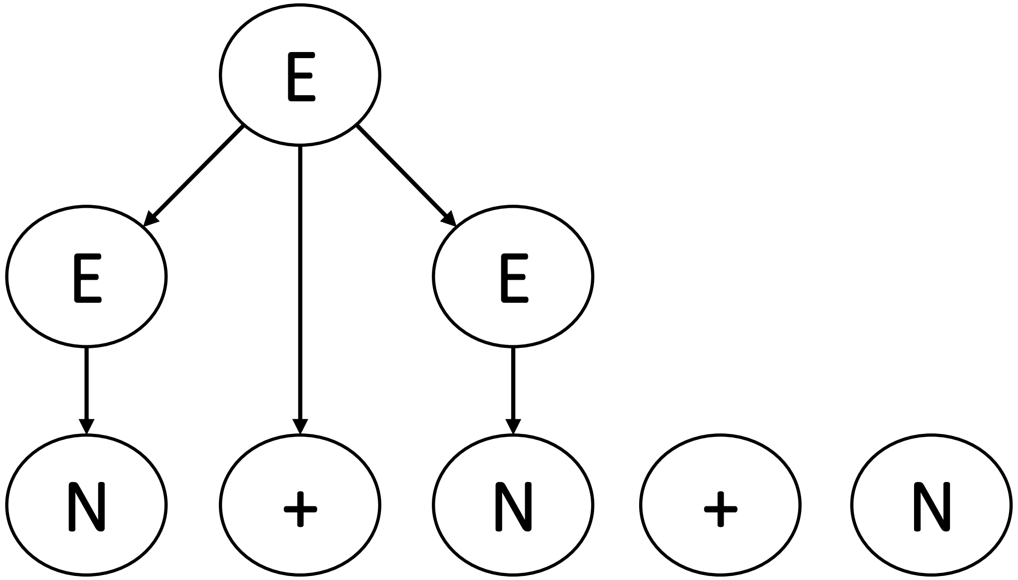




Input: $1 + 2 + 3\$$

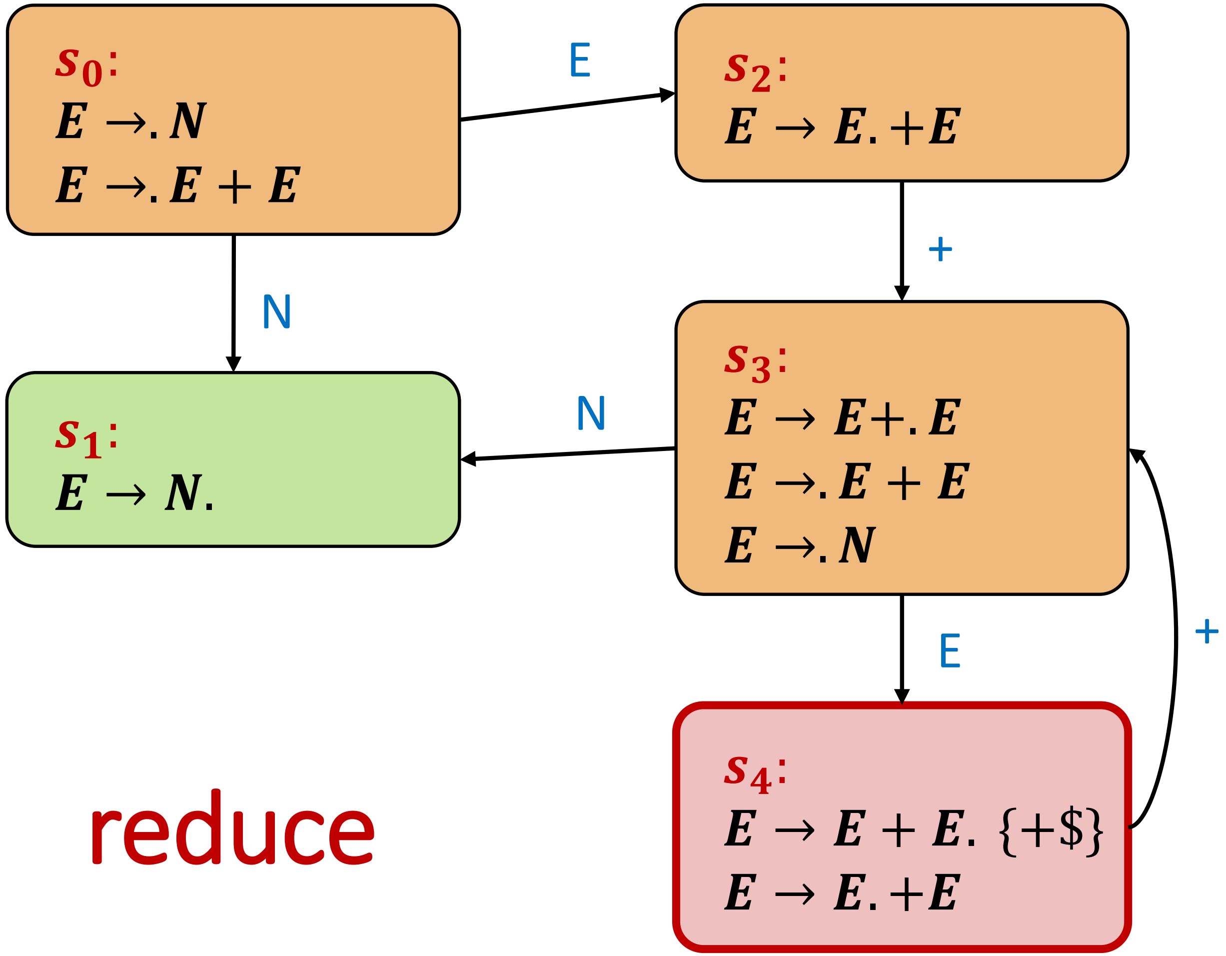
Stack: $s_0 E s_2 + s_3$

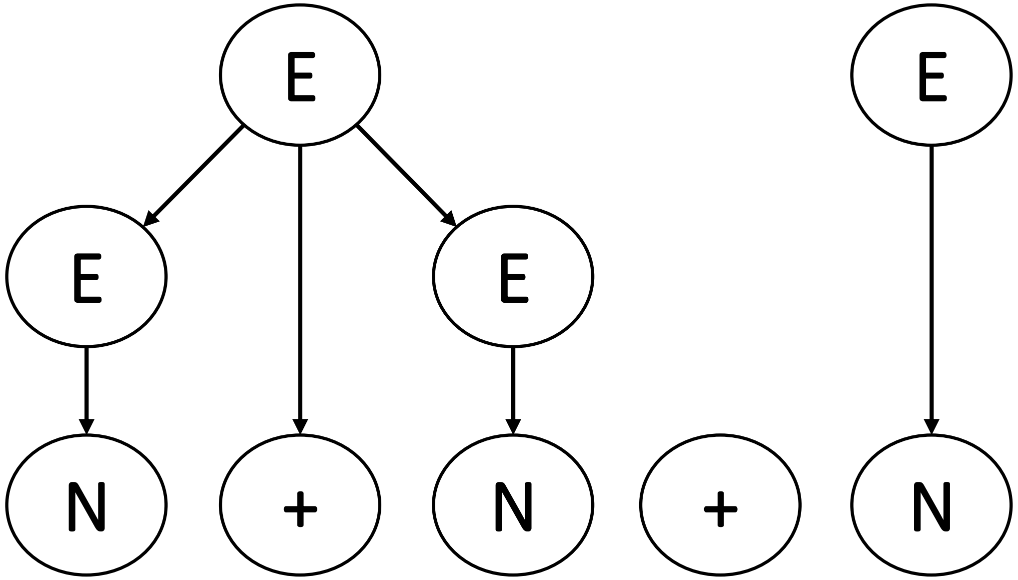




Input: $\mathbf{1 + 2 + 3}\$$

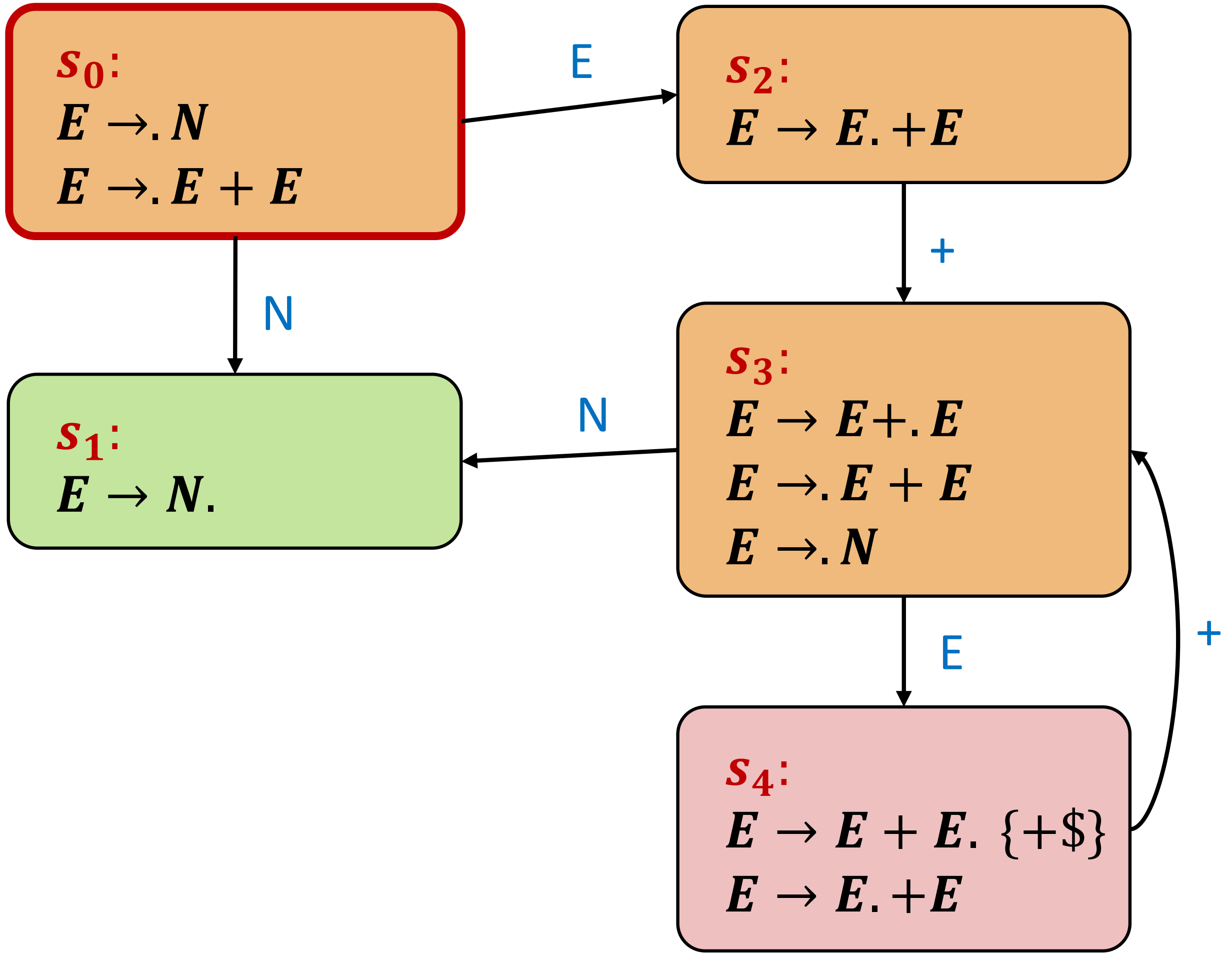
Stack: $s_0 E s_2 + s_3 N s_1$

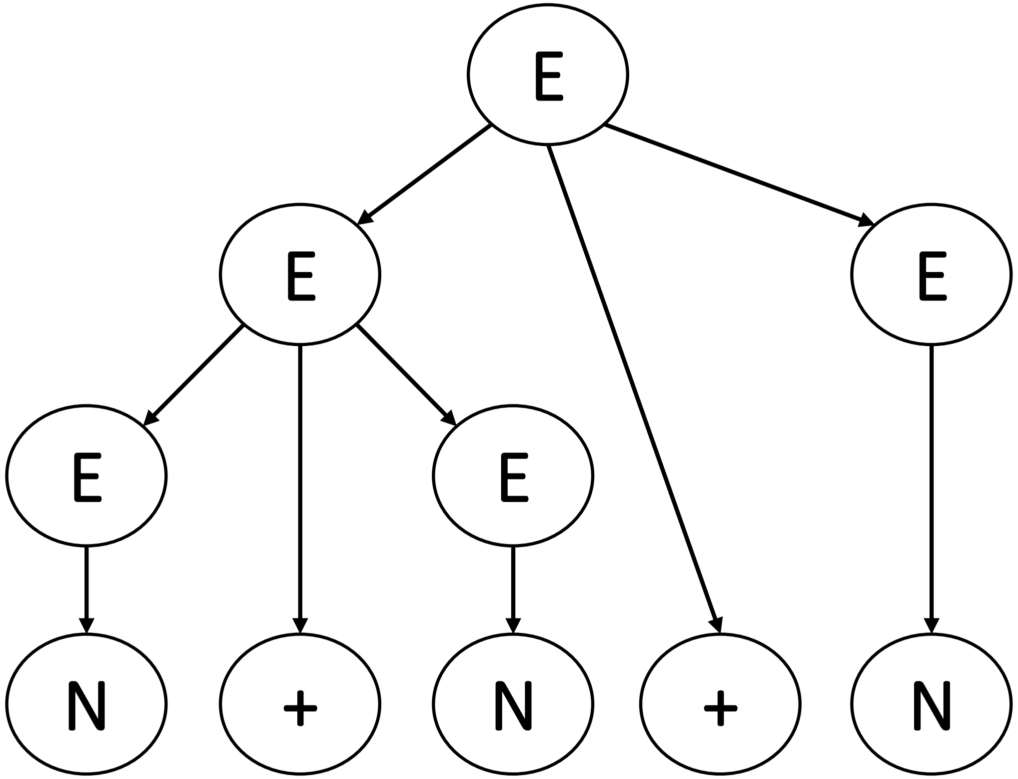




Input: $\mathbf{1 + 2 + 3}\$$

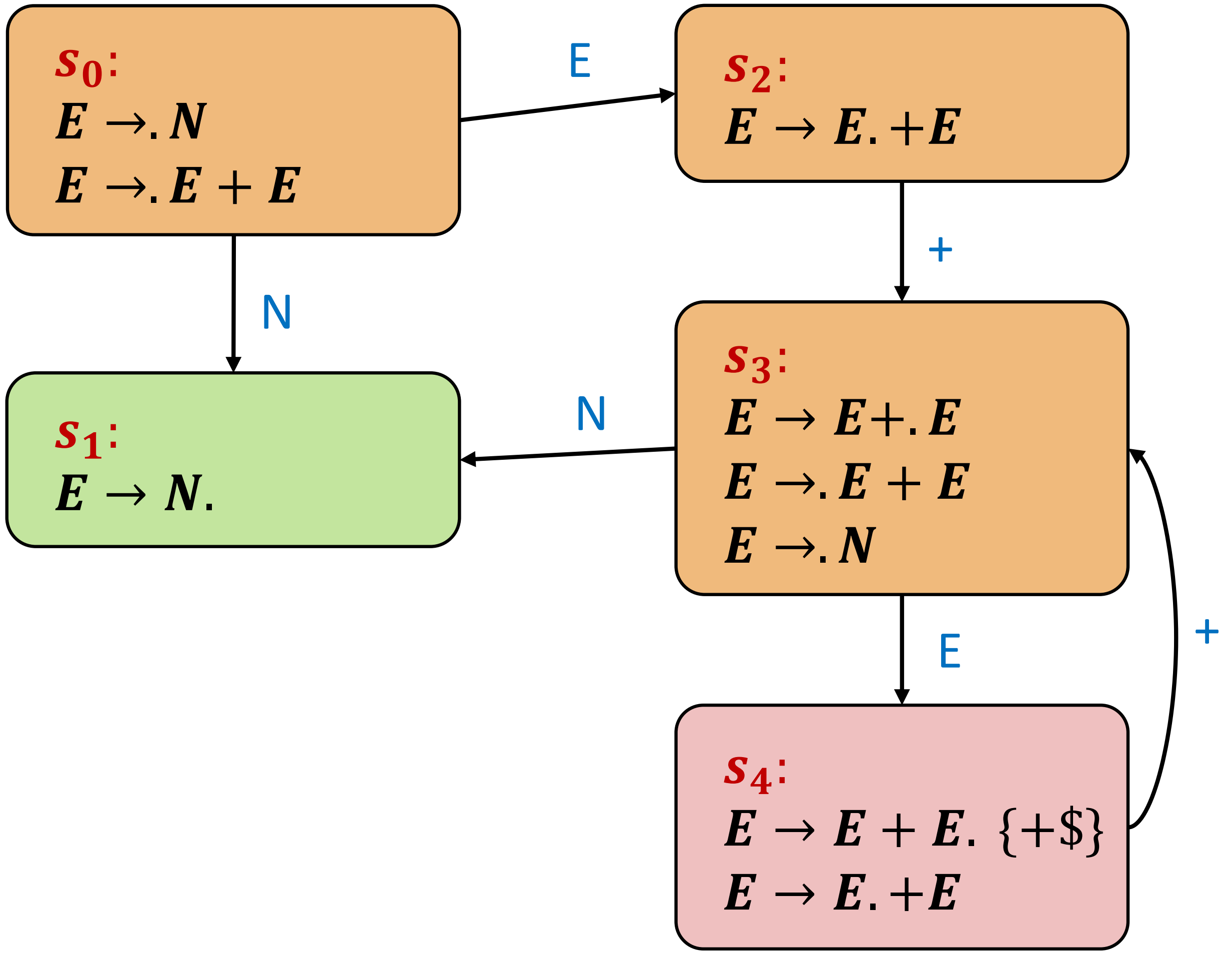
Stack: $s_0 E s_2 + s_3 E s_4$

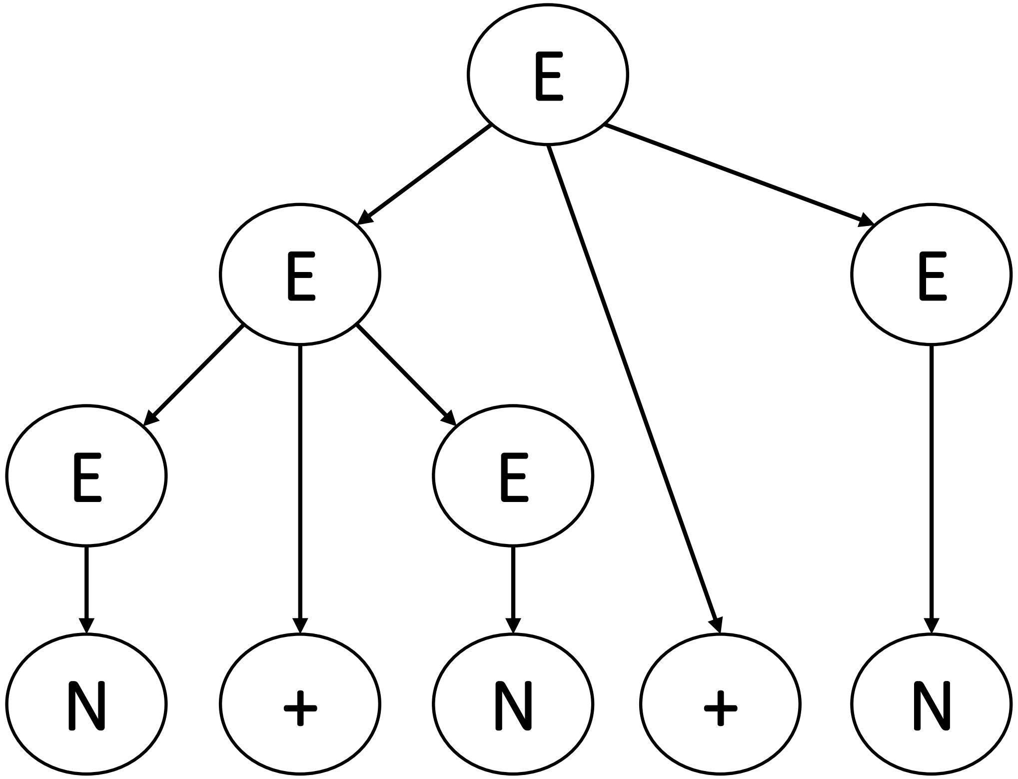




Input: $\mathbf{1 + 2 + 3}\$$

Stack: $s_0 E$





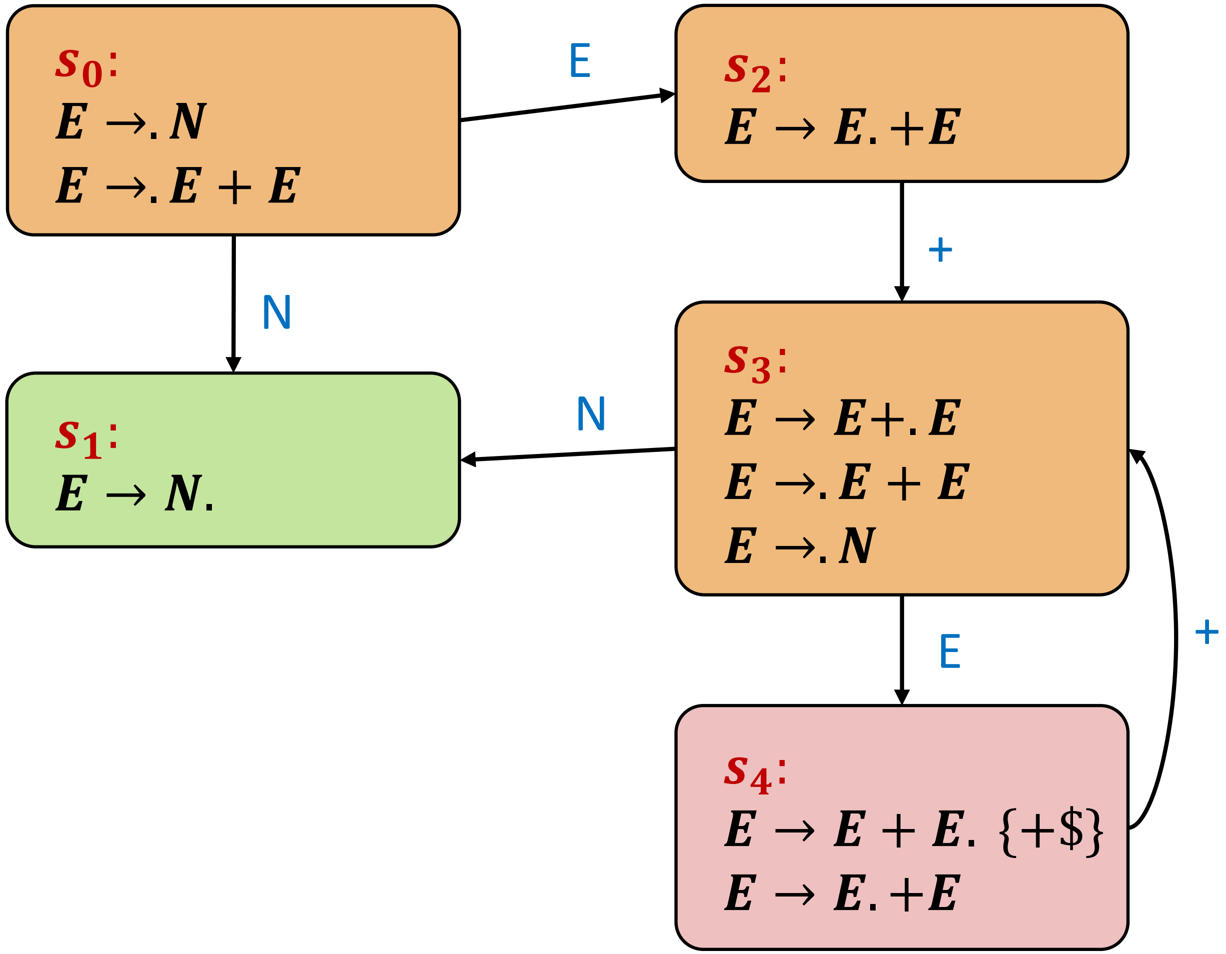
Input: $1 + 2 + 3\$$

Stack: $s_0 E$

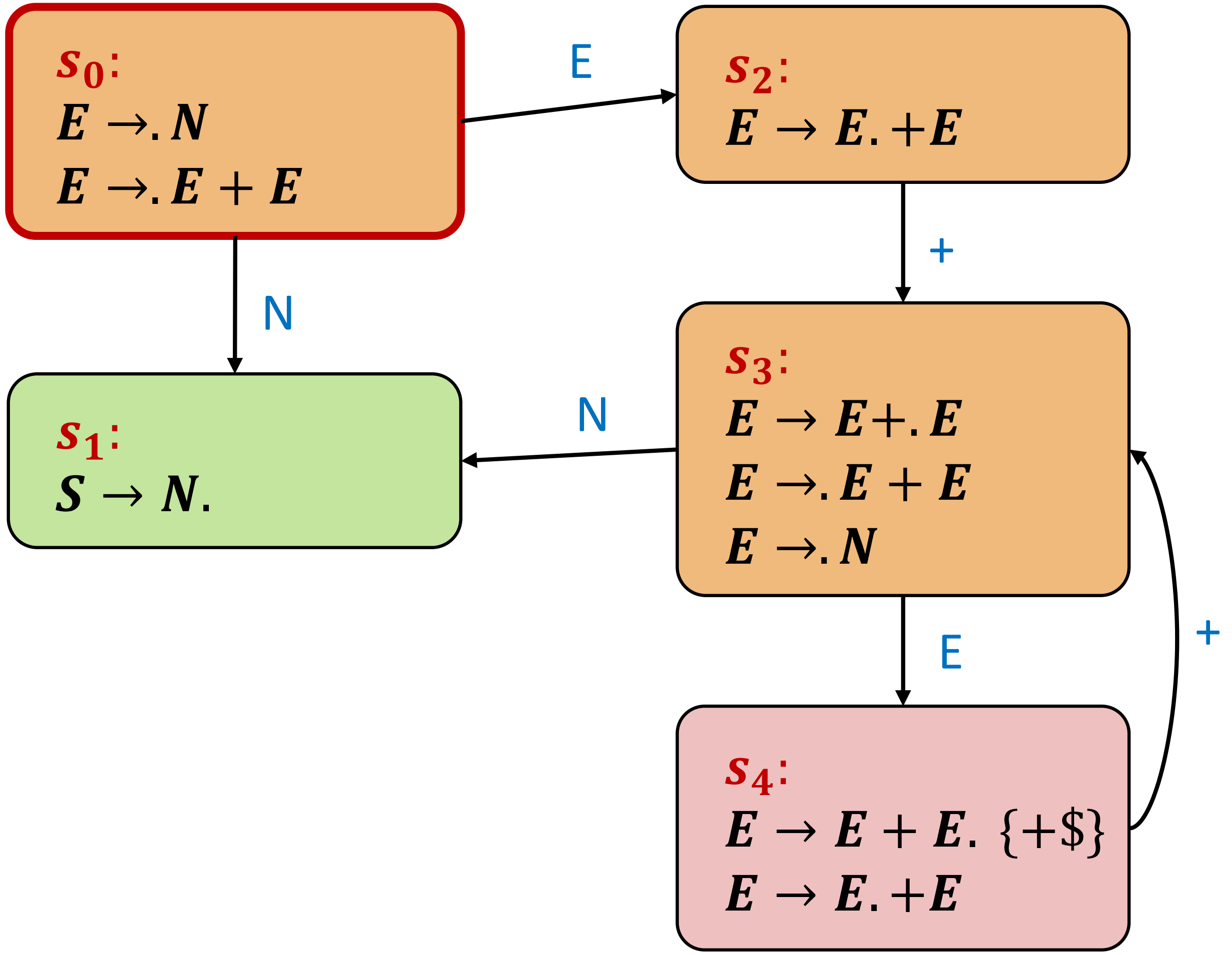
Resolving Conflicts

When resolving using the **shift** item:

- Right associative

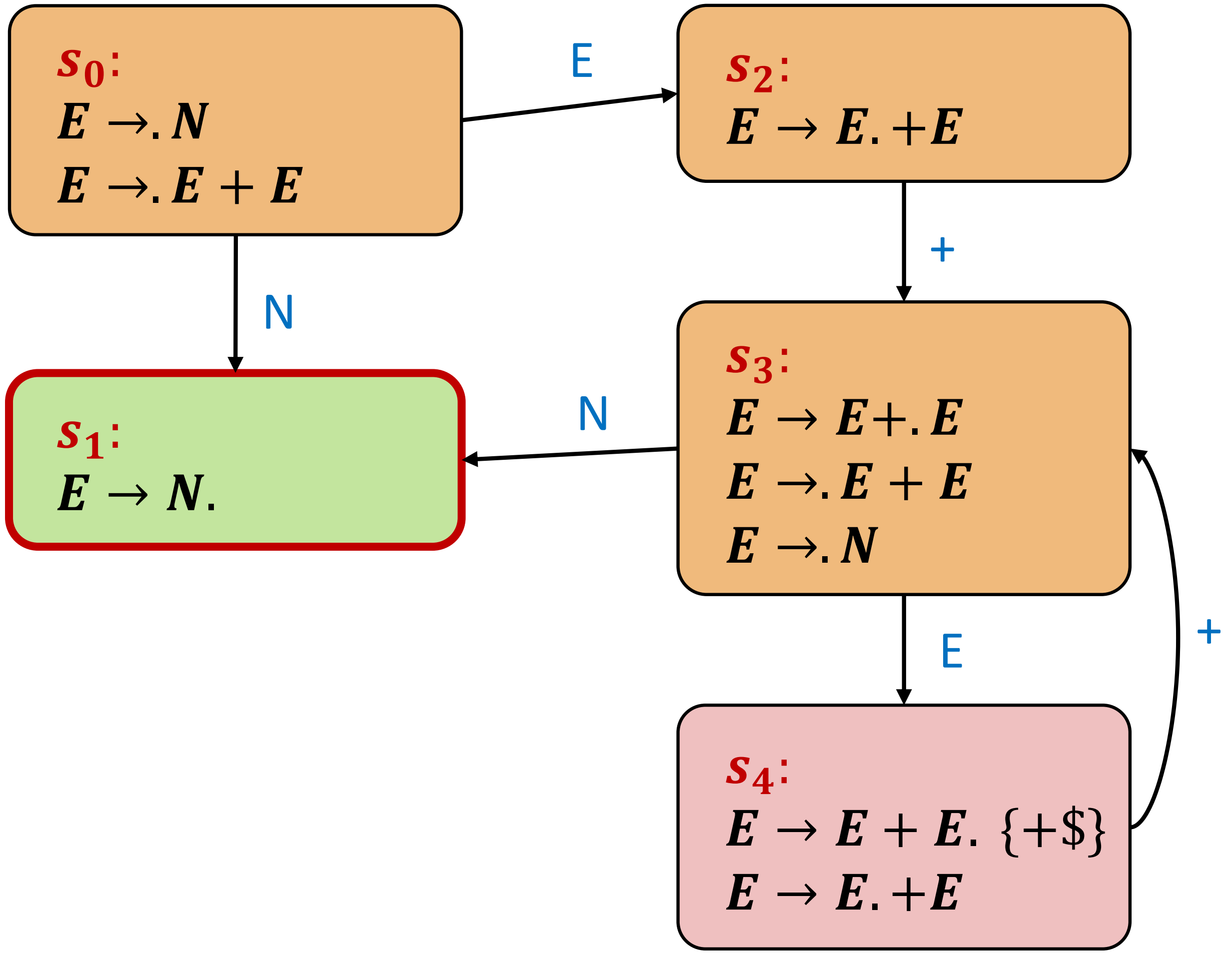


s_0:
$E \to .N$
$E \to .E + E$
E
s_2:
$E \to E.+E$
N
+
s_1:
$E \to N.$
N
s_3:
$E \to E+.E$
$E \to .E + E$
$E \to .N$
E
+
s_4:
$E \to E + E.\ \{+\$\}$
$E \to E.+E$



Input: $\mathbf{1 + 2 + 3\$}$

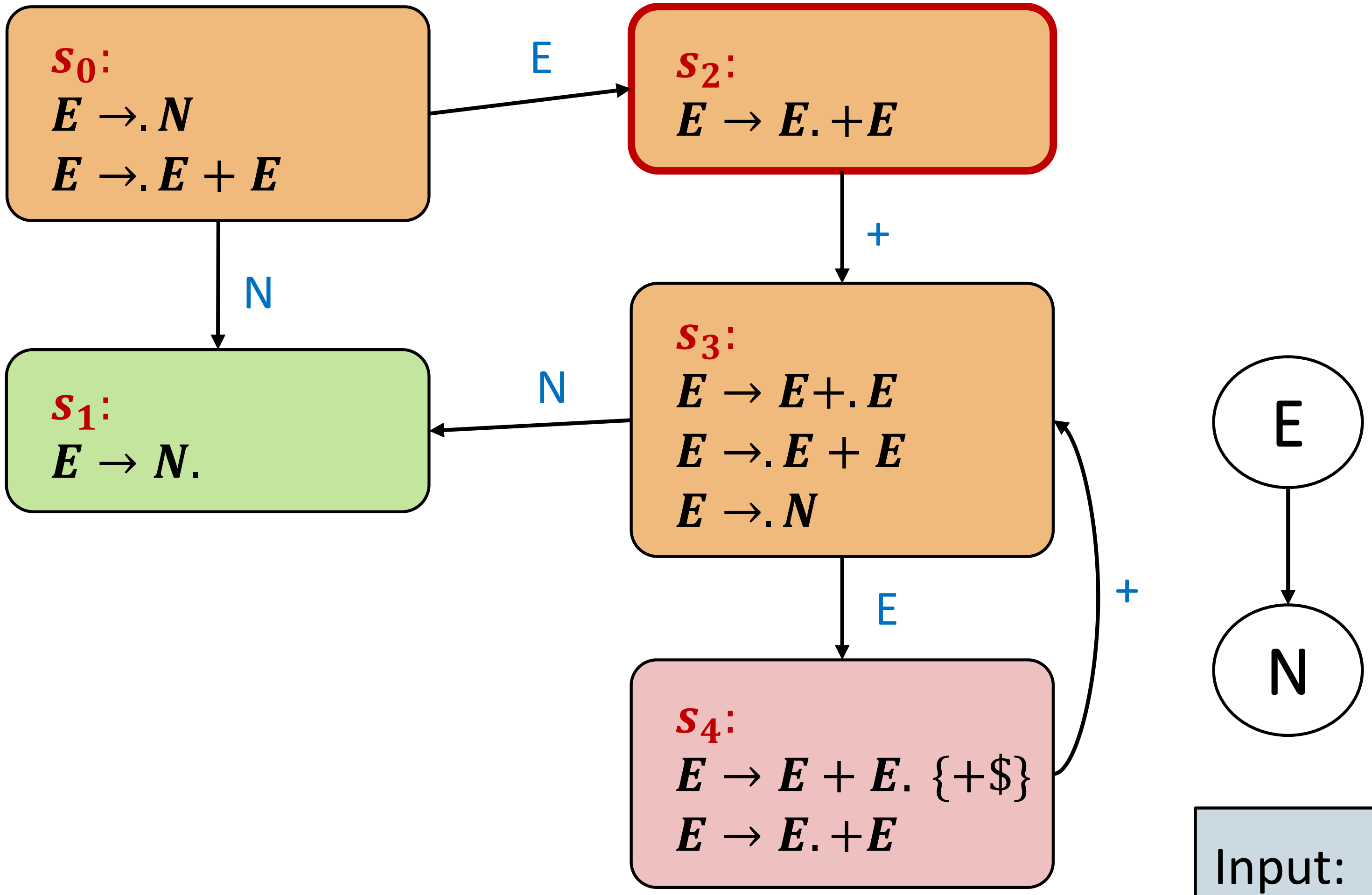
Stack: $\boldsymbol{s_0}$



N

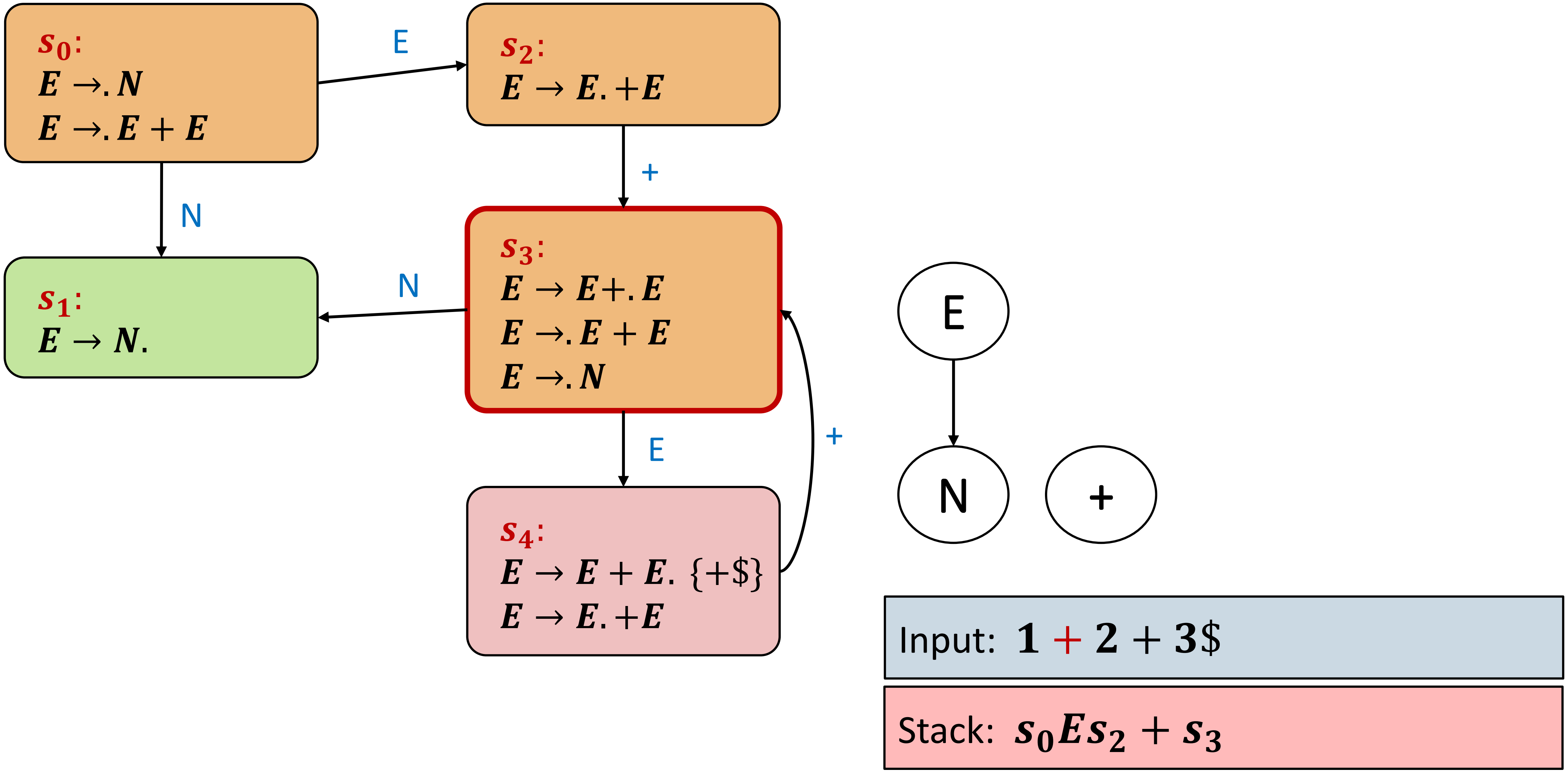
Input: $\mathbf{1} + 2 + 3\$$

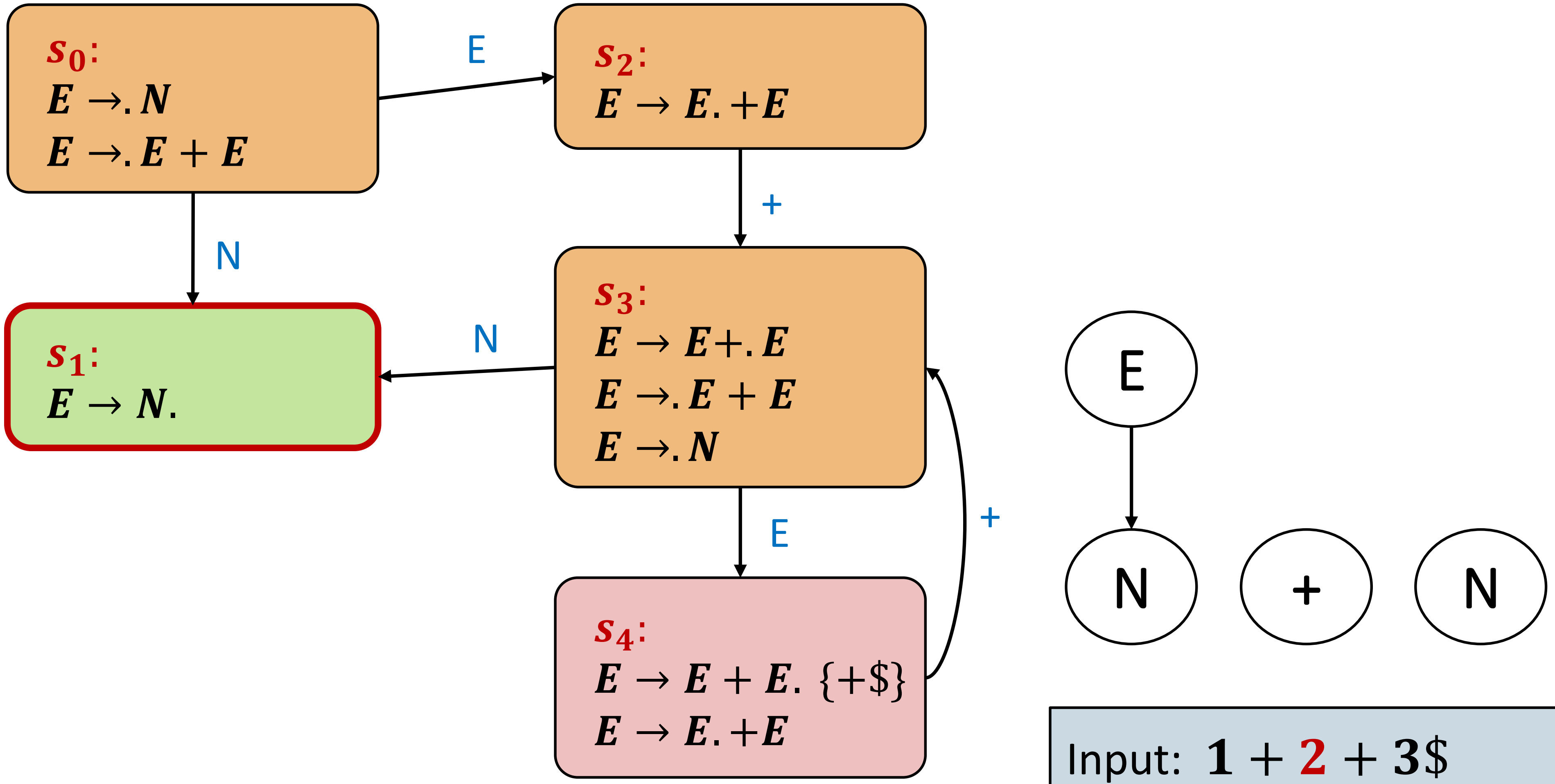
Stack: $s_0 N s_1$



Input: $\mathbf{1} + 2 + 3\$$

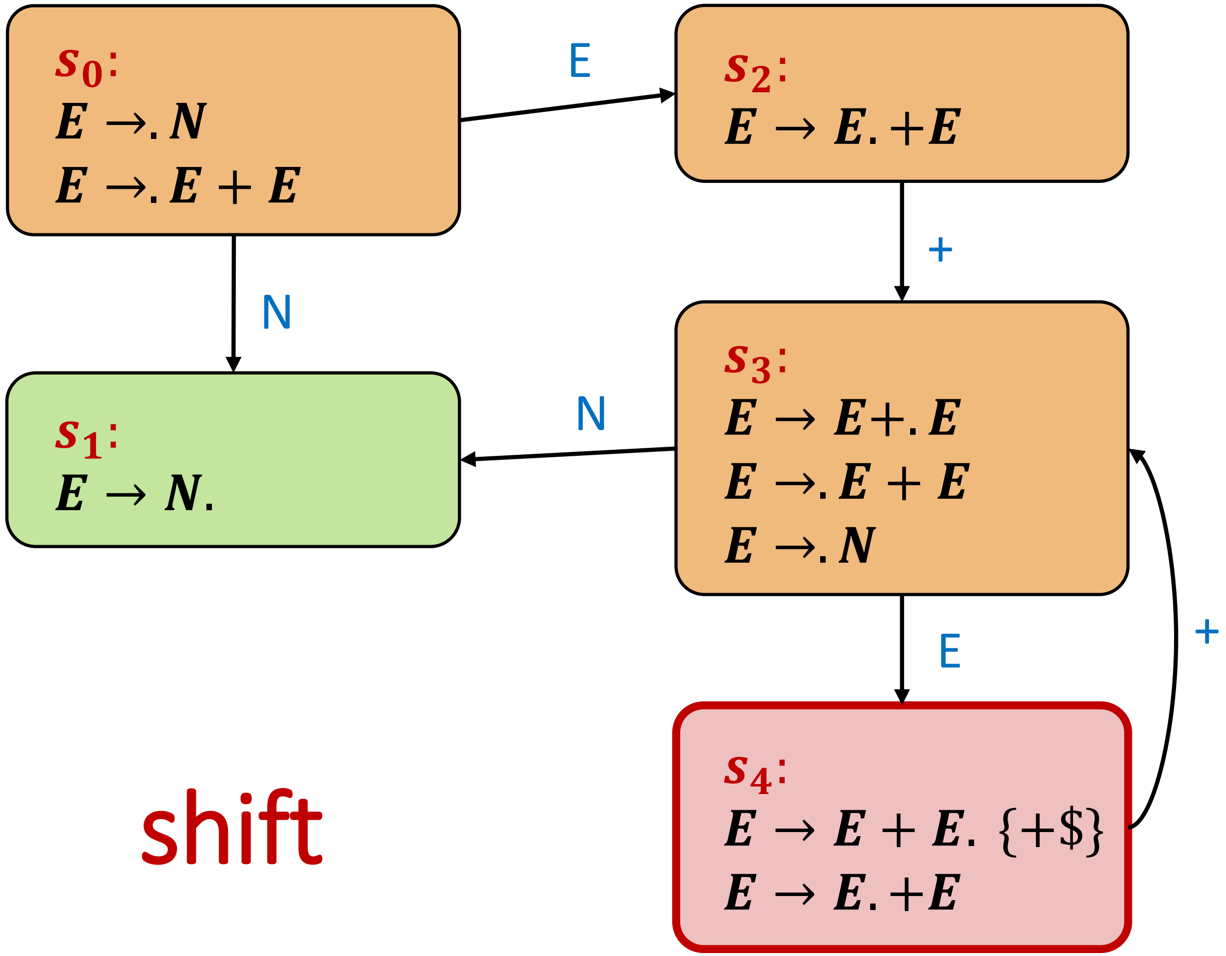
Stack: $s_0 E s_2$


s_0:
$E \to .N$
$E \to .E + E$
E
s_2:
$E \to E.+E$
+
N
s_1:
$E \to N.$
N
s_3:
$E \to E+.E$
$E \to .E + E$
$E \to .N$
E
+
s_4:
$E \to E + E.\ \{+\$\}$
$E \to E.+E$
E
N
+
Input: $1 + 2 + 3\$$
Stack: $s_0 E s_2 + s_3$

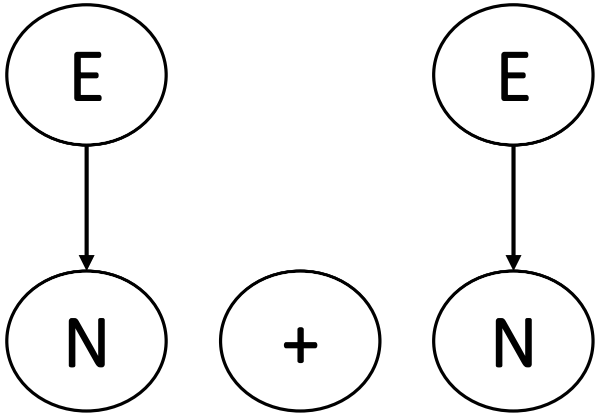


Input: $\mathbf{1 + 2 + 3\$}$

Stack: $\boldsymbol{s_0 E s_2 + s_3 N s_1}$

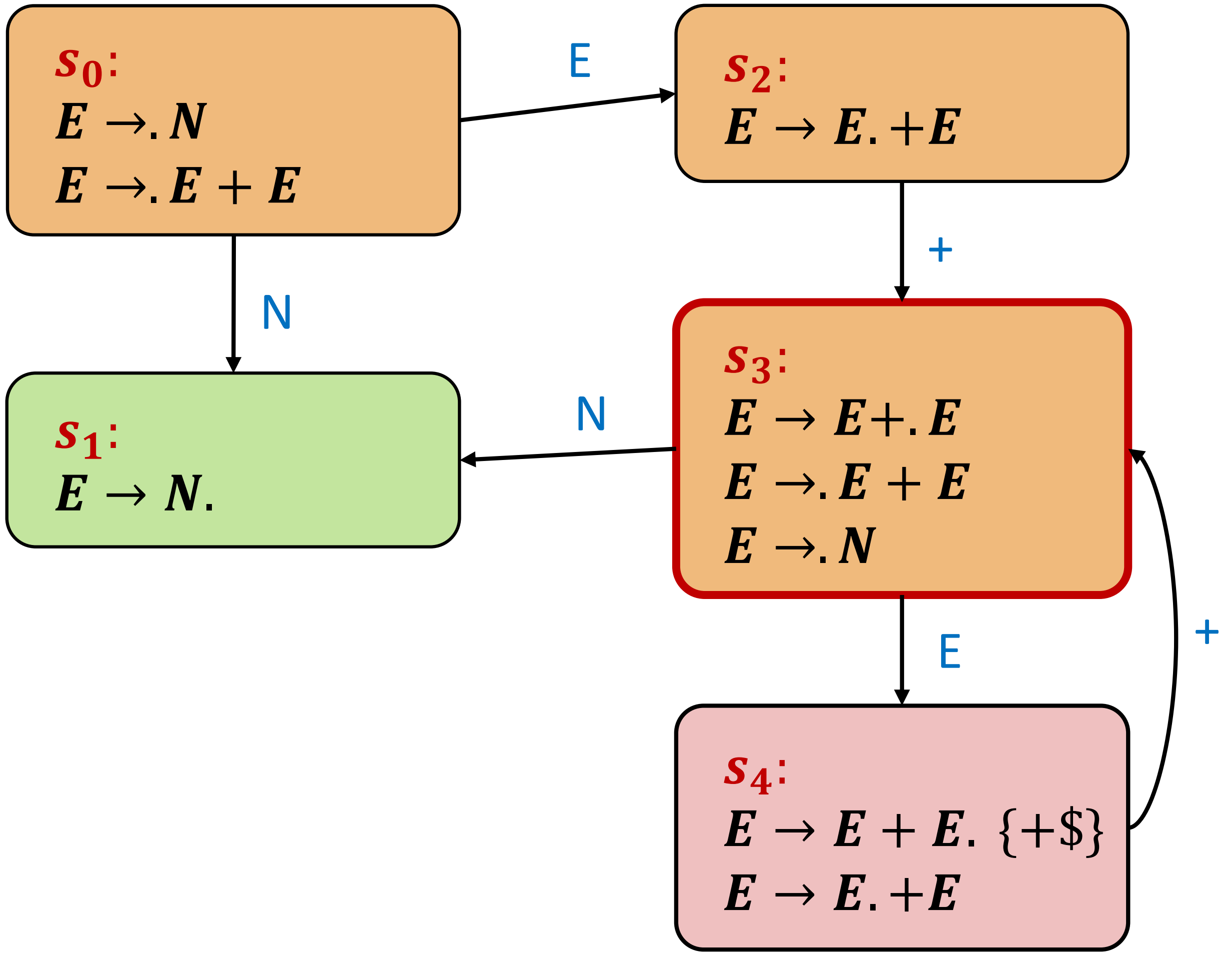


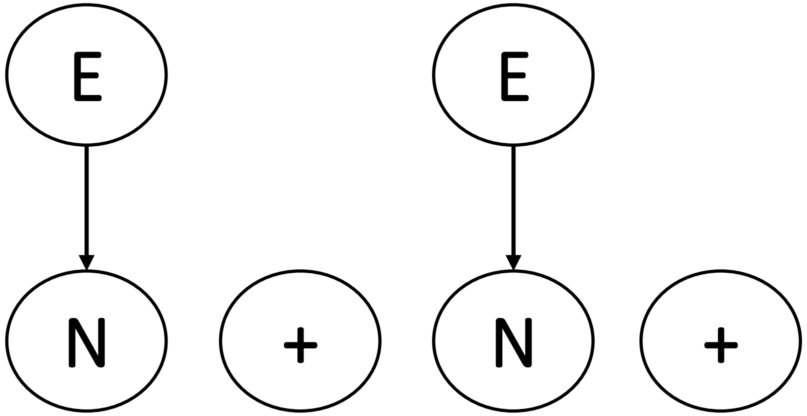
shift



Input: $\mathbf{1} + \mathbf{2} + \mathbf{3}\$$

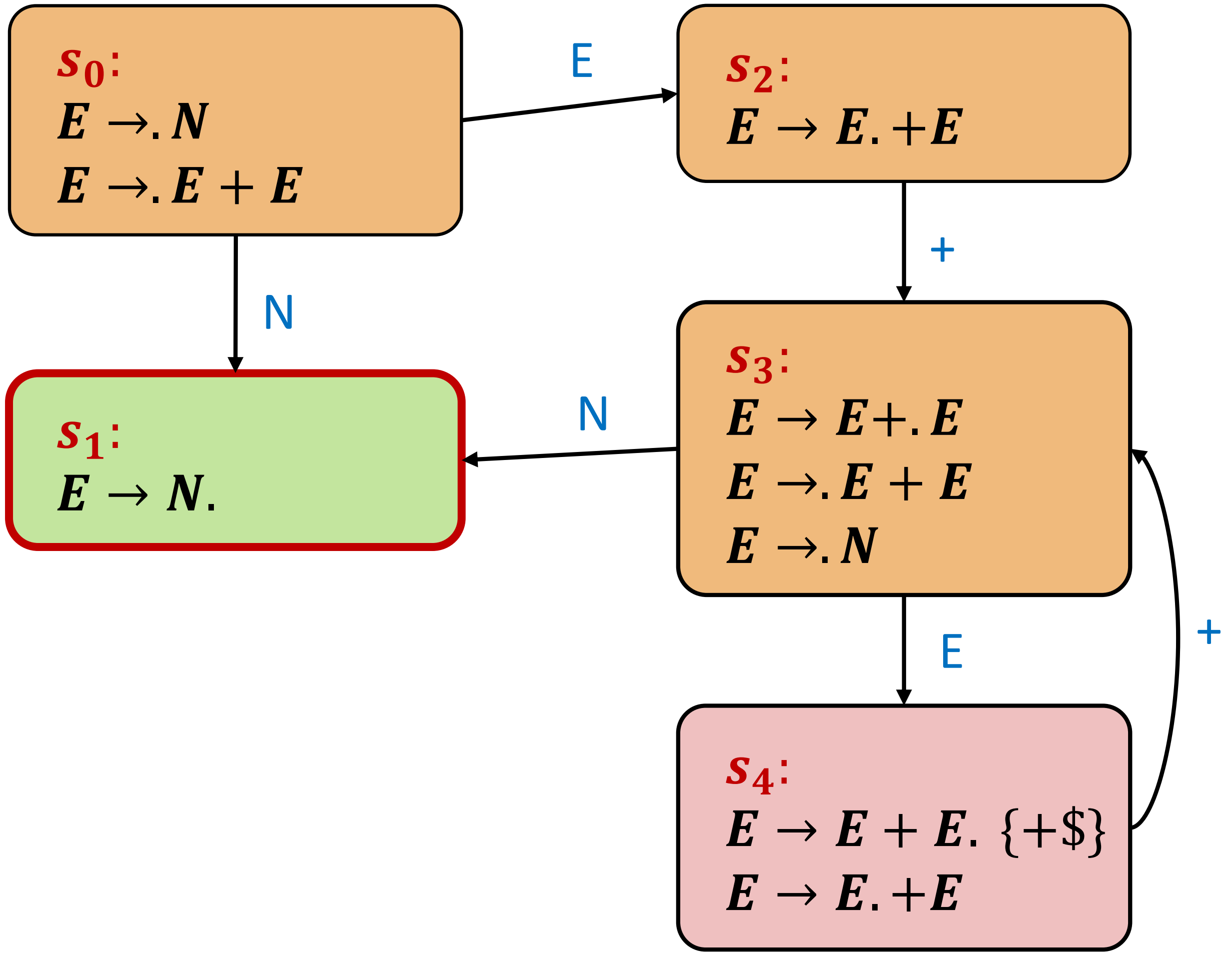
Stack: $s_0 E s_2 + s_3 E s_4$

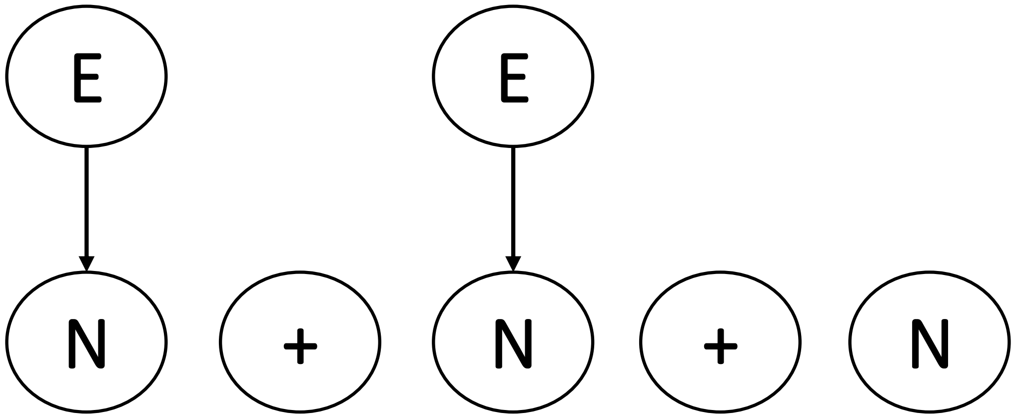




Input: $\mathbf{1 + 2 + 3\$}$

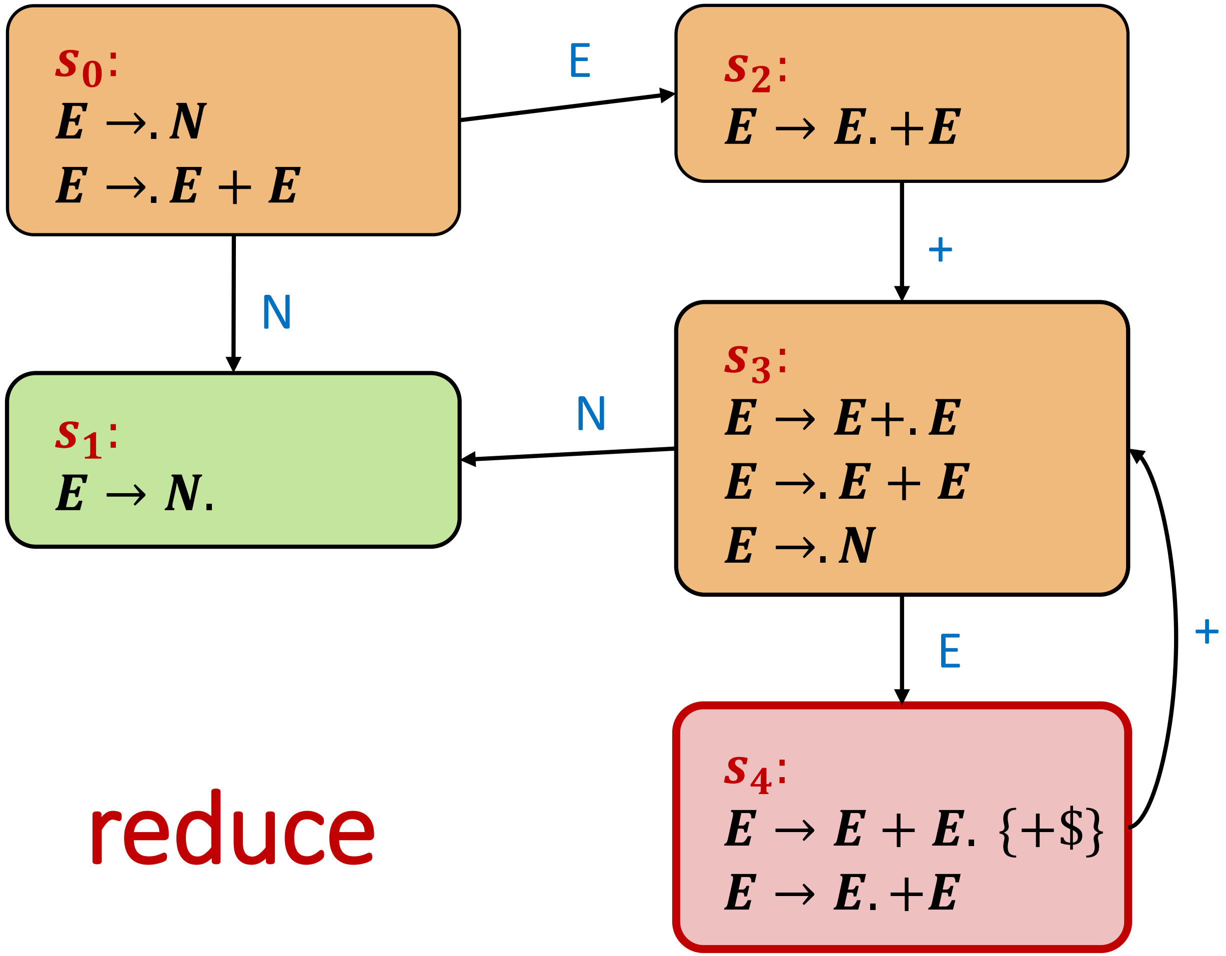
Stack: $s_0 E s_2 + s_3 E s_4 + s_3$



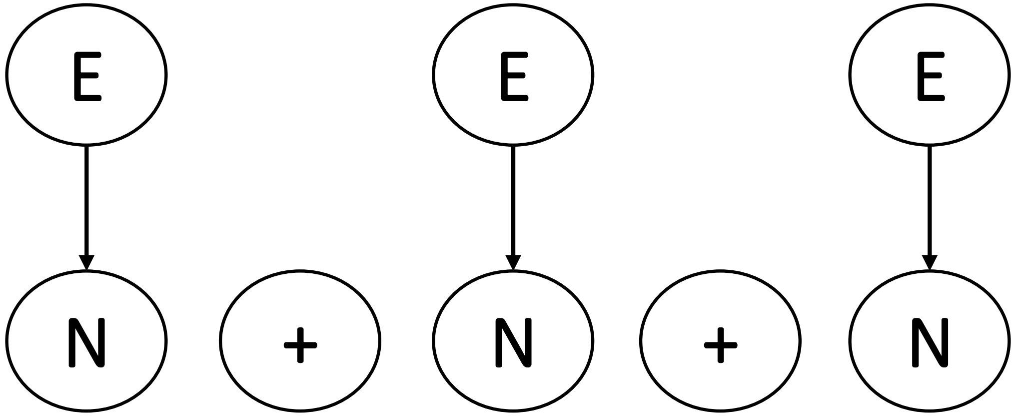


Input: $1 + 2 + 3\$$

Stack: $s_0 E s_2 + s_3 E s_4 + s_3 N s_1$

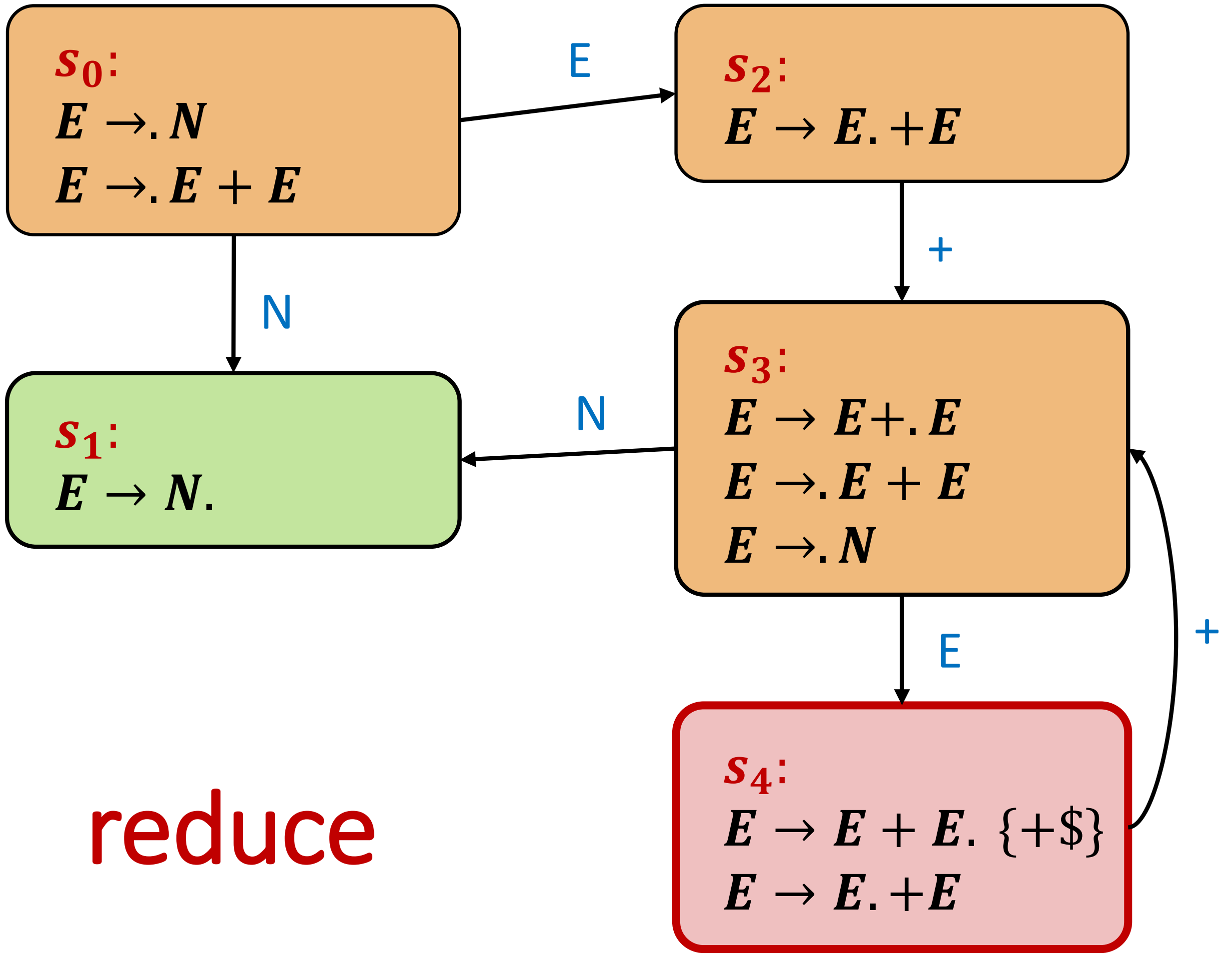


reduce

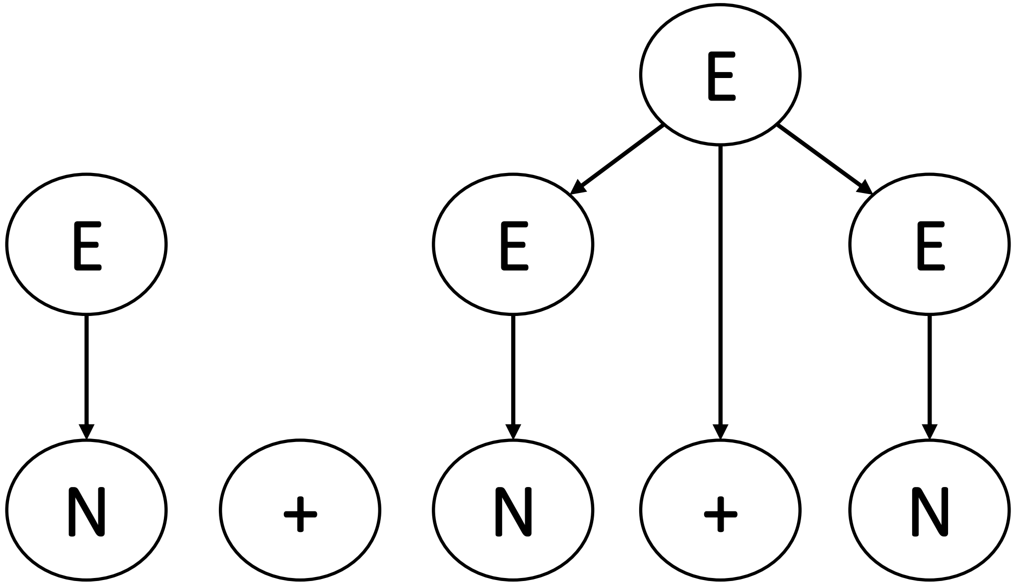


Input: $\mathbf{1 + 2 + 3}\$$

Stack: $s_0 E s_2 + s_3 E s_4 + s_3 E s_4$

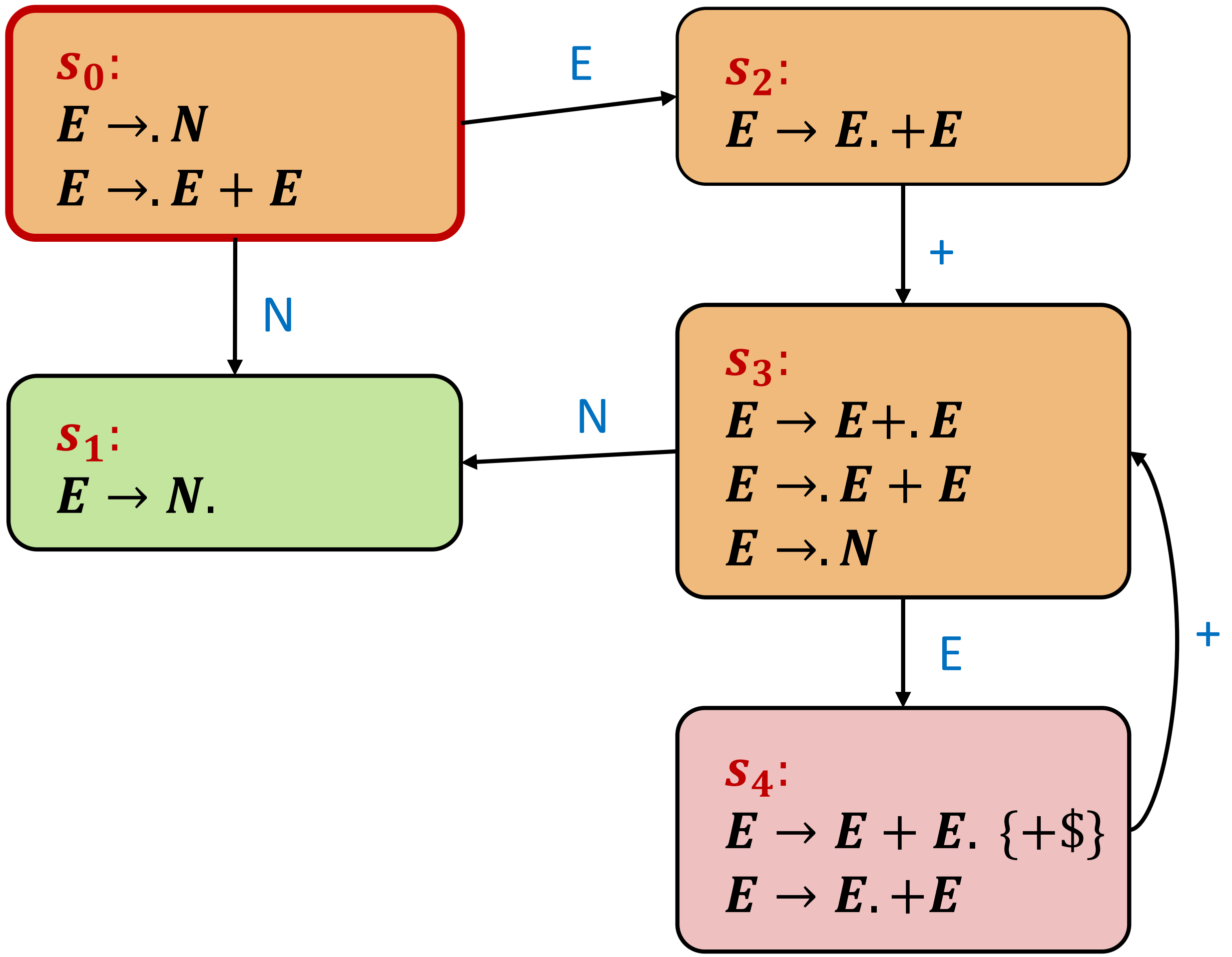


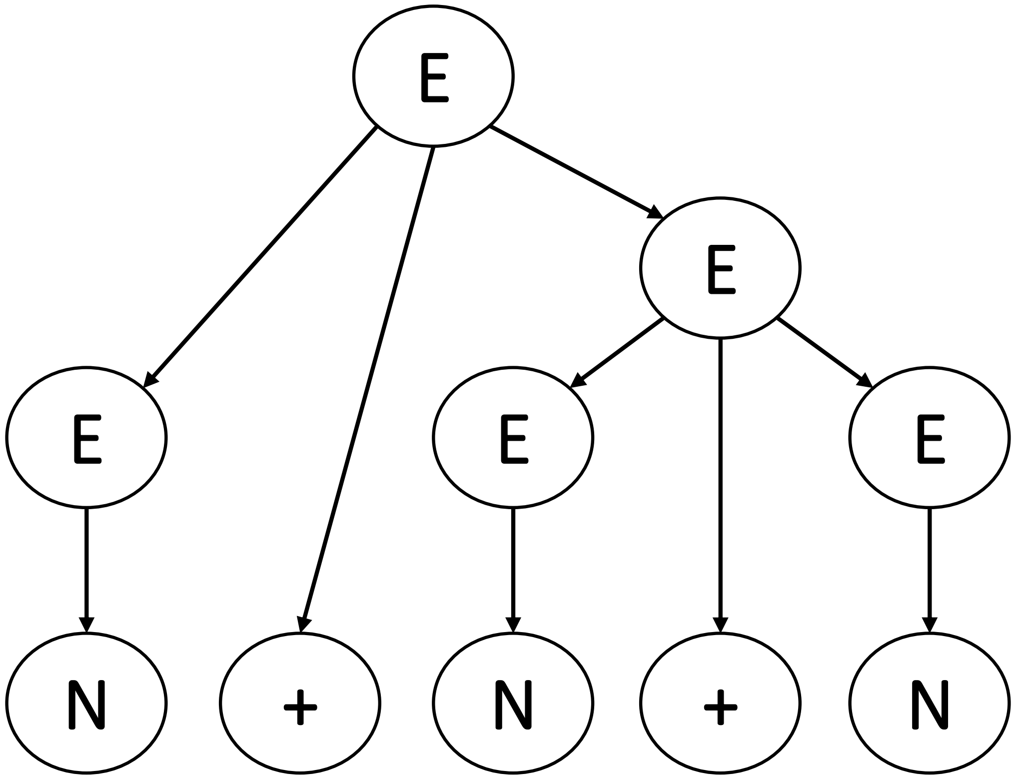
reduce



Input: $\mathbf{1 + 2 + 3\$}$

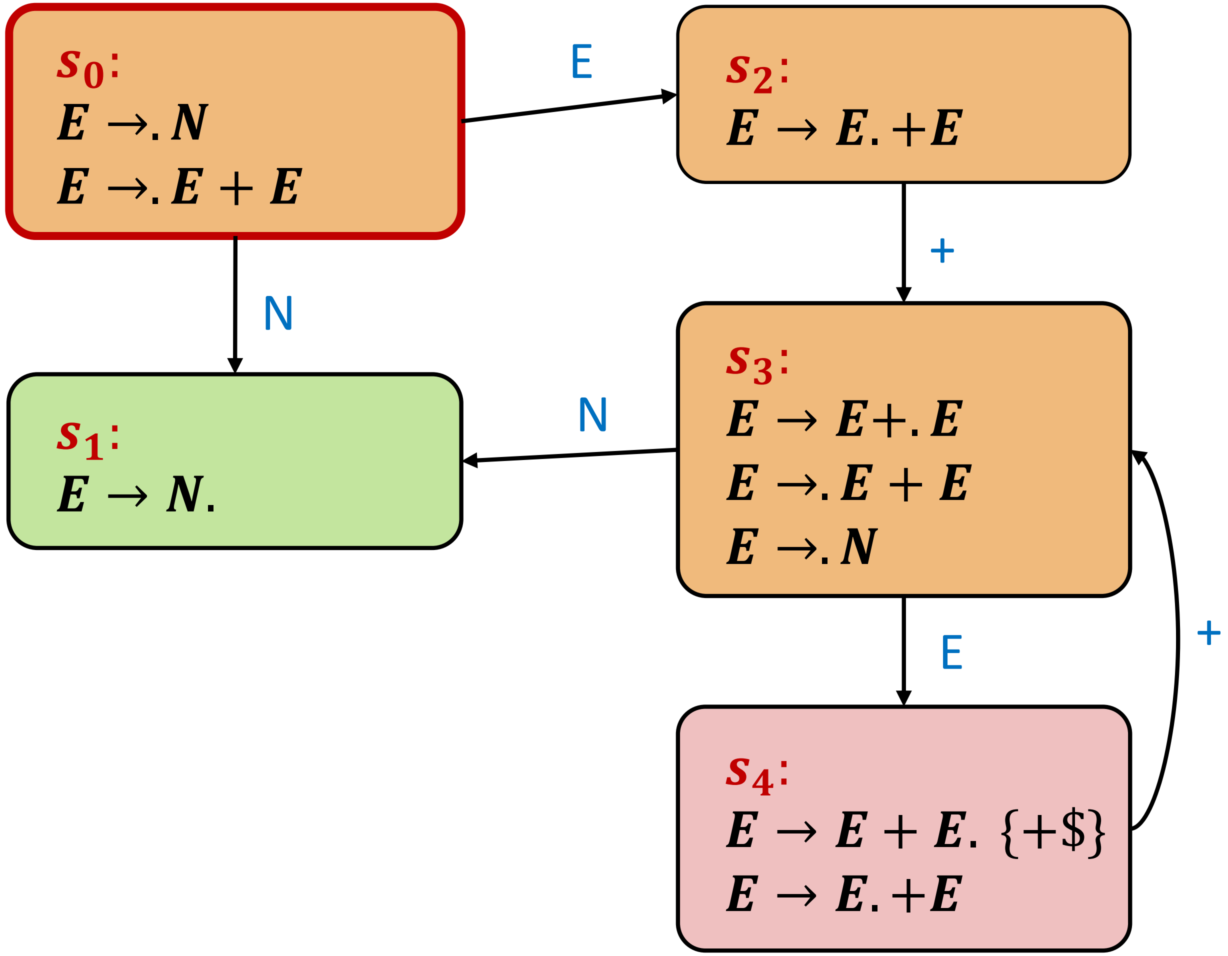
Stack: $s_0 E s_2 + s_3 E s_4$

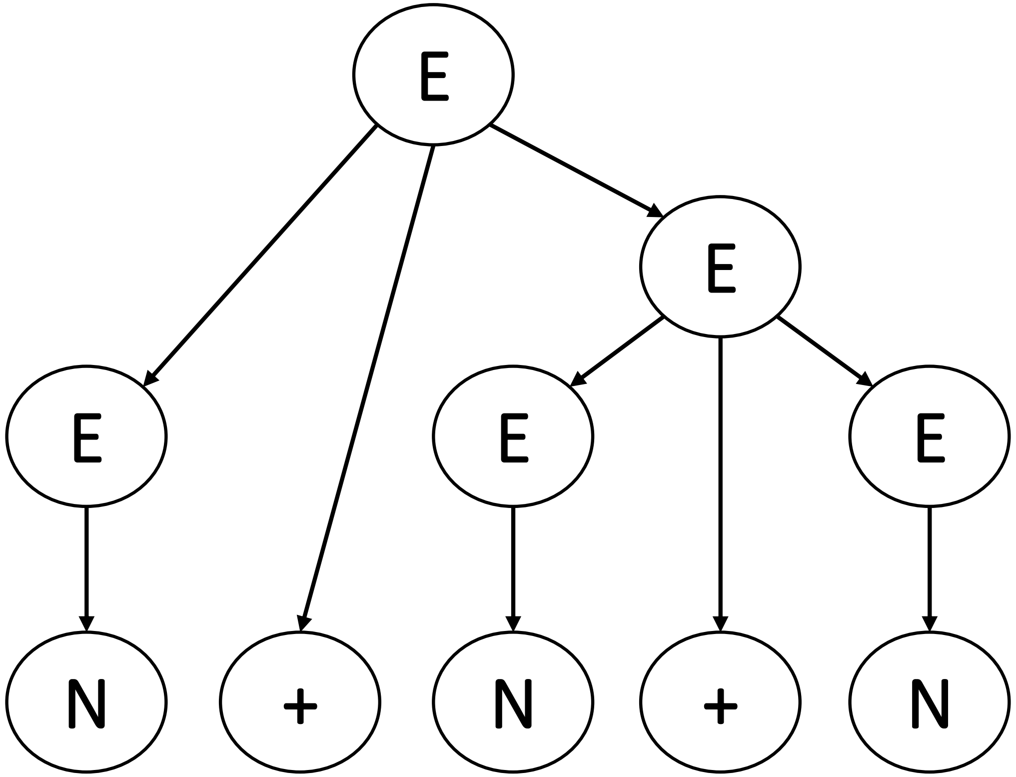




Input: $\mathbf{1} + \mathbf{2} + \mathbf{3}\$$

Stack: $s_0 E$





Input: $\mathbf{1+2+3\$}$

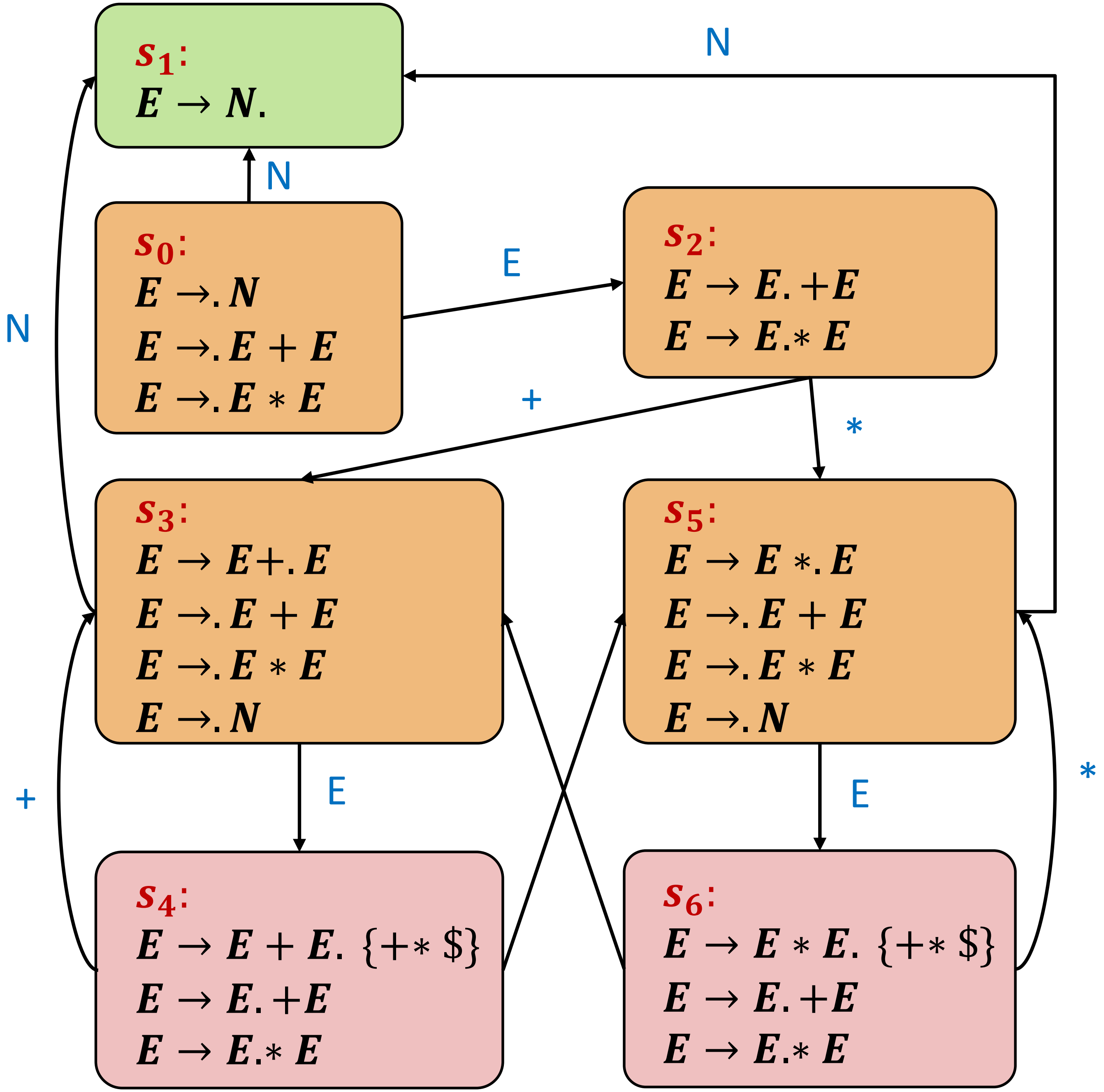
Stack: $s_0 E$

Resolving Conflicts

Consider the following CFG:

- $E \to N$
- $E \to E + E$
- $E \to E * E$

What will be the **transition system** of the SLR(1) parser for this CFG?



s_1:
$E \to N.$
s_0:
$E \to .N$
$E \to .E + E$
$E \to .E * E$
s_2:
$E \to E.+E$
$E \to E.* E$
s_3:
$E \to E+.E$
$E \to .E + E$
$E \to .E * E$
$E \to .N$
s_5:
$E \to E *.E$
$E \to .E + E$
$E \to .E * E$
$E \to .N$
s_4:
$E \to E + E. \{+ * \$\}$
$E \to E.+E$
$E \to E.* E$
s_6:
$E \to E * E. \{+ * \$\}$
$E \to E.+E$
$E \to E.* E$
N
N
N
E
+
*
E
E
+
*

Resolving Conflicts

When having a shift/reduce conflict:

- $E_1 \rightarrow \alpha_1 t_1 \beta_1 .$
- $E_2 \rightarrow \alpha_2 . t_2 \beta_2$

If t_1 has higher precedence, **reduce**
If t_2 has higher precedence, **shift**

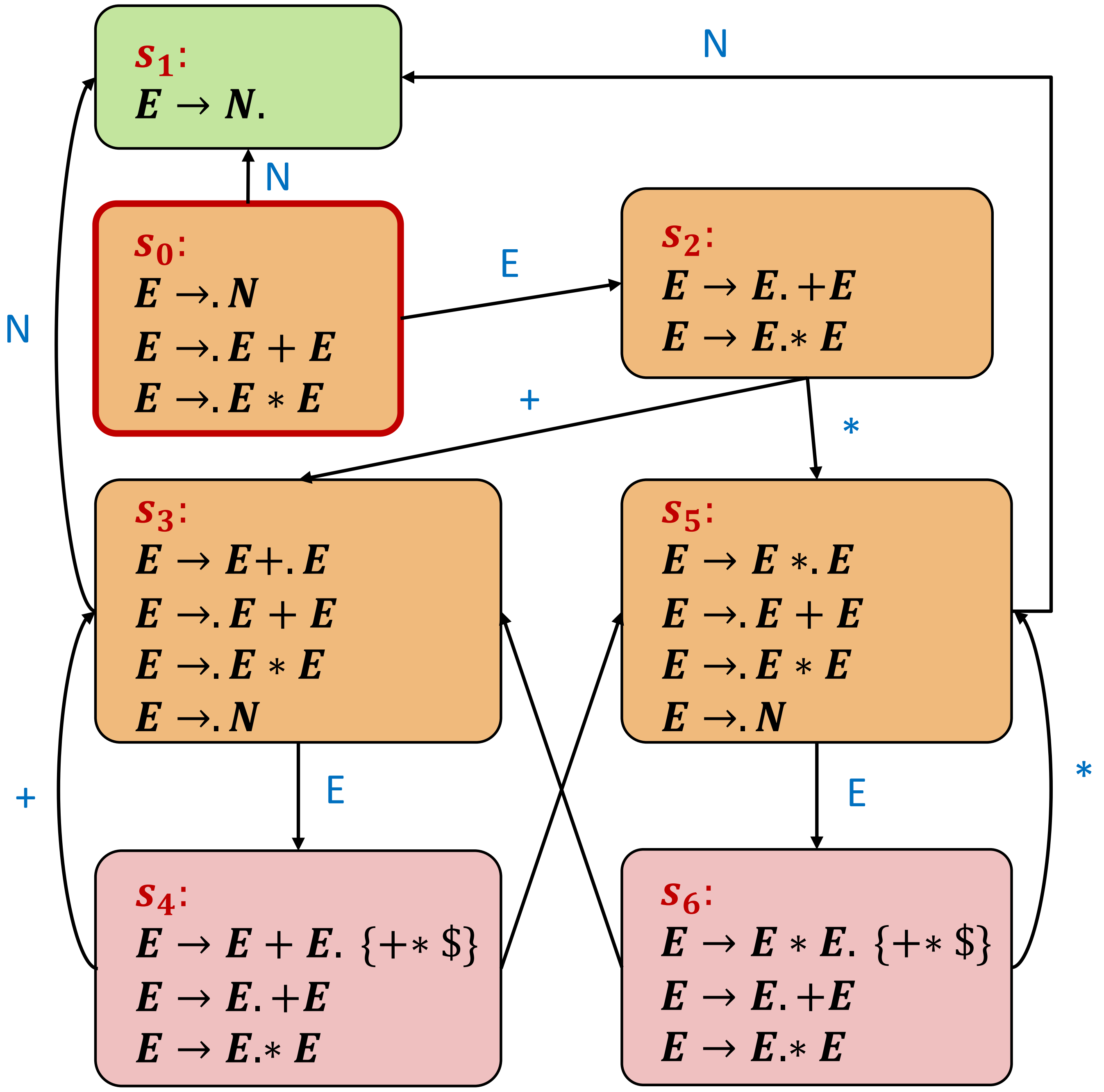
Resolving Conflicts

In our case:

- $E \to E + E.$
- $E \to E. * E$

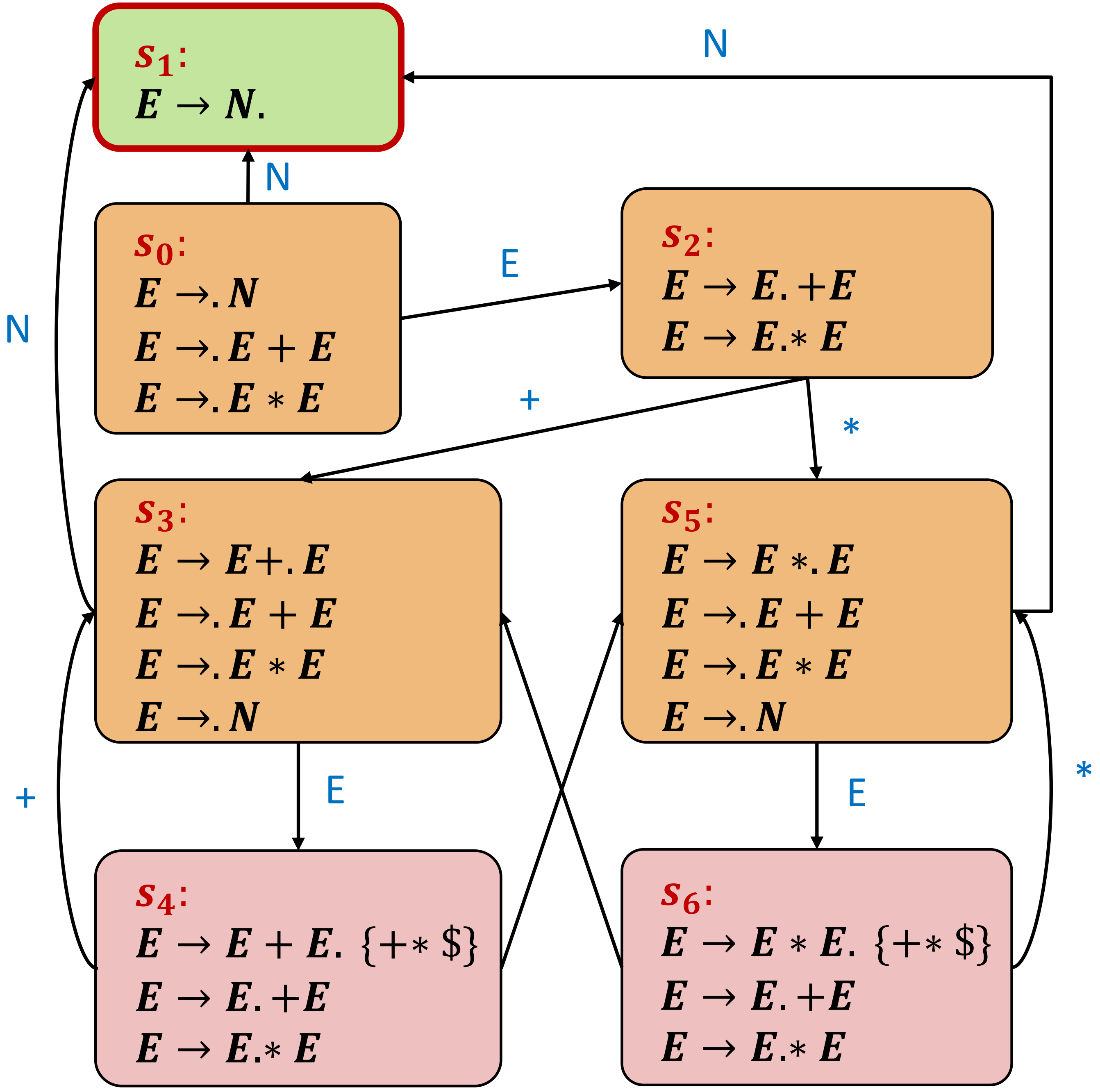
Assuming that multiplication has higher precedence:

- Resolve by Shift



Input: $3 + 4 * 8\$$

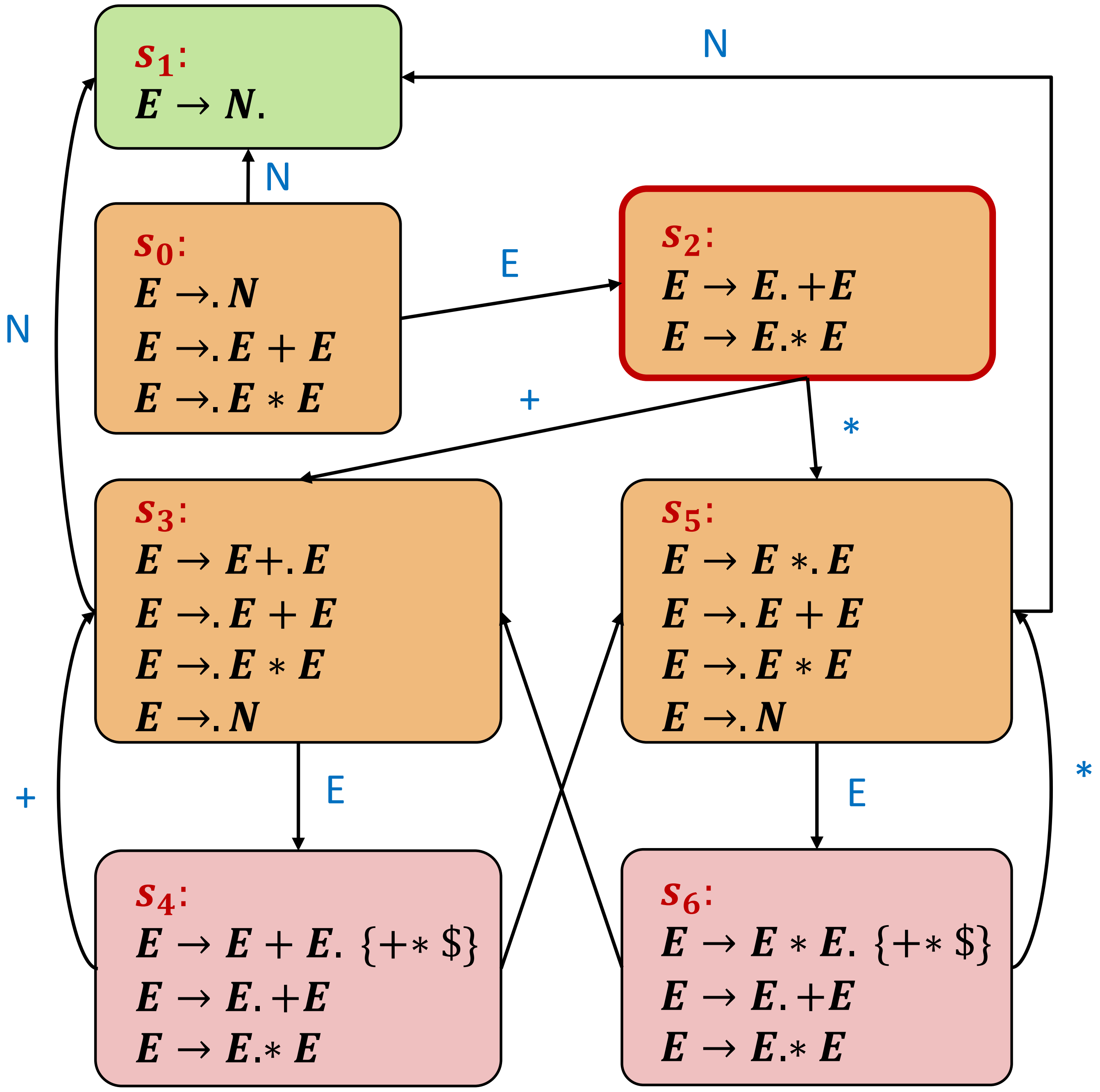
Stack: s_0

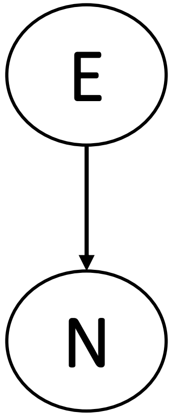


N

Input: $3 + 4 * 8\$$

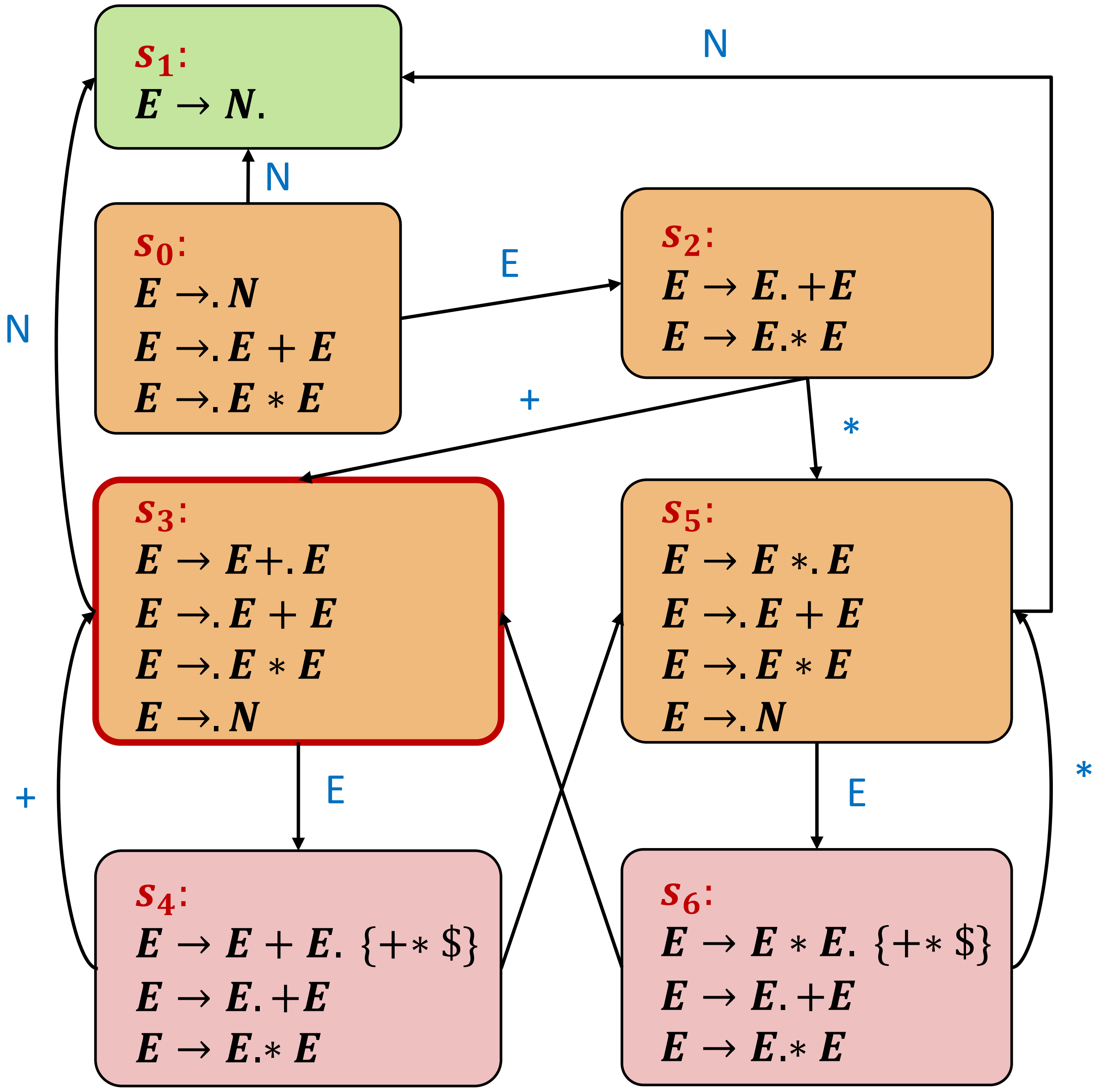
Stack: $s_0 N s_1$

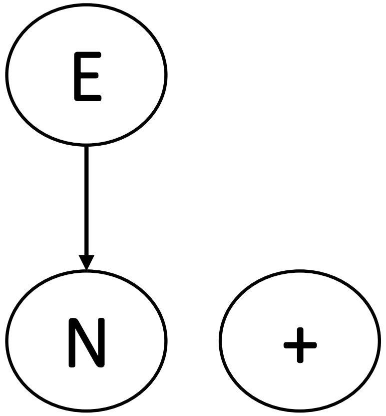




Input: $3 + 4 * 8\$$

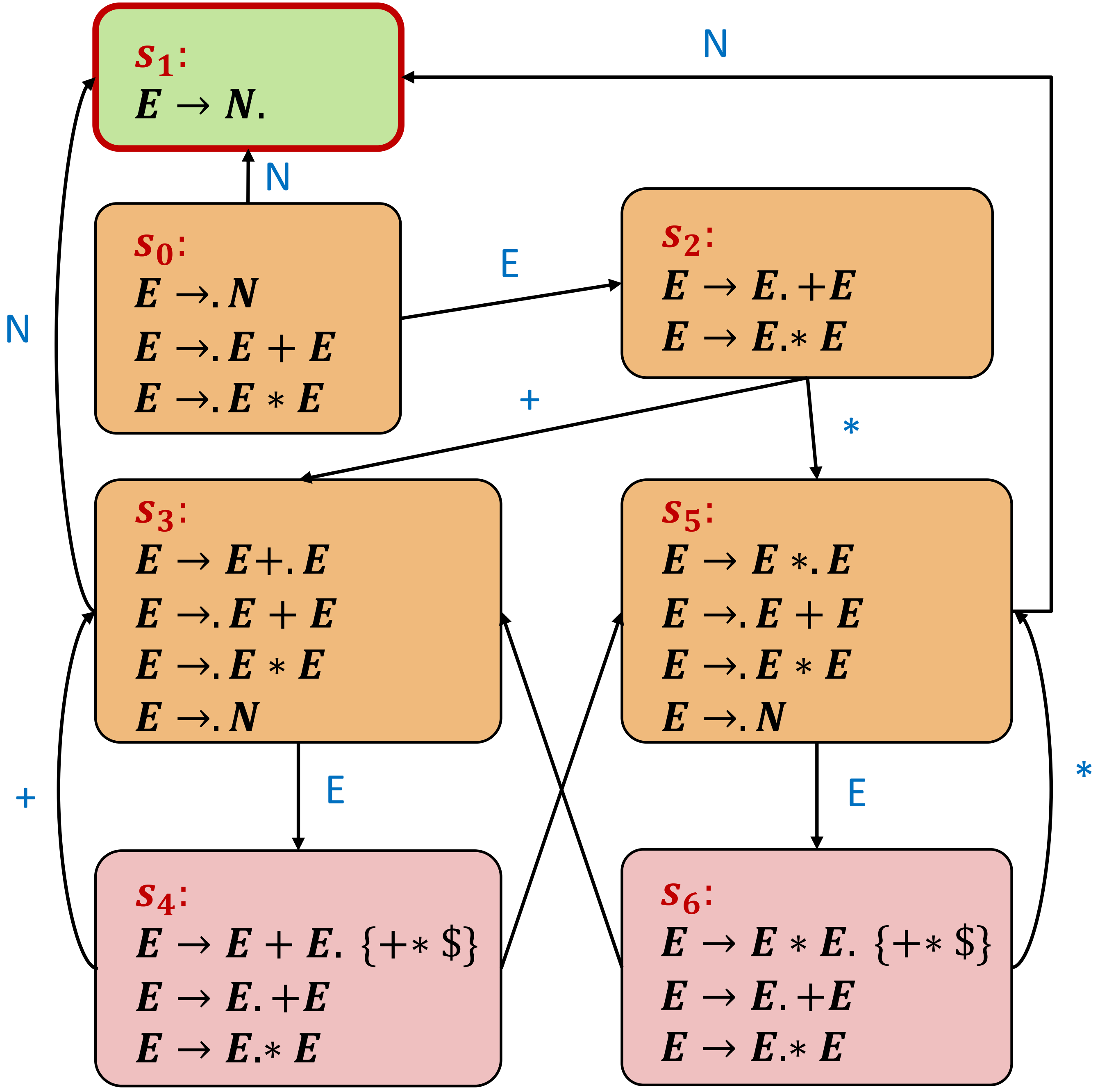
Stack: $s_0 E s_2$

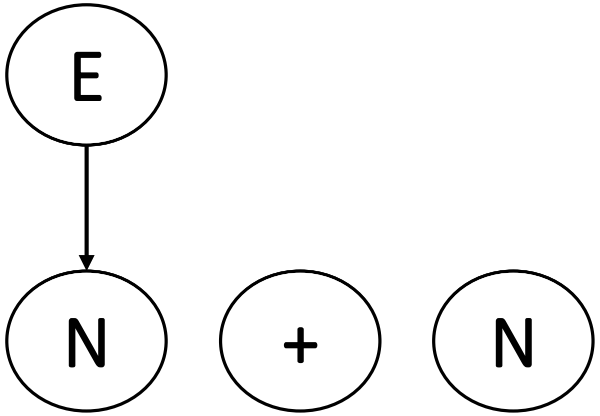




Input: $3 + 4 * 8\$$

Stack: $s_0 E s_2 + s_3$

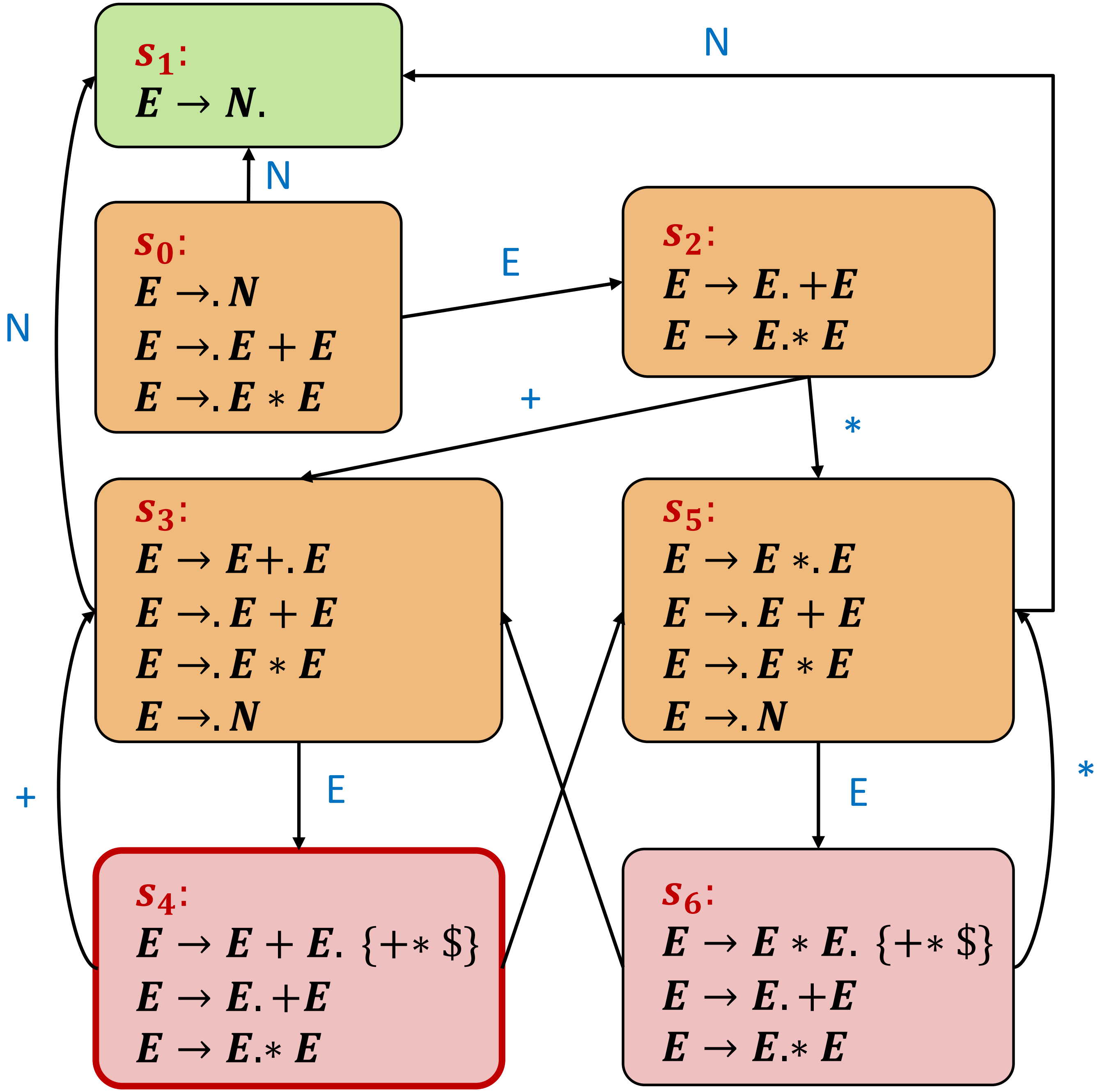


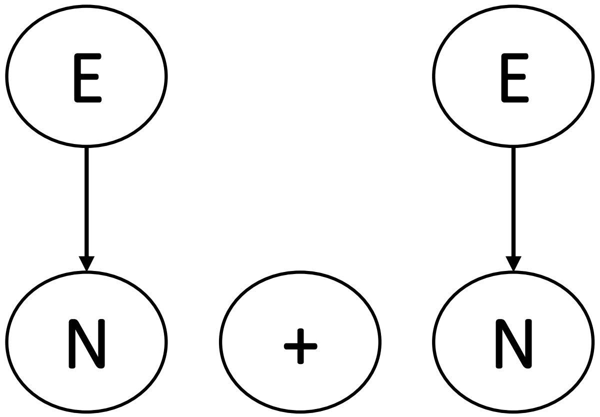


Input: $3 + 4 * 8\$$

Stack: $s_0 E s_2 + s_3 N s_1$

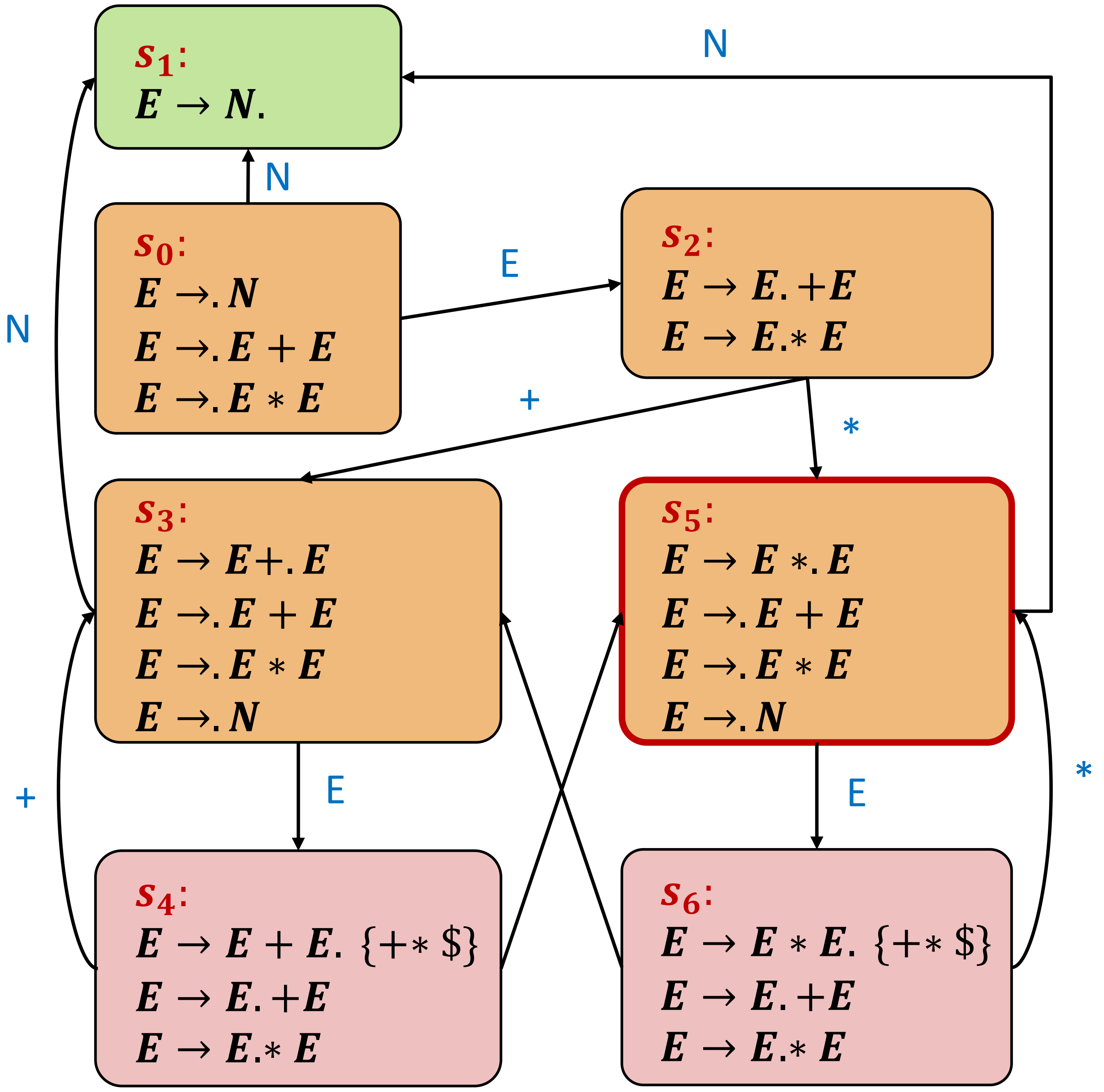
shift

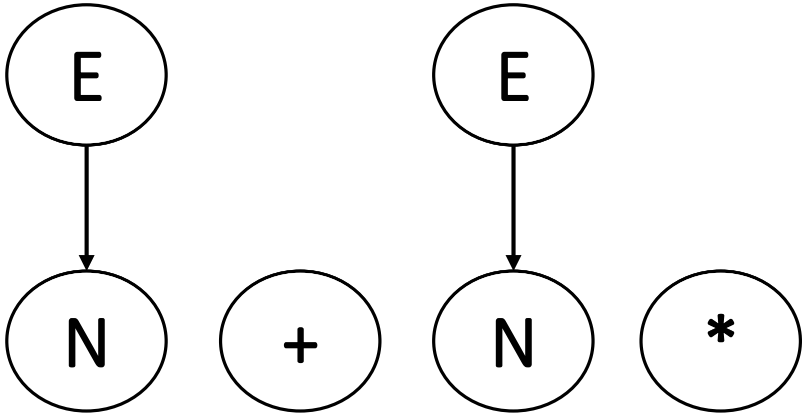




Input: $3 + 4 * 8\$$

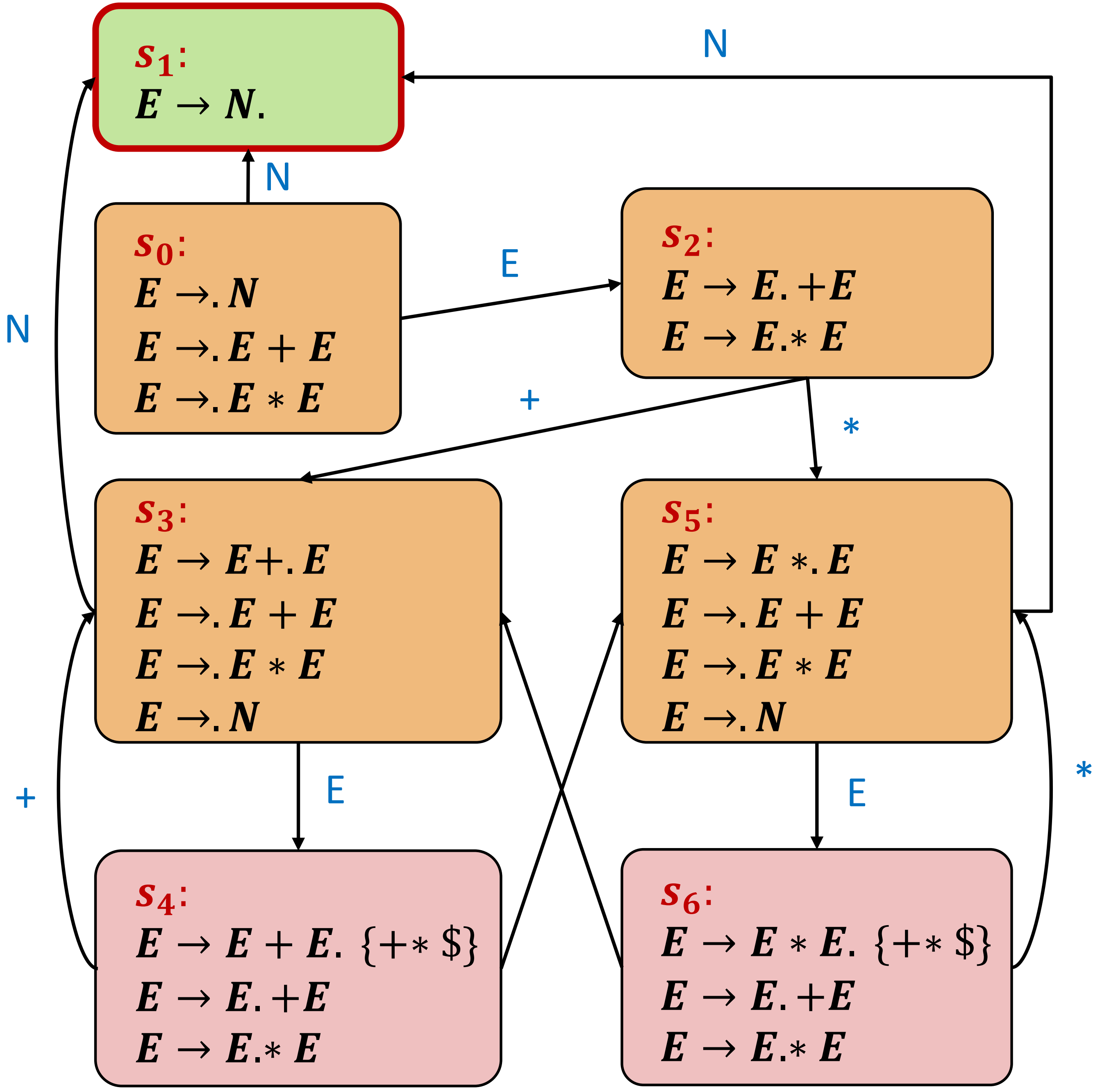
Stack: $s_0 E s_2 + s_3 E s_4$

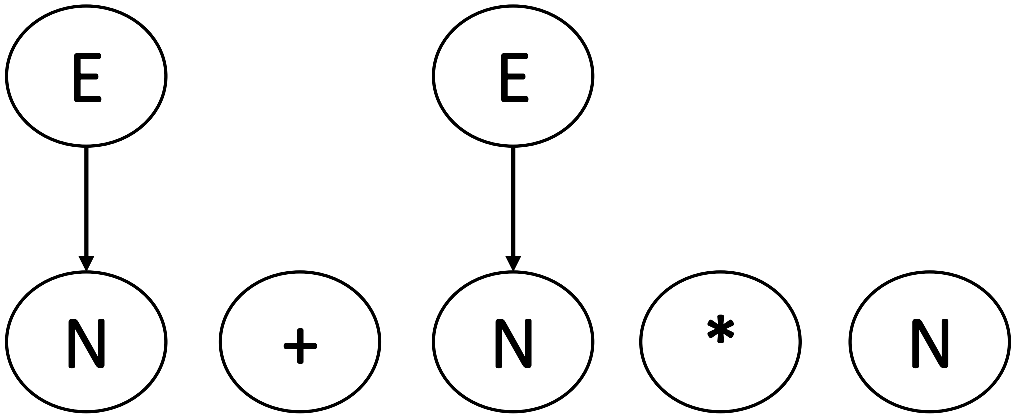




Input: $\mathbf{3 + 4 * 8\$}$

Stack: $s_0 E s_2 + s_3 E s_4 * s_5$

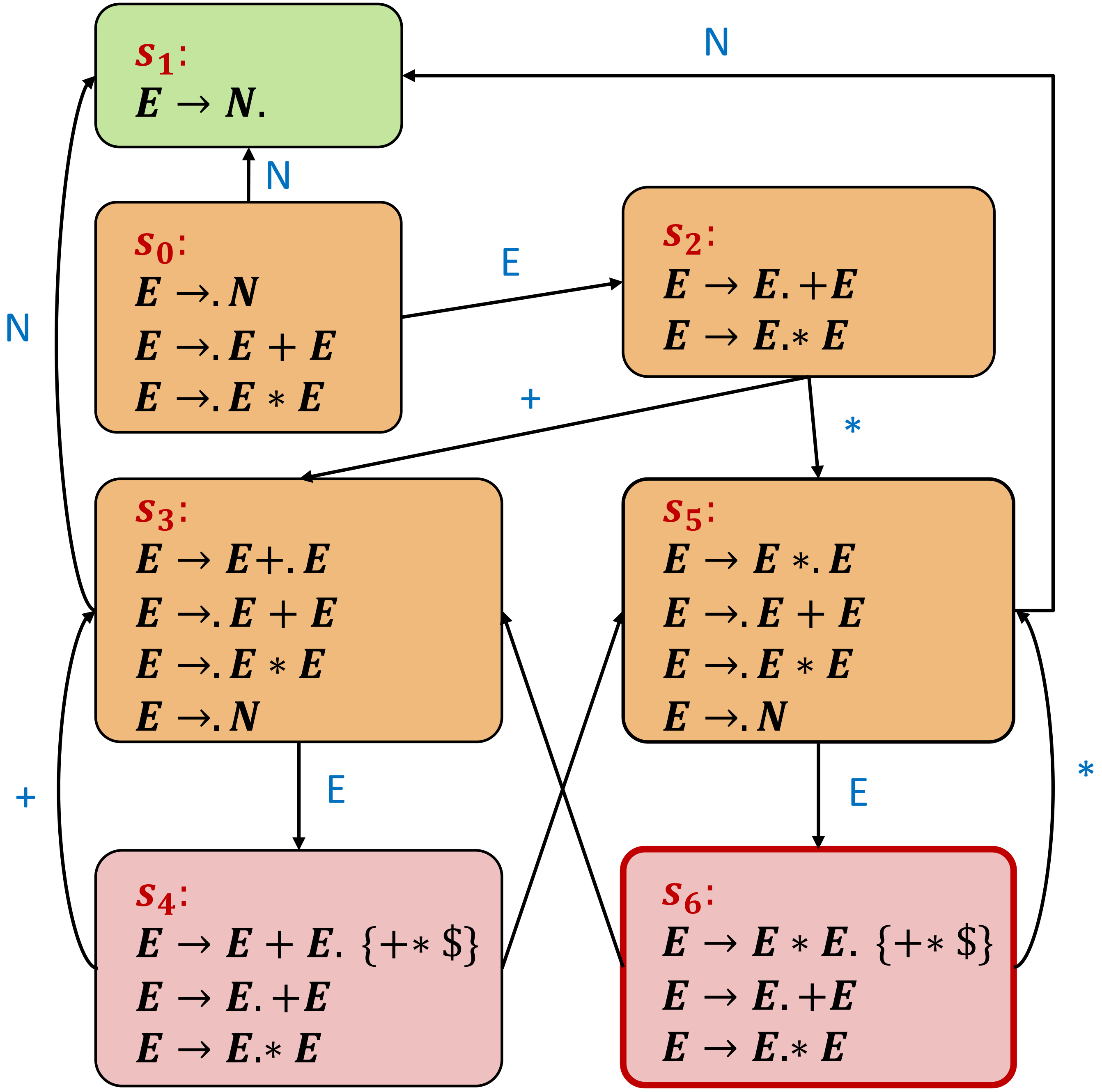


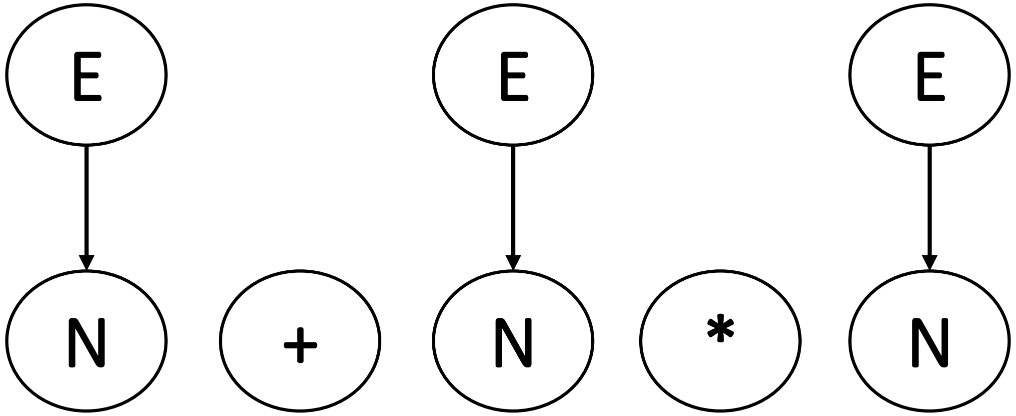


Input: $3 + 4 * \mathbf{8}\$$

Stack: $s_0 E s_2 + s_3 E s_4 * s_5 N s_1$

reduce

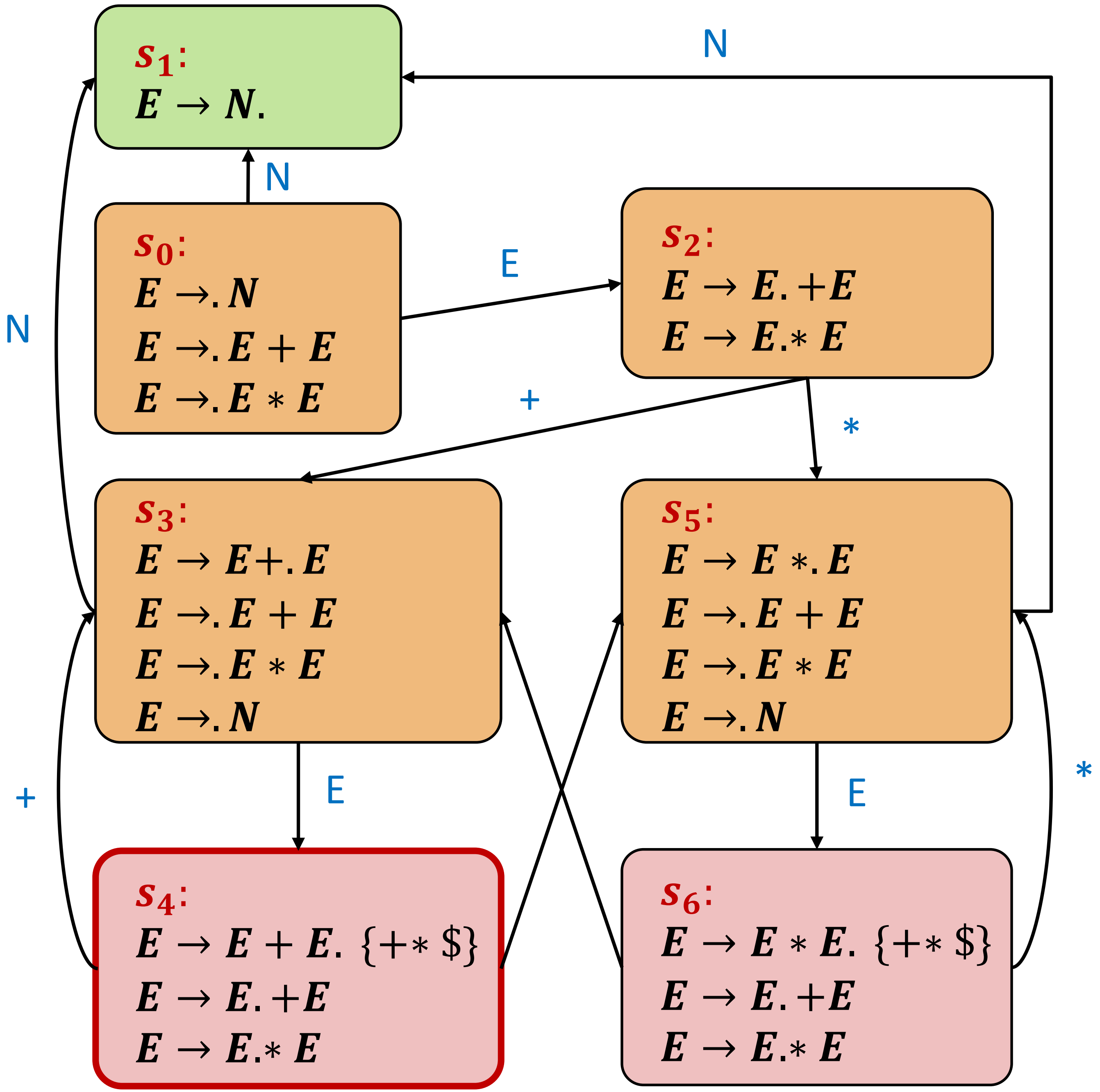


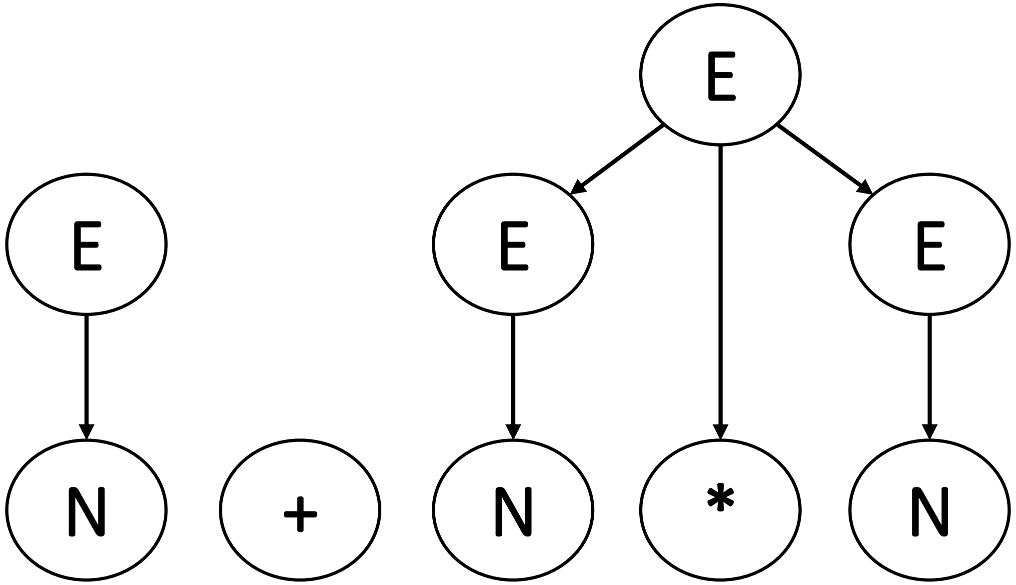


Input: $\mathbf{3 + 4 * 8\$}$

Stack: $s_0 E s_2 + s_3 E s_4 * s_5 E s_6$

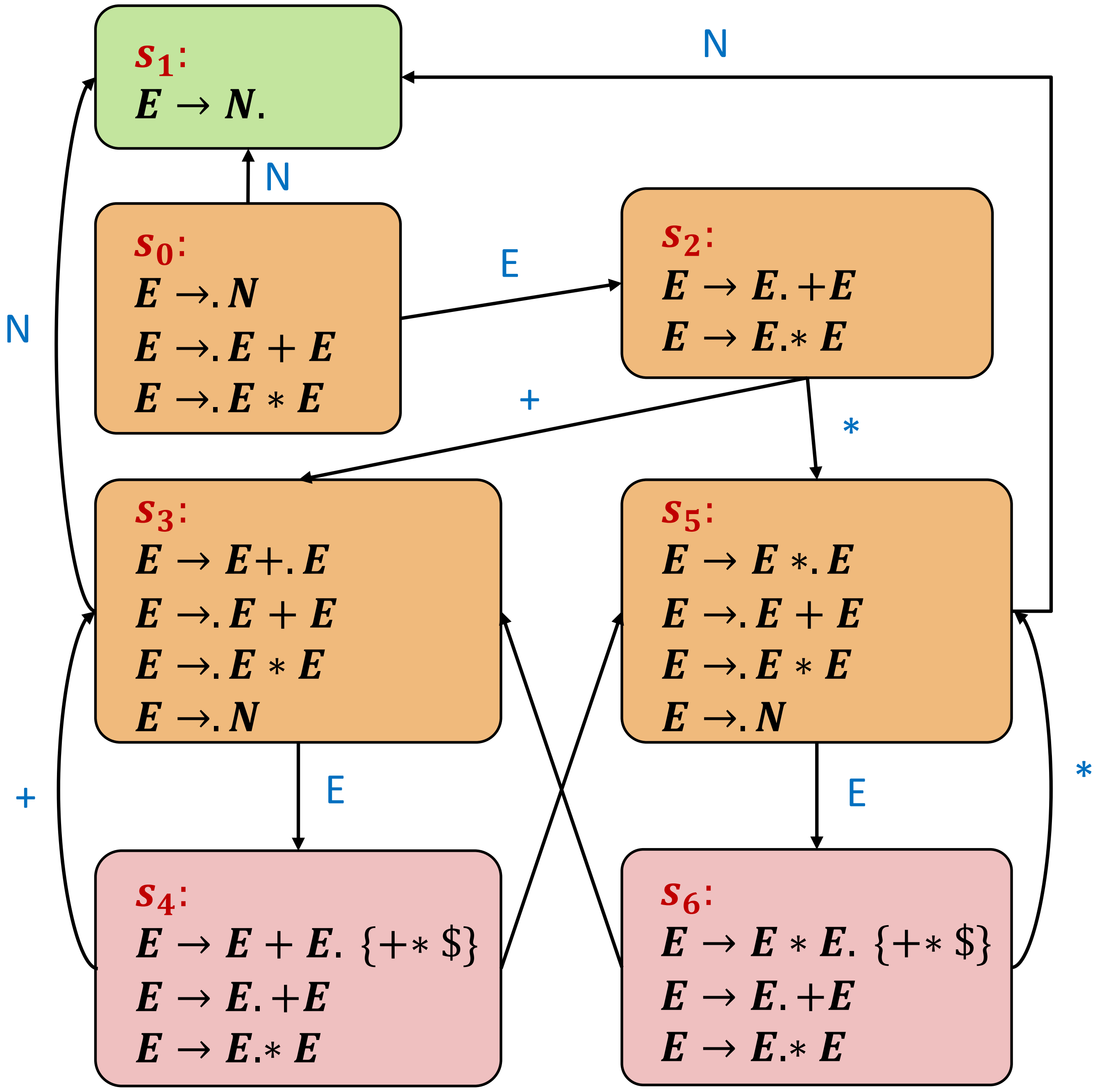
reduce

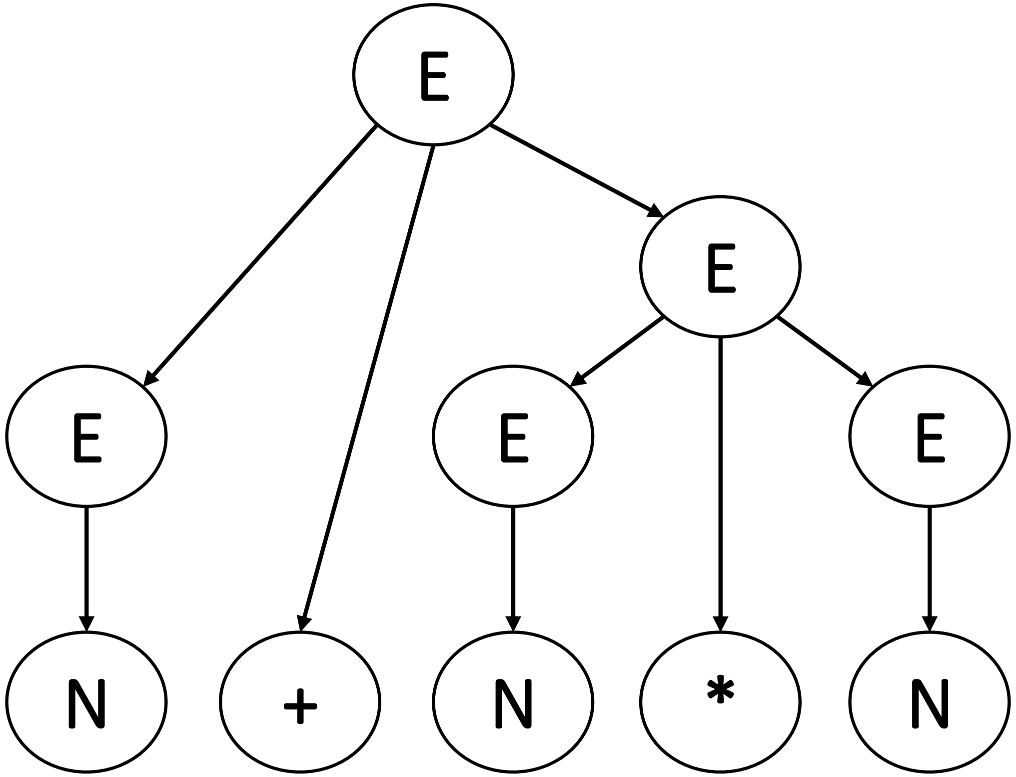




Input: $\mathbf{3 + 4 * 8\$}$

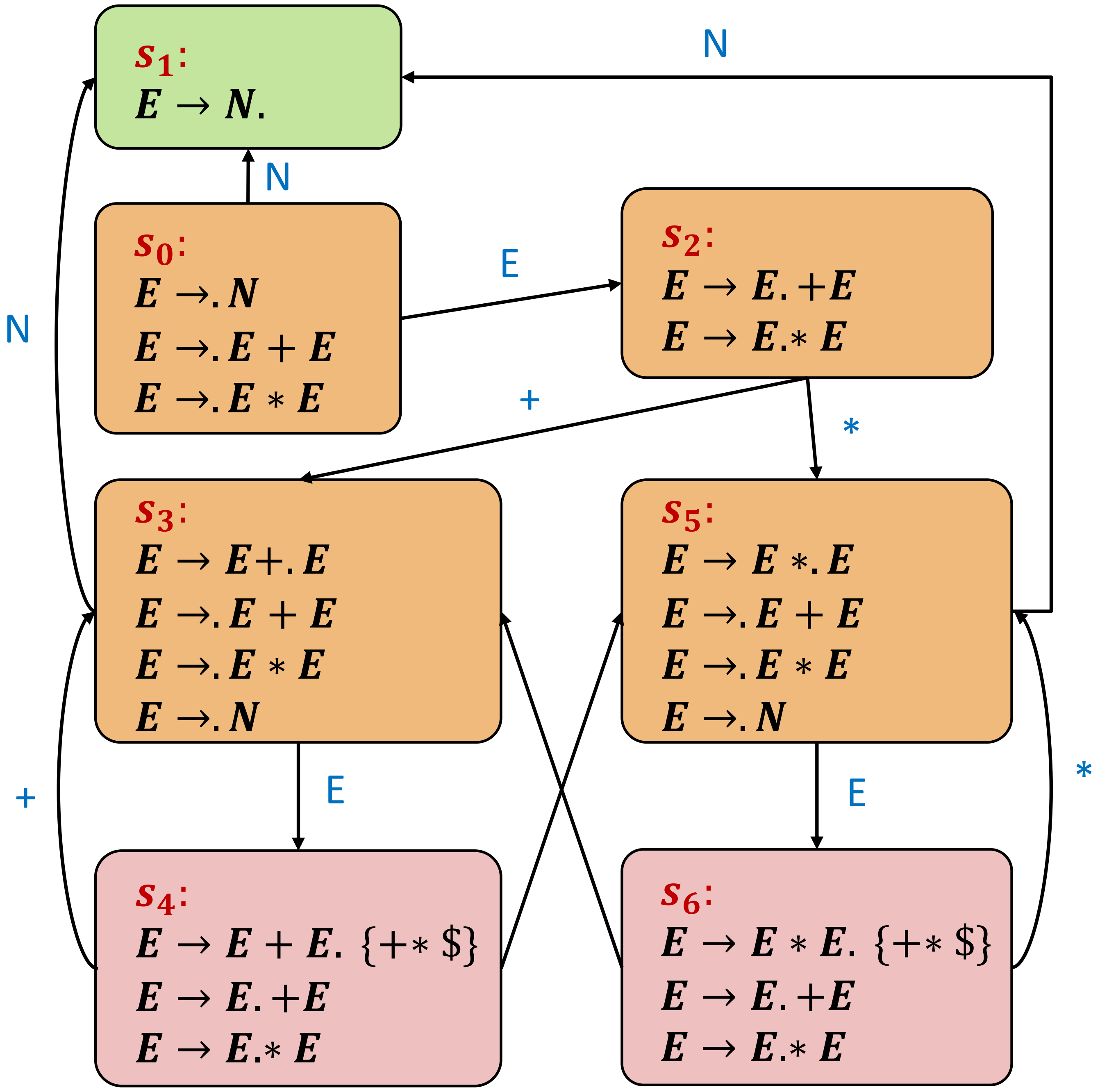
Stack: $s_0 E s_2 + s_3 E s_4$

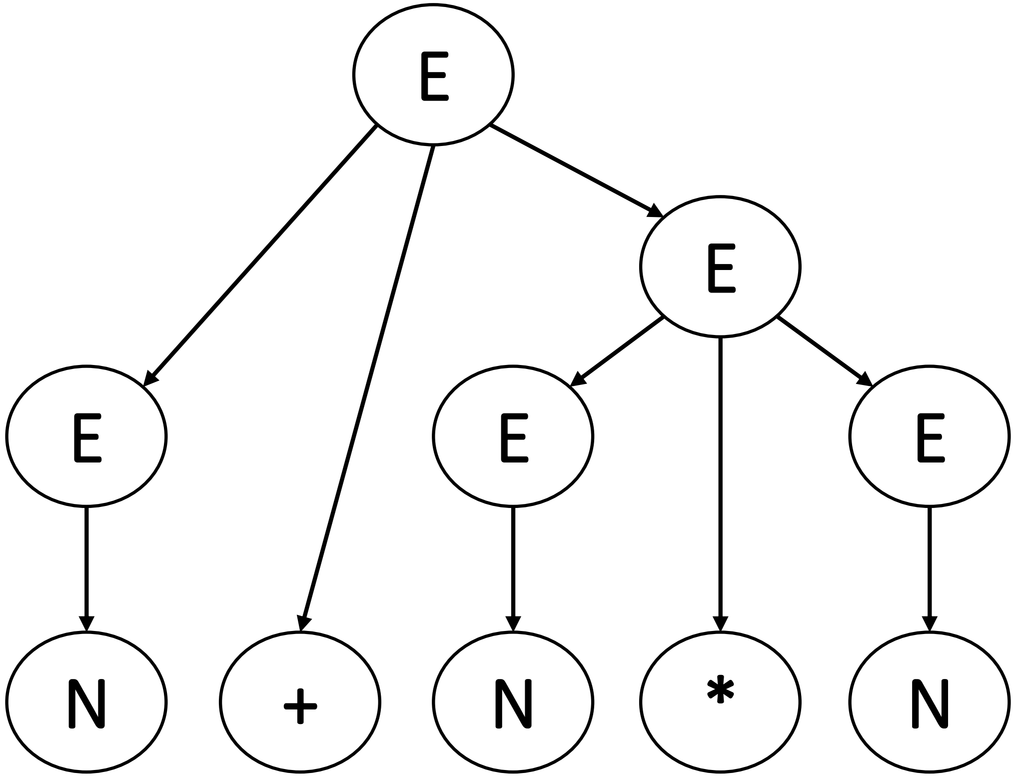




Input: $\mathbf{3} + \mathbf{4} * \mathbf{8}\$$

Stack: $s_0 E$





Input: $\mathbf{3 + 4 * 8}\$$

Stack: $s_0 E$