Proofs with Regular Expressions

Alyssa Lytle

Fall 2025

Recall

In general, we also discussed how some patterns are redundant, and therefore we can reduce our "set" of notations for patterns to be:

- Atomic patterns: $\{a, \boldsymbol{\epsilon}, \boldsymbol{\emptyset}\}$
- Compound patterns: $\{\cup, \circ, *\}$

Another way to say this is that every pattern α can be expressed in the following form. (Where β and γ are patterns.)

- $a \in \Sigma$
- $\boldsymbol{\epsilon}$
- $\boldsymbol{\emptyset}$
- $\beta \cup \gamma$
- $\beta \circ \gamma$
- β^*

Which lead us to the inductive definition of regular expressions.

Definition 1

The set of regular expressions can be defined *inductively* using atomic patterns and operators.

R is a regular expression if R is

1. $a \in \Sigma$
2. $\boldsymbol{\epsilon}$
3. $\boldsymbol{\emptyset}$
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
6. R_1^* where R_1 is a regular expression

Theorem 1

Let $A \subseteq \Sigma^*$. The following three statements are equivalent:

1. A is a regular language
2. $A = L(\alpha)$ for some pattern α
3. $A = L(\alpha)$ for some regular expression α

1 Proving Theorem 1

Since there are three equivalent statements, to prove this theorem, do you have to prove every combination of statements?

Not exactly! You can prove them in a "chain" if you will.

So we are going to prove $3 \to 2$, $2 \to 1$, and $1 \to 3$!

- $3 \to 2$: If $A = L(\alpha)$ for some regular expression α, then $A = L(\alpha)$ for some pattern α
- $2 \to 1$: If $A = L(\alpha)$ for some pattern α, then A is a regular language
- $1 \to 3$: If A is a regular language, then $A = L(\alpha)$ for some regular expression α

1.1 Proving $3 \to 2$

$3 \to 2$ is trivial because every regular expression *is* a pattern!

1.2 Proving $2 \to 1$

Want to Prove: If $A = L(\alpha)$ for some pattern α, then A is a regular language.

We can do this with a proof by induction!

We can use the (non redundant) atomic patterns for our base case a, $\boldsymbol{\epsilon}$, and $\boldsymbol{\emptyset}$

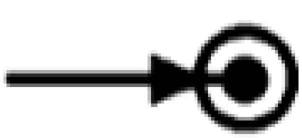
and the (non redundant) compound patterns $\cup$, $\circ$, and $*$

Base Cases: $\{\{a\}, \boldsymbol{\epsilon}, \boldsymbol{\emptyset}\}$

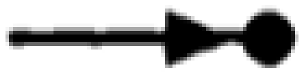
For each of these, we can just define an automaton that accepts each one!

- $a, a \in \Sigma$

- $\boldsymbol{\epsilon}$



- $\emptyset$



Inductive Step: $\{\cup, \circ, *\}$

Let our statement hold for smaller patterns β and γ. (Our inductive hypothesis)

More specifically, we use the IH that if $B = L(\beta)$ and $C = L(\gamma)$ for some pattern α, then B and C are regular languages.

Since every pattern α can be expressed in the following way:

- $\beta \cup \gamma$
- $\beta \circ \gamma$
- β^*

So, we can show that it holds for α by showing that it holds for $\beta \cup \gamma$, $\beta \circ \gamma$, and β^*. (Our WTP.)

More specifically, our WTP is in three parts:

- $L(\beta \cup \gamma)$ is a regular language
- $L(\beta \circ \gamma)$ is a regular language
- $L(\beta^*)$ is a regular language

As you can see this is a *proof by cases.* For the sake of time, we will not prove each of these, but for each case, you will do a *proof by construction.*

Let's prove the first case: $L(\beta \cup \gamma)$ is a regular language.

1. $B = L(\beta)$ and $C = L(\gamma)$ are regular languages (Inductive Hypothesis)
2. $L(\beta \cup \gamma) = L(\beta) \cup L(\gamma)$ (Compound Pattern Definition)
3. $L(\beta) \cup L(\gamma)$ is a regular language. (Needs to be proved!)
4. $L(\beta \cup \gamma)$ is a regular language. (Plugged in equivalence from line 2.)

As you can see, we need to prove if B and C are regular languages $B \cup C$ is a regular language. This would be your proof by construction. We already did a proof like this for $B \cap C$!

2 Proving $1 \rightarrow 3$

Want to Prove: If A is a regular language, then $A = L(\alpha)$ for some regular expression α

This is essentially saying each language accepted by a finite automaton can be represented by an equivalent regular expression.

So, we can prove this by converting a general finite automaton N to an equivalent regular expression! (A proof by construction!)