CSC236 Week 06: Automata and Languages

Hisbaan Noorani

October 14 – October 20, 2021

Contents

1 $L = \{x \in \{0,1\}^* \,|x$ begins and ends with a different bit$\}$ **1**

2 RE identities **2**

3 Turnstile finite-state machine **2**

4 Float machine **3**

5 States needed to classify a string **3**

6 Integer multiples of 3 **3**

7 Build an automaton with formalities **4**

1 $L = \{x \in \{0,1\}^* \,|x$ begins and ends with a different bit$\}$

The language $L'((0\,(0+1)^*\,1) \cup (1\,(1+0)^*\,0))$ should be the same language as the one listed above.

If we want to prove this, we can show that $\forall x \in L, x \in L' \wedge \forall x \in L', x \in L$. This is the same as showing that $L \subseteq L' \wedge L' \subseteq L$, which is equivalent to $L = L'$, what we are trying to show.

Proof: First we show that $L' \subseteq L$: Let $x \in L'$. Then either $x = 1y0$ where $y \in \{0,1\}^*$ or $x = 0w1$, where $w \in \{1,0\}^*$. Without loss of generality, assume $x = 1y0$, otherwise just replace 1 with 0, 0 with 1, and y with w.

Then $1 \in L(1), 0 \in L(0),$ and $y \in L(0+1)^*$, since it is the concatenation of 0 or more strings from $L(0+1)$. So $x \in L(1)L(0+1)^*L(0)$, so it begins with 1 and ends with 0, which are different, so $x \in L$.

Next we show that $L \subseteq L$: this is left as an exercise to the reader

So since we have shown that $L \subseteq L' \wedge L' \subseteq L$, $L = L'$. ■

2 RE identities

Some of these follow from set properties, others require some proof

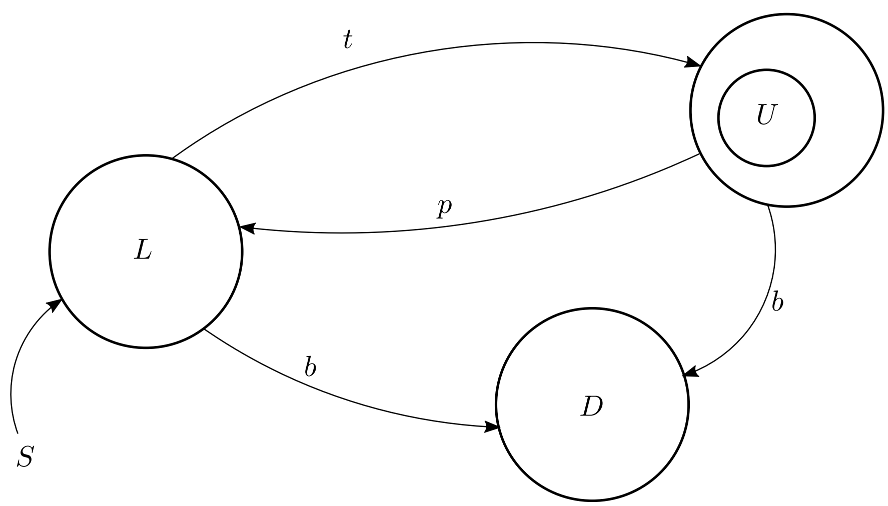
- Commutativity of union: $R + S \equiv S + R$
- Associativity of union: $(R + S) + T \equiv R + (S + T)$
- Associativity of concatenation: $(RS)T \equiv R(ST)$
- Left distributivity: $R(S + T) \equiv RS + RT$
- Right distributivity: $(S + T)R \equiv SR + ST$
- Identity for union: $R + \emptyset \equiv R$
- Identity for concatenation: $R\varepsilon \equiv R \equiv \varepsilon R$
- Annihilator for concatenation: $\emptyset R \equiv \emptyset \equiv R\emptyset$
- Idempotence of Kleene star: $(R^*)^* \equiv R^*$

3 Turnstile finite-state machine

Let out alphabet be $\{t, p, b\}$, where p denotes push, t denotes tap or token, and b denotes bicycle.

We will study three different states: $Q = \{U, L, D\}$. U denotes unlocked, L denotes locked, D stands for dead (or jammed or deactivated).

Σ^* is all strings over $\{t, p, b\} = \{t, p, b\}^*$



Is *tptppt* accepted? We start at L, go to U with the first t, go back to L with p and so on. When we have a p but we are at L, then nothing happens as pushing on a locked turnstile will do nothing. Similarly, when we have a t when we are at a U, then nothing will happen as using a tap/token on an unlocked turnstile will also do nothing. Following these rules, we arrive at the following:

$$L \xrightarrow{t} U \xrightarrow{p} L \xrightarrow{t} U \xrightarrow{p} L \xrightarrow{p} L \xrightarrow{t} U$$

And thus the combination is accepted.

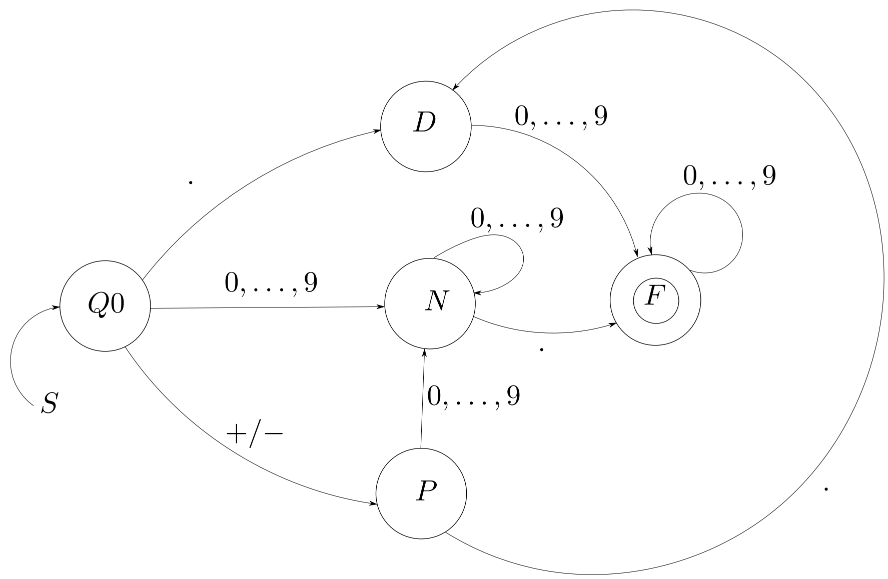
4 Float machine

Which strings are floats in Python?

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ., -, +\}$

Examples of accepted floats: $\{1.25, -0.3, .3, 125.\}$

Examples of rejected floats: $125, 1..2, 1.2.5, -., 1.25-$



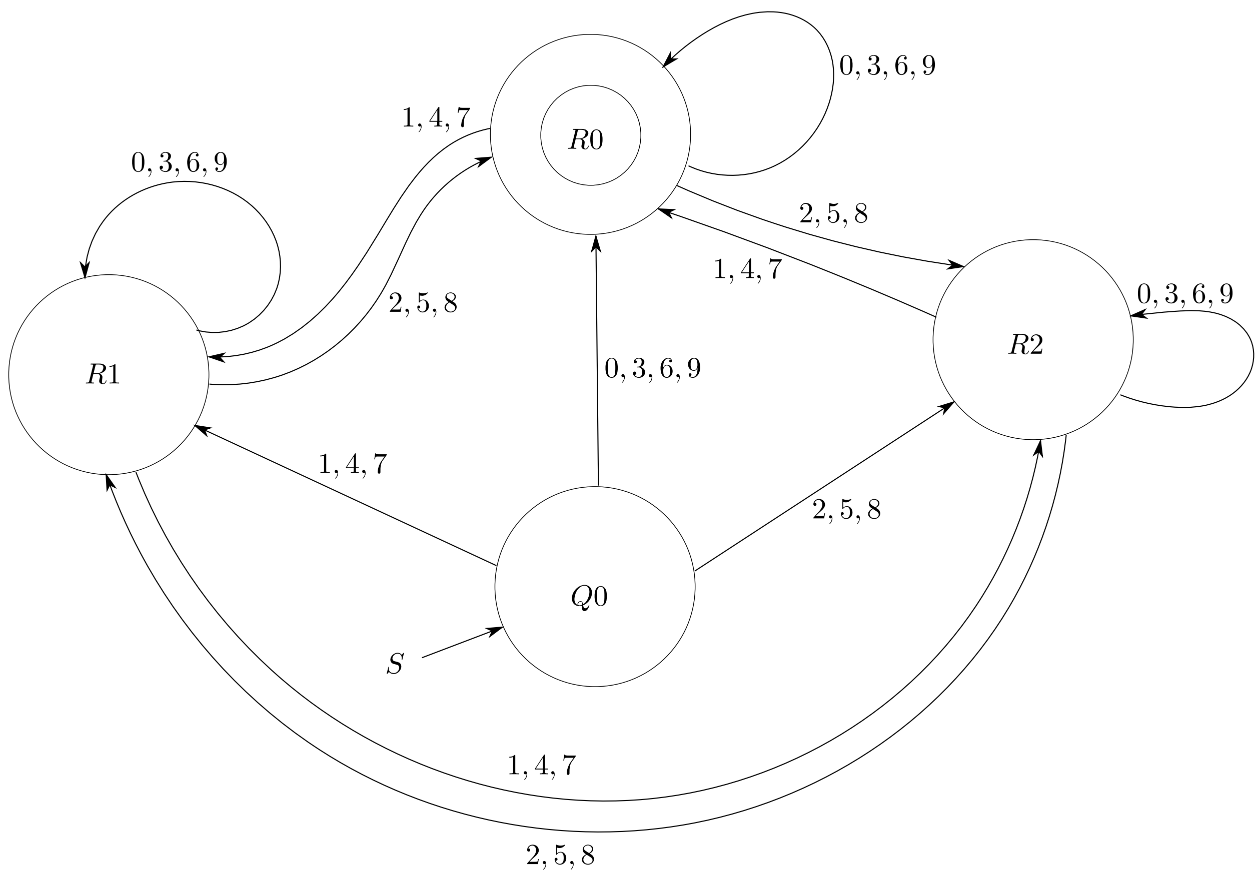
5 States needed to classify a string

What state is a stingy vending machine in, based on coins? Accepts only nickels, dimes, and quarters, no change given, and everything costs 30 cents.

δ	0	5	10	15	20	25	≥ 30
n	5	10	15	20	25	≥ 30	≥ 30
d	10	15	20	25	≥ 30	≥ 30	≥ 30
q	25	≥ 30	≥ 30	≥ 30	≥ 30	≥ 30	≥ 30

6 Integer multiples of 3

If an integer is divided by 3, it falls into one of three groups. Remainder 0, remainder 1, and remainder 2. Concatenating a number onto the end of another number can have interesting effects depicted in the diagram below.



7 Build an automaton with formalities

The idea motivating this is to be able to describe the complicated systems above without drawing out messy and time consuming diagrams.

Quintuple: $(Q, \Sigma, q_0, F, \delta)$

Q is a set of states, Σ is finite, non-empty alphabet, q_0 is the start state, F is the set of accepting states, and $\delta : Q \times \Sigma \mapsto Q$ is a transition function.

We can extend $\delta : Q \times \Sigma \mapsto Q$ to a transition function that tells us what state a string s takes the automaton to:

$$\delta^* : Q \times \Sigma^* \mapsto Q \qquad \delta^*(q, s) = \begin{cases} q & \text{if } s = \varepsilon \\ \delta(\delta^*(q, s'), a) & \text{if } s' \in \Sigma^*, a \in \Sigma, s = s'a \end{cases}$$

String s is accepted if and only if $\delta^*(q, s) \in F$, and it is rejected otherwise.

δ^* and δ are not in fact the same thing. δ takes a single valid character $\in \Sigma$, whereas δ^* takes any valid string of characters $\in \Sigma^*$. This is why $\delta : Q \times \Sigma \mapsto Q$ and $\delta^* : Q \times \Sigma^* \mapsto Q$.