Parse $(x + y) * z$ using the LR(1) parsing algorithm.

Answer:

$\mid (x + y) * z$	shift
$(\mid x + y) * z$	shift
$(x \mid + y) * z$	shift
$(F \mid + y) * z$	reduce using $F \longrightarrow x$
$(T \mid + y) * z$	reduce using $T \longrightarrow F$
$(E \mid + y) * z$	reduce using $E \longrightarrow T$
$(E+ \mid y) * z$	shift
$(E + y \mid) * z$	shift
$(E + F \mid) * z$	reduce using $F \longrightarrow y$
$(E + T \mid) * z$	reduce using $T \longrightarrow F$
$(E \mid) * z$	reduce using $E \longrightarrow E + T$
$(E) \mid * z$	shift
$F \mid * z$	reduce using $F \longrightarrow (E)$
$T \mid * z$	reduce using $T \longrightarrow F$
$T* \mid z$	shift
$T * z \mid$	shift
$T * F \mid$	reduce using $F \longrightarrow z$
$T \mid$	reduce using $T \longrightarrow T * F$
$E \mid$	reduce using $E \longrightarrow T$

Discussion:

The LR(1) parsing algorithm employs two actions, shift and reduce. Shift moves the current position one position to the right. Reduce applies only to the symbol(s) immediately to the left of the current position, and replaces those symbols (the right-hand side of a production) with the non-terminal on the left side of that production. The algorithm:

- Start with the current position (denoted |) on the left.
- Then, for $wx \mid y$ —
  - If there is a production of the form $A \longrightarrow x$ and no production of the form $A \longrightarrow xz$ (i.e. x is not the beginning of the right side of any other production), reduce: $wA \mid y$.

- If there is a production of the form $A \longrightarrow x$ and one or more productions of the form $A \longrightarrow xz$ (i.e. x is the beginning of the right side of at least one other production), reduce using $A \longrightarrow x$ if no z has the same first symbol as y and shift otherwise.
- If there is no production of the form $A \longrightarrow x$, shift.

To parse $(x + y) * z$, start with the current position on the left:

$$| \ (x + y) * z$$

No production has only ϵ on the right side, so shift:

$$(\ | \ x + y) * z$$

No production has only (on the right side, so shift:

$$(x \ | \ + y) * z$$

x is the right side of $F \longrightarrow x$ and does not begin the right side of any other productions, so reduce:

$$(F \ | \ + y) * z$$

F is the right side of $T \longrightarrow F$ and does not begin the right side of any other productions, so reduce:

$$(T \ | \ + y) * z$$

T is the right side of $E \longrightarrow T$ and does also begin the right side of $T \longrightarrow T * F$. Looking ahead one place past the current position yields +, which does not appear next in $T * F$, so reduce using $E \longrightarrow T$.

$$(E \ | \ + y) * z$$

No production has only E or $(E$ on the right side, so shift:

$$(E + \ | \ y) * z$$

No production has only +, $E+$, or $(E+$ on the right side, so shift:

$$(E + y \ | \ ) * z$$

y is the right side of $F \longrightarrow y$ and does not begin the right side of any other productions, so reduce:

$$(E + F \ | \ ) * z$$

F is the right side of $T \longrightarrow F$ and does not begin the right side of any other productions, so reduce:

$$(E + T \mid) * z$$

$E + T$ is the right side of $E \longrightarrow E + T$ and does not begin the right side of any other productions, so reduce:

$$(E \mid) * z$$

(T is also the right side of a production, but if T was reduced instead it would be impossible to match the + preceding it with anything else — reduce the longest match possible on the left side of the current position.)
No production has only E or $(E$ on the right side, so shift:

$$(E) \mid * z$$

(E) is the right side of $F \longrightarrow (E)$ and does not begin the right side of any other productions, so reduce:

$$F \mid * z$$

F is the right side of $T \longrightarrow F$ and does not begin the right side of any other productions, so reduce:

$$T \mid * z$$

T is the right side of $E \longrightarrow T$ and does also begin the right side of $T \longrightarrow T * F$. Looking ahead one place past the current position yields $*$, which *does* appear next in $T * F$, so shift instead of reducing.

$$T * \mid z$$

No production has only $*$ or $T*$ on the right side, so shift:

$$T * z \mid$$

z is the right side of $F \longrightarrow z$ and does not begin the right side of any other productions, so reduce:

$$T * F \mid$$

$T * F$ is the right side of $T \longrightarrow T * F$ and does not begin the right side of any other productions, so reduce:

$$T \mid$$

T is the right side of $E \longrightarrow T$ and does also begin the right side of $T \longrightarrow T * F$. Looking ahead one place past the current position yields ϵ (the current position is at the end of the string), which does not appear next in $T * F$, so reduce using $E \longrightarrow T$.

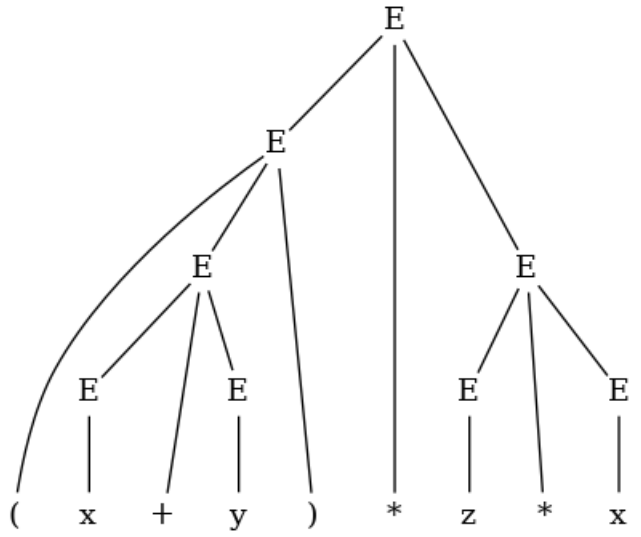
$$E \mid$$

And, having reached the start symbol and the end of the string being parsed, we are done.

Draw a parse tree for the string $(x + y) * z * x$ according to the grammar below.

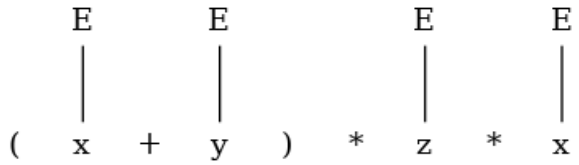
$$E \longrightarrow E + E$$
$$E \longrightarrow E * E$$
$$E \longrightarrow (E)$$
$$E \longrightarrow x$$
$$E \longrightarrow y$$
$$E \longrightarrow z$$

Answer:

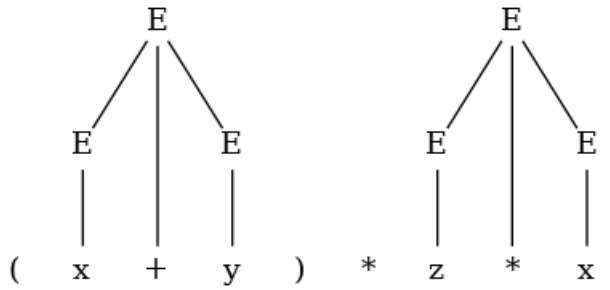


Discussion:

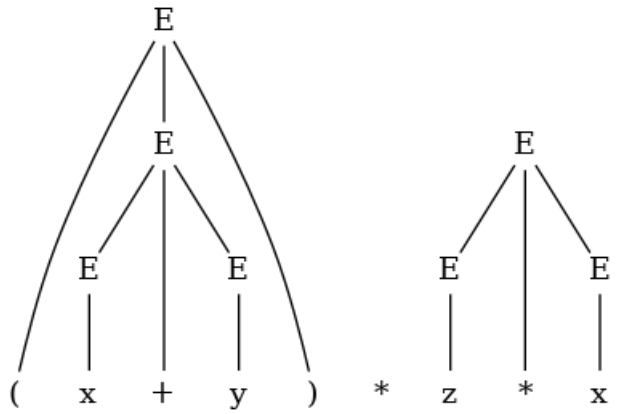
Start from the bottom. x could only have come from the production $E \longrightarrow x$. y and z are similar.



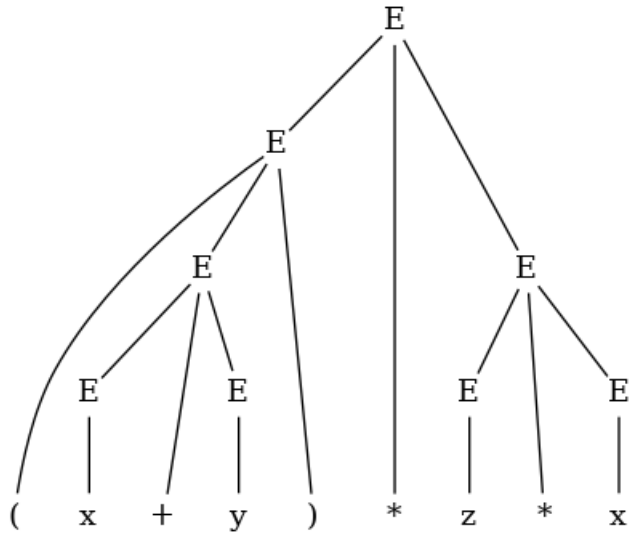
$E + E$ and $E * E$ could only have come from $E \longrightarrow E + E$ and $E \longrightarrow E * E$, respectively.



(E) comes from $E \longrightarrow (E)$.



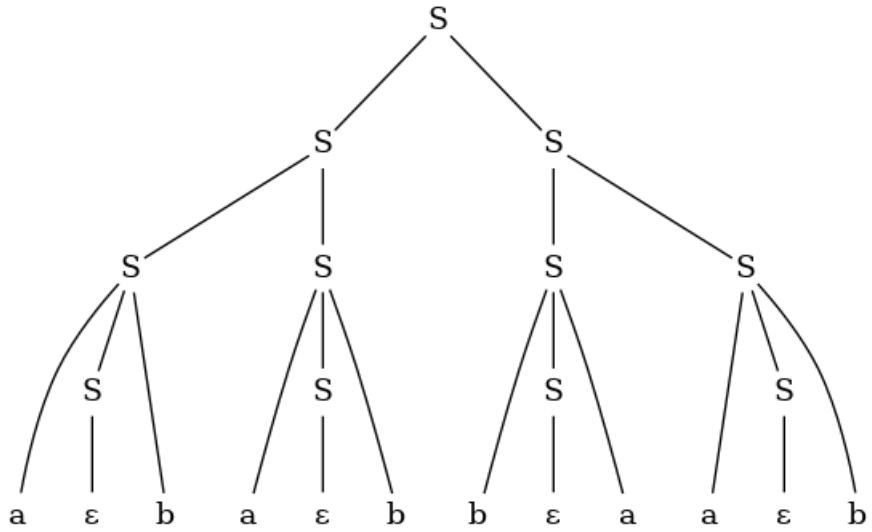
Finally, $E * E$ comes from $E \longrightarrow E * E$.

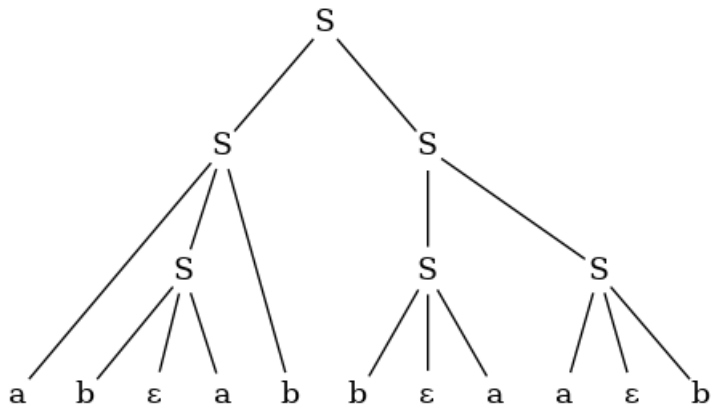


Draw two different parse trees for the string *ababbaab* based on the grammar below.

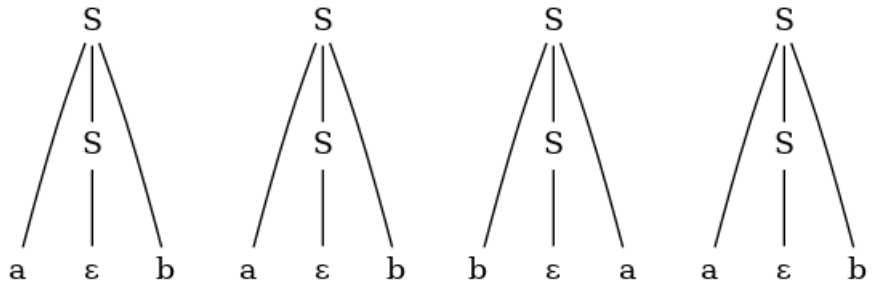
$$S \longrightarrow SS$$
$$S \longrightarrow aSb$$
$$S \longrightarrow bSa$$
$$S \longrightarrow \epsilon$$

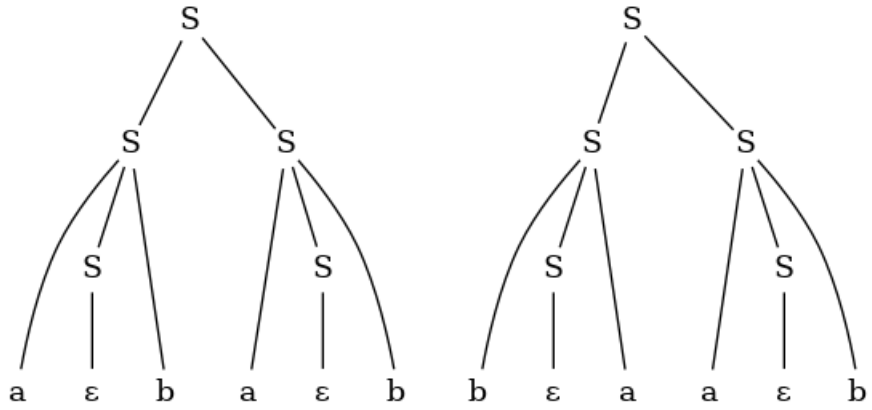
Answer:

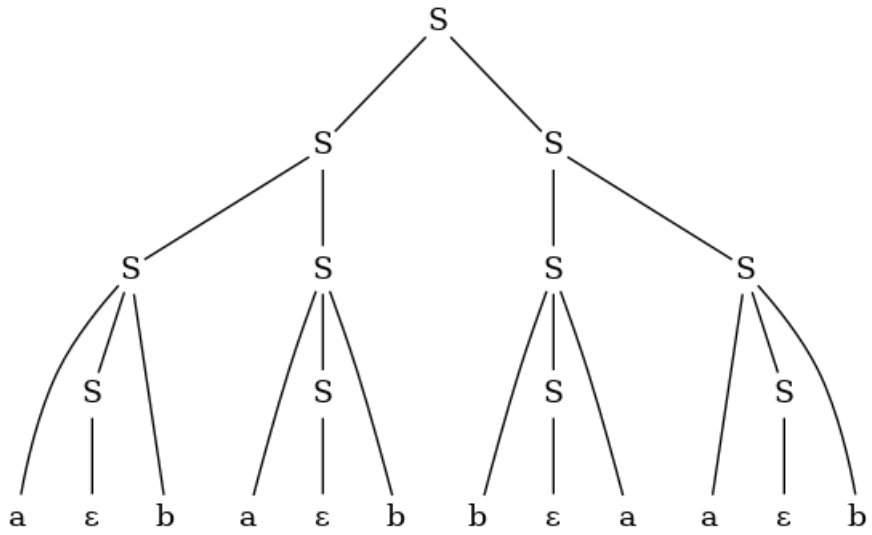




Discussion: A first step is to recognize that $S \Longrightarrow ab$ and $S \Longrightarrow ba$ because $S \longrightarrow \epsilon$.







For a second parse tree, at least one step has to be different. Observe that $S \Longrightarrow aSb \Longrightarrow abSab \Longrightarrow abab$.