CS301: Theory of Computation
Section 1.3 - Regular Expressions

Daniel White

Gettysburg College
Computer Science

Spring 2024

We can use the regular operations to build up expressions describing languages, which are called **regular expressions**.

$$\text{e.g. } (0 \cup 1)0^*$$

The value of this expression is the language consisting of all strings starting with a 0 or 1, followed by any number of 0s. Note:

- 0 and 1 are shorthand for the sets $\{0\}$ and $\{1\}$, respectively.
- The concatenation symbol $\circ$ is typically left implicit.

exercise: Let $\Sigma = \{0, 1\}$. Describe the languages given by $(0 \cup \varepsilon)1^*$ and $\Sigma^*0110\Sigma^*$.

exercise: Further, define $R^+ = RR^*$. Give a regular expression whose value is $\{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$.

Definition 1.52 (pg 64)

Say that R is a **regular expression** if R is

1. a for some a in the alphabet Σ,
2. ε (the language containing exactly the empty string),
3. $\varnothing$ (the empty language),
4. $R_1 \cup R_2$, where R_1 and R_2 are regular expressions,
5. $R_1 \circ R_2$, where R_1 and R_2 are regular expressions, or
6. R_1^*, where R_1 is a regular expression.

homework: 1.22

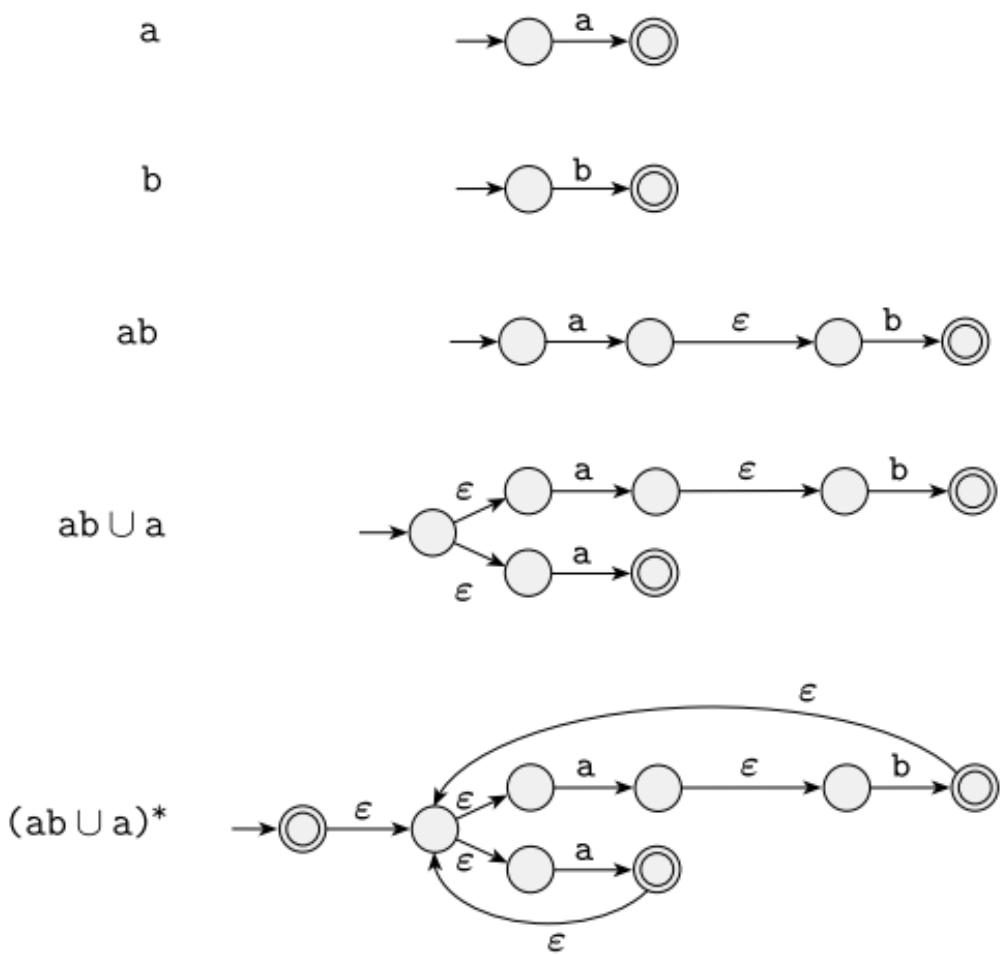
Theorem 1.54 (pg. 66)

A language is regular if and only if some regular expression describes it.

Proof. We first prove that if a language is described by a regular expression, then it is regular. Since the class of regular languages is closed under the union, concatenation, and star operations *(see 4-6 in the definition of regex)*, it will suffice to build NFAs recognizing the languages described by 1-3 in the definition of regex . . .

exercise: Complete this direction of the proof.

homework: 1.7f


a
a
b
b
ab
a
ε
b
ab ∪ a
ε
a
ε
b
ε
a
(ab ∪ a)*
ε
ε
a
ε
b
ε
ε
a
ε

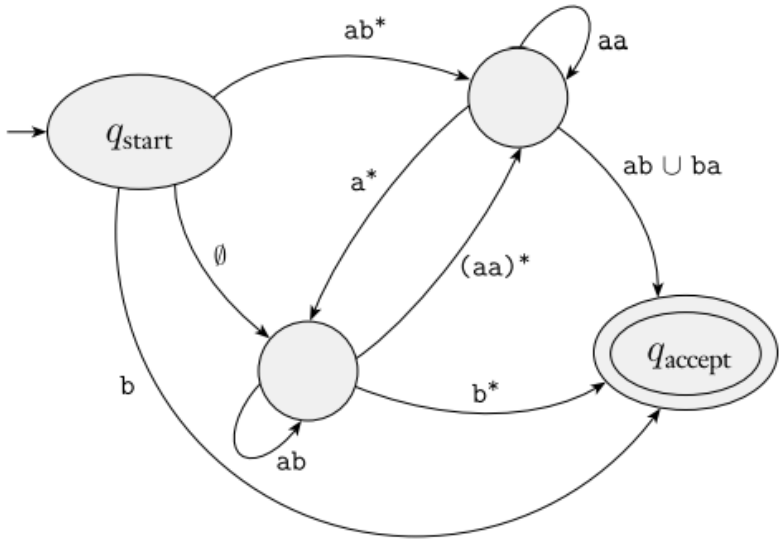
Theorem 1.54 (pg. 66)

A language is regular if and only if some regular expression describes it.

We now need to demonstrate that every regular language is described by some regular expression.

To do so, we'll first broaden the concept of an automaton similar to how we introduced NFAs in order to prove the closure theorems for regular languages.

Generalized nondeterministic finite automata (GNFA) are NFAs wherein the transition arrows may have any regular expressions as labels, instead of only members of the alphabet or ε.



exercise: Describe how G_1, the machine above, processes the input abbab.

For convenience, we require that GNFAs always be expressed in a special form that meets the following conditions:

1. the start state has transition arrows to every other state, but no incoming transition arrows,
2. the accept state is unique, with transition arrows from every other state, but no outgoing transition arrows, and
3. all other states have arrows to and from every other state, except for arrows from the accept state.

exercise: Does any given GNFA has an equivalent GNFA meeting the above conditions? Think about this briefly. We'll need some ideas on the next slide.

Lemma (pg. 71)

Each DFA is equivalent to some GNFA in the special form.

Proof. Given some DFA, apply the following changes.

1. Add new start state with an ε-transition to the old start state and new accept state with ε-transitions from old accept states.
2. If any transitions have multiple labels, we replace each with a single transition whose label is the union of the previous labels.
3. Finally, add transitions labeled with $\varnothing$ between states that had no transition arrows.

exercise: Let $A = \{w \mid w \text{ does not contain substring } \mathtt{aa}\}$ where $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Create DFA recognizing A, then convert to GNFA in special form.

Theorem (pgs. 71-74)

Each DFA is equivalent to some GNFA in special form with two states. In other words, every regular language is described by a regular expression.

Proof. It will suffice to prove that any GNFA, G, in special form with $k > 2$ states is equivalent to some GNFA in special form with $k - 1$ states. We do so by selecting a state of G, ripping it out of the machine, and repairing the remainder so that the same language is recognized. Let q_{rip} be any state of G that isn't the start or accept state ...

exercise: Finish this proof. *(pg. 72)*

homework: example 1.68