

Some additional notes before
Midterm 1

Resolving Ambiguity

If a grammar can be made unambiguous at all, it is usually made unambiguous through **layering**.

Have exactly one way to build each piece of the string.

Have exactly one way of combining those pieces back together.

-

Example: Balanced Parentheses

- Consider the language of all strings of balanced parentheses.
- Examples:
 - ϵ
 - $()$
 - $((())())$
 - $((()()))((())())$
- Here is one possible grammar for balanced parentheses:

$$P \rightarrow \epsilon \mid PP \mid (P)$$

Balanced Parentheses

- Given the grammar $P \rightarrow \epsilon \mid PP \mid (P)$
- How might we generate the string $((())())$?

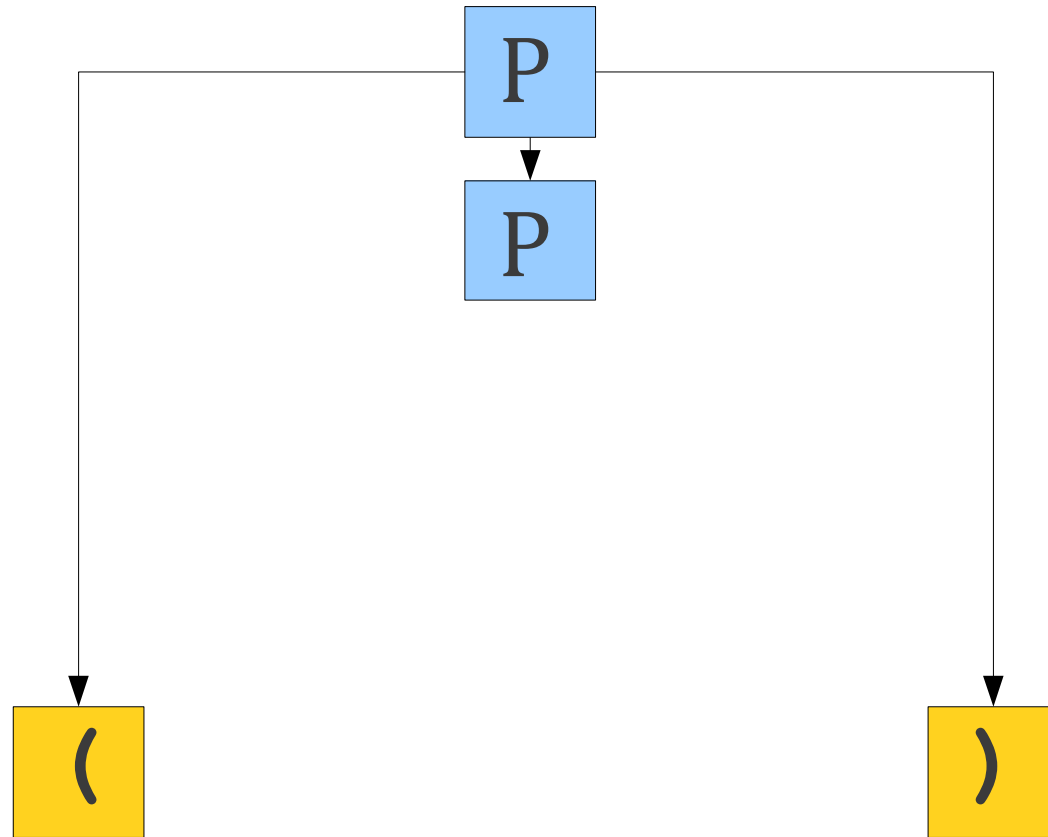
Balanced Parentheses

- Given the grammar $P \rightarrow \epsilon \mid PP \mid (P)$
- How might we generate the string $((())())$?

P

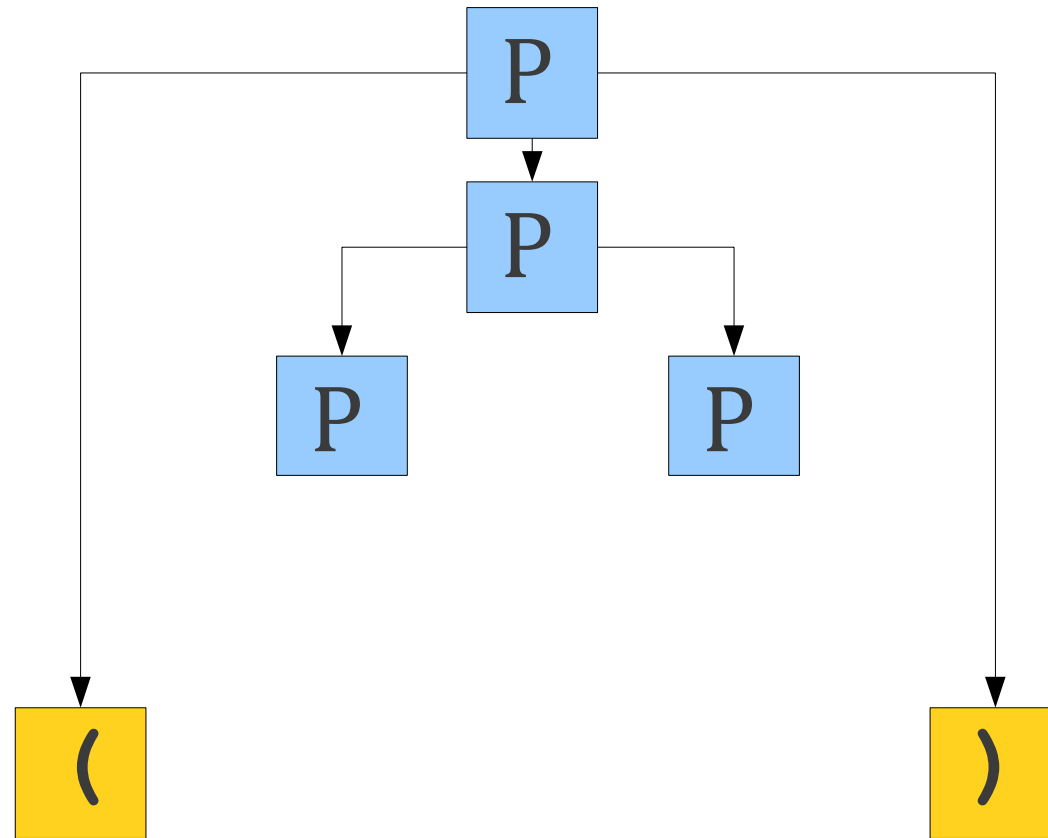
Balanced Parentheses

- Given the grammar $P \rightarrow \epsilon \mid PP \mid (P)$
- How might we generate the string $((())())$?



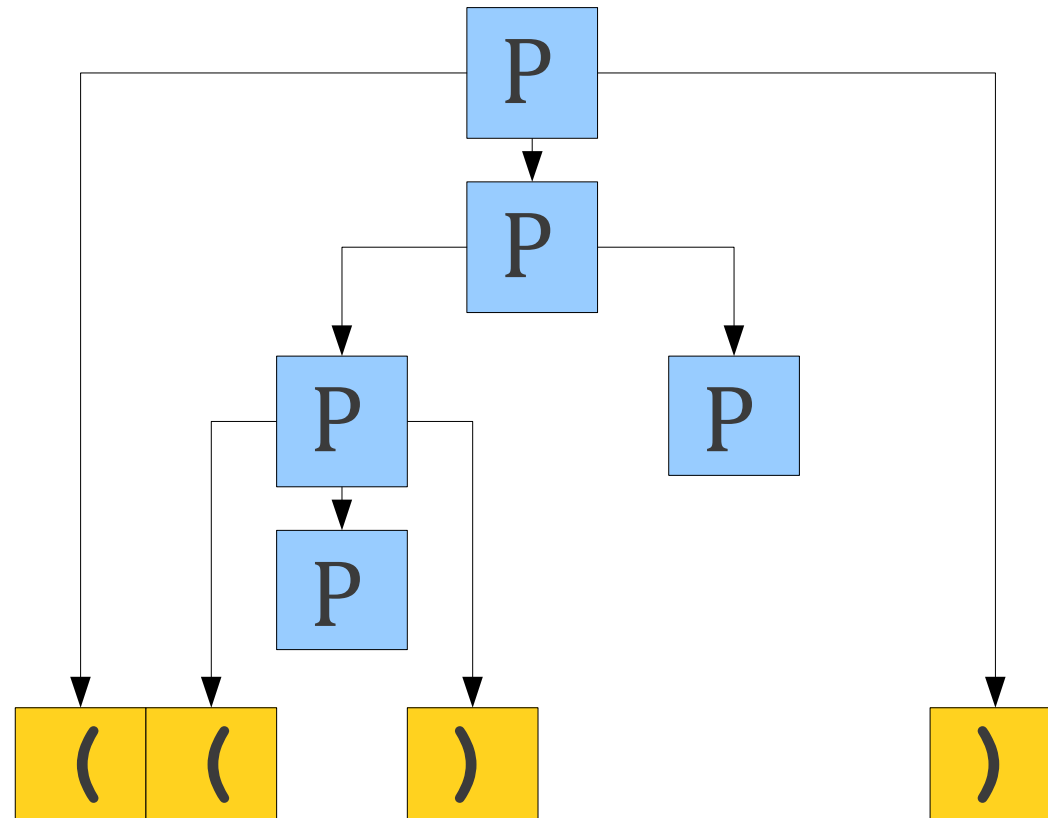
Balanced Parentheses

- Given the grammar $P \rightarrow \epsilon \mid PP \mid (P)$
- How might we generate the string $((())())$?



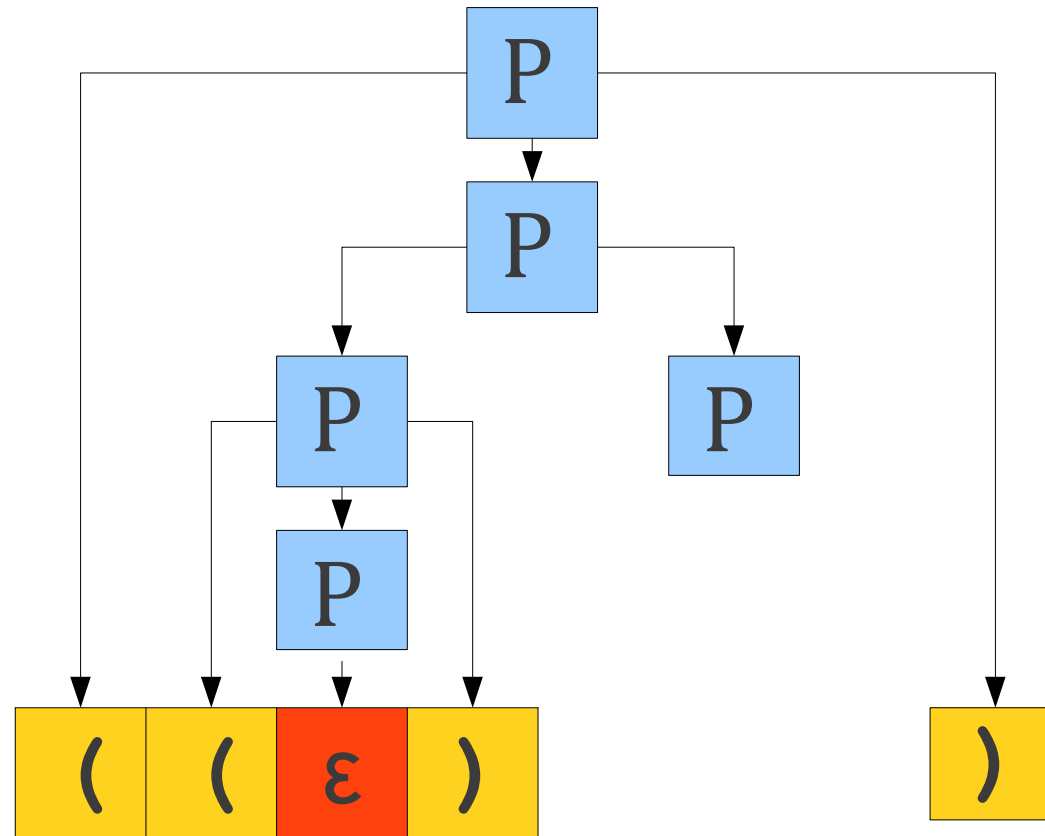
Balanced Parentheses

- Given the grammar $P \rightarrow \epsilon \mid PP \mid (P)$
- How might we generate the string $((()))$?



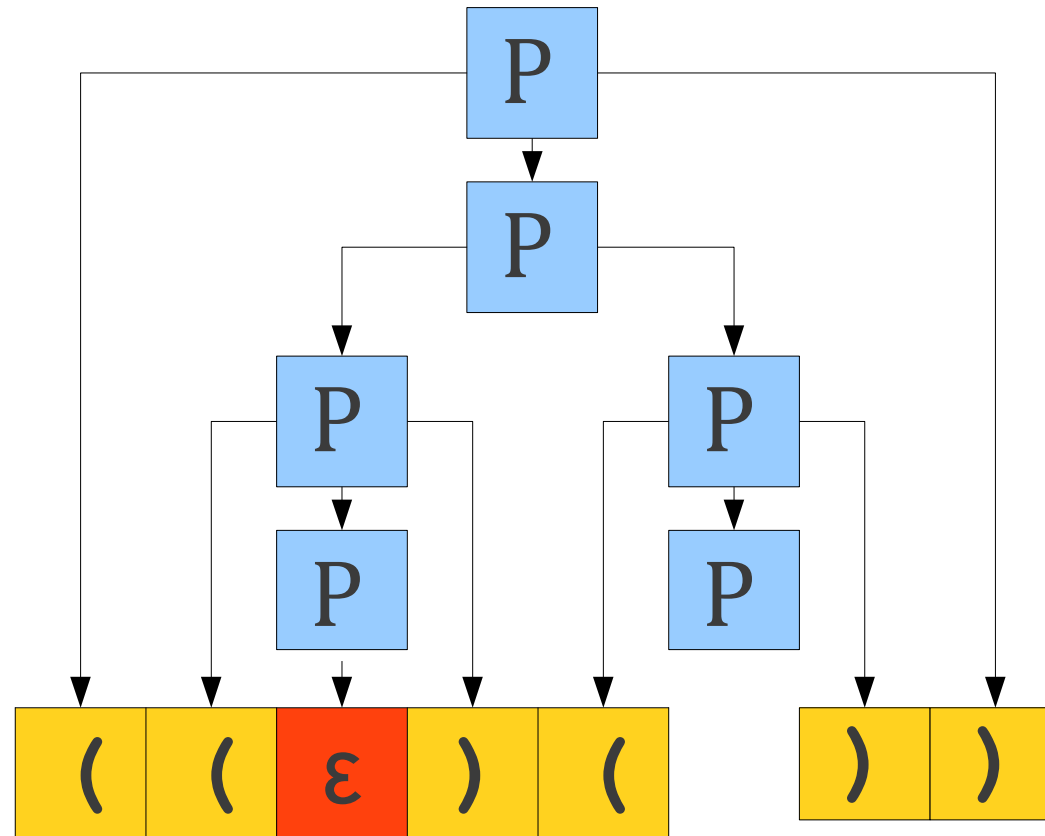
Balanced Parentheses

- Given the grammar $P \rightarrow \epsilon \mid PP \mid (P)$
- How might we generate the string $((())())$?



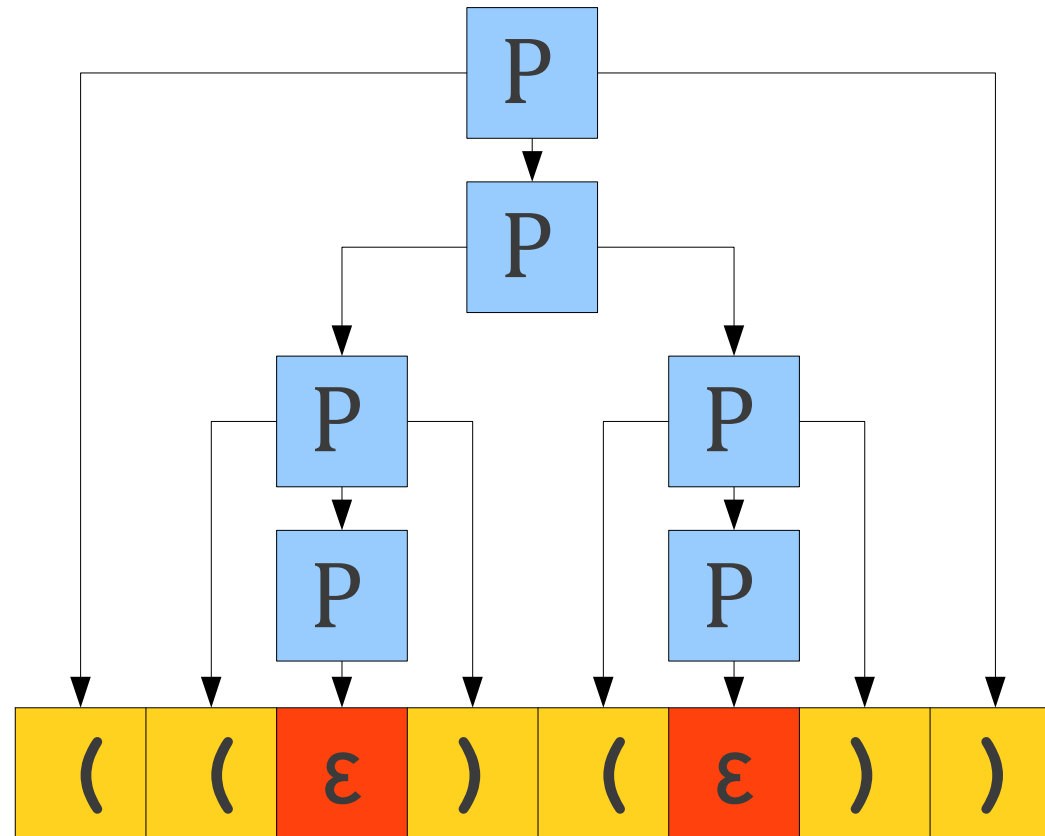
Balanced Parentheses

- Given the grammar $P \rightarrow \epsilon \mid PP \mid (P)$
- How might we generate the string $((\epsilon))$?

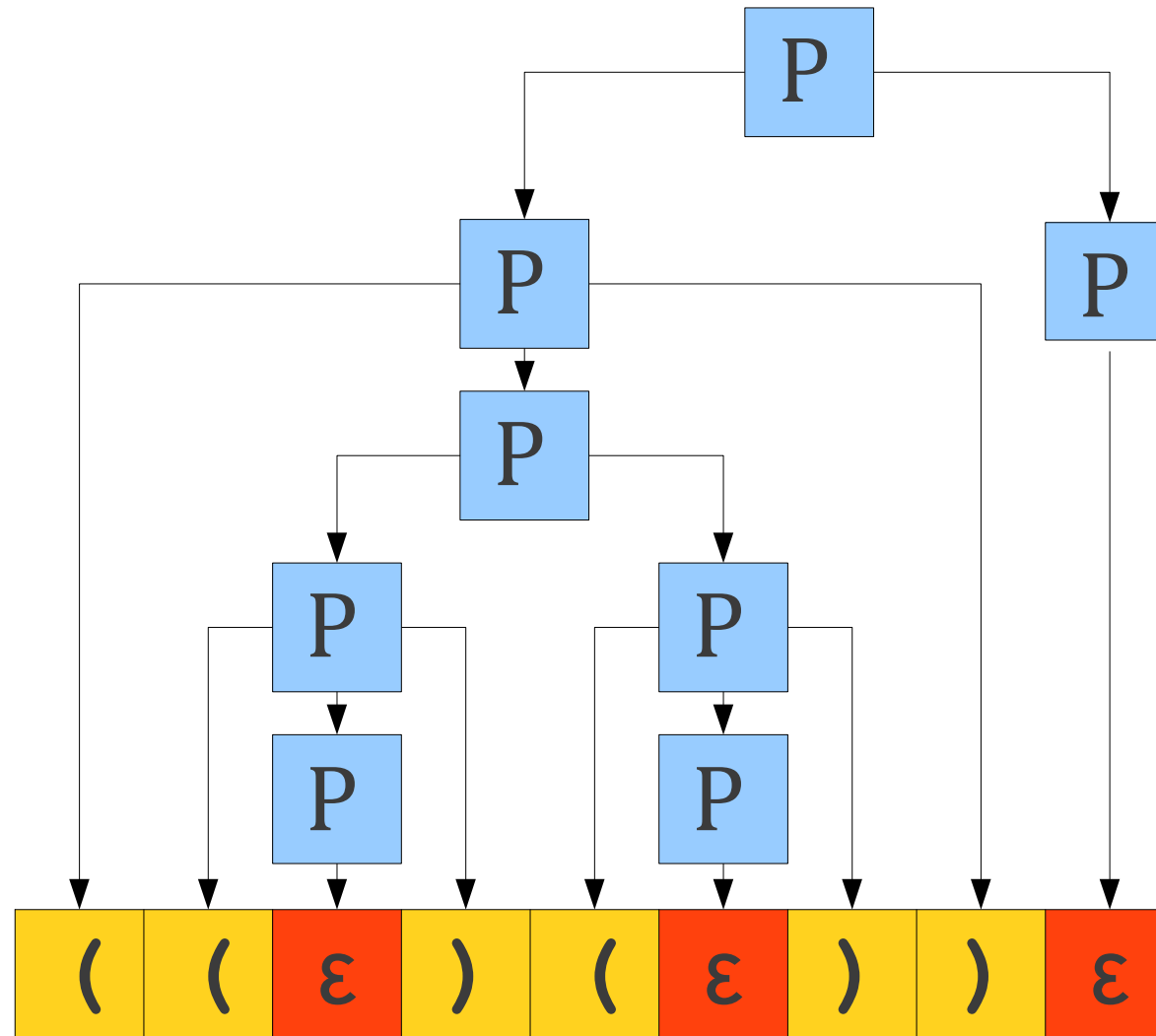


Balanced Parentheses

- Given the grammar $P \rightarrow \epsilon \mid PP \mid (P)$
- How might we generate the string $((\epsilon)(\epsilon))$?

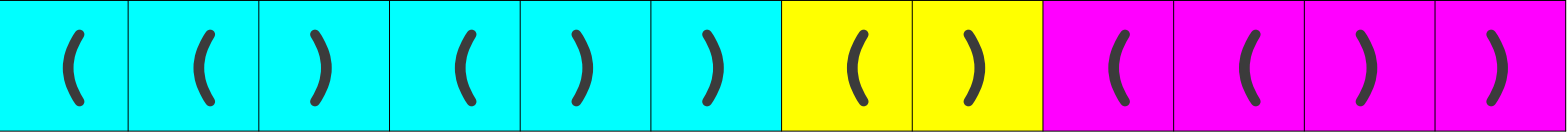


Balanced Parentheses



How to resolve this ambiguity?

(()	())	()	(())
---	---	---	---	---	---	---	---	---	---	---	---







Rethinking Parentheses

- A string of balanced parentheses is a sequence of strings that are themselves balanced parentheses.

To avoid ambiguity, we can build the string in

- two steps:
 - Decide how many different substrings we will glue together.
 - Build each substring independently.

Building Parentheses

- Spread a string of parentheses across the string.
There is exactly one way to do this for any number of parentheses.
- Expand out each substring by adding in parentheses and repeating.

$S \rightarrow P \mid S$

$P \rightarrow (S)$

Building Parentheses

S \rightarrow **P** **S** | ϵ

P \rightarrow (**S**)

S

\Rightarrow **PS**

\Rightarrow **PPS**

\Rightarrow **PP**

\Rightarrow (**S**) **P**

\Rightarrow (**S**) (**S**)

\Rightarrow (**PS**) (**S**)

\Rightarrow (**P**) (**S**)

\Rightarrow ((**S**)) (**S**)

\Rightarrow (()) (**S**)

\Rightarrow (()) ()

Context-Free Grammars

- A regular expression can be
 - Any letter
 - ϵ
 - The concatenation of regular expressions.
 - The union of regular expressions.
 - The Kleene closure of a regular expression.
 - A parenthesized regular expression.

Context-Free Grammars

- This gives us the following CFG:

$R \rightarrow a \mid b \mid c \mid \dots$

$R \rightarrow \epsilon$

$R \rightarrow RR$

$R \rightarrow R \mid R$

$R \rightarrow R^*$

$R \rightarrow (R)$

An Ambiguous Grammar

$R \rightarrow a \mid b \mid c \mid \dots$

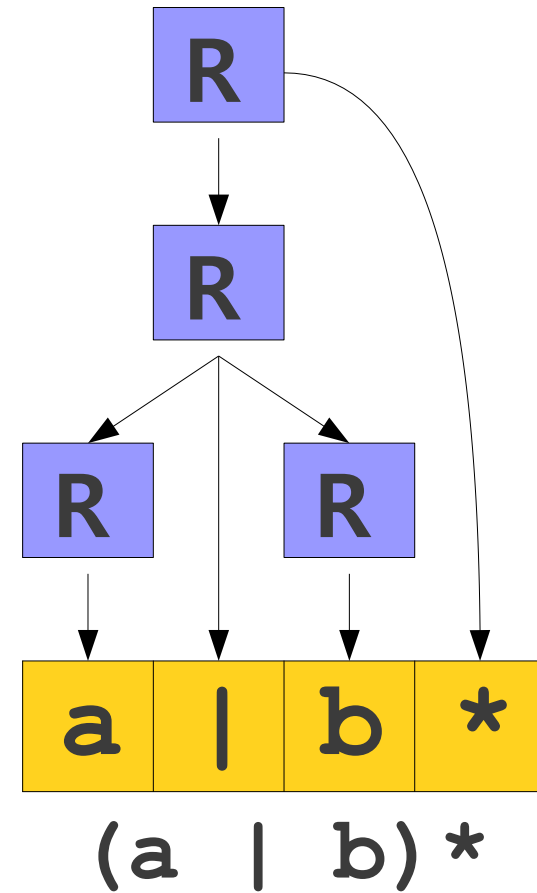
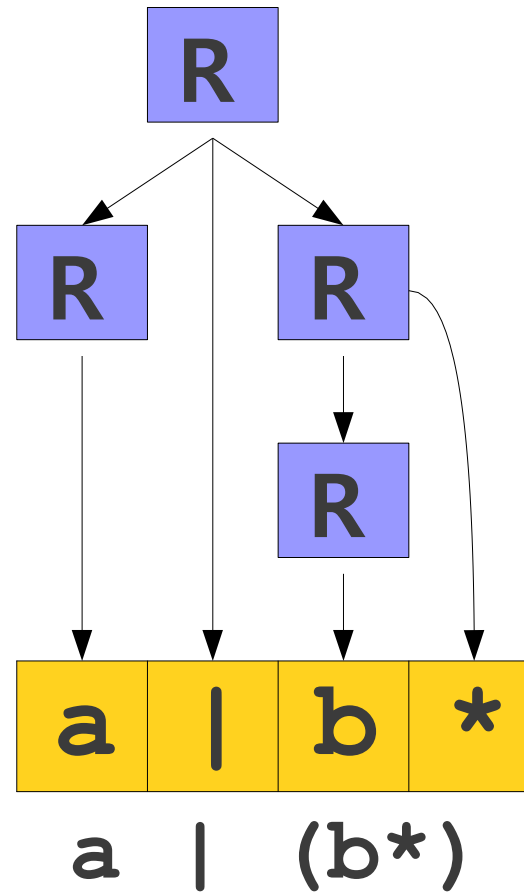
$R \rightarrow \epsilon$

$R \rightarrow RR$

$R \rightarrow R \mid R$

$R \rightarrow R^*$

$R \rightarrow (R)$



Resolving Ambiguity

- We can try to resolve the ambiguity via layering:

$R \rightarrow a \mid b \mid c \mid \dots$

$R \rightarrow \epsilon$

$R \rightarrow RR$

$R \rightarrow R \mid R$

$R \rightarrow R^*$

$R \rightarrow (R)$

a	a		b	*
---	---	--	---	---

Resolving Ambiguity

- We can try to resolve the ambiguity via layering:

$R \rightarrow a \mid b \mid c \mid \dots$

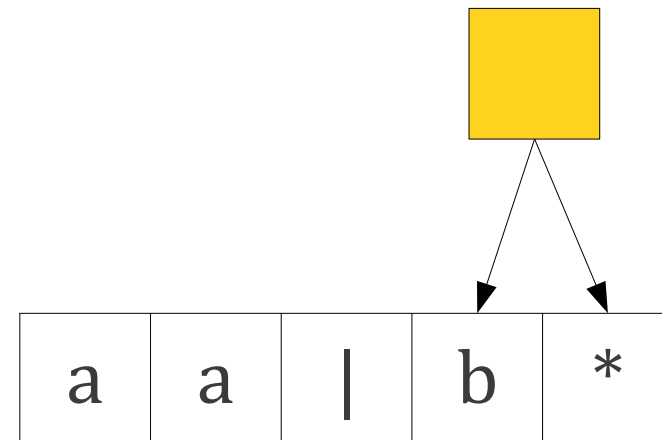
$R \rightarrow \epsilon$

$R \rightarrow RR$

$R \rightarrow R \mid R$

$R \rightarrow R^*$

$R \rightarrow (R)$



Resolving Ambiguity

- We can try to resolve the ambiguity via layering:

$R \rightarrow a \mid b \mid c \mid \dots$

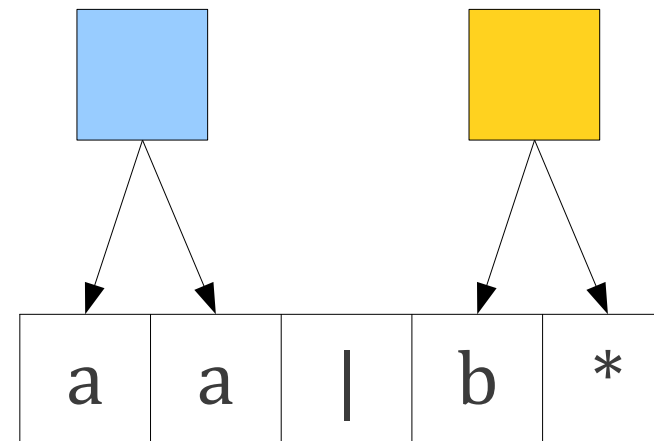
$R \rightarrow \epsilon$

$R \rightarrow RR$

$R \rightarrow R \mid R$

$R \rightarrow R^*$

$R \rightarrow (R)$



Resolving Ambiguity

- We can try to resolve the ambiguity via layering:

$R \rightarrow a \mid b \mid c \mid \dots$

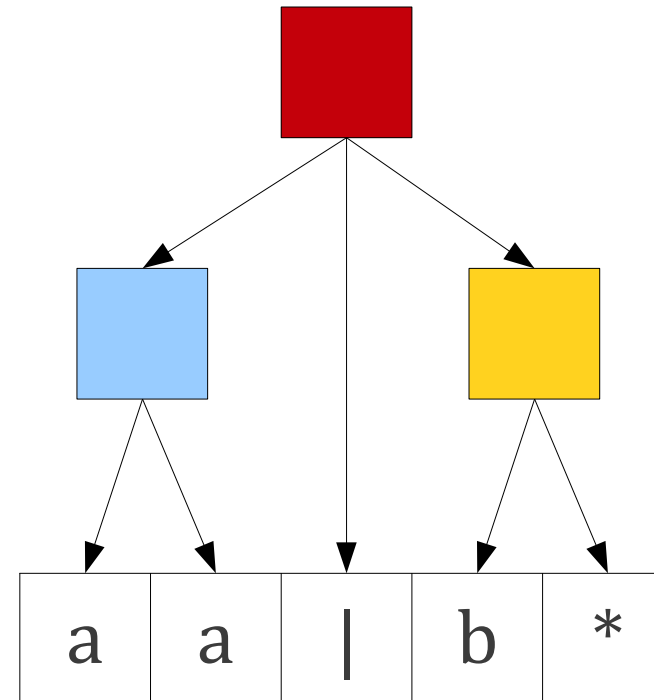
$R \rightarrow \epsilon$

$R \rightarrow RR$

$R \rightarrow R \mid R$

$R \rightarrow R^*$

$R \rightarrow (R)$



Resolving Ambiguity

- We can try to resolve the ambiguity via layering:

$R \rightarrow a \mid b \mid c \mid \dots$

$R \rightarrow \epsilon$

$R \rightarrow RR$

$R \rightarrow R \mid R$

$R \rightarrow R^*$

$R \rightarrow (R)$

$R \rightarrow S \mid R \mid S$

$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \epsilon$

$U \rightarrow (R)$

Why is this unambiguous?

$R \rightarrow S \mid R \text{ “} | \text{” } S$

$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid \dots$

$U \rightarrow \text{“} \epsilon \text{”}$

$U \rightarrow (R)$

Why is this unambiguous?

$R \rightarrow S \mid R \text{ “|” } S$

$S \rightarrow T \mid ST$

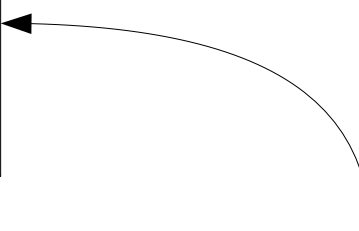
$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid \dots$

$U \rightarrow \text{“}\epsilon\text{”}$

$U \rightarrow (R)$

Only generates
“atomic” expressions



Why is this unambiguous?

$R \rightarrow S \mid R \text{ “|” } S$

$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid \dots$

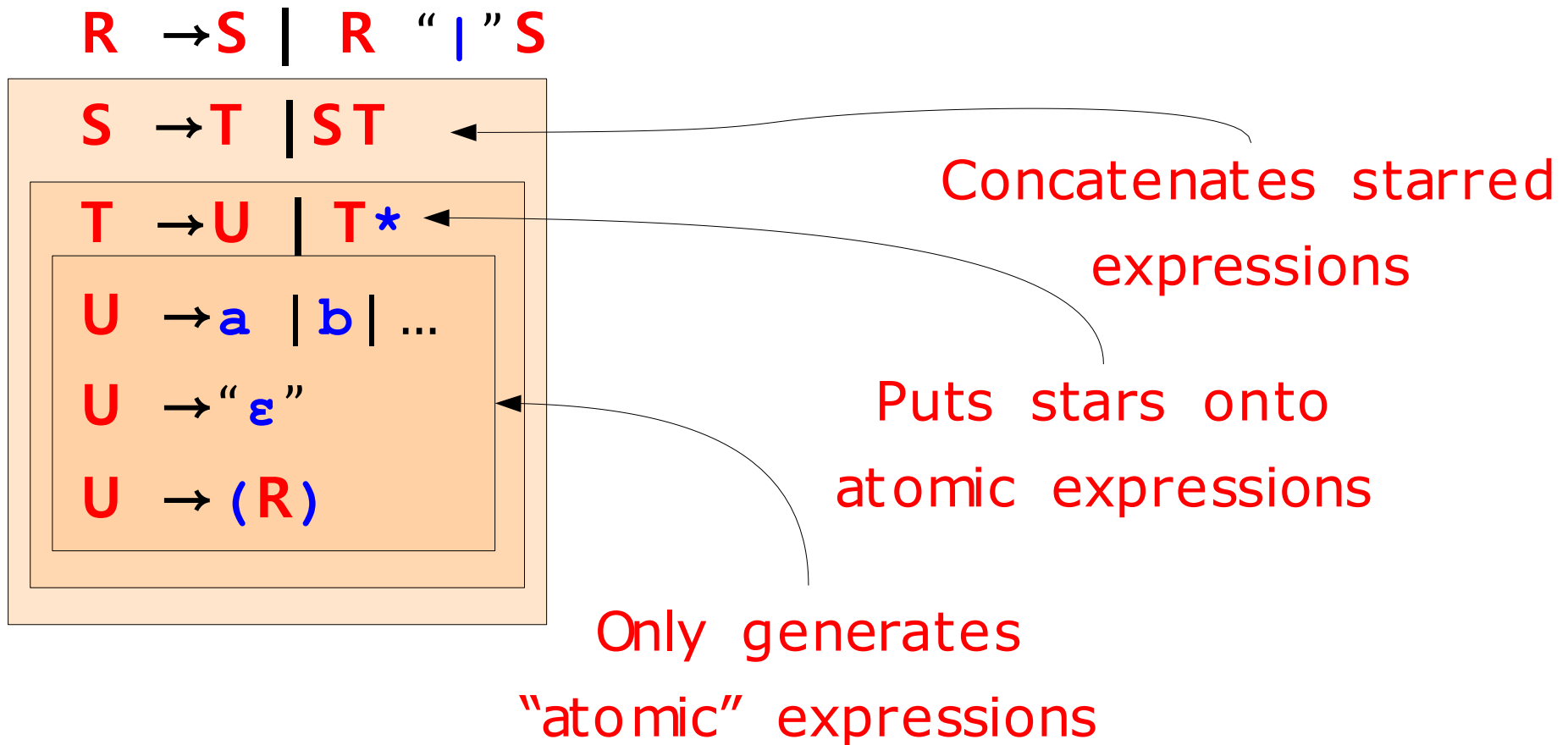
$U \rightarrow \text{“}\epsilon\text{”}$

$U \rightarrow (R)$

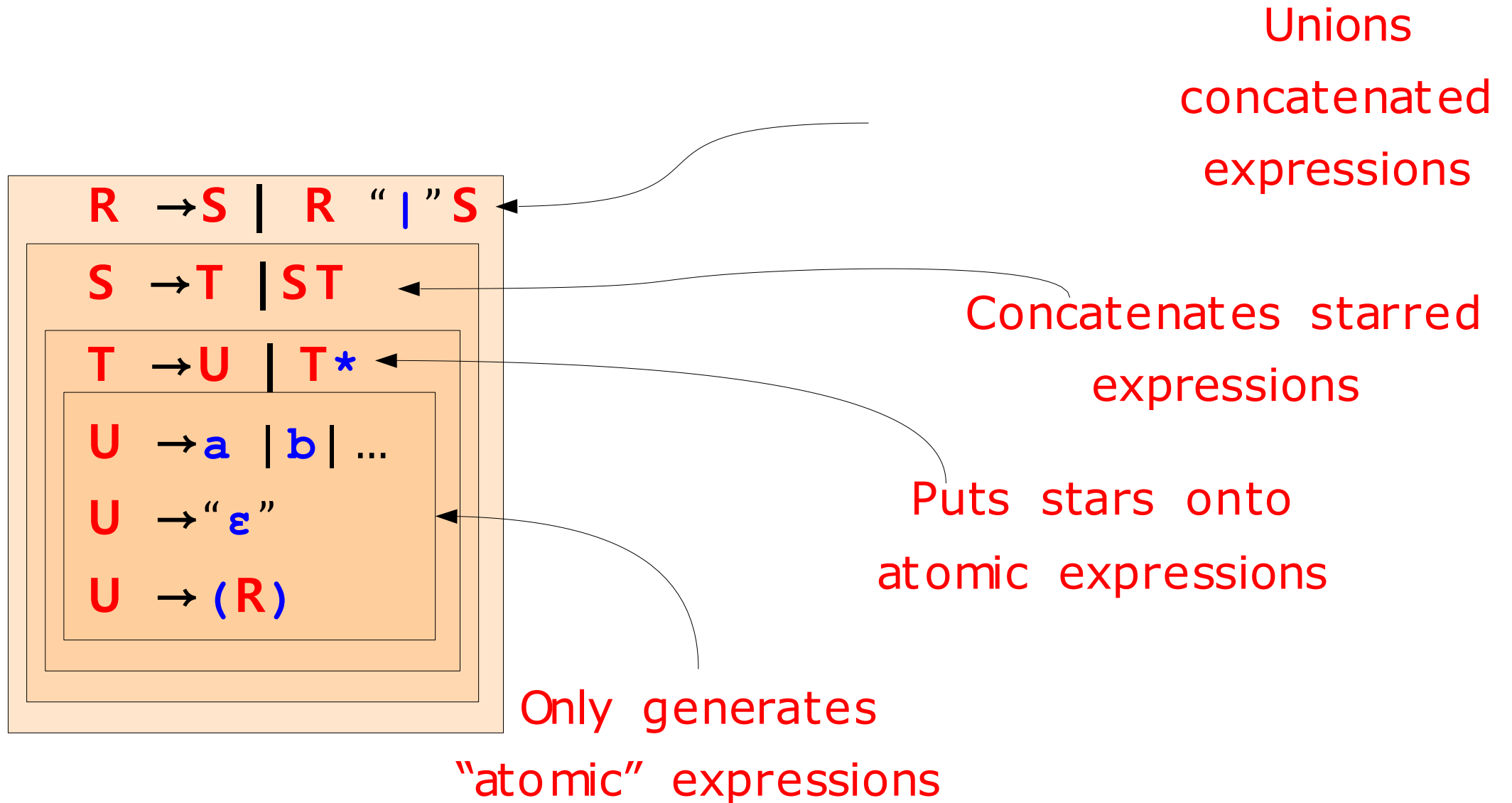
Puts stars onto
atomic expressions

Only generates
“atomic” expressions

Why is this unambiguous?



Why is this unambiguous?



R

$R \rightarrow S \mid R \text{ " | " } S$

$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \epsilon$

$U \rightarrow (R)$

a	b		c		a	*
---	---	--	---	--	---	---

$R \rightarrow S \mid R \text{ " | " } S$

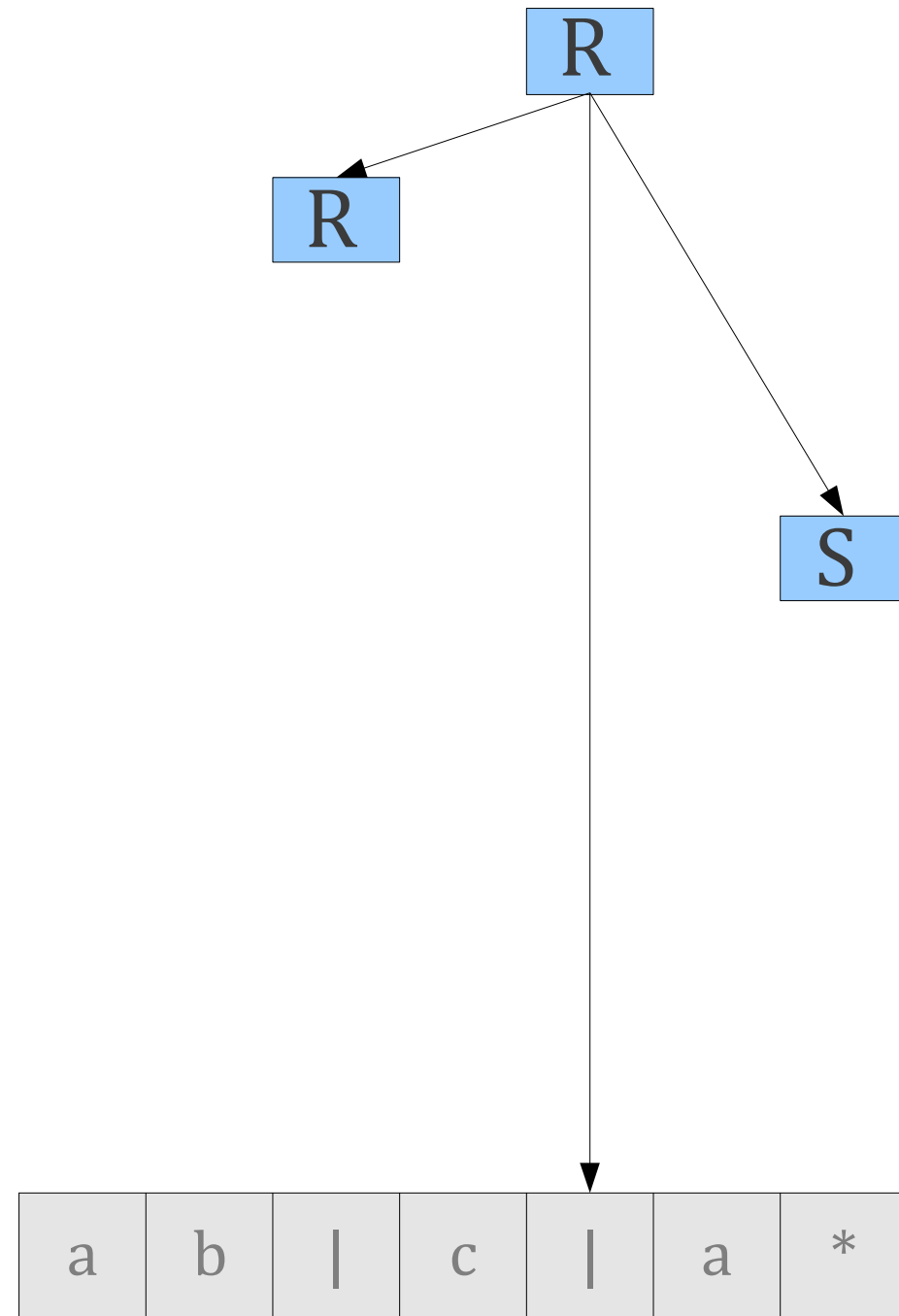
$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \epsilon$

$U \rightarrow (R)$



$R \rightarrow S \mid R \text{ " | " } S$

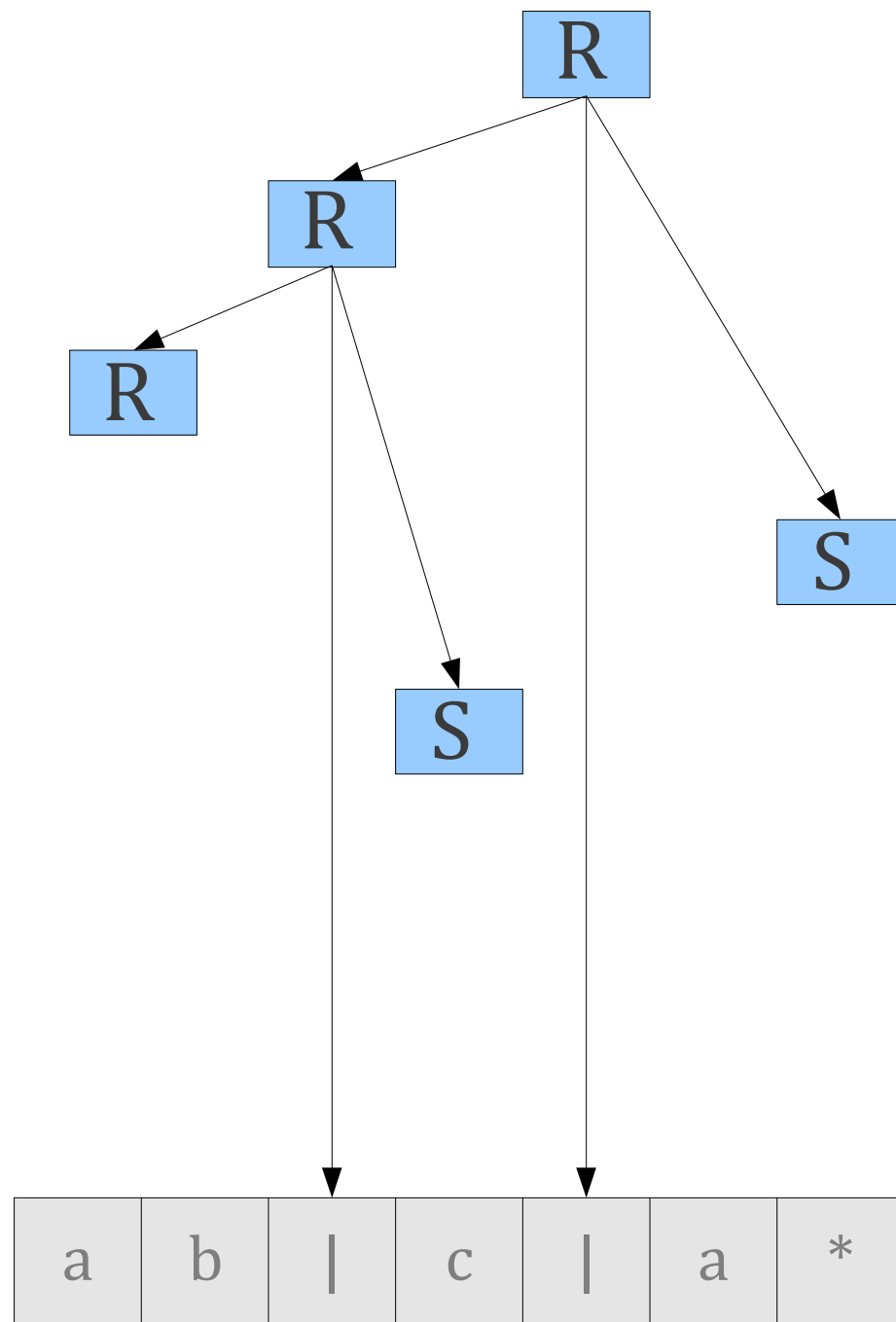
$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \epsilon$

$U \rightarrow (R)$



$R \rightarrow S \mid R \text{ " | " } S$

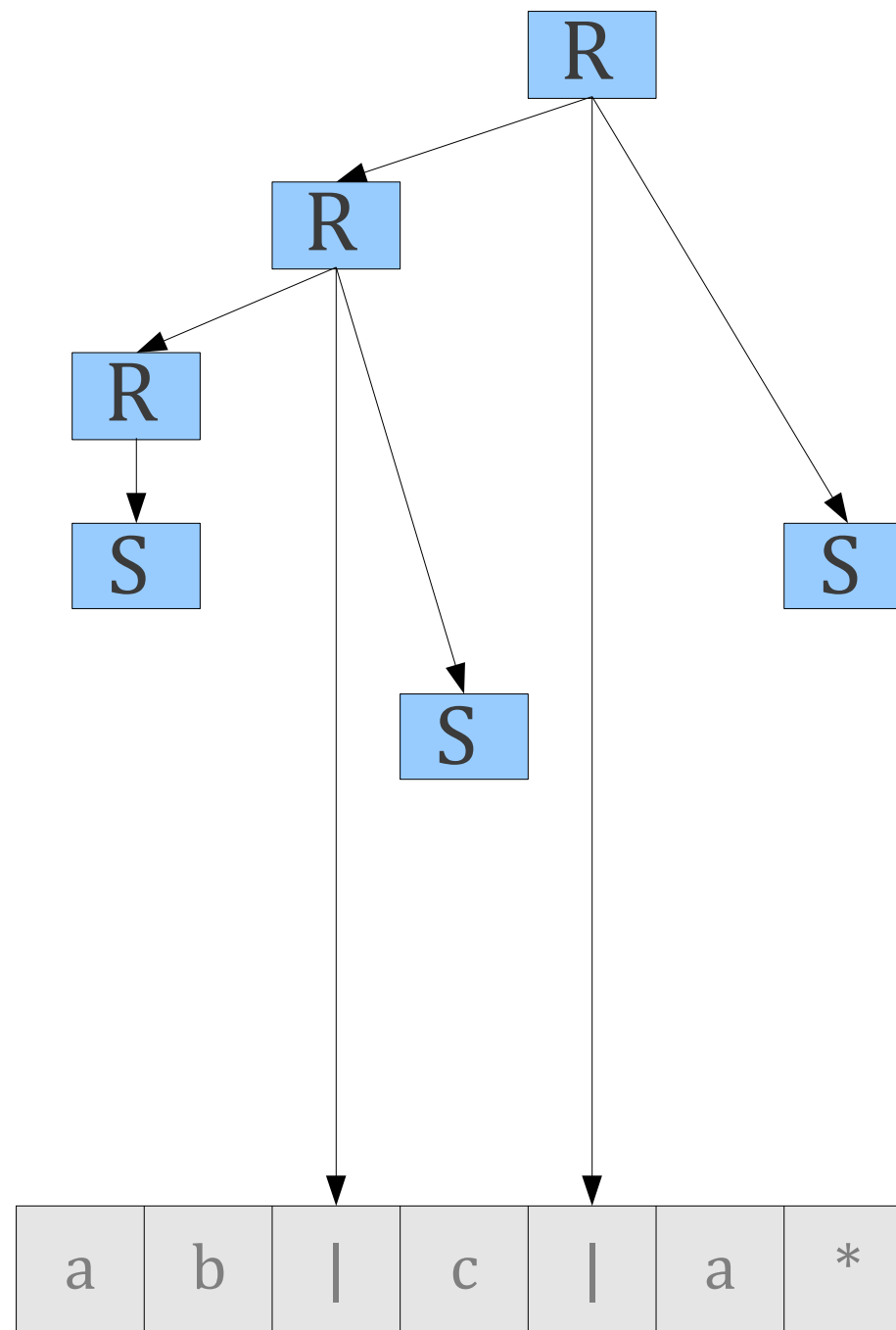
$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \epsilon$

$U \rightarrow (R)$



R \rightarrow **S** | **R** “ | ” **S**

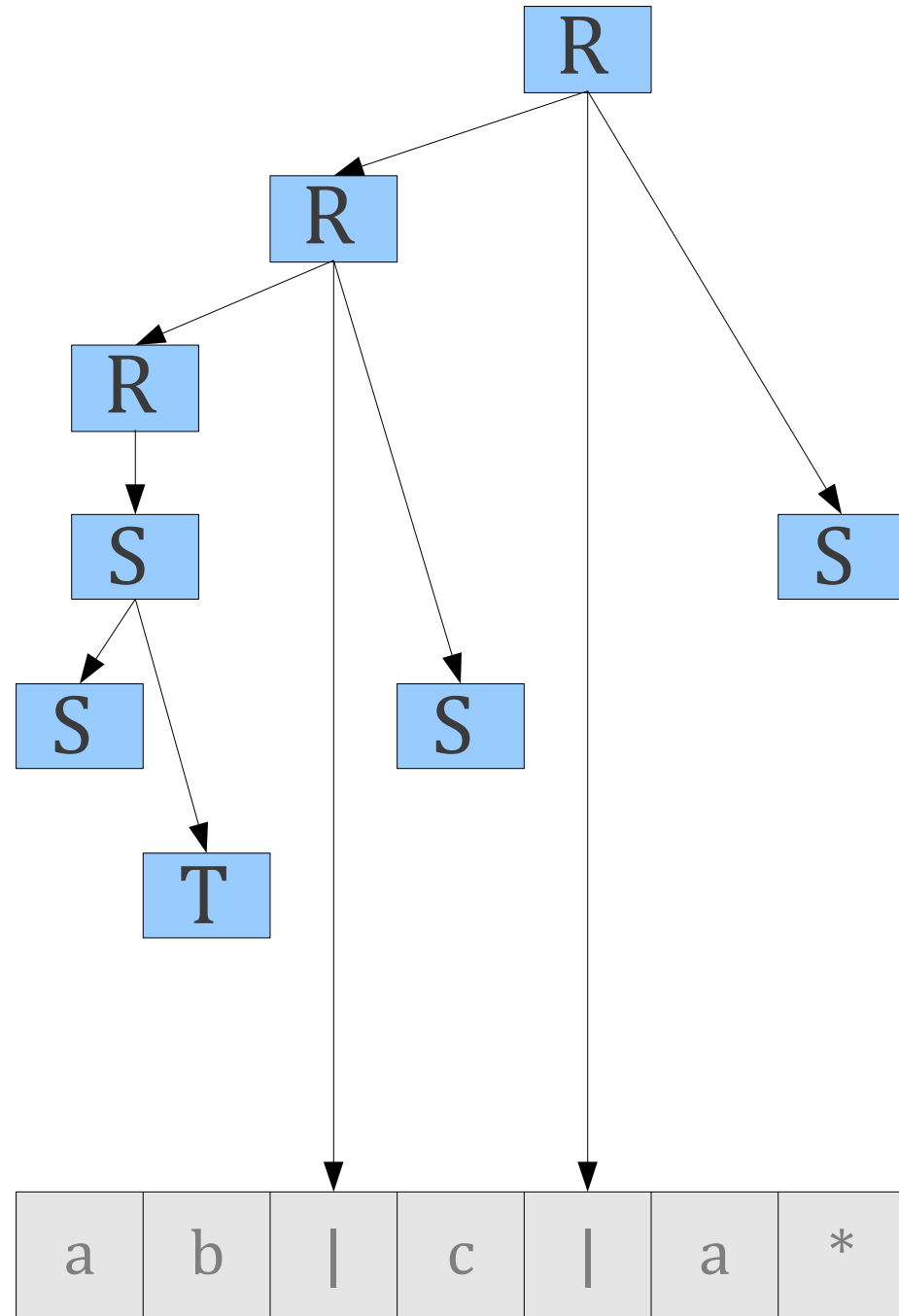
S \rightarrow **T** | **ST**

T → **U** | **T***

U \rightarrow **a** | **b** | **c** | ...

U → "ε"

U \rightarrow (**R**)



R \rightarrow **S** | **R** “ | ” **S**

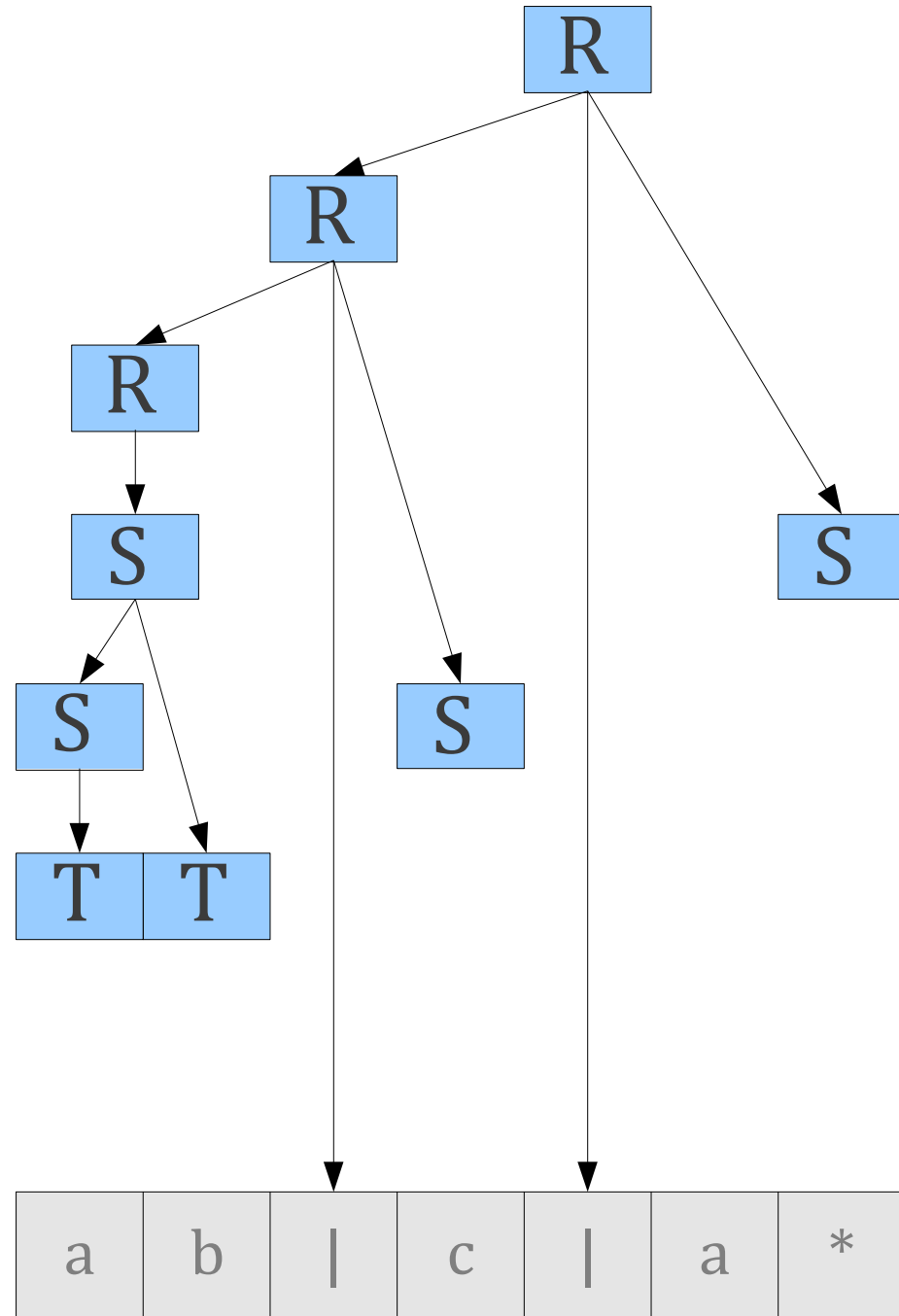
S \rightarrow **T** | **ST**

T → **U** | **T***

U → a | b | c | ...

U → "ε"

U \rightarrow (**R**)



R \rightarrow **S** | **R** “ | ” **S**

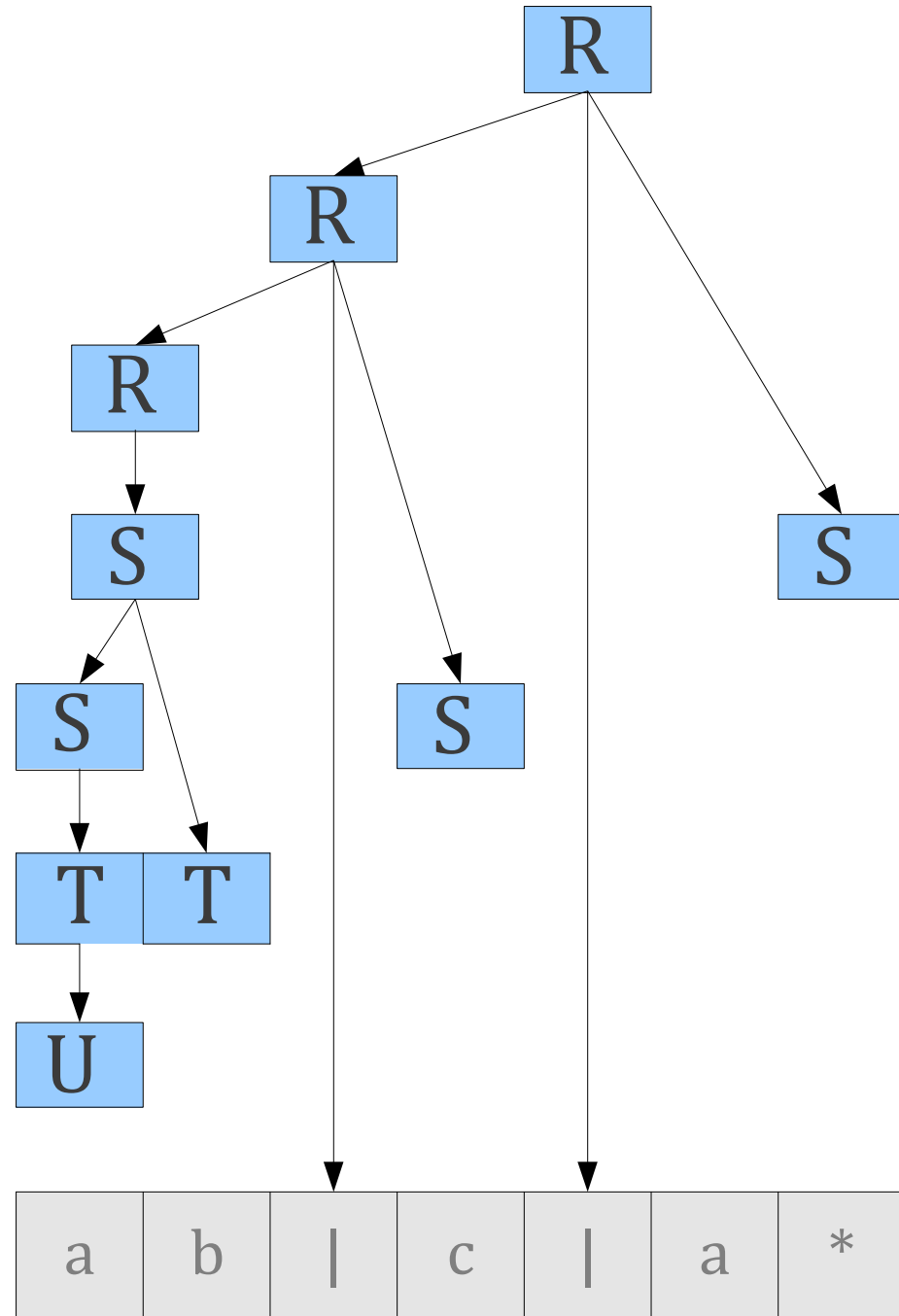
S \rightarrow **T** | **ST**

T → **U** | **T***

U \rightarrow **a** | **b** | **c** | ...

U → "ε"

U \rightarrow **(R)**



R \rightarrow **S** | **R** “ | ” **S**

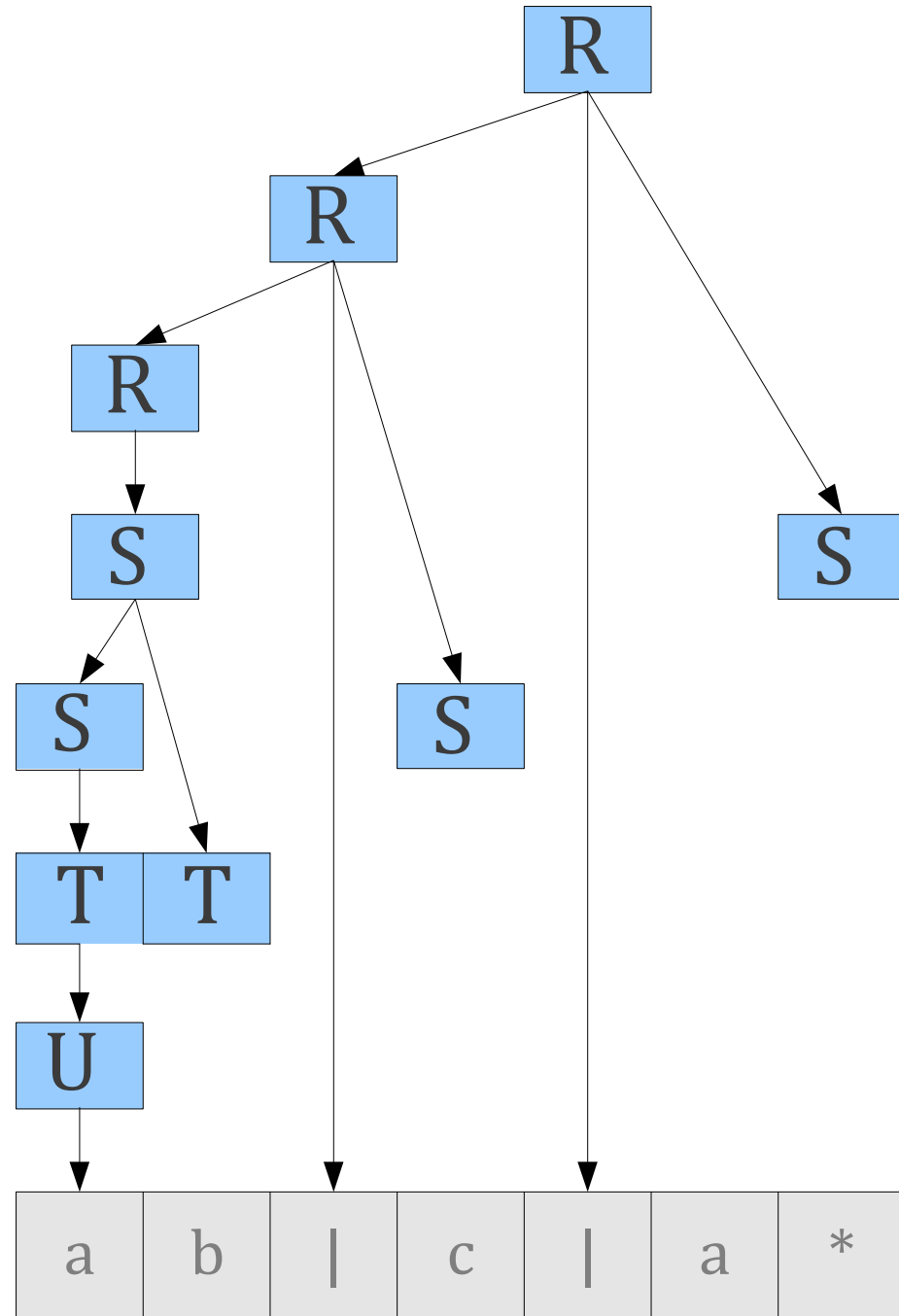
S \rightarrow **T** | **ST**

T → **U** | **T***

U \rightarrow **a** | **b** | **c** | ...

U → "ε"

U \rightarrow (**R**)



$R \rightarrow S \mid R \mid S$

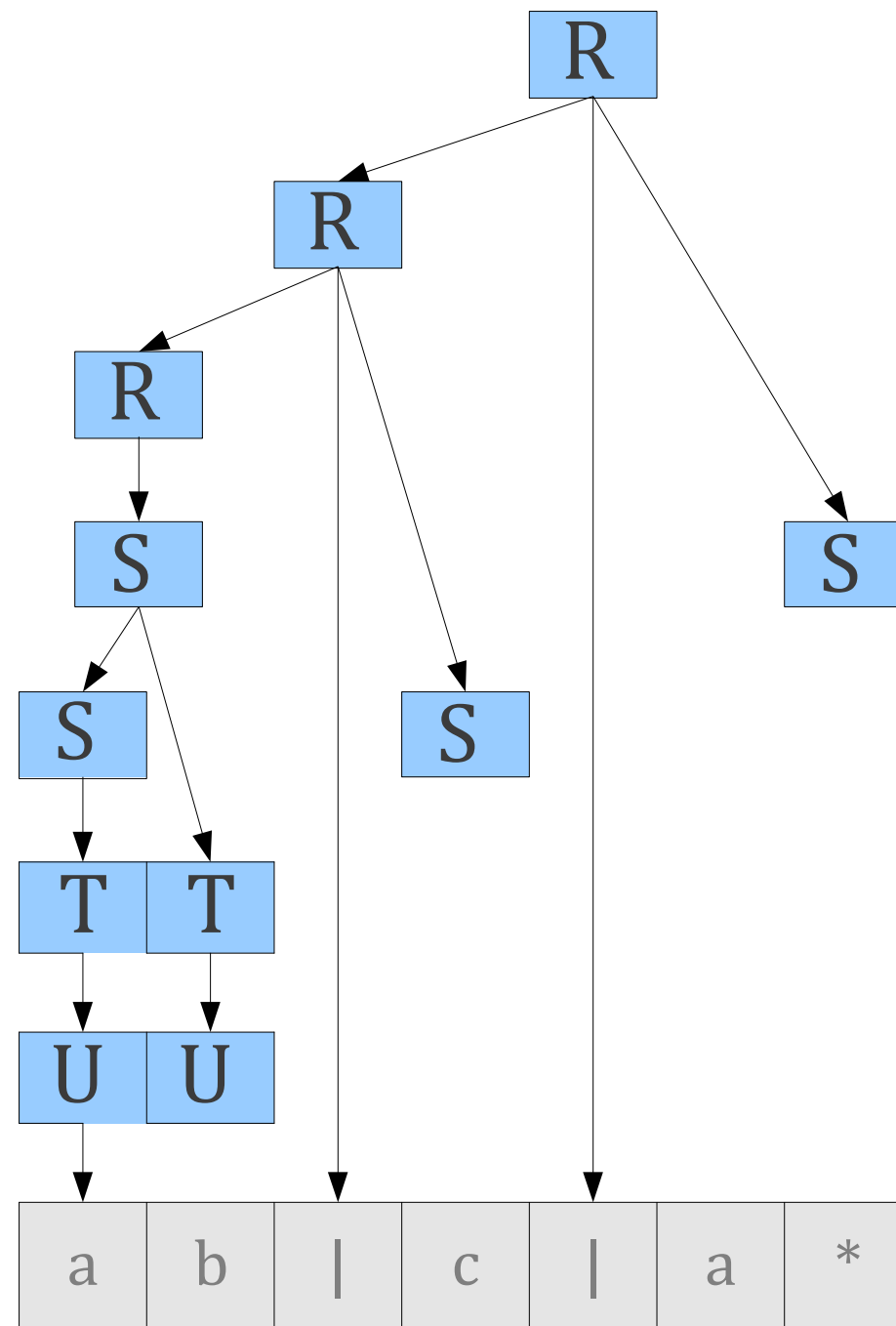
$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \epsilon$

$U \rightarrow (R)$



R \rightarrow **S** | **R** “ | ” **S**

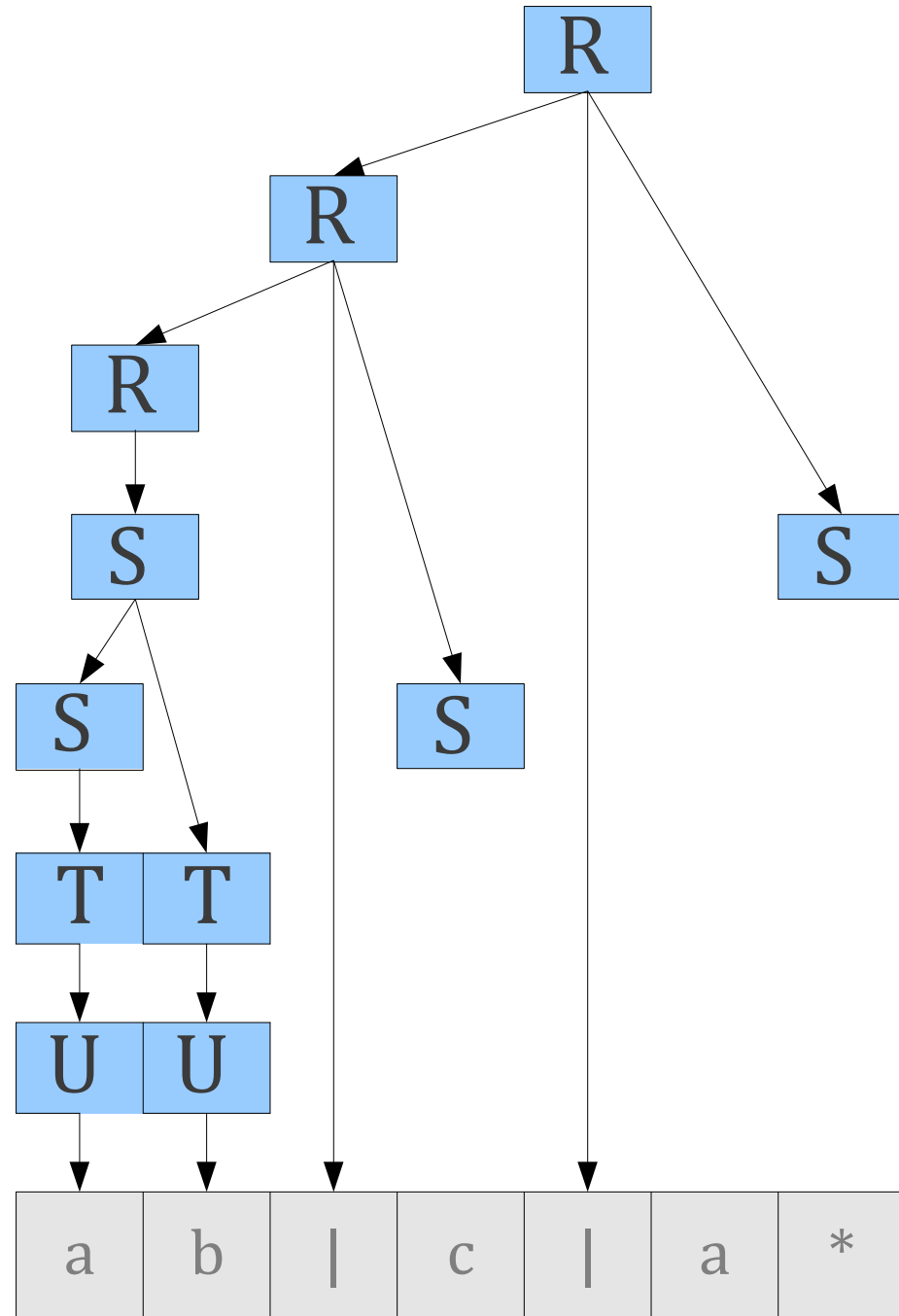
S \rightarrow **T** | **ST**

T → **U** | **T***

U \rightarrow **a** | **b** | **c** | ...

U → "ε"

U \rightarrow (**R**)



$R \rightarrow S \mid R \mid S$

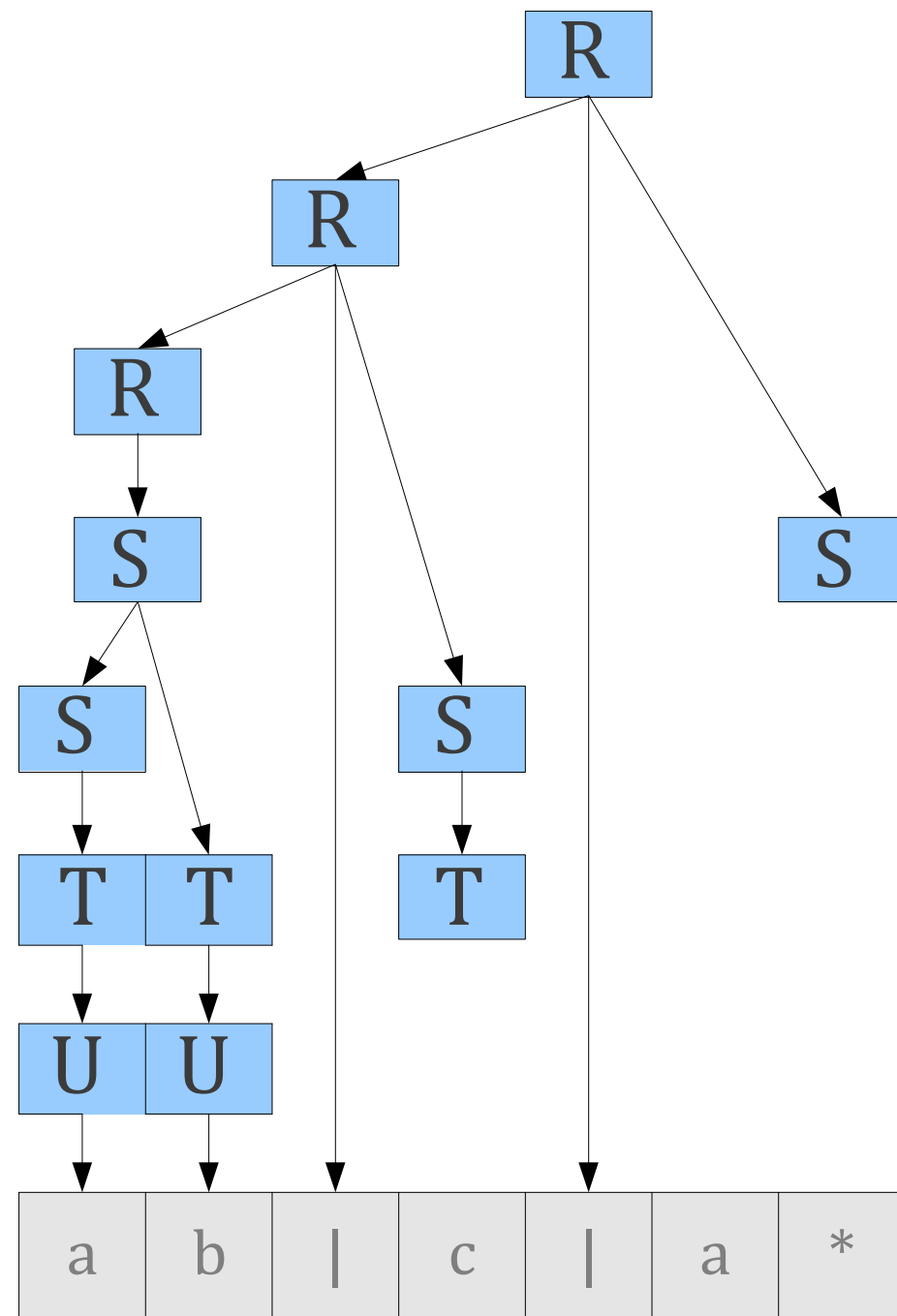
$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \epsilon$

$U \rightarrow (R)$



R \rightarrow **S** | **R** “ | ” **S**

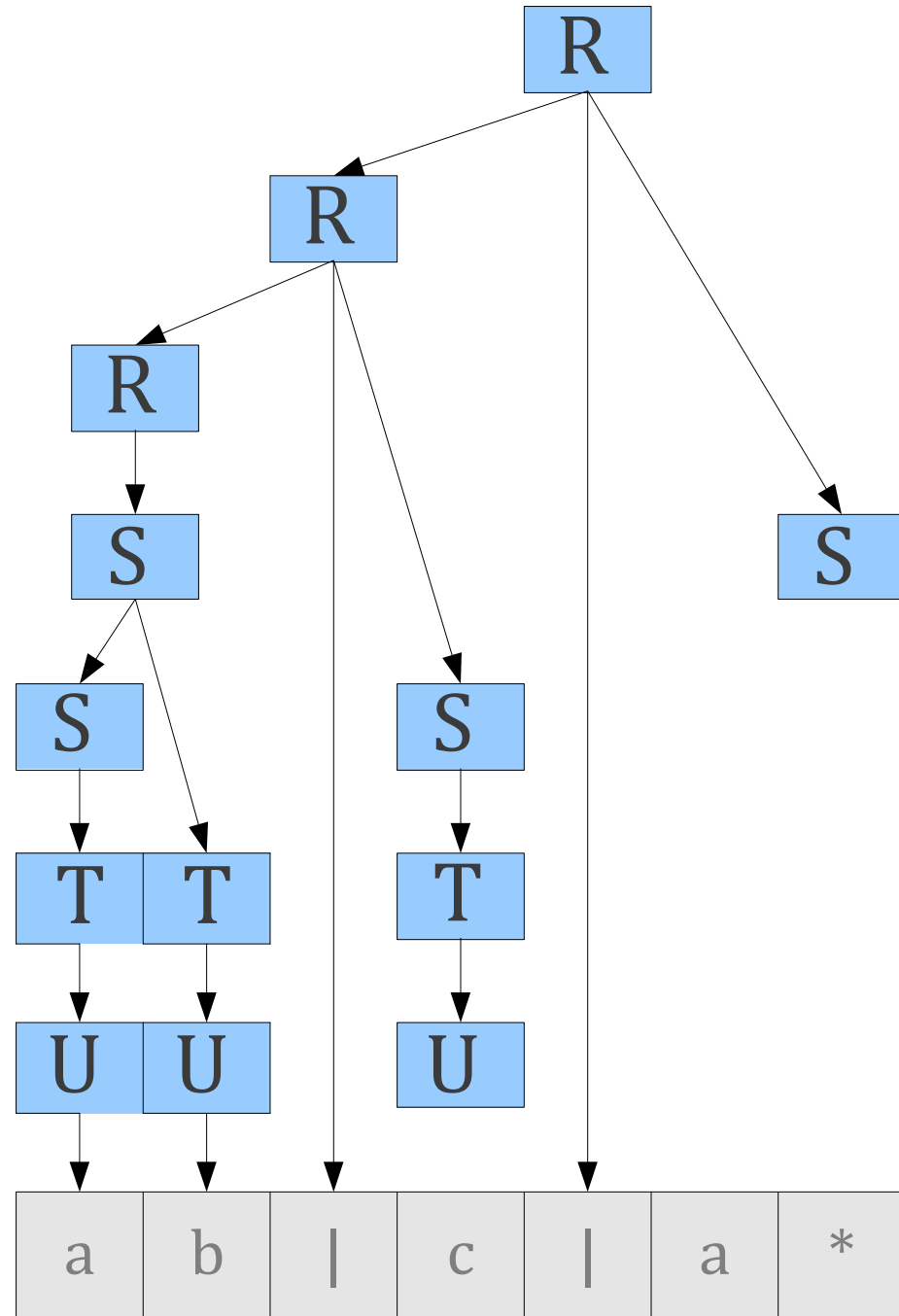
S \rightarrow **T** | **ST**

T → **U** | **T***

U \rightarrow a | b | c | ...

U → "ε"

U \rightarrow (**R**)



R \rightarrow **S** | **R** “ | ” **S**

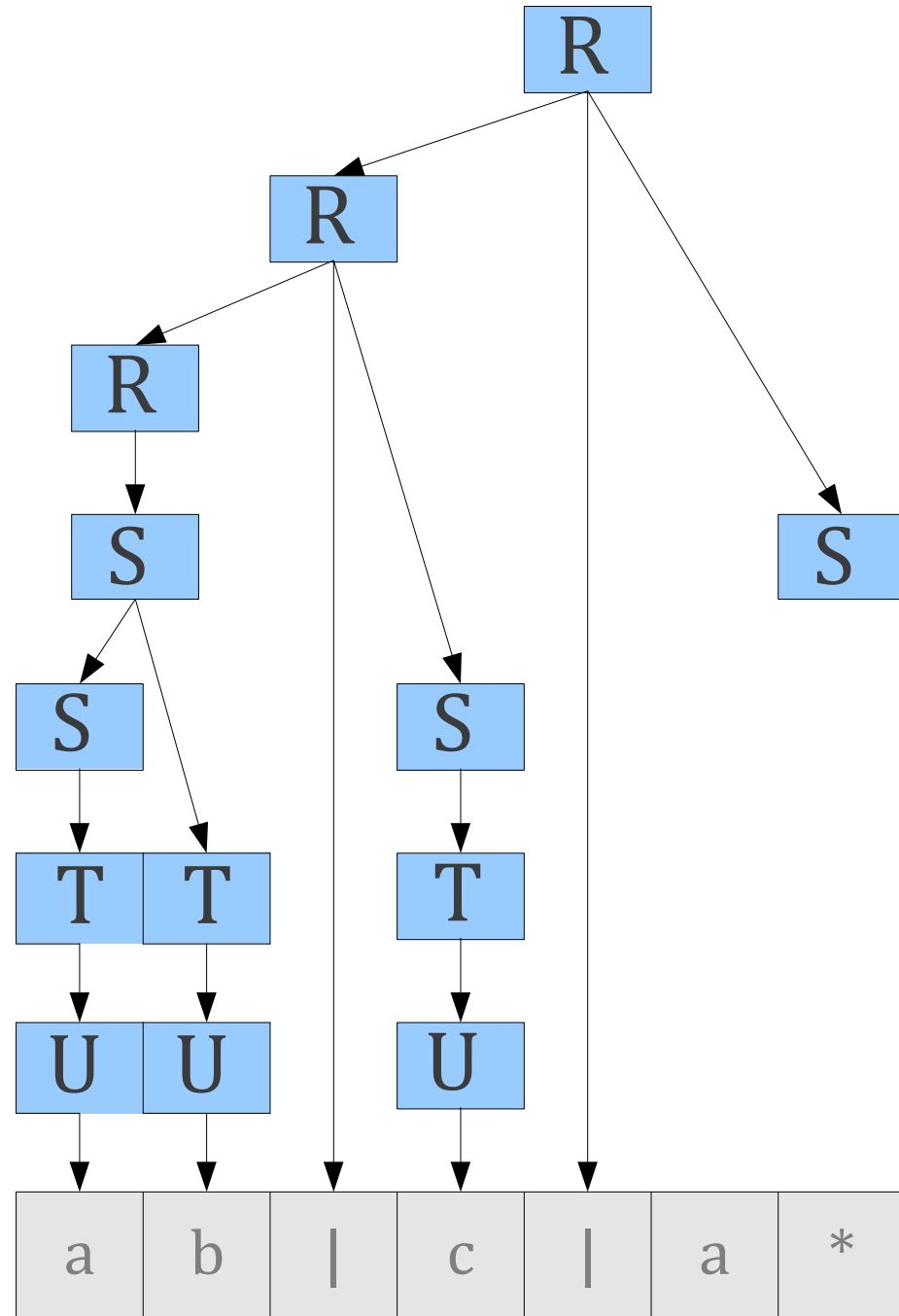
S \rightarrow **T** | **ST**

T → **U** | **T***

U → a | b | c | ...

U → "ε"

U \rightarrow (**R**)



R \rightarrow **S** | **R** “ | ” **S**

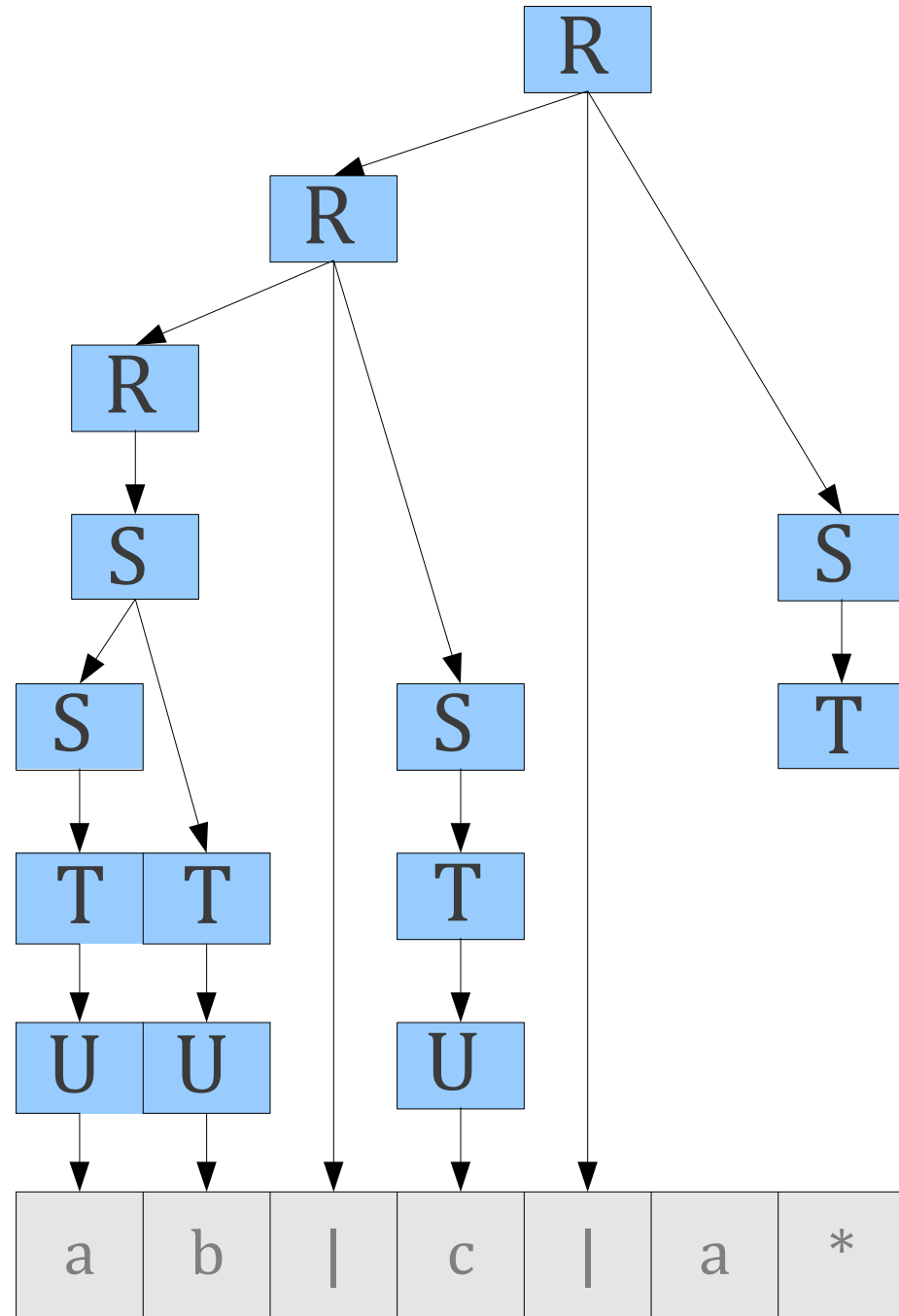
S \rightarrow **T** | **ST**

T → **U** | **T***

U \rightarrow a | b | c | ...

U → "ε"

U \rightarrow (**R**)



$R \rightarrow S \mid R \text{ " | " } S$

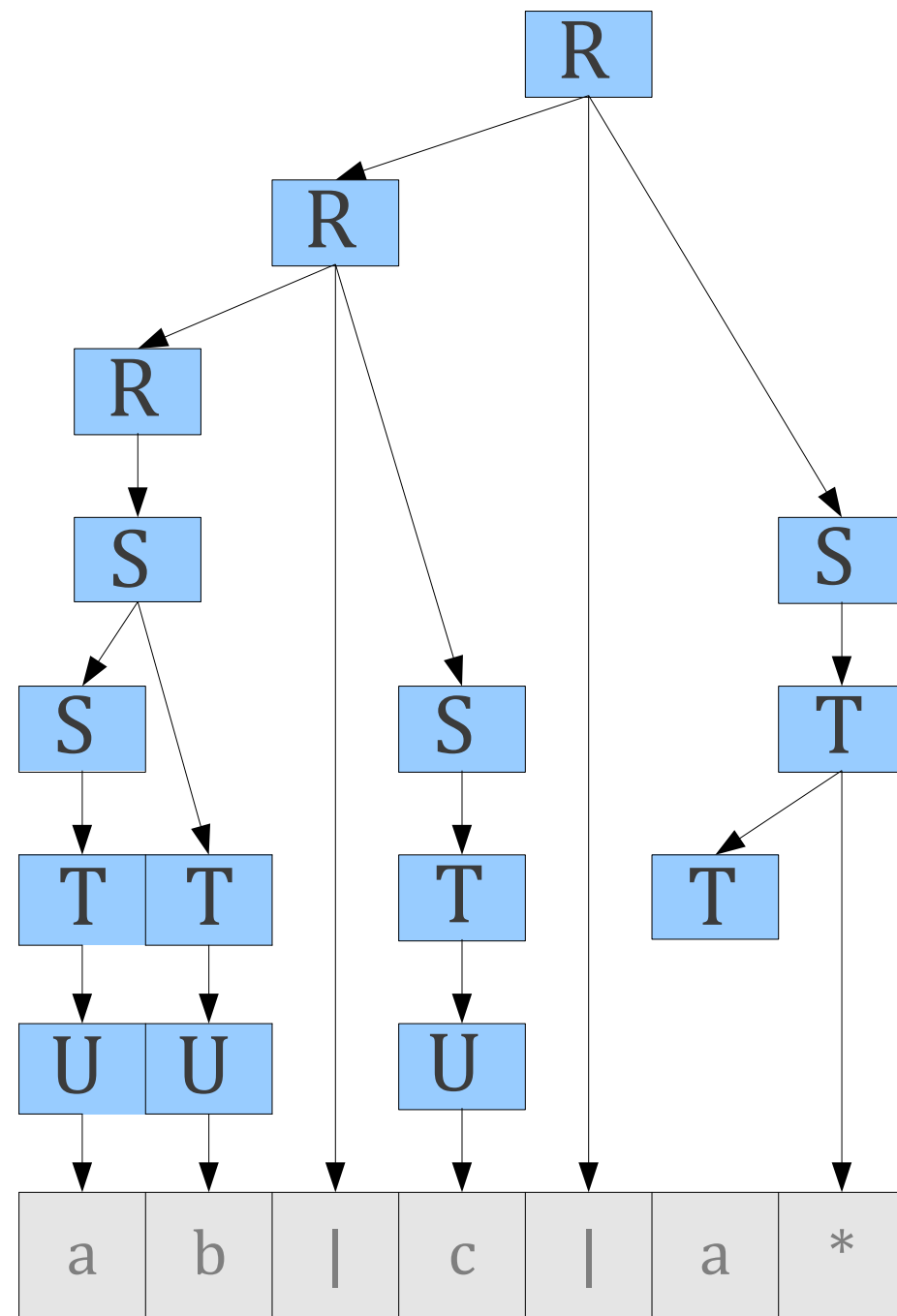
$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \text{"}\epsilon\text{"}$

$U \rightarrow (R)$



$R \rightarrow S \mid R \text{ " | " } S$

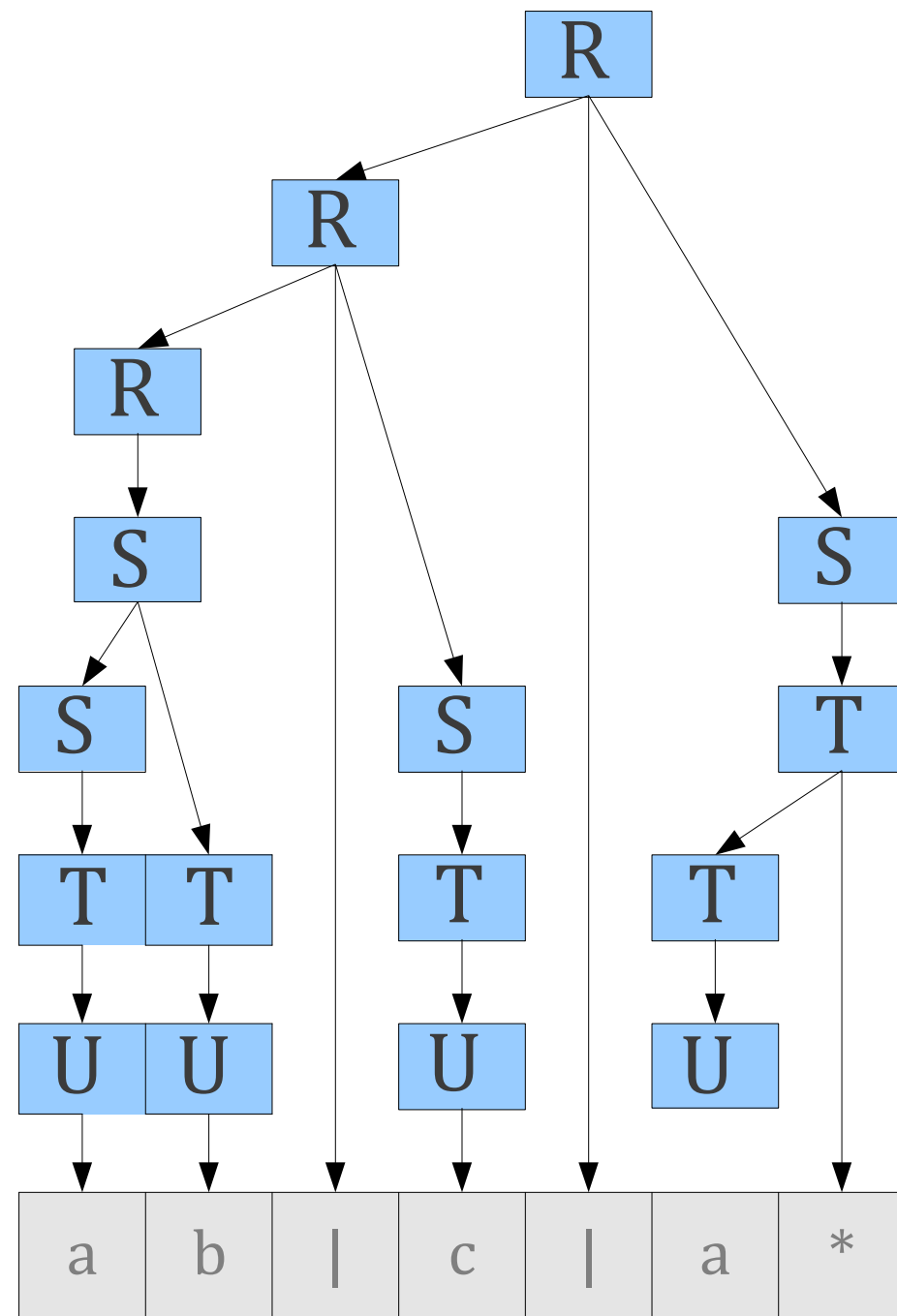
$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \text{"}\epsilon\text{"}$

$U \rightarrow (R)$



$R \rightarrow S \mid R \mid S$

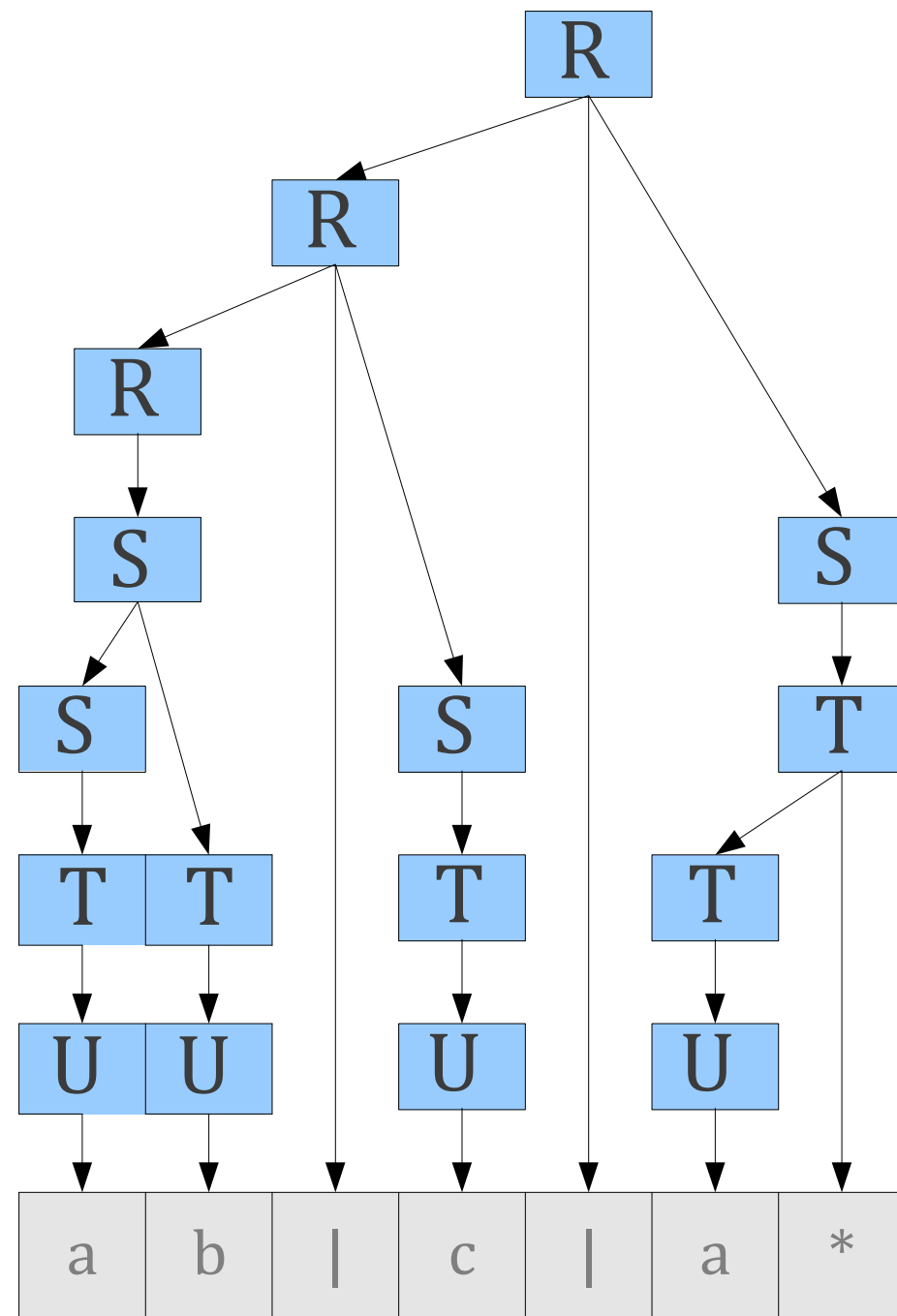
$S \rightarrow T \mid ST$

$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid c \mid \dots$

$U \rightarrow \epsilon$

$U \rightarrow (R)$



Example Questions

Consider the following grammars

$$1. \langle S \rangle \rightarrow a \langle S \rangle \mid \langle S \rangle a \mid \varepsilon$$

$$2. \langle S \rangle \rightarrow (\langle A \rangle) \mid a$$
$$\langle A \rangle \rightarrow \langle A \rangle \langle S \rangle \mid \langle S \rangle$$

$$3. \langle S \rangle \rightarrow \langle A \rangle \langle A \rangle$$
$$\langle A \rangle \rightarrow \langle A \rangle a \mid \langle A \rangle b \mid \varepsilon$$

$$4. \langle S \rangle \rightarrow \langle S \rangle \langle S \rangle \mid \langle A \rangle \langle B \rangle$$
$$\langle A \rangle \rightarrow \langle A \rangle a \langle A \rangle \mid a$$
$$\langle B \rangle \rightarrow \langle B \rangle b \langle B \rangle \mid b$$

$$5. \langle S \rangle \rightarrow a \mid \# \langle S \rangle \mid \langle S \rangle \langle S \rangle$$

a) Indicate which of these grammars are ambiguous.

b) Choose one of the ambiguous grammars above, and prove the ambiguity. Clearly specify your reasons, and show the entire representation.

Consider the following grammars

1. $\langle S \rangle \rightarrow a \langle S \rangle \mid \langle S \rangle a \mid \varepsilon$ Yes aa $S \rightarrow a \langle S \rangle \rightarrow a \langle S \rangle a \rightarrow aa$ or $S \rightarrow \langle S \rangle a \rightarrow a \langle S \rangle a \rightarrow aa$

2. $\langle S \rangle \rightarrow (\langle A \rangle) \mid a$
 $\langle A \rangle \rightarrow \langle A \rangle \langle S \rangle \mid \langle S \rangle$ No $\langle S \rangle \rightarrow (\langle A \rangle) \rightarrow (\langle A \rangle \langle S \rangle) \rightarrow (\langle S \rangle \langle S \rangle) \rightarrow (aa)$

3. $\langle S \rangle \rightarrow \langle A \rangle \langle A \rangle$
 $\langle A \rangle \rightarrow \langle A \rangle a \mid \langle A \rangle b \mid \varepsilon$ Yes a $\langle S \rangle \rightarrow \langle A \rangle \langle A \rangle \rightarrow \langle A \rangle a$ empty \rightarrow empty a empty
 or $\langle S \rangle \rightarrow \langle A \rangle \langle A \rangle \rightarrow$ empty $\langle A \rangle a \rightarrow$ empty empty a

4. $\langle S \rangle \rightarrow \langle S \rangle \langle S \rangle \mid \langle A \rangle \langle B \rangle$
 $\langle A \rangle \rightarrow \langle A \rangle a \langle A \rangle \mid a$
 $\langle B \rangle \rightarrow \langle B \rangle b \langle B \rangle \mid b$ ababab $\langle S \rangle \rightarrow \langle S \rangle \langle S \rangle \rightarrow \langle S \rangle \langle S \rangle \langle A \rangle \langle B \rangle \rightarrow$
 $\langle S \rangle \rightarrow \langle S \rangle \langle S \rangle \rightarrow \langle A \rangle \langle B \rangle \langle S \rangle \langle S \rangle$

5. $\langle S \rangle \rightarrow a \mid \# \langle S \rangle \mid \langle S \rangle \langle S \rangle$
 Yes $\langle S \rangle \rightarrow \langle S \rangle \langle S \rangle \rightarrow \# \langle S \rangle \langle S \rangle \rightarrow \#aa$ or $\langle S \rangle \rightarrow \# \langle S \rangle \rightarrow \# \langle S \rangle \langle S \rangle \rightarrow \#aa$

a) Indicate which of these grammars are ambiguous.

b) Choose one of the ambiguous grammars above, and prove the ambiguity. Clearly specify your reasons, and show the entire representation.

Consider the following grammar in BNF for a language with three infix operators represented by \$, % and #, and a prefix operator !.

```
<exp> → ! <exp>
      | <exp> $ <exp>
      | <exp> % <exp>
      | <exp> # <exp>
      | x
      | y
```

Construct an unambiguous grammar for this language by implementing the following precedence and associativity rules.

•

Prefix operator ! has the highest priority.

Precedence order of the infix operators from highest to lowest: %, \$, #.

Associativity of the operators: \$ and % are left, # is right associative.

The hexadecimal number system uses sixteen digits/alphabets: {0,1,2,3,4,5,6,7,8,9} and {A,B,C,D,E,F} with the base number as 16. Here, A-F of the hexadecimal system means the numbers 10-15 of the decimal number system respectively. In C, Java and Python, these hexadecimals are represented as constants that start with 0x, which consist of the digits 0-9, A-F. For example: 0x02BFCD, 0x6F, 0x4B2A are some examples of numbers in the hexadecimal number system.

Write the regular expression that will match to the hexadecimal numbers

0x([A-F]|[0-9])+

Consider the following lexical analysis file:

```
%% definitions
```

```
binary [01]
```

```
decimal [0-9]
```

```
hexadecimal [0-9A-F]
```

```
alphabetic [a-zA-Z]
```

```
%% regular expressions and the actions when regular expressions are matched
```

```
{alphabetic}{binary}* printf("Valid");
```

```
{alphabetic}* printf("X");
```

```
{binary}* printf("Y");
```

```
{decimal}* printf("W");
```

```
{hexadecimal}* printf("Z");
```

```
%%
```

a) What is the output for the input **afdeb091001**? XW

b) What is the output for the input **00f02AGD**? YValidZX

c) What is the output for the input **AbC13Ed**? XZValid

Consider the following pseudocode:

```
void main{
int z = 12;
int x = 1;
    void subG(){
        int x=3;
        int y=4;
        void subF(){
            print z;
            z := x + y;..... (1)
        }
        void subH(){
            int z=3;
            int y=2;
            subF();
            print z;..... (2)
        }
        subH();
    }
subG();
}
```

	visible	hidden
Point 1	x, y (subG), z (main)	x(main)
Point 2	y, z (subH), x(subG)	y (subG), x(main), z(main)

1. Suppose that the language uses *static scoping*. What is the output? **12 3**
2. Which names are visible and which names are hidden in the referencing environment Point 1, and 2 when static scoping is used?

Consider the following pseudocode:

```
void main{
  int z = 12;
  int x = 1;
  void subG(){
    int x=3;
    int y=4;
    void subF(){
      print z;
      z := x + y;..... (1)
    }
    void subH(){
      int z=3;
      int y=2;
      subF();
      print z;..... (2)
    }
    subH();
  }
  subG();
}
```

	visible	hidden
Point 1	x (subG), y, z (subH)	x, z (main), y (subG)
Point 2	z, y (subH), x (subG)	y (subG), x, z (main)

1. Suppose that the language uses *dynamic scoping*. What is the output? **3 5**
2. Which names are visible and which names are hidden in the referencing environment Point 1, and 2 when dynamic scoping is used?