Context-Free Languages

Announcements

- Problem Set 6 due Friday at 2:15 PM.
  - Stop by OH with questions!
  - Email us at **cs103@cs.stanford.edu** with questions!
- Problem session tonight in 370-370 from 7-8PM.
  - Alternate time: Tomorrow from 7-8PM in Gates 100
  - Covers nonregular languages and CFLs.

The Limits of Regular Languages

- The **pumping lemma for regular languages** can be used to establish limits on what languages are regular.
- If we want to describe more complex languages, we need a more powerful formalism.

Propositional Logic

- Consider the alphabet $\Sigma = \{ \wedge, \vee, \rightarrow, \leftrightarrow, \neg, \top, \bot, p, q, r, (,) \}$.
- Let PL $\in \Sigma^*$ be the language

  { w | w is a legal propositional logic statement }
- L is **not** a regular language.
  - Intuition – need to track balanced parentheses.
  - Can prove this using pumping lemma or homomorphisms.
- What formalism would we use for *PL*?

Describing Propositional Logic

- p is a propositional statement.
- q is a propositional statement.
- r is a propositional statement.
- $\top$ is a propositional statement.
- $\bot$ is a propositional statement.
- If φ is a propositional statement, then (φ) is a propositional statement.
- If φ is a propositional statement, then ¬φ is a propositional statement.
- If φ and ψ are propositional statements, then φ ∧ ψ is a propositional statement.
- If φ and ψ are propositional statements, then φ ∨ ψ is a propositional statement.
- If φ and ψ are propositional statements, then φ → ψ is a propositional statement.
- If φ and ψ are propositional statements, then φ ↔ ψ is a propositional statement.

Describing Propositional Logic

- p **is a propositional statement**.
- q **is a propositional statement**.
- r **is a propositional statement**.
- ⊤ **is a propositional statement**.
- ⊥ **is a propositional statement**.
- If φ **is a propositional statement**, then (φ) **is a propositional statement**.
- If φ **is a propositional statement**, then ¬φ **is a propositional statement**.
- If φ and ψ **are propositional statements**, then φ ∧ ψ **is a propositional statement**.
- If φ and ψ **are propositional statements**, then φ ∨ ψ **is a propositional statement**.
- If φ and ψ **are propositional statements**, then φ → ψ **is a propositional statement**.
- If φ and ψ **are propositional statements**, then φ ↔ ψ **is a propositional statement**.

Describing Propositional Logic

- p **∈ PL**.
- q **∈ PL**.
- r **∈ PL**.
- $\top$ **∈ PL**.
- $\bot$ **∈ PL**.
- If φ **∈ PL**, (φ) **∈ PL**.
- If φ **∈ PL**, then ¬φ **∈ PL**.
- If φ, ψ **∈ PL**, then φ ∧ ψ **∈ PL**.
- If φ, ψ **∈ PL**, then φ ∨ ψ **∈ PL**.
- If φ, ψ **∈ PL**, then φ → ψ **∈ PL**.
- If φ, ψ **∈ PL**, then φ ↔ ψ **∈ PL**.

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \land \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \lor \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ “$\rightarrow$” S
- $S \rightarrow S \leftrightarrow S$

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ “$\rightarrow$” S
- $S \rightarrow S \leftrightarrow S$

S

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- **S → S "→" S**
- $S \rightarrow S \leftrightarrow S$

S

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ "$\rightarrow$" S
- $S \rightarrow S \leftrightarrow S$

$$S \qquad \Rightarrow$$

$$S \rightarrow S$$

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- **$S \rightarrow S \wedge S$**
- $S \rightarrow S \vee S$
- $S \rightarrow S$ "$\rightarrow$" S
- $S \rightarrow S \leftrightarrow S$

$S \Rightarrow$

S $\rightarrow S$

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ “$\rightarrow$” S
- $S \rightarrow S \leftrightarrow S$

$$S \Rightarrow$$
$$S \rightarrow S \Rightarrow$$
$$S \wedge S \rightarrow S$$

Describing Propositional Logic

- p ∈ PL.
- q ∈ PL.
- r ∈ PL.
- ⊤ ∈ PL.
- ⊥ ∈ PL.
- If φ ∈ PL, (φ) ∈ PL.
- If φ ∈ PL, then ¬φ ∈ PL.
- If φ, ψ ∈ PL, then φ ∧ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ∨ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ → ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ↔ ψ ∈ PL.

- **S → p**
- S → q
- S → r
- S → ⊤
- S → ⊥
- S → (S)
- S → ¬S
- S → S ∧ S
- S → S ∨ S
- S → S "→" S
- S → S ↔ S

S ⇒
S → S ⇒
S ∧ S → S

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ “$\rightarrow$” S
- $S \rightarrow S \leftrightarrow S$

$$S \Rightarrow$$
$$S \rightarrow S \Rightarrow$$
$$S \wedge S \rightarrow S \Rightarrow$$
$$p \wedge S \rightarrow S$$

Describing Propositional Logic

- p ∈ PL.
- q ∈ PL.
- r ∈ PL.
- ⊤ ∈ PL.
- ⊥ ∈ PL.
- If φ ∈ PL, (φ) ∈ PL.
- If φ ∈ PL, then ¬φ ∈ PL.
- If φ, ψ ∈ PL, then φ ∧ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ∨ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ → ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ↔ ψ ∈ PL.

- S → p
- **S → q**
- S → r
- S → ⊤
- S → ⊥
- S → (S)
- S → ¬S
- S → S ∧ S
- S → S ∨ S
- S → S "→" S
- S → S ↔ S

S ⇒
S → S ⇒
S ∧ S → S ⇒
p ∧ **S** → S

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ "$\rightarrow$" S
- $S \rightarrow S \leftrightarrow S$

$$
\begin{aligned}
&S \Rightarrow \\
&S \rightarrow S \Rightarrow \\
&S \wedge S \rightarrow S \Rightarrow \\
&p \wedge S \rightarrow S \Rightarrow \\
&p \wedge q \rightarrow S
\end{aligned}
$$

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- **$S \rightarrow r$**
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ "$\rightarrow$" S
- $S \rightarrow S \leftrightarrow S$

$S \Rightarrow$

$S \rightarrow S \Rightarrow$

$S \wedge S \rightarrow S \Rightarrow$

$p \wedge S \rightarrow S \Rightarrow$

$p \wedge q \rightarrow$ **S**

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \to \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \to p$
- $S \to q$
- $S \to r$
- $S \to \top$
- $S \to \bot$
- $S \to (S)$
- $S \to \neg S$
- $S \to S \wedge S$
- $S \to S \vee S$
- $S \to S$ "$\to$" S
- $S \to S \leftrightarrow S$

$$
\begin{aligned}
&S \Rightarrow \\
&S \to S \Rightarrow \\
&S \wedge S \to S \Rightarrow \\
&p \wedge S \to S \Rightarrow \\
&p \wedge q \to S \Rightarrow \\
&p \wedge q \to r
\end{aligned}
$$

Describing Propositional Logic

- p ∈ PL.
- q ∈ PL.
- r ∈ PL.
- ⊤ ∈ PL.
- ⊥ ∈ PL.
- If φ ∈ PL, (φ) ∈ PL.
- If φ ∈ PL, then ¬φ ∈ PL.
- If φ, ψ ∈ PL, then φ ∧ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ∨ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ → ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ↔ ψ ∈ PL.

- S → p
- S → q
- S → r
- S → ⊤
- S → ⊥
- S → (S)
- S → ¬S
- S → S ∧ S
- S → S ∨ S
- S → S "→" S
- S → S ↔ S

S

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- **$S \rightarrow S \vee S$**
- $S \rightarrow S$ "$\rightarrow$" S
- $S \rightarrow S \leftrightarrow S$

S

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ "$\rightarrow$" S
- $S \rightarrow S \leftrightarrow S$

$$S \Rightarrow$$
$$S \vee S$$

Describing Propositional Logic

- p ∈ PL.
- q ∈ PL.
- r ∈ PL.
- ⊤ ∈ PL.
- ⊥ ∈ PL.
- If φ ∈ PL, (φ) ∈ PL.
- If φ ∈ PL, then ¬φ ∈ PL.
- If φ, ψ ∈ PL, then φ ∧ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ∨ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ → ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ↔ ψ ∈ PL.

- S → p
- S → q
- S → r
- S → ⊤
- S → ⊥
- S → (S)
- **S → ¬S**
- S → S ∧ S
- S → S ∨ S
- S → S "→" S
- S → S ↔ S

S ⇒

S ∨ **S**

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ "$\rightarrow$" S
- $S \rightarrow S \leftrightarrow S$

$$S \Rightarrow$$
$$S \vee S \Rightarrow$$
$$S \vee \neg S$$

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- **$S \rightarrow r$**
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- $S \rightarrow S$ "$\rightarrow$" S
- $S \rightarrow S \leftrightarrow S$

$S \Rightarrow$

$S \vee S \Rightarrow$

$S \vee \neg$**S**

Describing Propositional Logic

- p ∈ PL.
- q ∈ PL.
- r ∈ PL.
- ⊤ ∈ PL.
- ⊥ ∈ PL.
- If φ ∈ PL, (φ) ∈ PL.
- If φ ∈ PL, then ¬φ ∈ PL.
- If φ, ψ ∈ PL, then φ ∧ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ∨ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ → ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ↔ ψ ∈ PL.

- S → p
- S → q
- S → r
- S → ⊤
- S → ⊥
- S → (S)
- S → ¬S
- S → S ∧ S
- S → S ∨ S
- S → S "→" S
- S → S ↔ S

S ⇒

S ∨ S ⇒

S ∨ ¬S ⇒

S ∨ ¬r

Describing Propositional Logic

- p ∈ PL.
- q ∈ PL.
- r ∈ PL.
- ⊤ ∈ PL.
- ⊥ ∈ PL.
- If φ ∈ PL, (φ) ∈ PL.
- If φ ∈ PL, then ¬φ ∈ PL.
- If φ, ψ ∈ PL, then φ ∧ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ∨ ψ ∈ PL.
- If φ, ψ ∈ PL, then φ → ψ ∈ PL.
- If φ, ψ ∈ PL, then φ ↔ ψ ∈ PL.

- S → p
- **S → q**
- S → r
- S → ⊤
- S → ⊥
- S → (S)
- S → ¬S
- S → S ∧ S
- S → S ∨ S
- S → S "→" S
- S → S ↔ S

S ⇒
S ∨ S ⇒
S ∨ ¬S ⇒
S ∨ ¬r

Describing Propositional Logic

- $p \in PL$.
- $q \in PL$.
- $r \in PL$.
- $\top \in PL$.
- $\bot \in PL$.
- If $\varphi \in PL$, $(\varphi) \in PL$.
- If $\varphi \in PL$, then $\neg\varphi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \wedge \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \vee \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \rightarrow \psi \in PL$.
- If $\varphi, \psi \in PL$, then $\varphi \leftrightarrow \psi \in PL$.

- $S \rightarrow p$
- $S \rightarrow q$
- $S \rightarrow r$
- $S \rightarrow \top$
- $S \rightarrow \bot$
- $S \rightarrow (S)$
- $S \rightarrow \neg S$
- $S \rightarrow S \wedge S$
- $S \rightarrow S \vee S$
- S → S "→" S
- $S \rightarrow S \leftrightarrow S$

$$
\begin{aligned}
&S && \Rightarrow \\
&S \vee S && \Rightarrow \\
&S \vee \neg S && \Rightarrow \\
&S \vee \neg r && \Rightarrow \\
&q \vee \neg r
\end{aligned}
$$

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look down. Look DIR. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look down. Look DIR. LP NP ⇒
Look down. Look up. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look down. Look DIR. LP NP ⇒
Look down. Look up. LP NP ⇒
Look down. Look up. NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look down. Look DIR. LP NP ⇒
Look down. Look up. LP NP ⇒
Look down. Look up. NP ⇒
Look down. Look up. Now look at me. I'm on a NOUN

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look down. Look DIR. LP NP ⇒
Look down. Look up. LP NP ⇒
Look down. Look up. NP ⇒
Look down. Look up. Now look at me. I'm on a NOUN ⇒
Look down. Look up. Now look at me. I'm on a horse

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look DIR. Look DIR. Look DIR. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look DIR. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look DIR. Look DIR. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look DIR. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look DIR. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look DIR. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look up. LP NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look DIR. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look up. LP NP ⇒
Look into your soul. Look sharp. Look up. NP

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look DIR. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look up. LP NP ⇒
Look into your soul. Look sharp. Look up. NP ⇒
Look into your soul. Look sharp. Look up. Now look at me. I'm on a NOUN.

Describing Language

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

SENTENCE ⇒
LP NP ⇒
Look DIR. LP NP ⇒
Look DIR. Look DIR. LP NP ⇒
Look DIR. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look DIR. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look DIR. LP NP ⇒
Look into your soul. Look sharp. Look up. LP NP ⇒
Look into your soul. Look sharp. Look up. NP ⇒
Look into your soul. Look sharp. Look up. Now look at me. I'm on a NOUN. ⇒
Look into your soul. Look sharp. Look up. Now look at me. I'm on a roll.

Context-Free Grammars

- A **context-free grammar** (CFG) is a system used to describe languages.
- Formally, a CFG is a four-tuple (V, Σ, R, S), where
  - V is a set of **variables** (or **nonterminals**)
  - Σ is a set of **terminals** (with no overlap with V)
  - R is a set of **rules** (or **productions**)
  - $S \in V$ is the **start symbol**.

Context-Free Grammars

- Consider the CFG (V, Σ, R, S) defined as
  - $V = \{ A \}$
  - $\Sigma = \{ 0, 1 \}$
  - $R = \{ A \rightarrow 0A1, A \rightarrow \varepsilon \}$
  - $S = A$
- One string **generated** by the grammar is 0011:
  - $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 0011$
- Another is 000111:
  - $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000111$

General Terminology

- We use upper-case letters to denote nonterminals: A, B, C, D, …
- We use lower-case letters at the start of the alphabet to denote terminals: a, b, c, d
- We use lower-case letters at the end of the alphabet to denote strings: x, y, z

Yields and Derives

- Given a string wAx, if A → *y* is a production, then we say that wAx **yields** wyx.
- We denote this by wAx ⇒ wyx
- Given a string w, we say that w **derives** x if there is a sequence $u_1, u_2, \dots, u_n$ such that

$$w \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \dots \Rightarrow u_n \Rightarrow x$$

- Note that w derives w for any string w.
- We write w ⇒* x if w derives x.

The Language of a Grammar

- If $G = (V, \Sigma, R, S)$ is a CFG, then the **language** of G is the set

$$\mathcal{L}(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

- That is, the set of strings derivable from the start symbol.
- If L is a language and there is some CFG G such that $L = \mathcal{L}(G)$, then we say that L is a **context-free language** (CFL).

The Language of a Grammar

- If G = (V, Σ, R, S) is a CFG, then the **language** of G is the set

$$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

- That is, the set of strings derivable from the start symbol.
- If L is a language and there is some CFG G such that L = *L*(G), then we say that L is a **context-free language** (CFL).

The Language of a Grammar

- If $G = (V, \Sigma, R, S)$ is a CFG, then the **language** of G is the set

$$\mathcal{L}(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$$

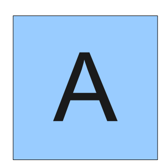
- That is, the set of strings derivable from the start symbol.
- If L is a language and there is some CFG G such that $L = \mathcal{L}(G)$, then we say that L is a **context-free language** (CFL).

Context-Free Grammars

- Consider the CFG G = (V, Σ, R, S) defined as
  - V = { A }
  - Σ = { 0, 1 }
  - R = { $A \to 0A1$, $A \to \varepsilon$ }
  - S = A
- Then $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$

Context-Free Grammars

- Consider the CFG G = (V, Σ, R, S) defined as
  - V = { A }
  - Σ = { 0, 1 }
  - R = { $A \rightarrow 0A1$, $A \rightarrow \varepsilon$ }
  - S = A
- Then L(G) = $\{0^n 1^n \mid n \in \mathbb{N}\}$



Context-Free Grammars

- Consider the CFG G = (V, Σ, R, S) defined as
  - V = { A }
  - Σ = { 0, 1 }
  - R = { A → 0A1, A → ε }
  - S = A
- Then $L(G) = \{0^n 1^n \mid n \in \mathbb{N}\}$

0	A	1

Context-Free Grammars

- Consider the CFG G = (V, Σ, R, S) defined as
  - V = { A }
  - Σ = { 0, 1 }
  - R = { A → 0A1, A → ε }
  - S = A
- Then $L(G) = \{0^n1^n \mid n \in \mathbb{N}\}$

0	0	A	1	1

Context-Free Grammars

- Consider the CFG $G = (V, \Sigma, R, S)$ defined as
  - $V = \{ A \}$
  - $\Sigma = \{ 0, 1 \}$
  - $R = \{ A \rightarrow 0A1, A \rightarrow \varepsilon \}$
  - $S = A$
- Then $\mathcal{L}(G) = \{0^n 1^n \mid n \in \mathbb{N} \}$

0	0	0	A	1	1	1

Context-Free Grammars

- Consider the CFG G = (V, Σ, R, S) defined as
  - V = { A }
  - Σ = { 0, 1 }
  - R = { A → 0A1, A → ε }
  - S = A
- Then *L*(G) = $\{0^n1^n \mid n \in \mathbb{N}\}$

0	0	0	0	A	1	1	1	1

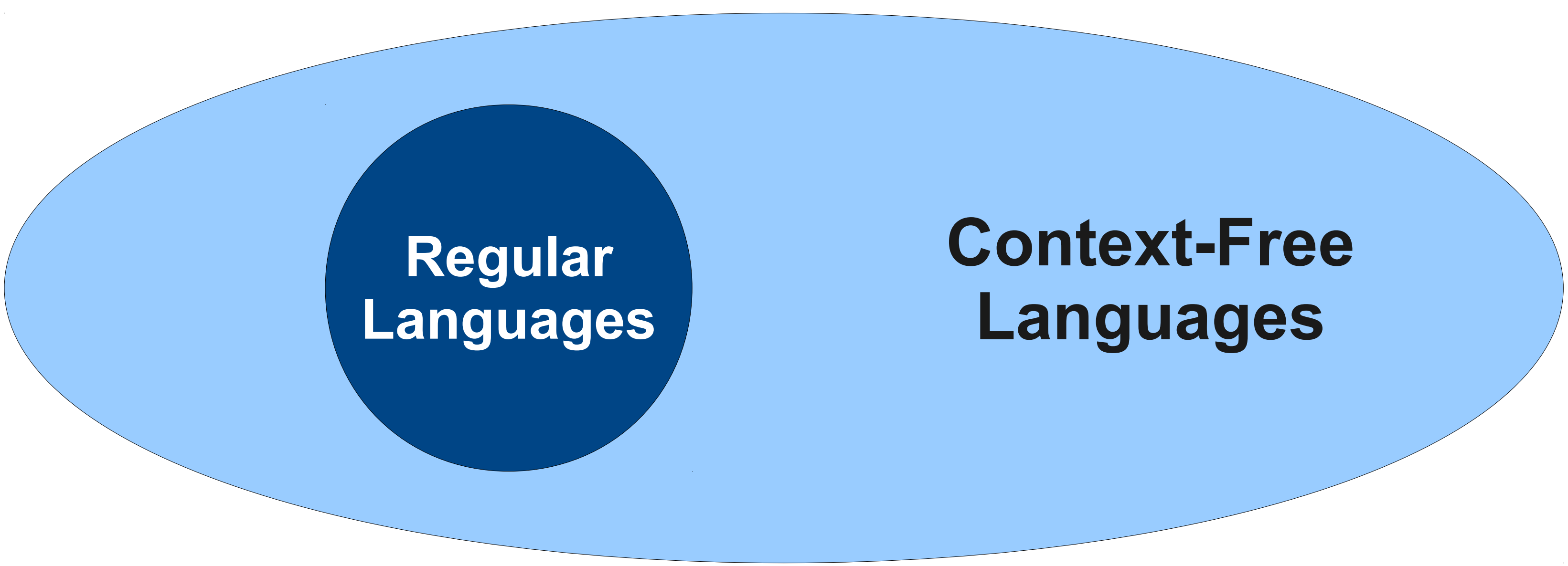
Context-Free Grammars

- Consider the CFG $G = (V, \Sigma, R, S)$ defined as
  - $V = \{ A \}$
  - $\Sigma = \{ 0, 1 \}$
  - $R = \{ A \rightarrow 0A1, A \rightarrow \varepsilon \}$
  - $S = A$
- Then $\mathcal{L}(G) = \{0^n1^n \mid n \in \mathbb{N} \}$

0	0	0	0	1	1	1	1

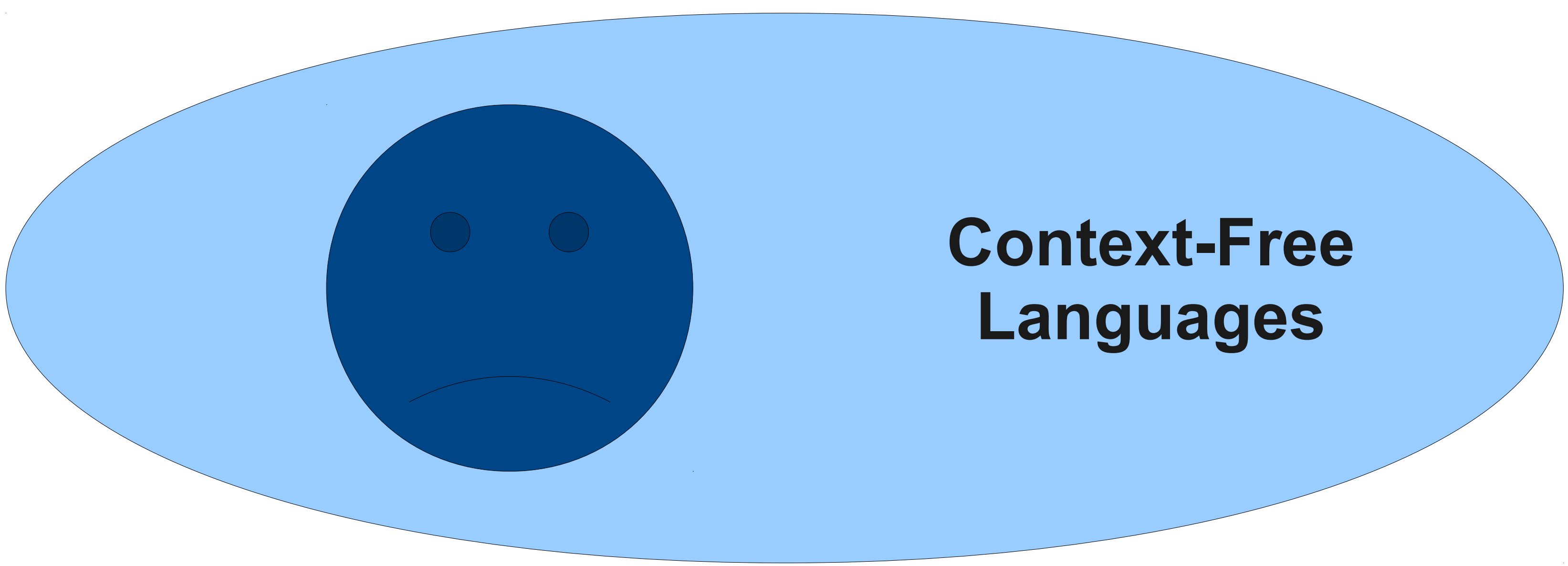
Regular and Context-Free Languages

- Context-free languages are a **strict superset** of the regular languages.
- Every regular language is context-free, but not necessarily the other way around.
- We'll see a proof of this next time.



Regular and Context-Free Languages

- Context-free languages are a **strict superset** of the regular languages.
- Every regular language is context-free, but not necessarily the other way around.
- We'll see a proof of this next time.



Some Shorthand

SENTENCE → LP NP
LP → Look DIR. LP
LP → ε
DIR → up
DIR → down
DIR → into your soul
DIR → sharp
DIR → before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse
NOUN → velociraptor
NOUN → whale
NOUN → roll

Some Shorthand

SENTENCE → LP NP
LP → Look DIR. LP | ε

DIR → up | down | into your soul | sharp | before you leap

NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

Some Shorthand

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

Some Differences from Regexps

- The symbol | still means “or.”
- ε still means “the empty string”
- Parentheses, *, and Ø are **not** part of CFGs.

Leftmost Derivations

```
BLOCK → STMT
      | { STMTS }

STMTS → ε
      | STMT STMTS

STMT  → EXPR;
      | if (EXPR) BLOCK
      | while (EXPR) BLOCK
      | do BLOCK while (EXPR)
      | BLOCK
      | …

EXPR  → id
      | const
      | EXPR = EXPR
      | EXPR + EXPR
      | EXPR – EXPR
      | EXPR * EXPR
      | ...
```

Leftmost Derivations

BLOCK → STMT
  | { STMTS }

STMTS → ε
  | STMT STMTS

STMT → EXPR;
  | **if** (EXPR) BLOCK
  | **while** (EXPR) BLOCK
  | **do** BLOCK **while** (EXPR)
  | BLOCK
  | …

EXPR → **id**
  | **const**
  | EXPR = EXPR
  | EXPR + EXPR
  | EXPR – EXPR
  | EXPR * EXPR
  | …

CFGs are often used to describe programming languages.

Take a compilers course for more information on how and why!

Leftmost Derivations

```
BLOCK → STMT
      | { STMTS }

STMTS → ε
      | STMT STMTS

STMT  → EXPR;
      | if (EXPR) BLOCK
      | while (EXPR) BLOCK
      | do BLOCK while (EXPR)
      | BLOCK
      | ...

EXPR  → id
      | const
      | EXPR = EXPR
      | EXPR + EXPR
      | EXPR – EXPR
      | EXPR * EXPR
      | ...
```

```
  BLOCK
⇒ { STMTS }
⇒ { STMT STMTS }
⇒ { EXPR; STMTS }
⇒ { EXPR = EXPR; STMTS }
⇒ { id = EXPR; STMTS }
⇒ { id = EXPR + EXPR; STMTS }
⇒ { id = id + EXPR; STMTS }
⇒ { id = id + const; } STMTS }
⇒ { id = id + const; }
```

Leftmost Derivations

- A **leftmost derivation** expands the leftmost nonterminal first.
- A **rightmost derivation** expands the rightmost nonterminal first.
- If you take a compilers course (**which you should!**), these concepts are invaluable when developing parsers.

Context-Free Grammars

- Interestingly, the set of regular expressions is **not** regular!
- However, it is context-free!
- A regular expression can be
  - Any letter
  - ε
  - Ø
  - The concatenation of regular expressions.
  - The disjunction of regular expressions.
  - The Kleene closure of a regular expression.
  - A parenthesized regular expression.

Context-Free Grammars

- Interestingly, the set of regular expressions is **not** regular!
- However, it is context-free!
- A regular expression can be
  - R → a | b | c | …
  - R → "ε"
  - R → Ø
  - R → RR
  - R → R "|" R
  - R → R*
  - R → (R)

CFG for Regular Expressions

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

$R \rightarrow$ “ε”

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ “|” R

$R \rightarrow R^*$

$R \rightarrow (R)$

CFG for Regular Expressions

R → a | b | c | …

R → “ε”

R → Ø

R → RR

R → R “|” R

R → R*

R → (R)

R

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

$\Rightarrow R \mid R$

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

$\Rightarrow R \mid R$

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

$\Rightarrow$ R | **R**

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \varnothing$

R → RR

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

⇒ R | **R**

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

R → RR

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

⇒ R | **R**

⇒ R | RR

CFG for Regular Expressions

$R \rightarrow$ a | b | c | …

$R \rightarrow$ "ε"

$R \rightarrow$ Ø

$R \rightarrow$ RR

$R \rightarrow$ R "|" R

$R \rightarrow$ R*

$R \rightarrow$ (R)

R

⇒ R | **R**

⇒ R | RR

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

⇒ R | **R**

⇒ R | R**R**

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

⇒ R | **R**

⇒ R | R**R**

CFG for Regular Expressions

R → a | b | c | …

R → "ε"

R → Ø

R → RR

R → R "|" R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | RR*

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | RR*

CFG for Regular Expressions

$R \to a \mid b \mid c \mid \ldots$

$R \to$ “ε”

$R \to \varnothing$

$R \to RR$

$R \to R$ “|” R

$R \to R^*$

$R \to (R)$

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

CFG for Regular Expressions

R → a | b | c | …

R → "ε"

R → Ø

R → RR

R → R "|" R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

CFG for Regular Expressions

R → a | b | c | …

R → “ε”

R → Ø

R → RR

R → R “|” R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ R | cR*

CFG for Regular Expressions

R → a | b | c | …

R → "ε"

R → ∅

R → RR

R → R "|" R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ R | cR*

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ **R** | cR*

CFG for Regular Expressions

R → a | b | c | …

R → "ε"

R → Ø

R → RR

R → R "|" R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ **R** | cR*

CFG for Regular Expressions

R → a | b | c | …

R → "ε"

R → Ø

R → RR

R → R "|" R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ **R** | cR*

⇒ a | cR*

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \emptyset$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ **R** | cR*

⇒ a | cR*

CFG for Regular Expressions

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

$R \rightarrow$ "ε"

$R \rightarrow \varnothing$

$R \rightarrow RR$

$R \rightarrow R$ "|" R

$R \rightarrow R^*$

$R \rightarrow (R)$

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ **R** | cR*

⇒ a | c**R***

CFG for Regular Expressions

R → a | b | c | …

R → “ε”

R → ∅

R → RR

R → R “|” R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ **R** | cR*

⇒ a | c**R***

CFG for Regular Expressions

R → a | b | c | …

R → "ε"

R → ∅

R → RR

R → R "|" R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ **R** | cR*

⇒ a | c**R***

⇒ a | cb*

CFG for Regular Expressions

R → a | b | c | …

R → "ε"

R → Ø

R → RR

R → R "|" R

R → R*

R → (R)

R

⇒ R | **R**

⇒ R | R**R**

⇒ R | **R**R*

⇒ **R** | cR*

⇒ a | c**R***

⇒ a | cb*

Parse Trees

$R \to a \mid b \mid c \mid \ldots$

$R \to$ “ε”

$R \to Ø$

$R \to RR$

$R \to R$ “|” R

$R \to R^*$

$R \to (R)$

R

Parse Trees

R → a | b | c | …

R → “ε”

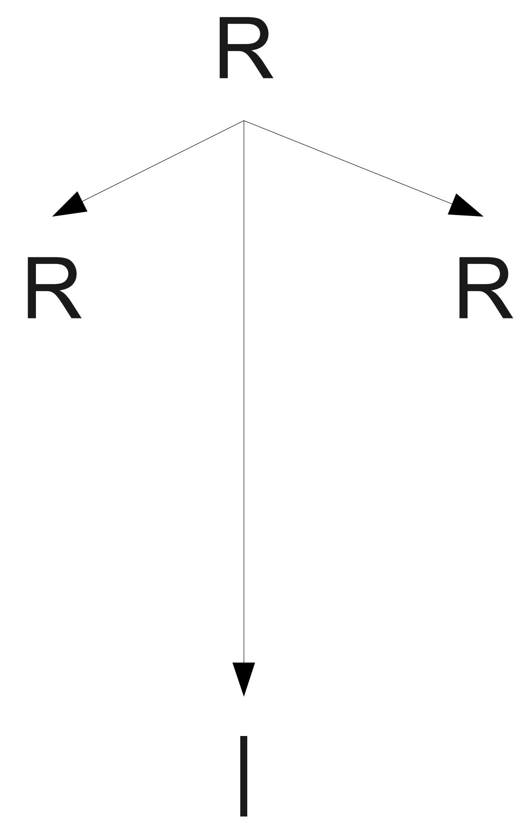
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



Parse Trees

R → a | b | c | …

R → "ε"

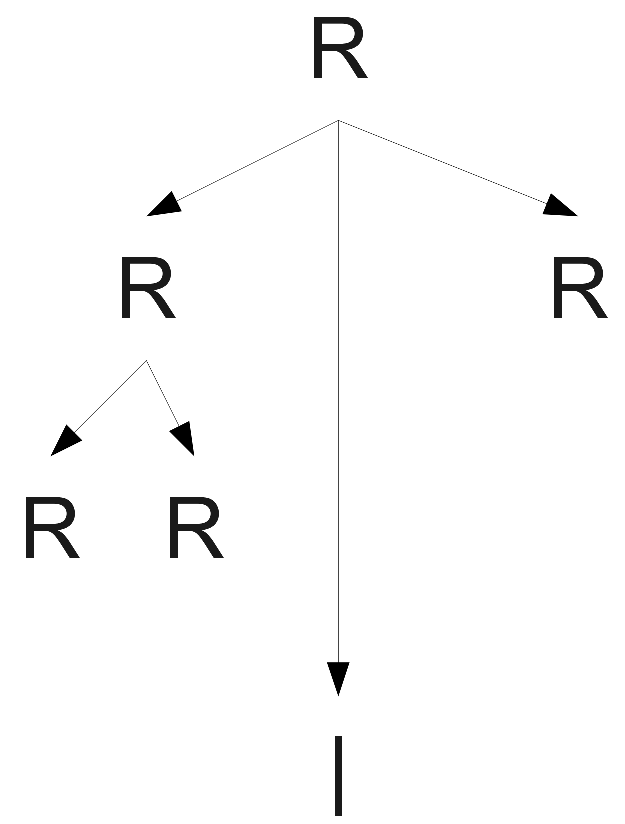
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Parse Trees

R → a | b | c | …

R → “ε”

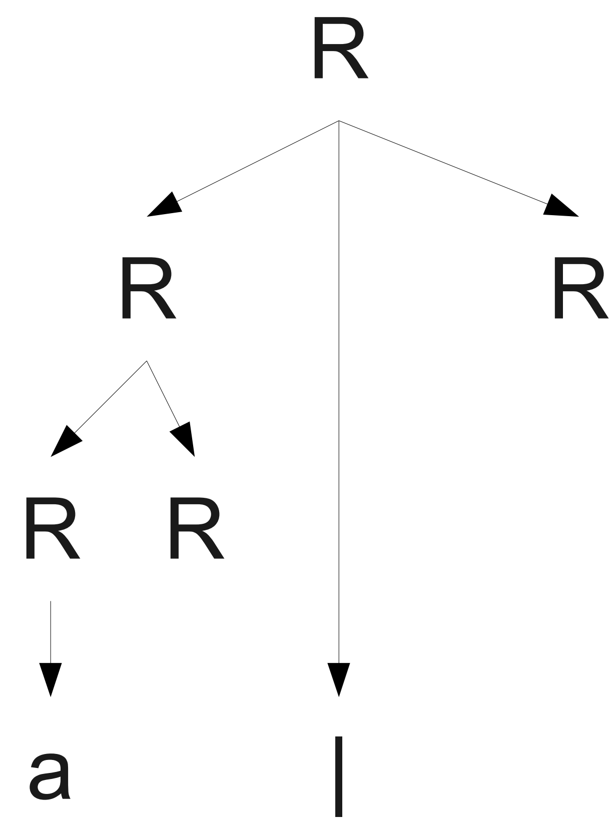
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



Parse Trees

R → a | b | c | …

R → "ε"

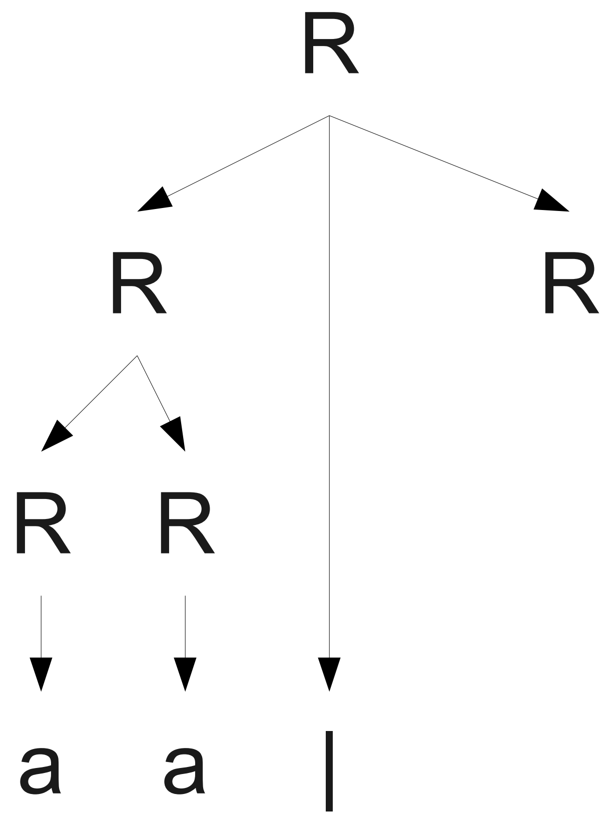
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Parse Trees

R → a | b | c | …

R → “ε”

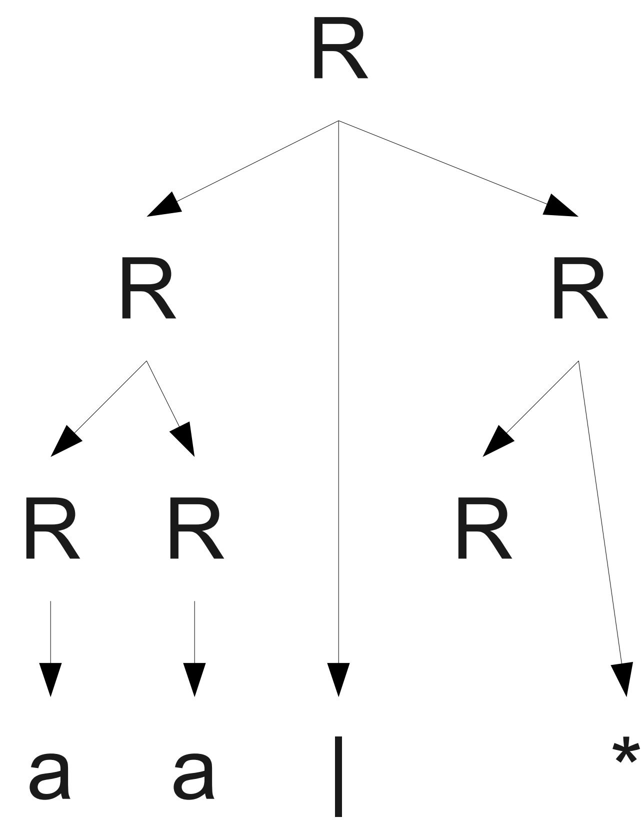
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



Parse Trees

R → a | b | c | …

R → "ε"

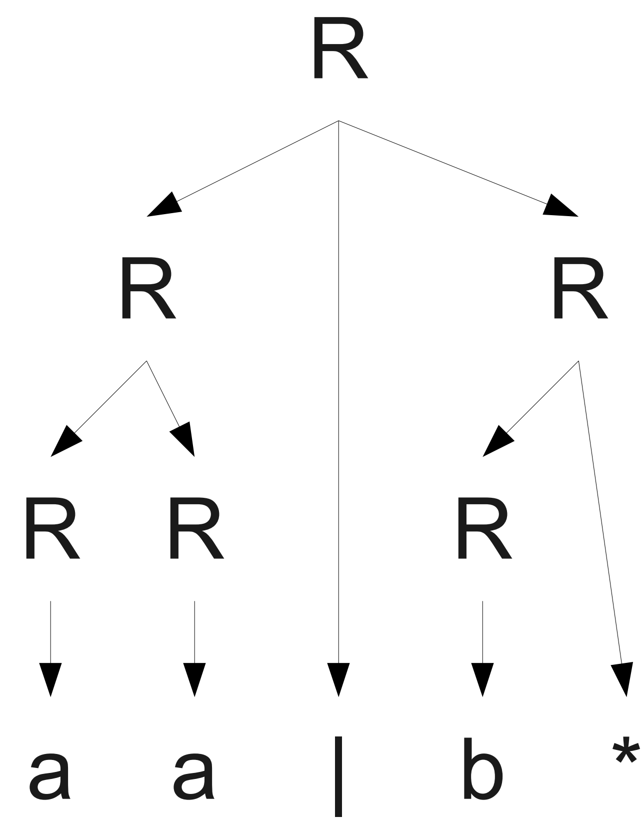
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Parse Trees

R → a | b | c | ...

R → “ε”

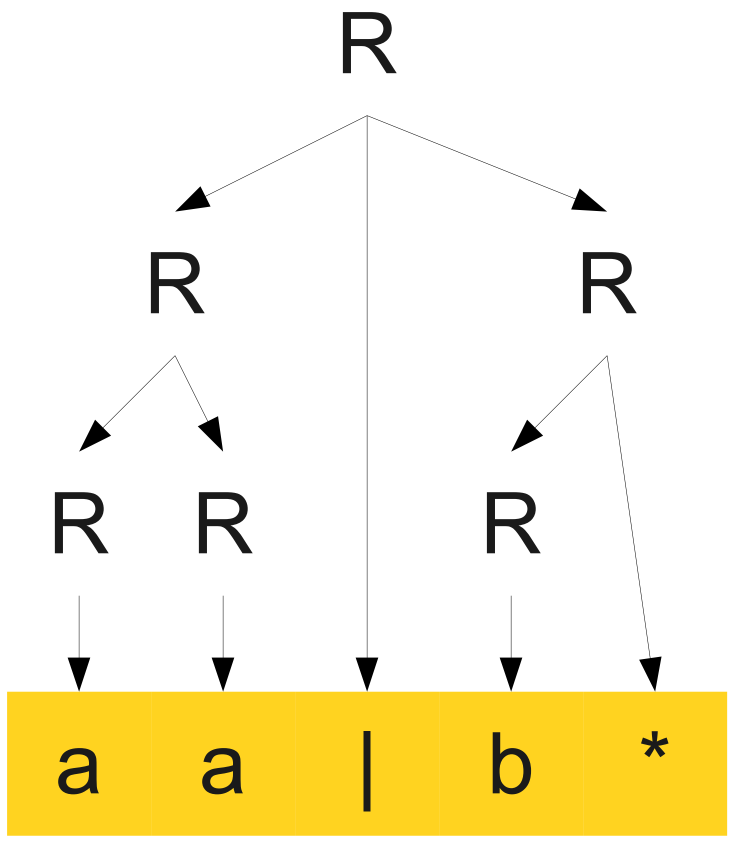
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



Parse Trees

- Tree structure encoding the steps in a derivation.
- Inorder walk of the leaves contains the generated string.
- A derivation encodes **how** to produce the input.
- A parse tree encodes the **structure** of the input.

Parse Trees

R → a | b | c | …

R → “ε”

R → Ø

R → RR

R → R “|” R

R → R*

R → (R)

R

Parse Trees

$R \to a \mid b \mid c \mid \ldots$

$R \to$ "ε"

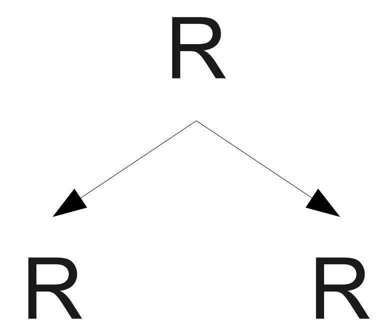
$R \to \emptyset$

$R \to RR$

$R \to R$ "|" R

$R \to R^*$

$R \to (R)$



Parse Trees

R → a | b | c | …

R → "ε"

R → Ø

R → RR

R → R "|" R

R → R*

R → (R)

Parse Trees

R → a | b | c | …

R → “ε”

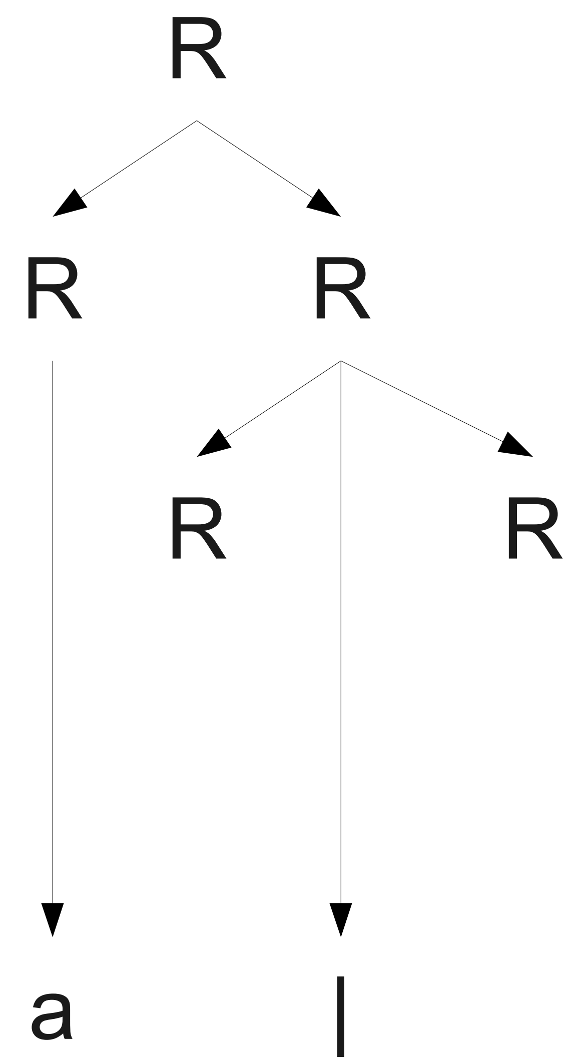
R → Ø

R → RR

R → R “|” R

R → R*

R → (R)



Parse Trees

R → a | b | c | …

R → "ε"

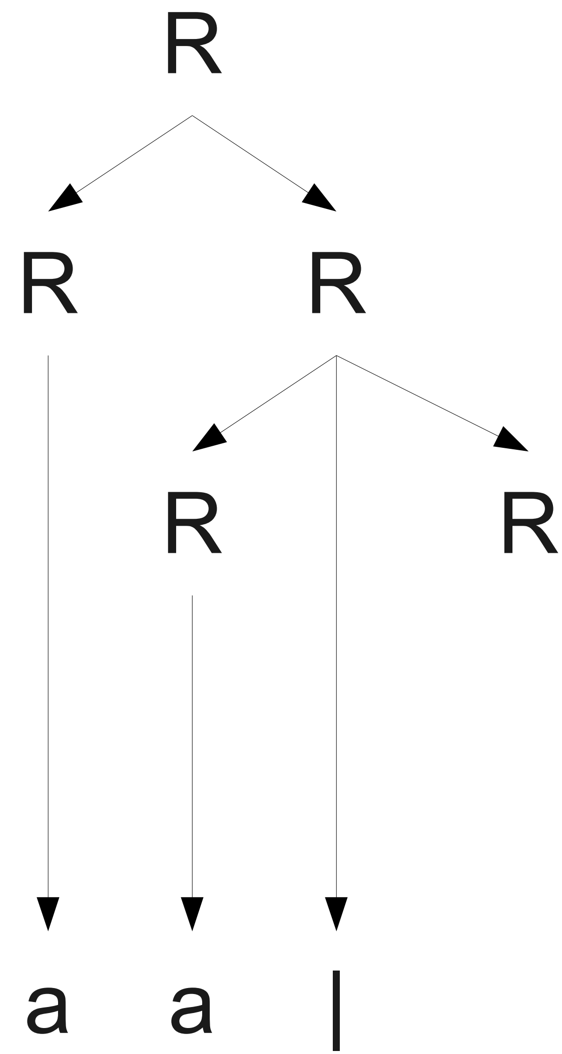
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Parse Trees

R → a | b | c | …

R → "ε"

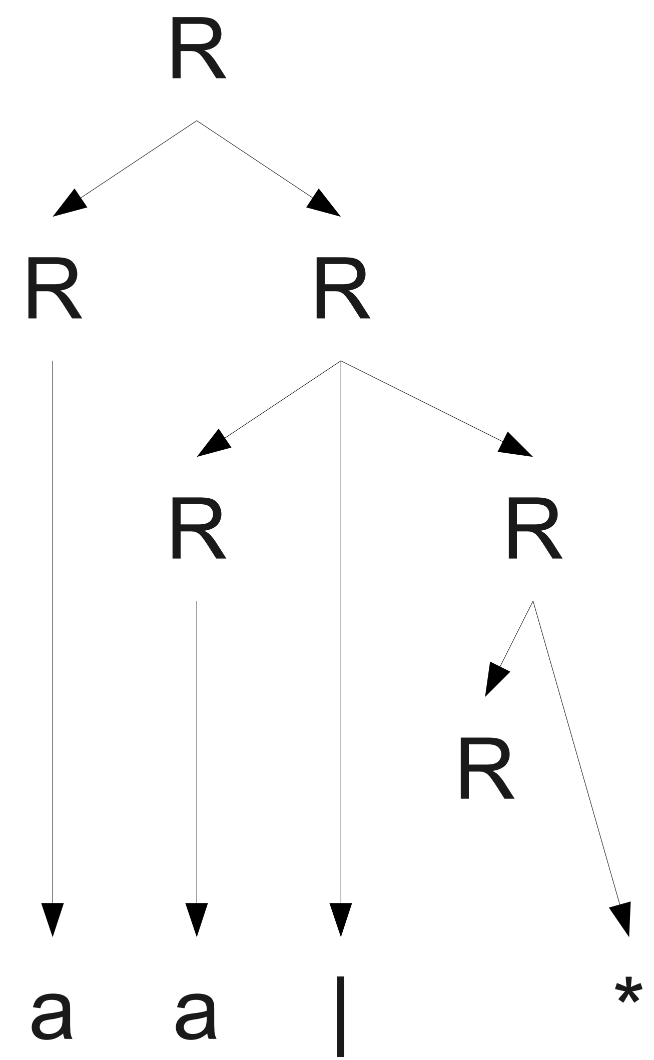
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Parse Trees

R → a | b | c | …

R → "ε"

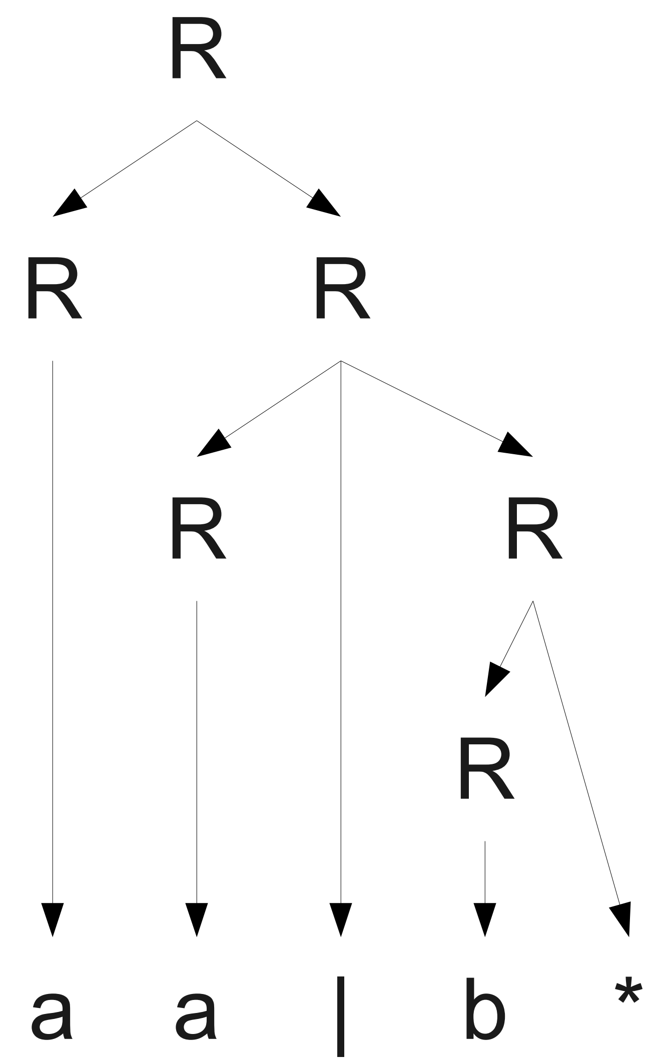
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Parse Trees

$R \to a \mid b \mid c \mid \ldots$

$R \to$ "ε"

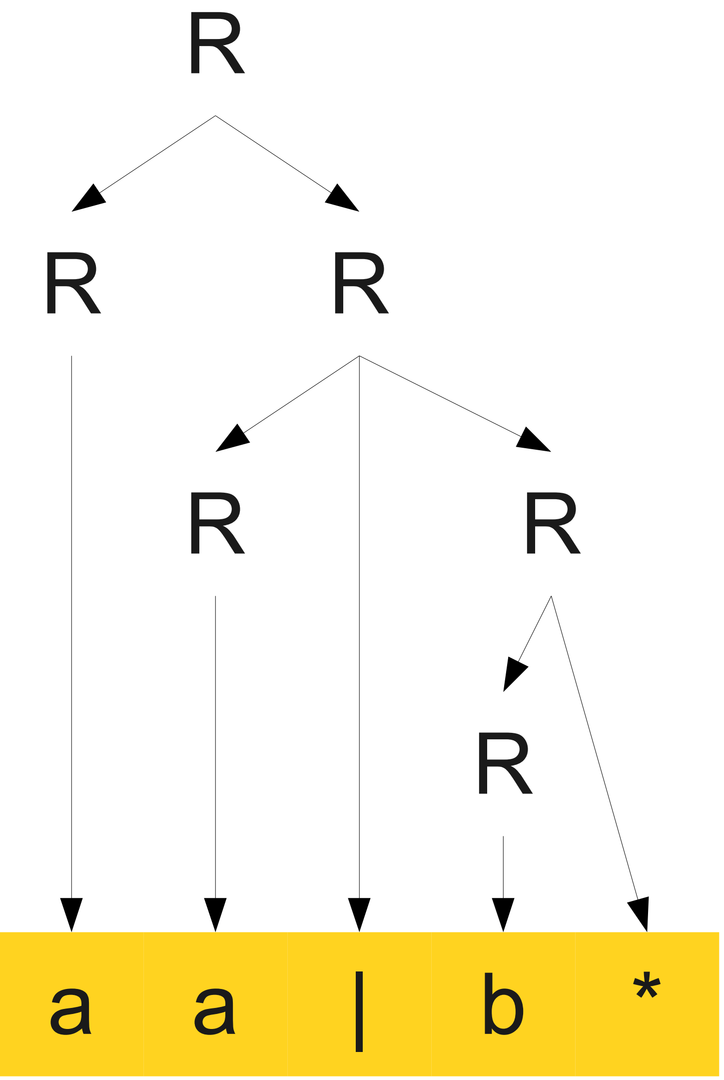
$R \to \emptyset$

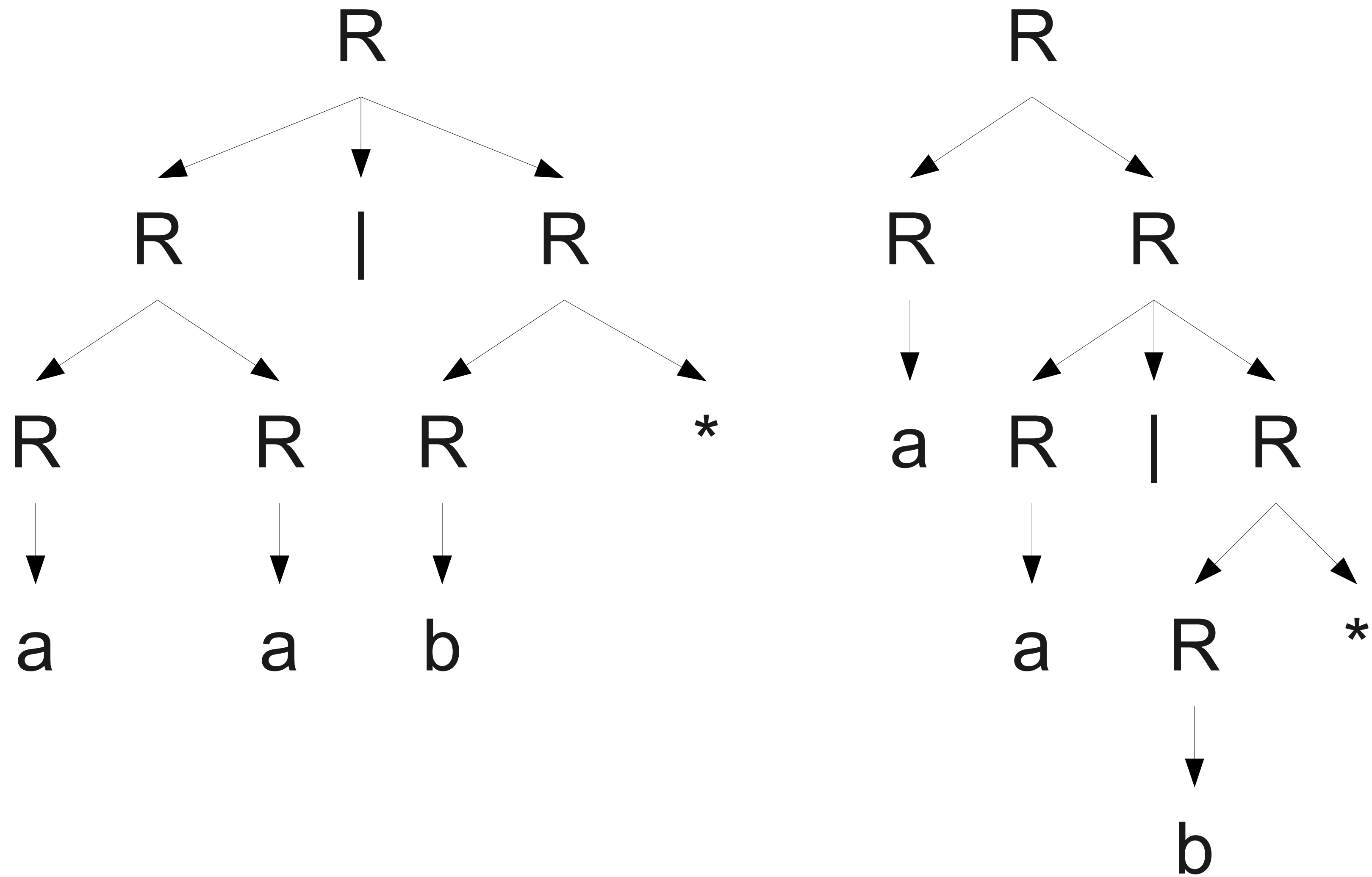
$R \to RR$

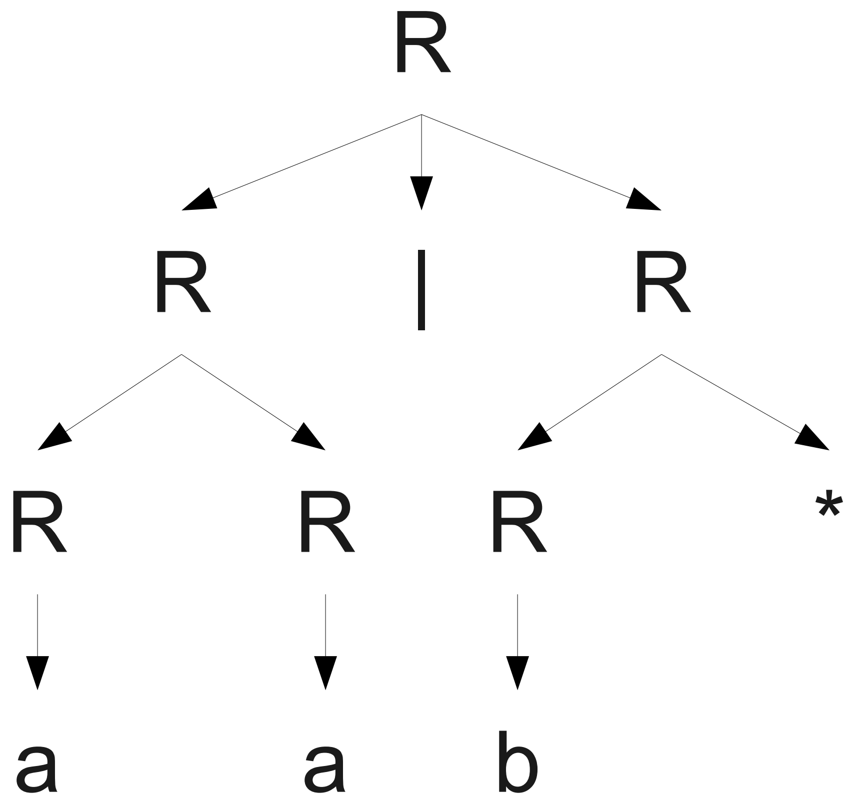
$R \to R$ "|" R

$R \to R^*$

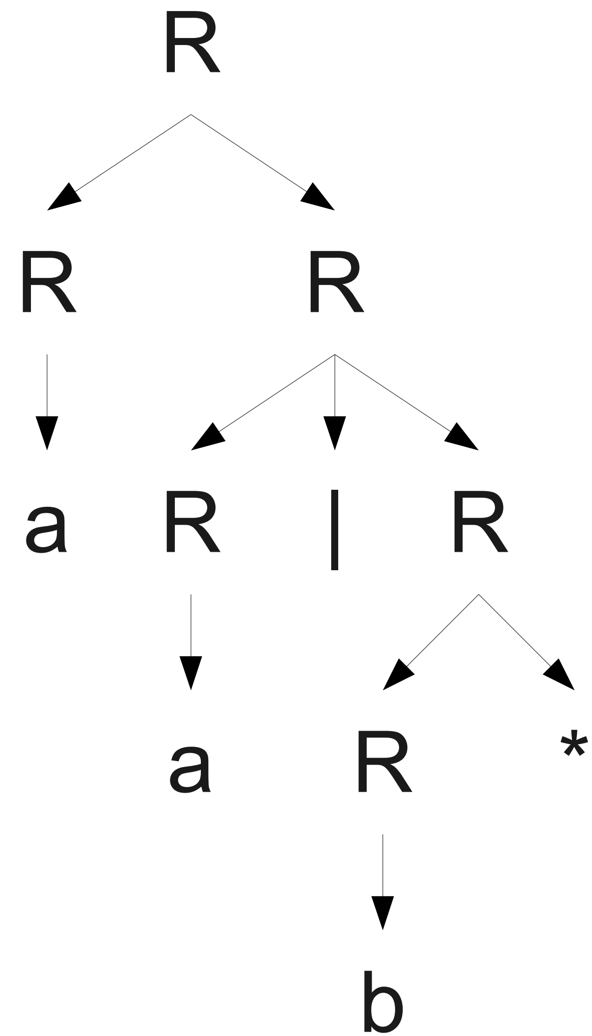
$R \to (R)$



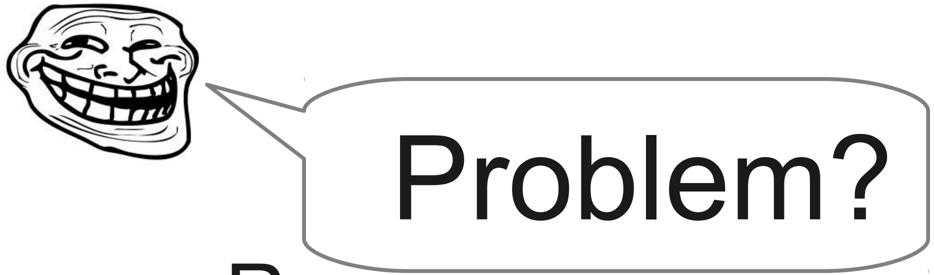

R
R
|
R
R
R
R
*
a
a
b
R
R
R
a
R
|
R
a
R
*
b

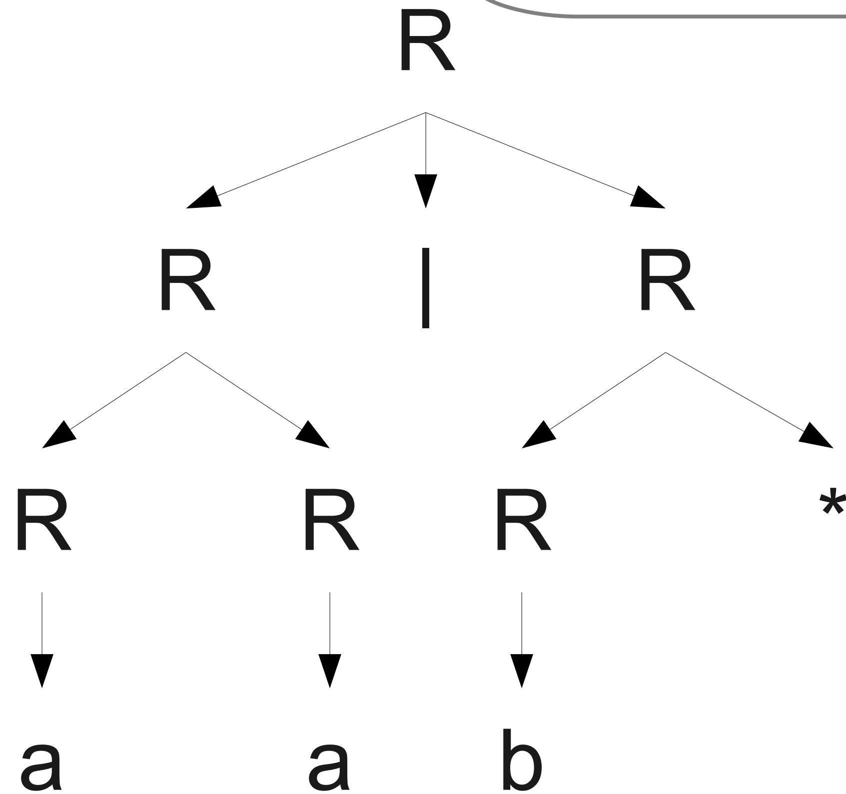


(aa) | (b*)

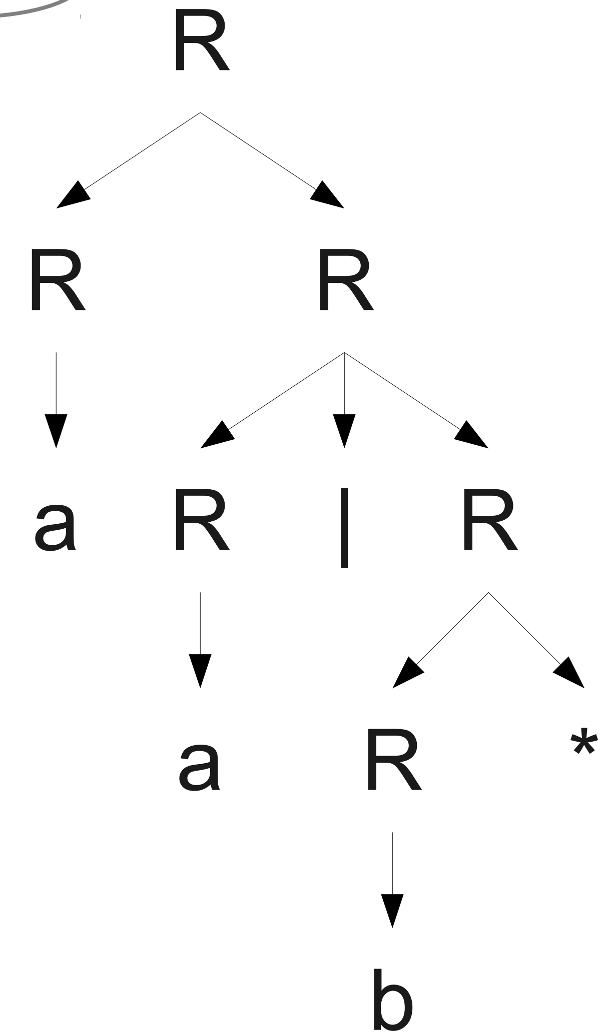


a(a | (b*))





(aa) | (b*)



a(a | (b*))

Ambiguity

- A CFG is said to be **ambiguous** if there is at least one string with two or more parse trees.
- Unfortunately:
  - There is **no algorithm** to definitively check whether a CFG is ambiguous.
  - There are some languages that are **inherently** ambiguous (any CFG for the language must be ambiguous)

Is Ambiguity a Problem?

Is Ambiguity a Problem?

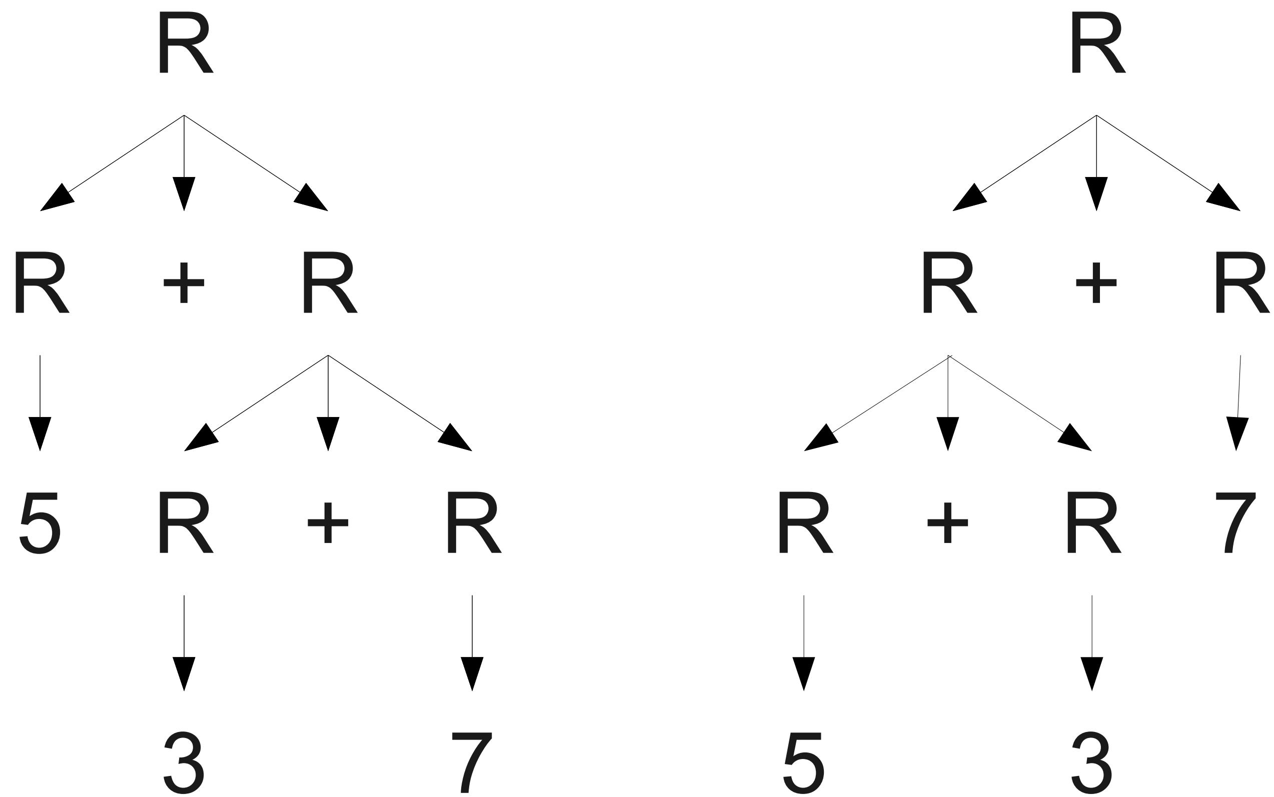
- Depends on **semantics**.

Is Ambiguity a Problem?

- Depends on **semantics**.
- $E \rightarrow$ **int** | E + E

Is Ambiguity a Problem?

- Depends on **semantics**.
- E → **int** | E + E

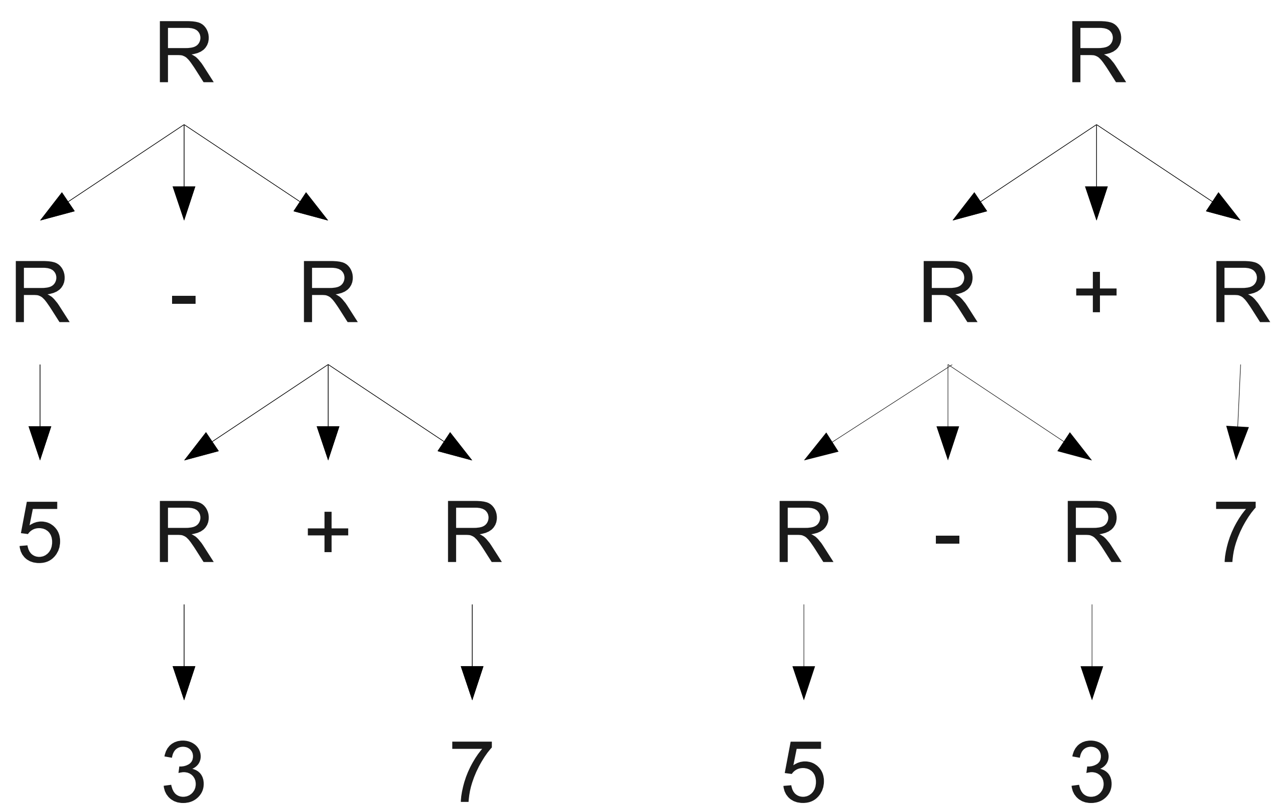


Is Ambiguity a Problem?

- Depends on **semantics**.
- E → **int** | E + E | E – E

Is Ambiguity a Problem?

- Depends on **semantics**.
- E → **int** | E + E | E – E



Resolving Ambiguity

- Ambiguous grammars can often be rewritten to be unambiguous.
- Return to our CFG for regular expressions:

R → a | b | c | …

R → “ε”

R → Ø

R → RR

R → R “|” R

R → R*

R → (R)

<table><tr><td>a</td><td>a</td><td>|</td><td>b</td><td>*</td></tr></table>

Resolving Ambiguity

- Ambiguous grammars can often be rewritten to be unambiguous.
- Return to our CFG for regular expressions:

  R → a | b | c | …

  R → “ε”

  R → Ø

  R → RR

  R → R “|” R

  R → R*

  R → (R)

a	a			b	*

Resolving Ambiguity

- Ambiguous grammars can often be rewritten to be unambiguous.
- Return to our CFG for regular expressions:

R → a | b | c | …

R → "ε"

R → Ø

R → RR

R → R "|" R

R → R*

R → (R)

a	a	|	b	*

Resolving Ambiguity

- Ambiguous grammars can often be rewritten to be unambiguous.
- Return to our CFG for regular expressions:

  R → a | b | c | …

  R → "ε"

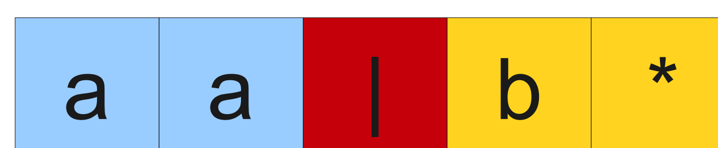
  R → Ø

  R → RR

  R → R "|" R

  R → R*

  R → (R)



Resolving Ambiguity

- Ambiguous grammars can often be rewritten to be unambiguous.
- Return to our CFG for regular expressions:

  R → a | b | c | …

  R → “ε”

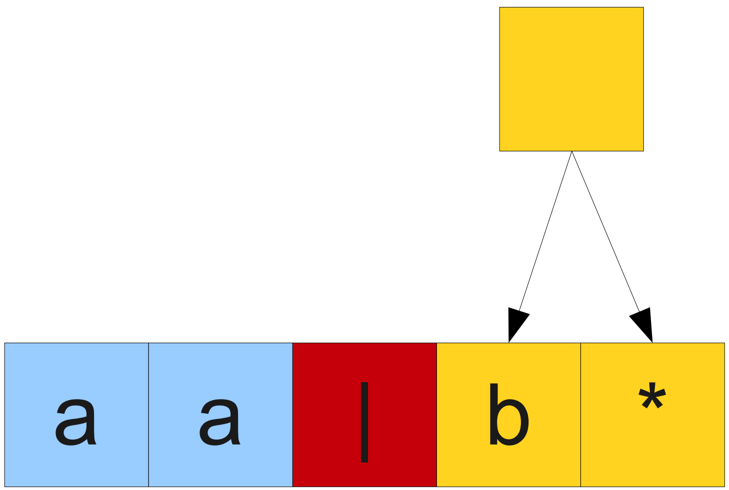
  R → Ø

  R → RR

  R → R “|” R

  R → R*

  R → (R)



Resolving Ambiguity

- Ambiguous grammars can often be rewritten to be unambiguous.
- Return to our CFG for regular expressions:

R → a | b | c | …

R → "ε"

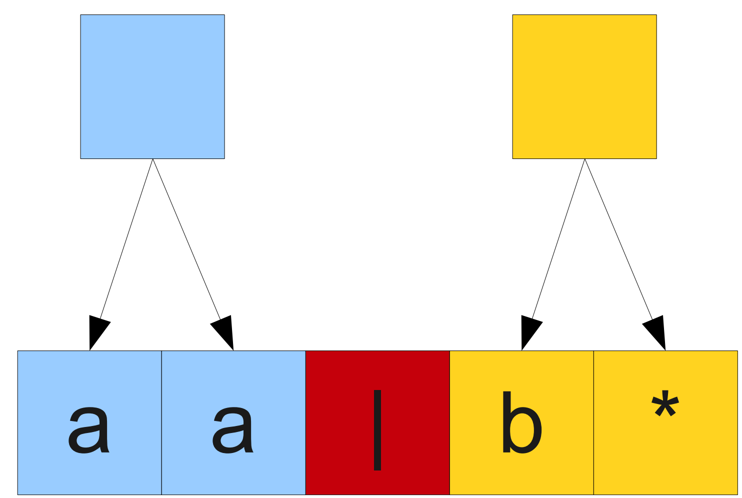
R → Ø

R → RR

R → R "|" R

R → R*

R → (R)



Resolving Ambiguity

- Ambiguous grammars can often be rewritten to be unambiguous.
- Return to our CFG for regular expressions:

  R → a | b | c | …

  R → "ε"

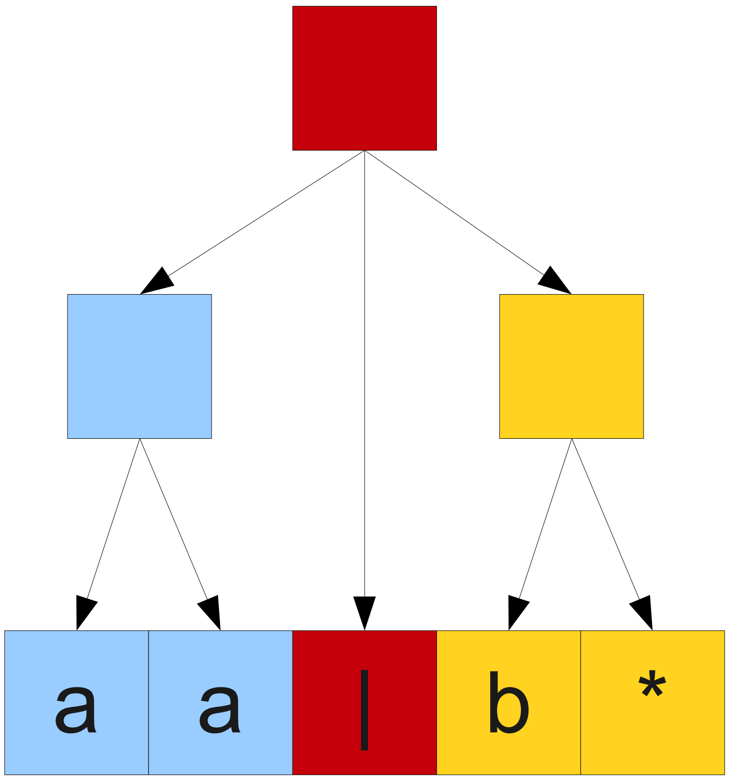
  R → Ø

  R → RR

  R → R "|" R

  R → R*

  R → (R)



Resolving Ambiguity

- Ambiguous grammars can often be rewritten to be unambiguous.
- Return to our CFG for regular expressions:

R → a | b | c | …

R → "ε"

R → Ø

R → RR

R → R "|" R

R → R*

R → (R)

R → S | R "|" S

S → T | ST

T → U | T*

U → a | b | c | …

U → "ε"

U → Ø

U → (R)

Why is this unambiguous?

R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)

Why is this unambiguous?

R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)

Only generates "atomic" expressions

Why is this unambiguous?

R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)

Puts stars onto atomic expressions

Only generates "atomic" expressions

Why is this unambiguous?

R → S | R "|" S

S → T | ST

T → U | T*

U → a | b | …

U → "ε"

U → Ø

U → (R)

Concatenates starred expressions

Puts stars onto atomic expressions

Only generates "atomic" expressions

Why is this unambiguous?

R → S | R "|" S

S → T | ST

T → U | T*

U → a | b | …

U → "ε"

U → Ø

U → (R)

Unions concatenated expressions

Concatenates starred expressions

Puts stars onto atomic expressions

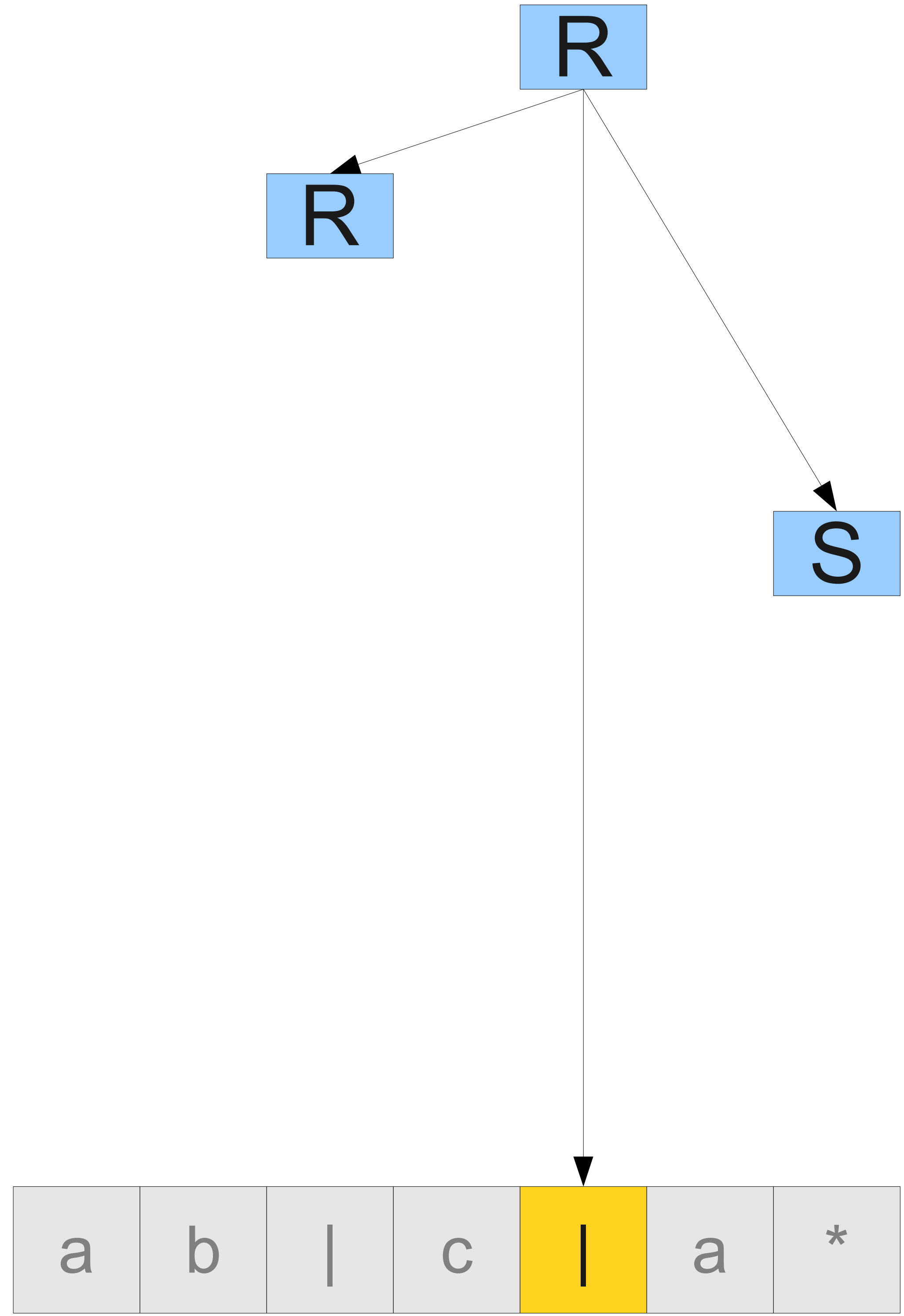
Only generates "atomic" expressions

R

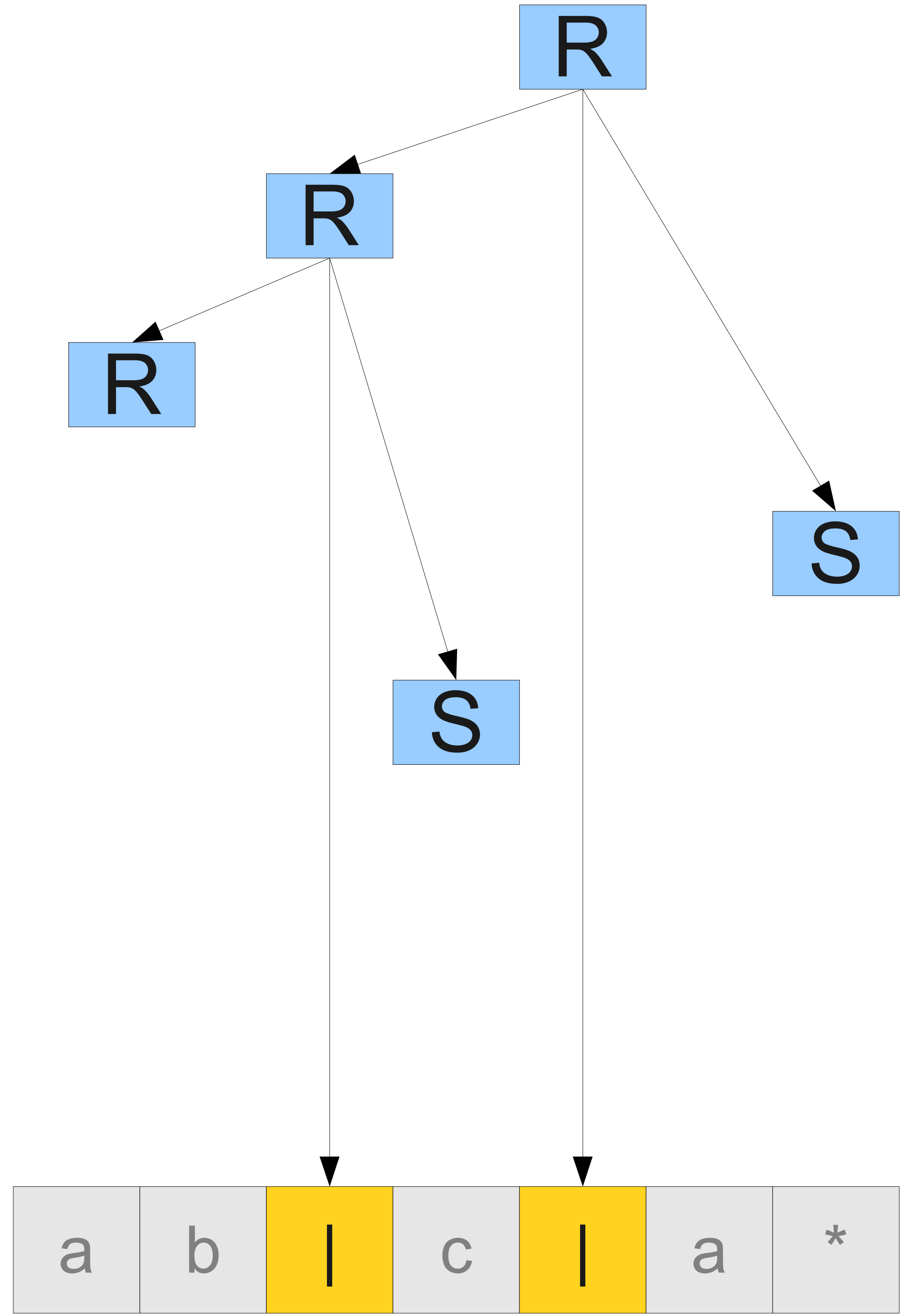
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)

a	b	|	c	|	a	*

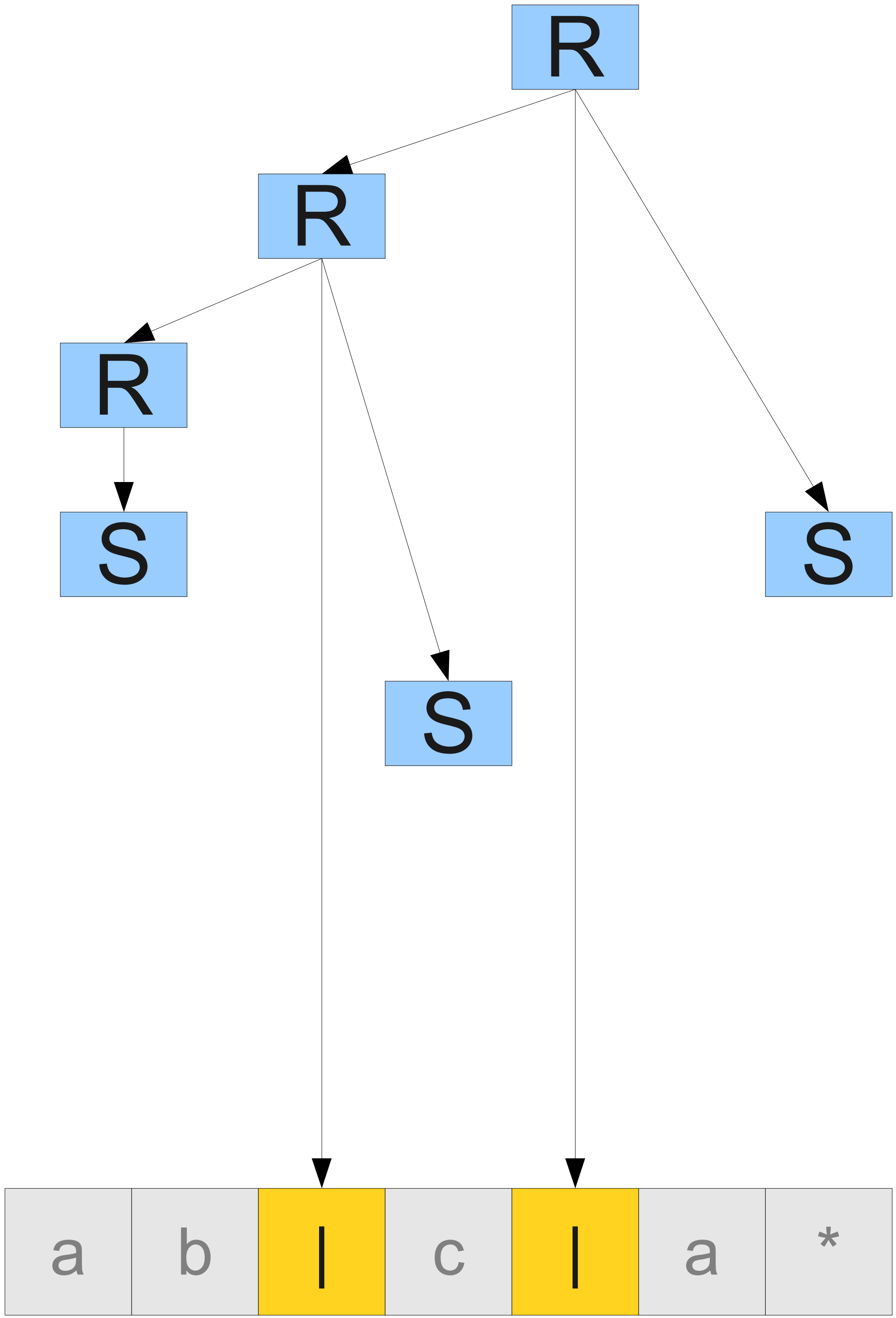
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



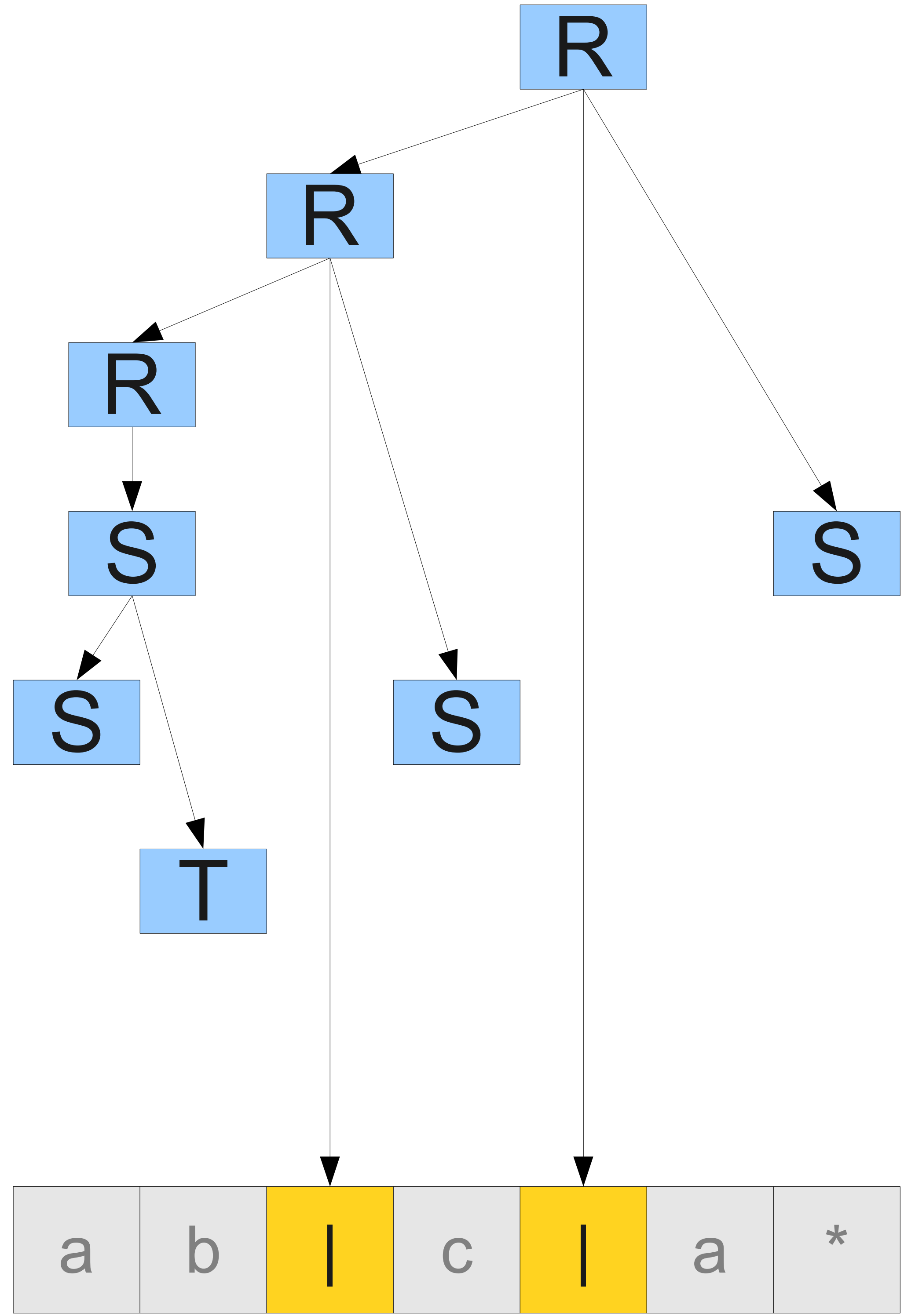
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



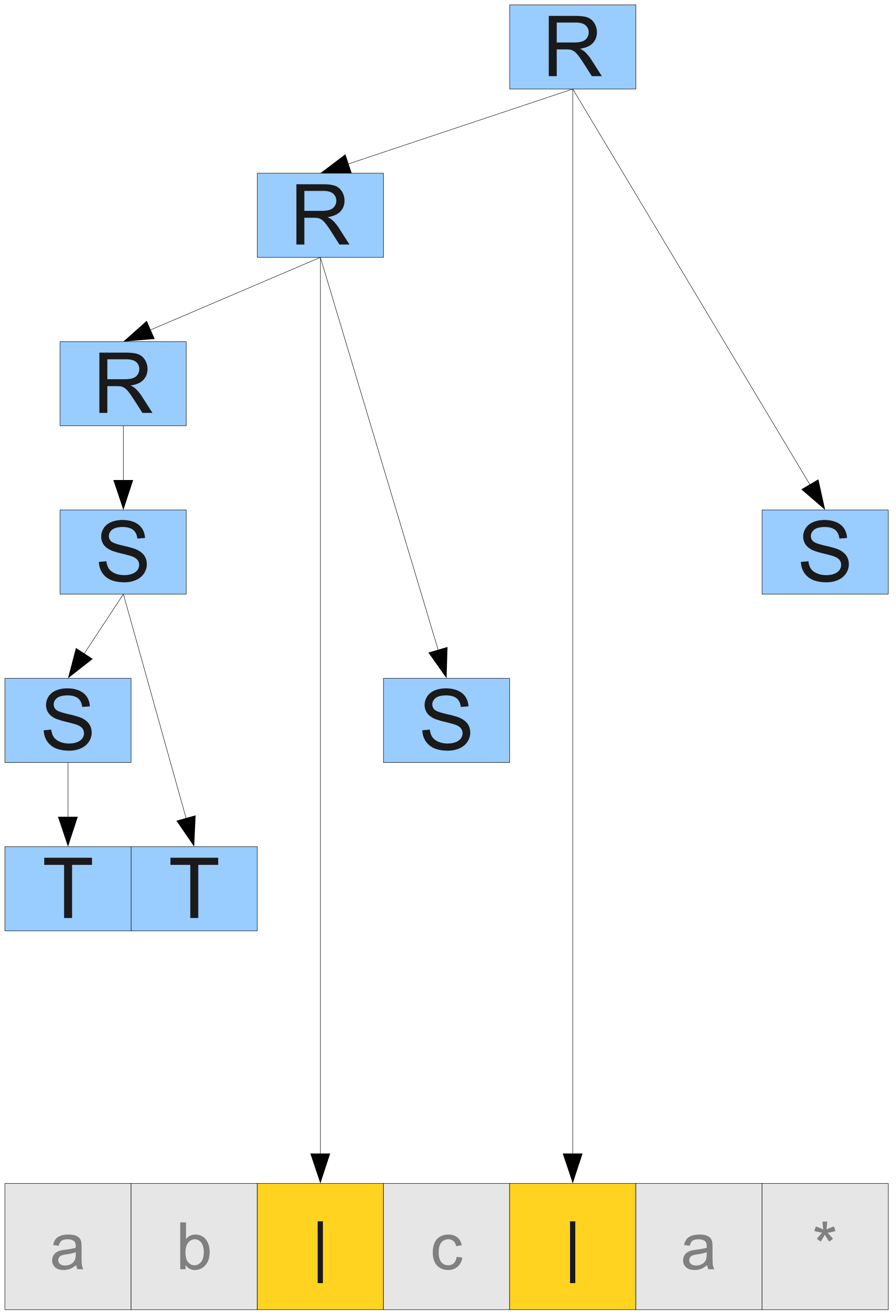
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



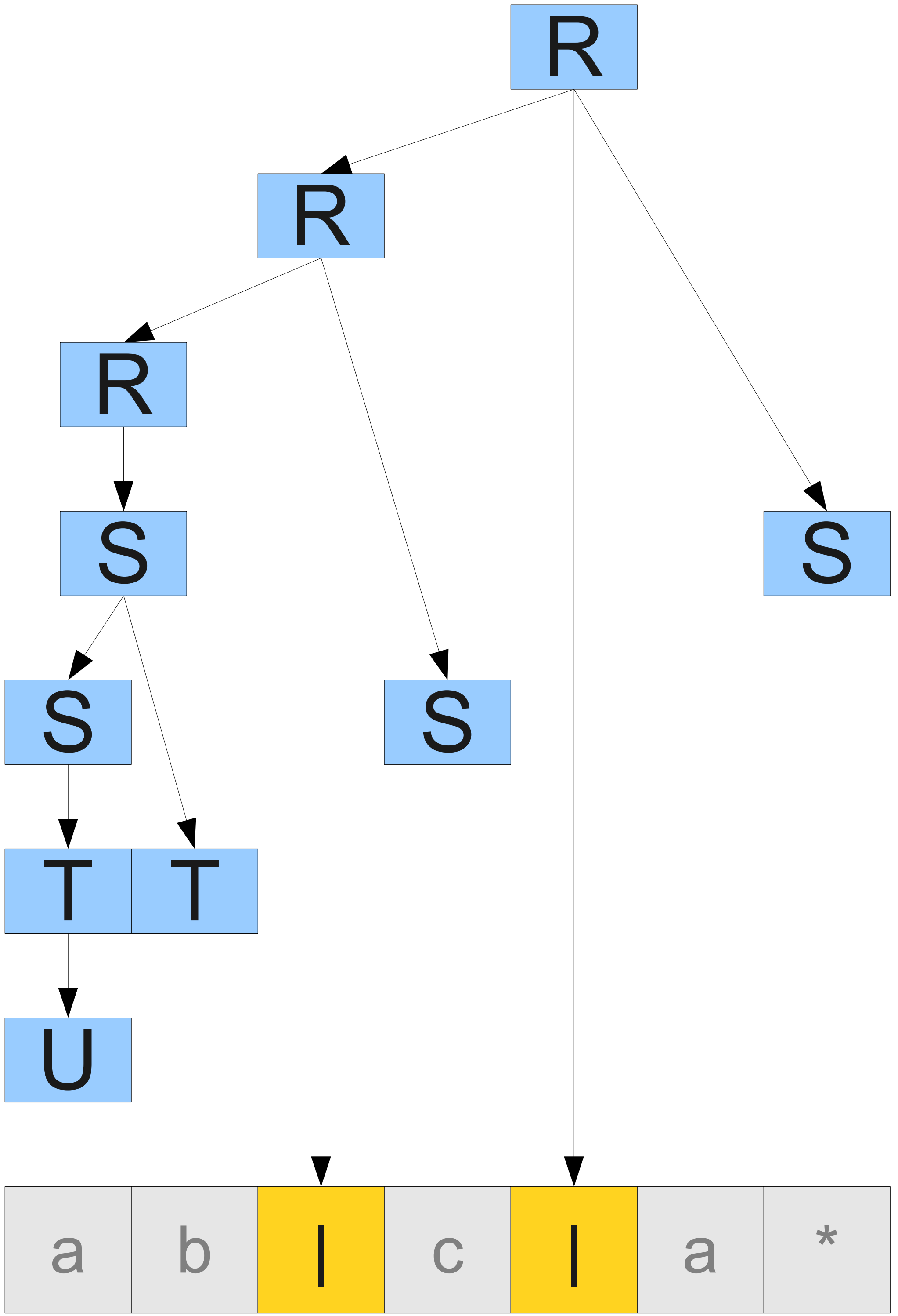
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



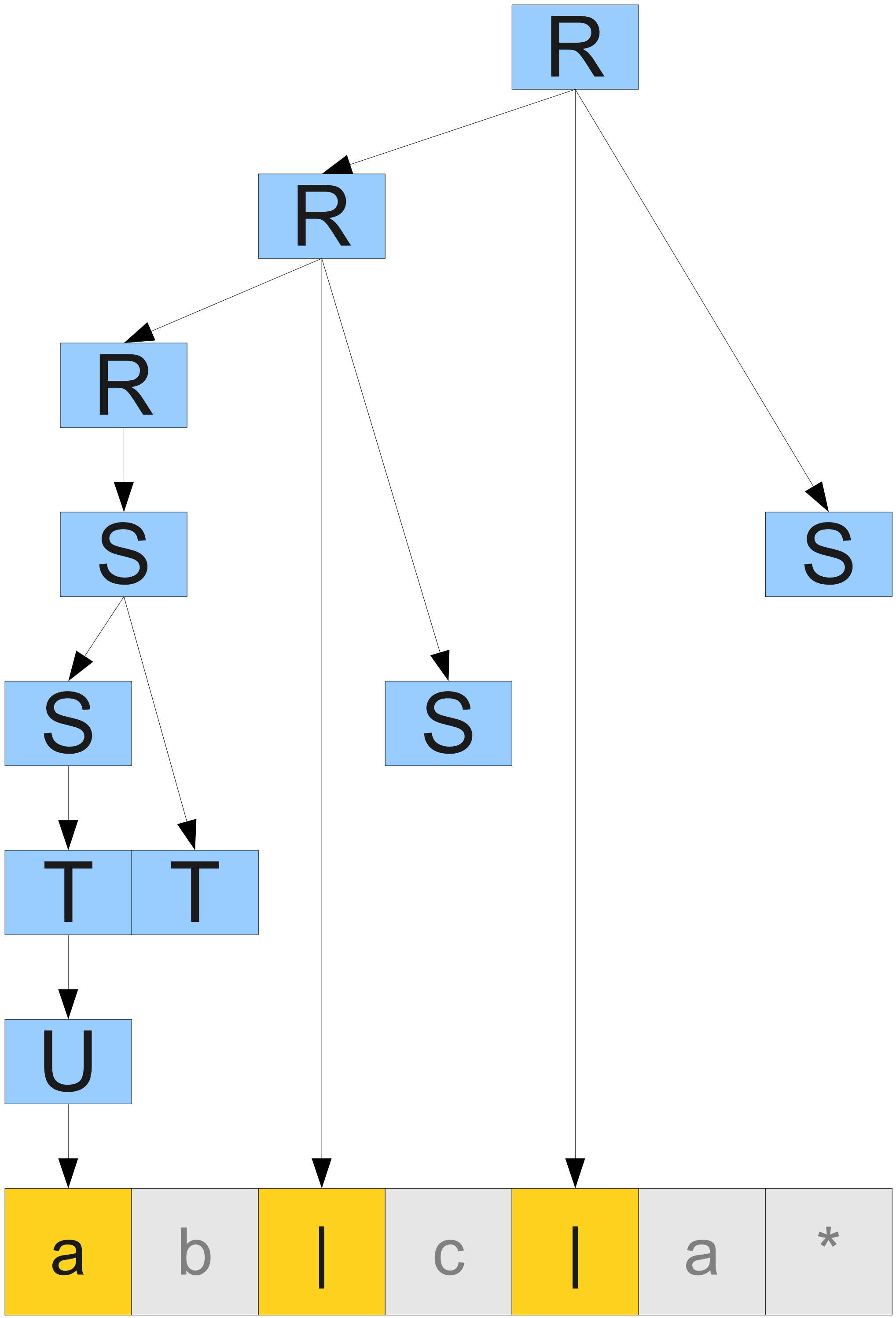
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



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

$S \rightarrow T \mid ST$

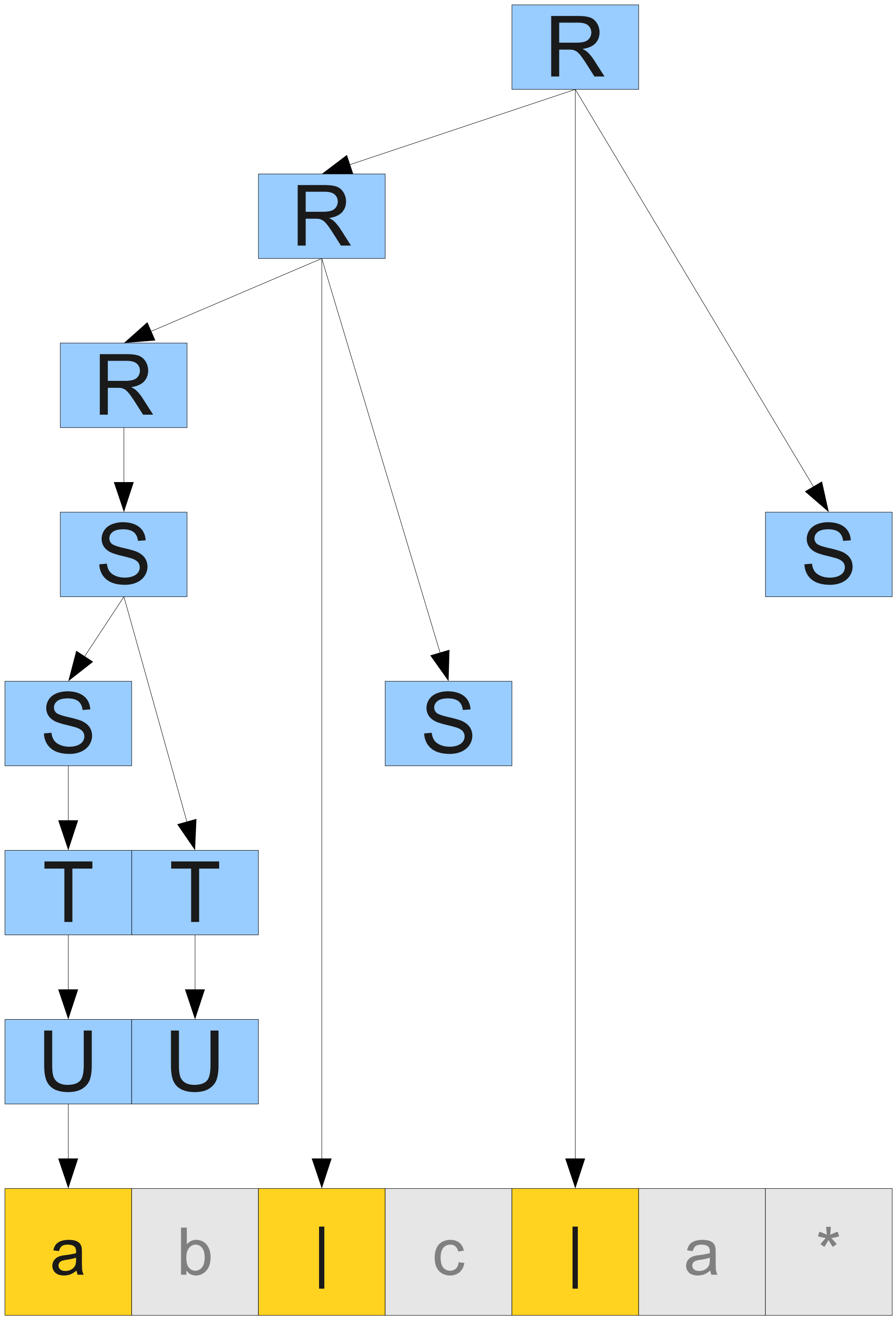
$T \rightarrow U \mid T^*$

$U \rightarrow a \mid b \mid \ldots$

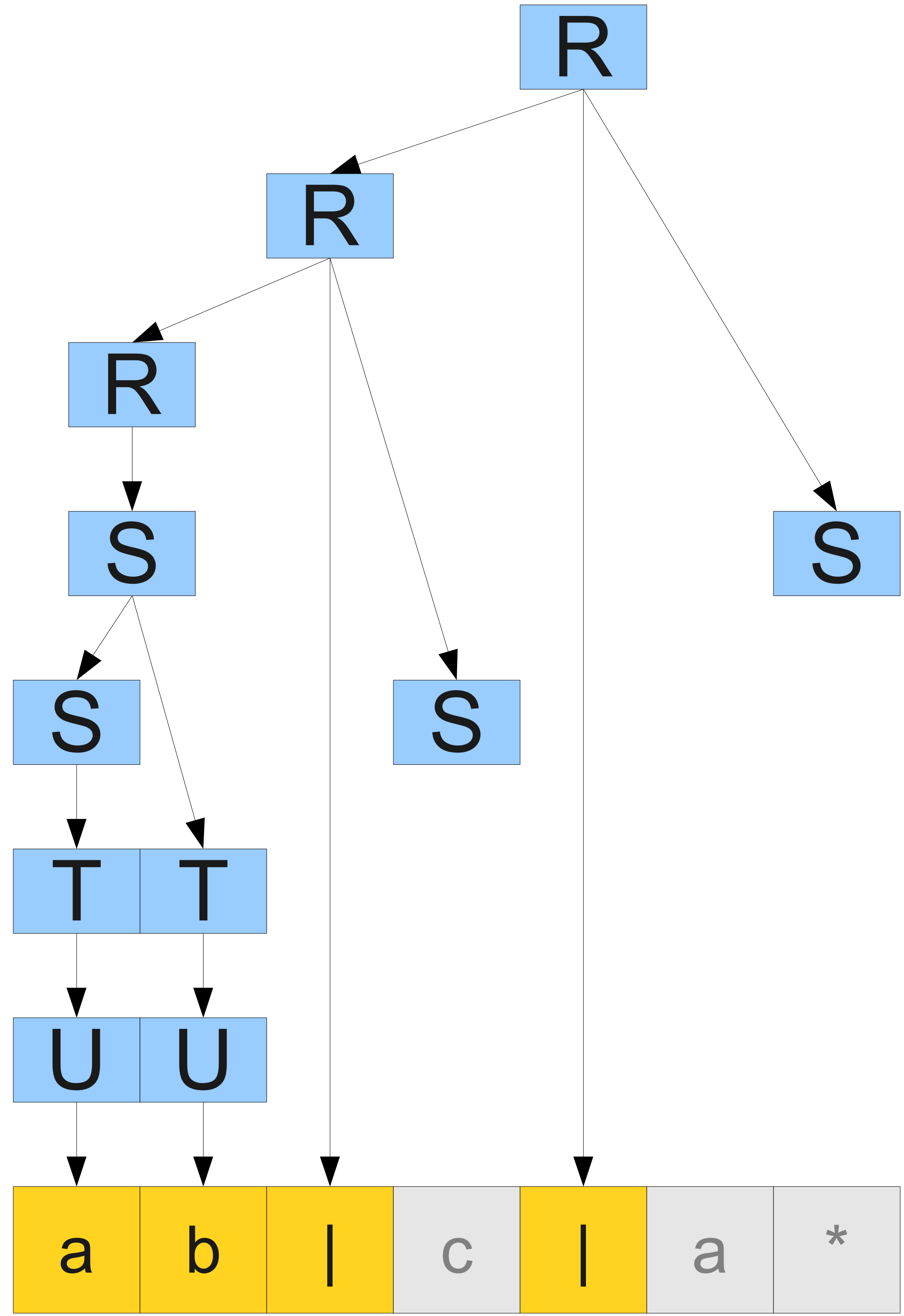
$U \rightarrow \text{"}\varepsilon\text{"}$

$U \rightarrow \varnothing$

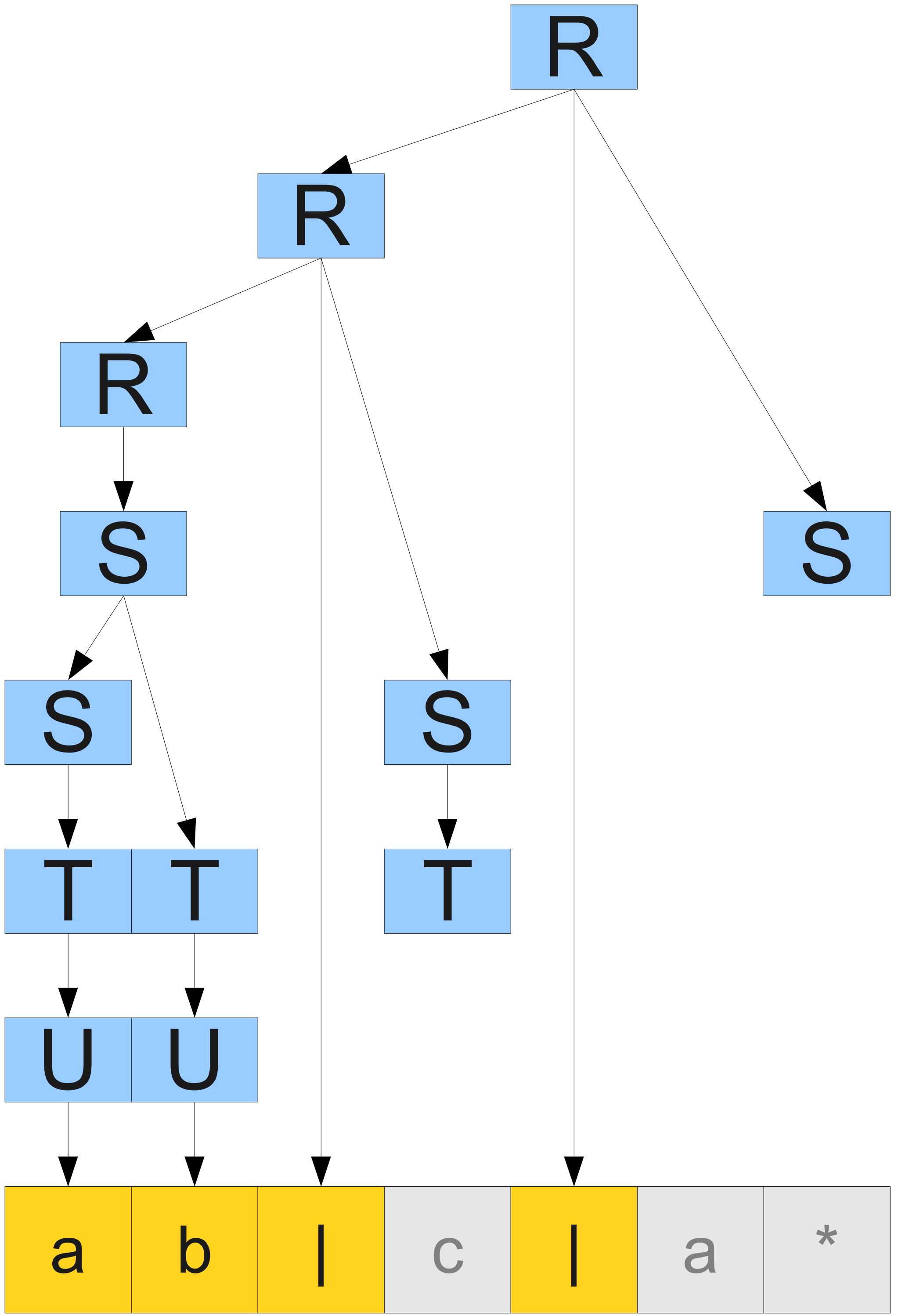
$U \rightarrow (R)$



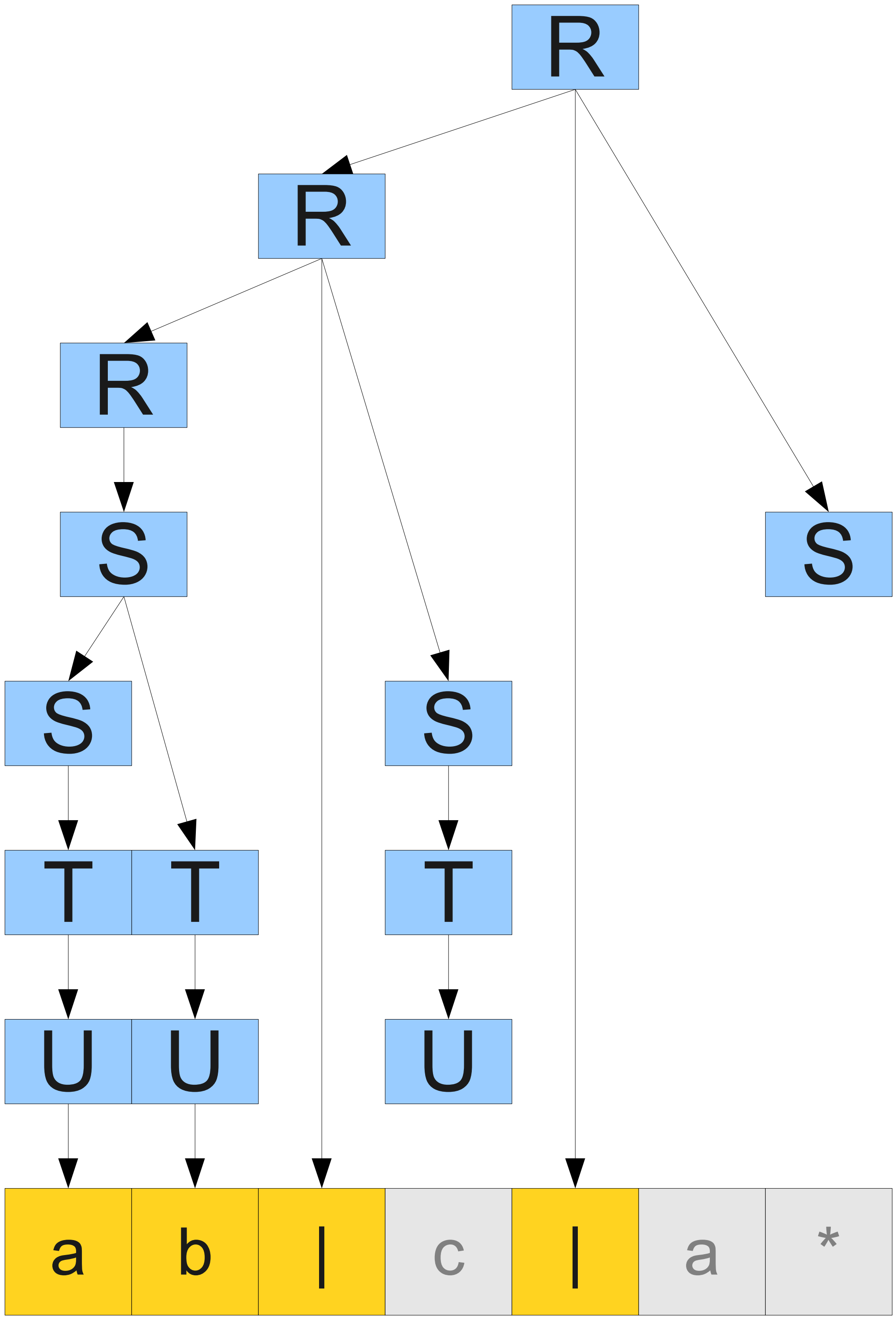
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



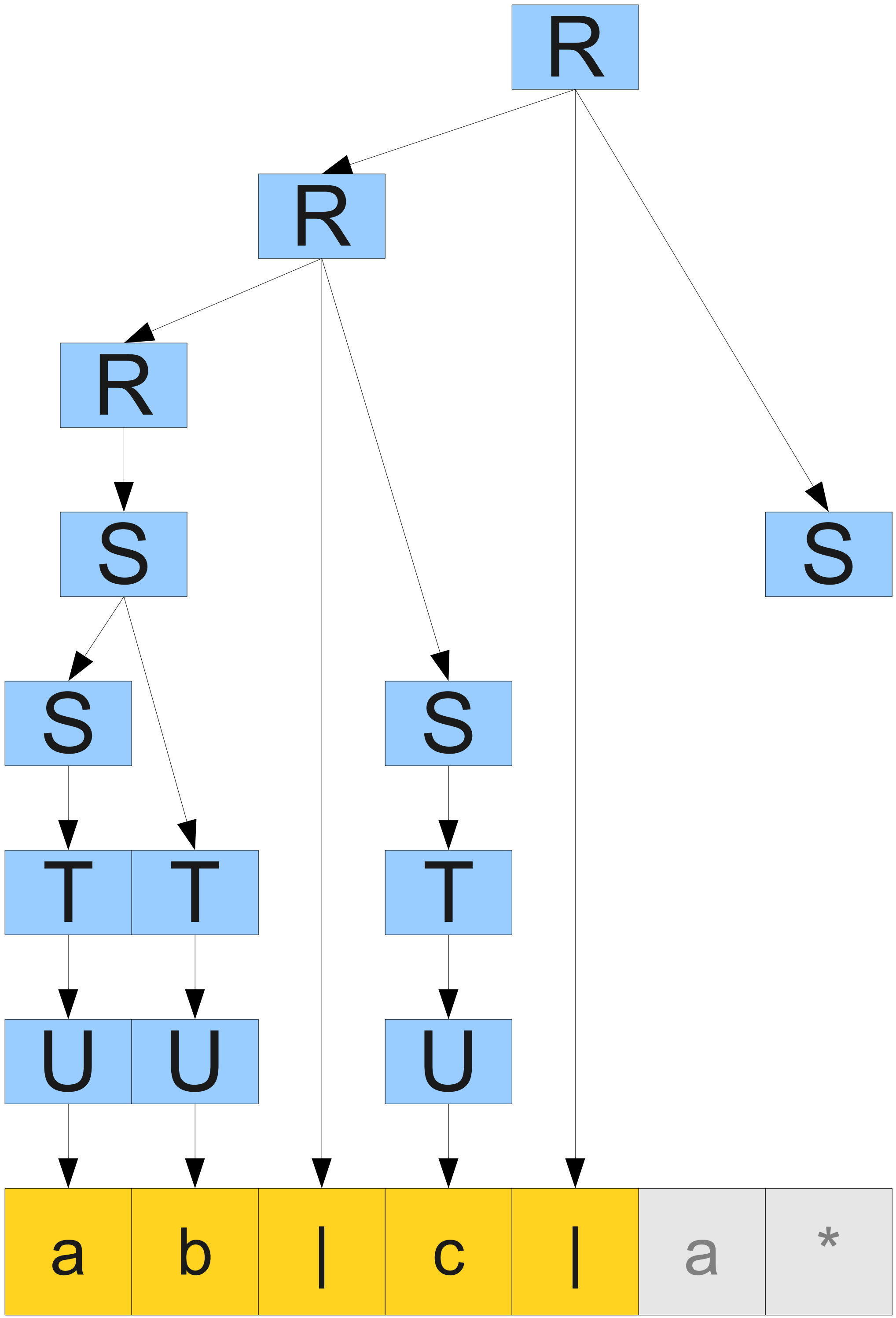
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



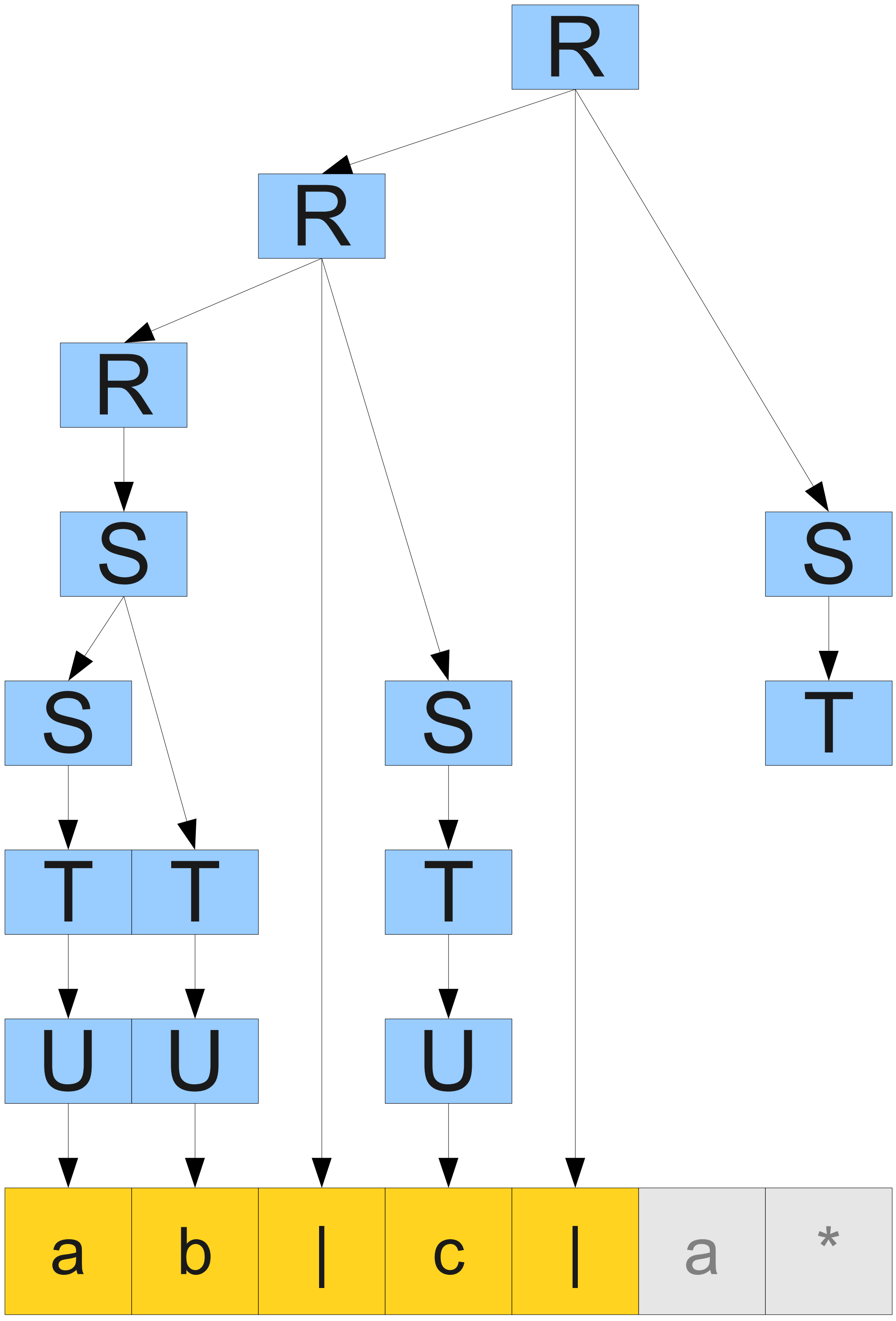
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



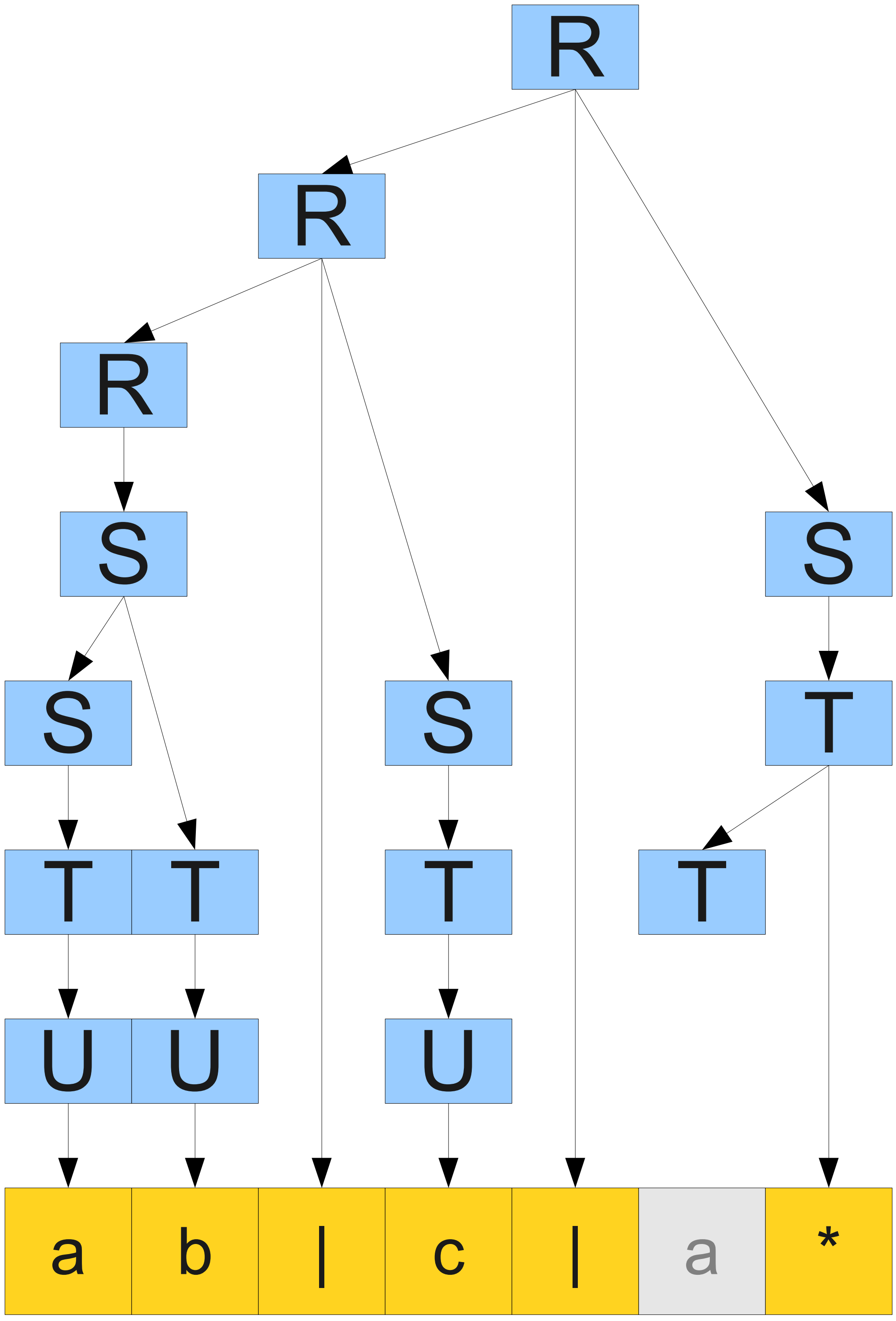
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



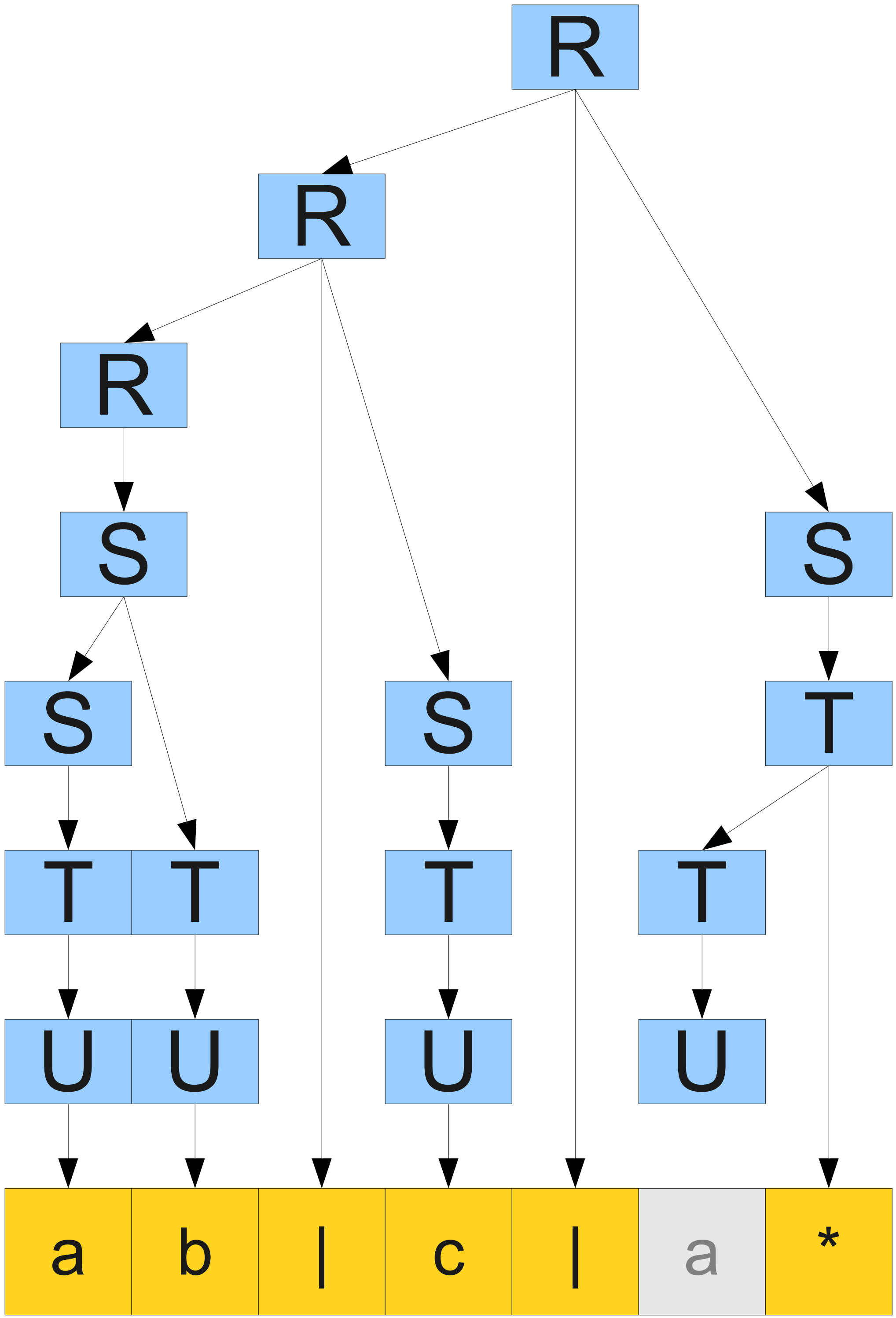
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



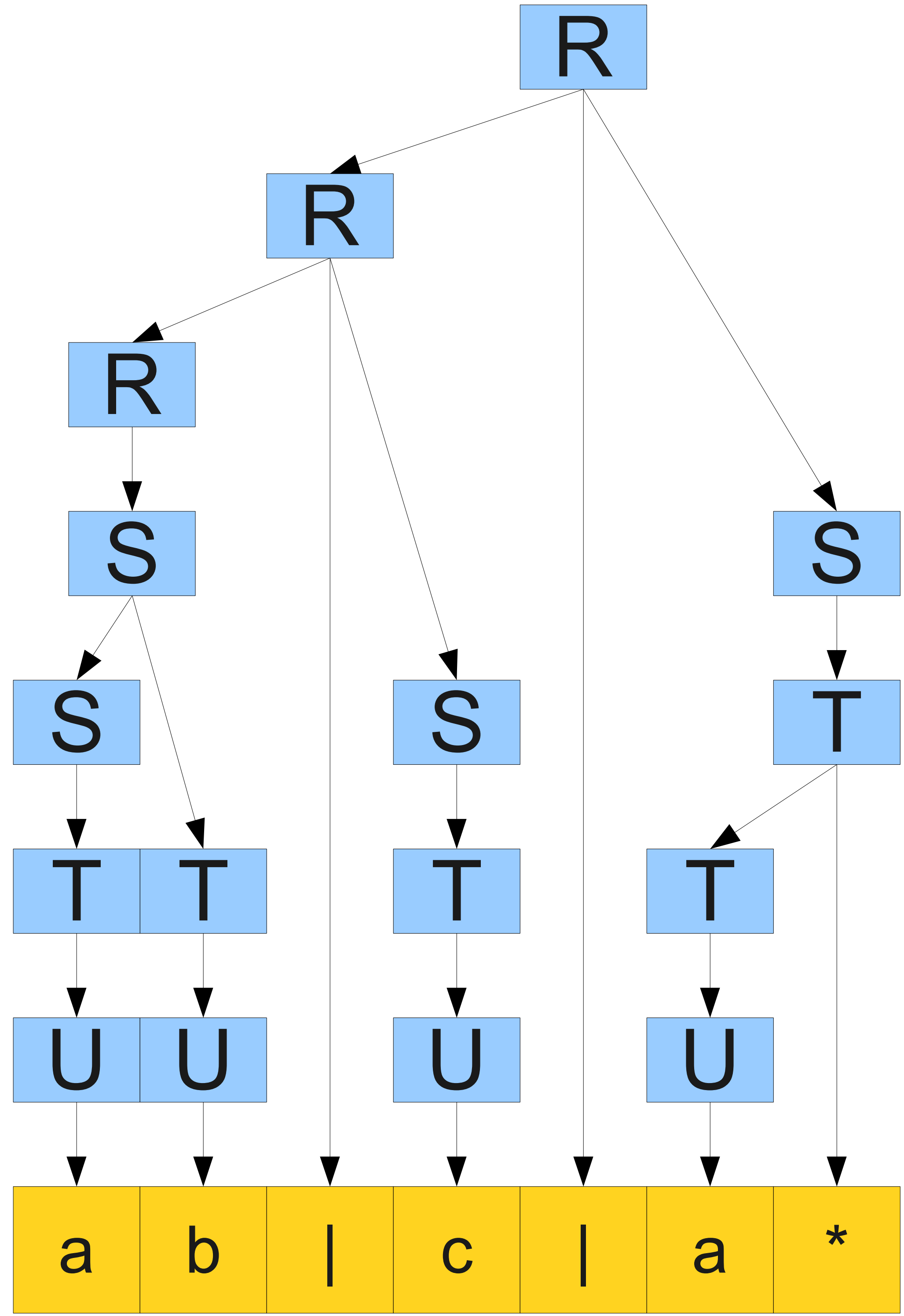
R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



R → S | R "|" S
S → T | ST
T → U | T*
U → a | b | …
U → "ε"
U → Ø
U → (R)



Closure Properties of Context-Free Languages

Closure Properties

- If L_1 and L_2 are regular, then
  - $L_1 \cup L_2$ is regular.
  - $L_1 \cap L_2$ is regular.
  - $L_1 – L_2$ is regular.
  - L_1L_2 is regular.
  - L_1^* is regular.
  - $h(L_1)$ is regular.
  - $\overline{L_1}$ is regular.
- How many of these properties still hold for context-free languages?

The Union of CFLs

- Suppose that L_1 and L_2 are **context-free** languages.
- Is $L_1 \cup L_2$ a context-free language?

The Union of CFLs

The Union of CFLs

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

The Union of CFLs

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

R → a | “ε” | Ø
R → RR | R “|” R | R* | (R)

The Union of CFLs

S → SENTENCE | R

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

R → a | "ε" | Ø
R → RR | R "|" R | R* | (R)

The Union of CFLs

- Suppose that L_1 and L_2 are **context-free** languages.
- Is $L_1 \cup L_2$ a context-free language?
- **Yes!**
- Let $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ be CFGs for L_1 and L_2, respectively, with V_1 and V_2 having no overlap.
- Then $G_{1 \cup 2} = (V', \Sigma, R', S)$, where
  - $S \notin V_1$, $S \notin V_2$
  - $V' = V_1 \cup V_2 \cup \{S\}$
  - $R' = R_1 \cup R_2 \cup \{ S \to S_1, S \to S_2 \}$

The Concatenation of CFLs

The Concatenation of CFLs

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

The Concatenation of CFLs

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

R → a | "ε" | Ø
R → RR | R "|" R | R* | (R)

The Concatenation of CFLs

S → SENTENCE R

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

R → a | "ε" | Ø
R → RR | R "|" R | R* | (R)

The Kleene Closure of CFLs

The Kleene Closure of CFLs

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

The Kleene Closure of CFLs

S → SENTENCE S | ε

SENTENCE → LP NP
LP → Look DIR. LP | ε
DIR → up | down | into your soul | sharp | before you leap
NP → Now look at me. I'm on a NOUN.
NOUN → horse | velociraptor | whale | roll

Closure Properties of CFLs

- If L_1 and L_2 are context-free languages, then
  - $L_1 \cup L_2$ is context-free.
  - $L_1 L_2$ is context-free.
  - L_1^* is context-free.
  - $h(L_1)$ is context-free.
- However:
  - $L_1 \cap L_2$ is **not necessarily** context-free.
  - $\overline{L_1}$ is **not necessarily** context-free.
  - $L_1 - L_2$ is **not necessarily** context-free.

Next Time

- **Pushdown Automata**
  - Automata for recognizing CFLs.
  - A beautiful generalization of DFAs and NFAs.
  - An easy proof that any regular language is context-free.