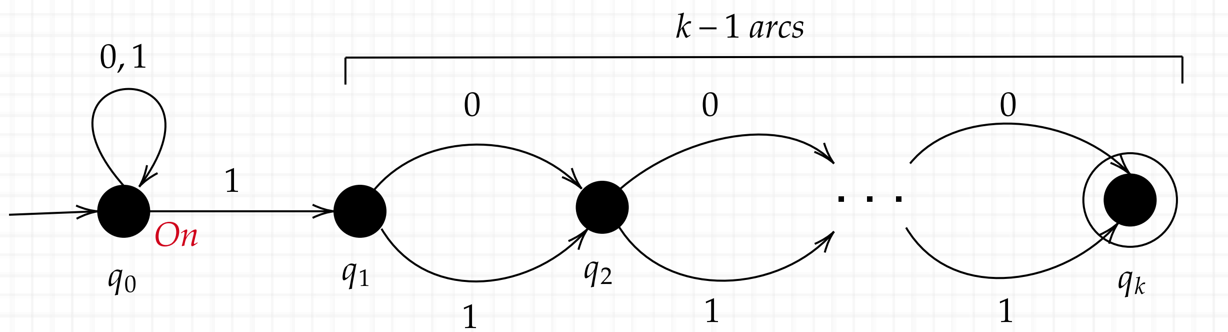
The Main Theorem About Regular Expressions and Finite Automata

Theorem: For any language A over an alphabet Σ, the following statements are equivalent:

1. There is a regular expression α such that $A = L(\alpha)$.
2. There is an NFA N such that $A = L(N)$.
3. There is a DFA M such that $A = L(M)$.

Example (moved up from last time): The "Leap of Faith" NFAs N_k for any $k > 1$:



Example: $x = 101101$ *On* *On* *On*

$$L(N_k) = (0+1)^*1(0+1)^{k-1}$$
$$= \left\{x \in \{0,1\}^* : \textit{the kth bit of } x \textit{ from the end is a } 1\right\}.$$

Fact (will be proved later): Whereas the NFA N_k has only $k+1$ states, the smallest DFA M_k such that $L(M_k) = L(N_k)$ requires 2^k states.
This is a case of **exponential blowup** in the NFA-to-DFA algorithm.
For now, we just care that an equivalent DFA **can** be built.

From NFA to DFA

Theorem (part two of Kleene's Theorem): Given any NFA $N = (Q, \Sigma, \delta, s, F)$ we can build a DFA $M = (\mathcal{Q}, \Sigma, \Delta, S, \mathcal{F})$ such that $L(M) = L(N)$.

Notice that s got capitalized to S, which hints that S is a *set* rather than a single element. And δ got capitalized to Δ. Q and F were already sets, but they got...curlier. What does that mean? Well, that they are "of an even higher order"---sets of sets, for instance. An important set of sets is:

$\mathcal{P}(Q)$, also written 2^Q, called the *power set* of Q and defined as $\{R : R \subseteq Q\}$.

Unlike what textbooks tend to say, we will not necessarily make $\mathbb{Q}$ be all of $\mathcal{P}(Q)$, just those subsets R that are *reachable* from S. What this means is that the states of the DFA will be sets of states of the NFA---the states that are *possible* upon *processing* a given part of the input string x.

This suggests the question, which states (of N) are possible before we process any chars in x? Obviously the start state s of N is possible, but are there any others? Yes, if there are ϵ-transitions out of s. Define $E(s)$ to be the set of states of N that are reachable this way. If N has no ϵ-arcs (out of s or overall), then $E(s)$ is just $\{s\}$. Thus we begin building M by taking $S = E(s)$. We could have said "S" in place of "$E(s)$" to begin with, but the notation is useful to define

$$E(R) = \{r : \textit{for some } q \in R,\ N \textit{ can process } \epsilon \textit{ from } q \textit{ to } r\}$$

for any subset R of states. This is called the *epsilon-closure* of R. If $E(R) = R$ then R is already *epsilon-closed*. It sounds "weeny" technical, but we will only need to use subsets that are ϵ-closed. The insight is that *the states of the DFA are the* ***possible*** *subsets of states of the NFA*.

To make the DFA equivalent to the NFA, at least in terms of the language it accepts, we need to build on the correspondence we started with s and S. Let $x \in \Sigma^*$ be some input of length n. For $i = 0, 1, \ldots, n-1, n$ the design goal $G(i)$ for M is to arrange that:

M upon reading $x_1 x_2 \cdots x_i$ is in the state $R_i = \{r : N \textit{ can process } x_1 x_2 \cdots x_i \textit{ from } s \textit{ to } r\}$.

Now when $i = 0$, the initial portion $x_1 x_2 \cdots x_i$ is ϵ (more "Zen" reasoning), so R_0 turns out to be just another name for $E(s)$. By setting $S = E(s)$, what we've done is achieve the property $G(0)$. We can now use this as the basis for an induction $G(i-1) \implies G(i)$ which we build Δ to achieve. This will give us the final property $G(n)$, which states:

M upon reading all of x is in the state $R_n = \{r : N \textit{ can process } x \textit{ from } s \textit{ to } r\}$.

Now N accepts x if and only if R_n includes at least one accepting state $f \in F$, i.e., $R_n \cap F \neq \emptyset$. Thus when we regard a possible subset R as a state of M, we should call it accepting if and only if $R \cap F \neq \emptyset$. Thus the property $G(n)$ will imply $x \in L(M) \iff x \in L(N)$, and getting this for all n and x of length n will yield the conclusion $L(M) = L(N)$. So thus far we have defined:

- $\mathbb{Q} = \{\textit{possible } R \subseteq Q\}$;
- $S = E(s)$;
- $\mathcal{F} = \{R \in \mathbb{Q} : R \cap F \neq \emptyset\}$.

And Σ is the same. The only component of M left to define is Δ. For any $P \in \mathbb{Q}$ and $c \in \Sigma$ define

$$\Delta(P, c) \;=\; \{r:\; for\ some\ p\ \in P,\ N\ can\ process\ c\ from\ p\ to\ r\}.$$

This set is automatically ϵ-closed, since $c \cdot \epsilon^* \;=\; c$ so any trailing ϵ-arcs can count as part of processing c. If we assume $G(i-1)$ as our induction hypothesis, take the set R_{i-1} which the property $G(i-1)$ refers to, and define $R_i \;=\; \Delta(R_{i-1}, x_i)$, then we only need to show that R_i has the property required for the conclusion $G(i)$. This is that R_i equals the set of states that N can process the bits $x_1 \cdots x_i$ to. The core of the proof is finally to observe that:

> N can process $x_1 x_2 \cdots x_{i-1} x_i$ *from s to r* if and only if there is a state p such that N can process $x_1 x_2 \cdots x_{i-1}$ from s to p (which by IH $G(i-1)$ includes p into R_{i-1}) and such that N can process the char x_i from p to r.

How does this finish the proof? Let's see... We can make a small change to the definition of $\Delta(P, c)$ that makes it quicker and less error-prone to calculate M from N, by a process that examples will view as an instance of *breadth-first search.*

Example

We make a slight change to the heart of the proof where we left off. The change saves some time in executing the NFA-to-DFA construction when ϵ-arcs are present and reduces errors. First define

$$\underline{\delta}(p, c) \;=\; E(\{q:\; (p, c, q) \;\in\; \delta\})$$

for any state $p \;\in\; Q$ and char c. Recall $E(\cdot)$ is ϵ-closure. So what this means in simple terms is:

1. **First** take arc(s) on c out of the state p.
   - If there are none, **stop** and put $\underline{\delta}(p, c) \;=\; \emptyset$.
   - Else collect all states q reached on those arc(s).
2. **Then,** for each state q reached by processing c, add states reached on any series of ϵ-arcs out of q, if there are any.

Now we can give a new definition of the DFA's transition function Δ: for any $P \;\subseteq\; Q$ and $c \;\in\; \Sigma$,

$$\Delta(P, c) \;=\; \bigcup_{p \in P} \underline{\delta}(p, c)\ .$$

The difference is that we avoid worrying about initial ϵ-arcs that could come before processingf c. We only have to track *trailing* ones in a machine diagram. The reason is that the trailing arcs at the previous step already took care of any initial ones now. Initializing the start state S of the DFA M to have all states reached by ϵ-arcs out of s in N sets this in motion. We need to prove for all i:

$$G(i):\; \Delta^*(S, x_1 \cdots x_i) \;=\; \{r:\; N\ can\ process\ x_1 \cdots x_i\ from\ s\ to\ r\}.$$

Here we have *extended* Δ, a function of a state and a char, to Δ^* which is a function of a state and a *string*, by the basis $\Delta^*(R, \epsilon) = R$ for all $R \in \mathbb{Q}$ and for $i \geq 1$,

$$\Delta^*(R, x_1 \cdots x_{i-1} x_i) = \Delta\left(\Delta^*(R, x_1 \cdots x_{i-1}),\ x_i\right).$$

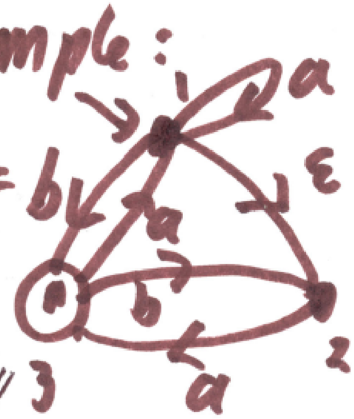
So let R_{i-1} stand for $\Delta^*(S, x_1 \cdots x_{i-1})$. Then by the inductive hypothesis $G(i-1)$, R_{i-1} equals the set of states q such that N can process $x_1 \cdots x_{i-1}$ from s to q. Now put $R_i = \Delta(R_{i-1}, x_i)$.

- Let $r \in R_i$. Then $r \in \underline{\delta}(q, x_i)$ for some $q \in R_{i-1}$. By IH $G(i-1)$, N can process $x_1 \cdots x_{i-1}$ from s to q. And N can process x_i from q to r by definition of $r \in \underline{\delta}(q, x_i)$. So N can process $x_1 \cdots x_i$ from s to r.
- Suppose N can process $x_1 \cdots x_i$ from s to r. Then---and this is the key point---the processing goes to some state q just before the char x_i is processed. By IH $G(i-1)$, q belongs to R_{i-1}. Moreover, $r \in \underline{\delta}(q, x_i)$ because we first do the step that processed the char x_i at q, then any trailing ϵ-arcs. Thus $r \in \Delta(R_{i-1}, x_i)$, which means $r \in R_i$.

Thus we have established that R_i equals the set of states r such that N can process $x_1 \cdots x_i$ from s to r. This is the statement $G(i)$, which is what we had to prove to make the induction go through. This finally proves the NFA-to-DFA part of Kleene's Theorem. ☒

Example:

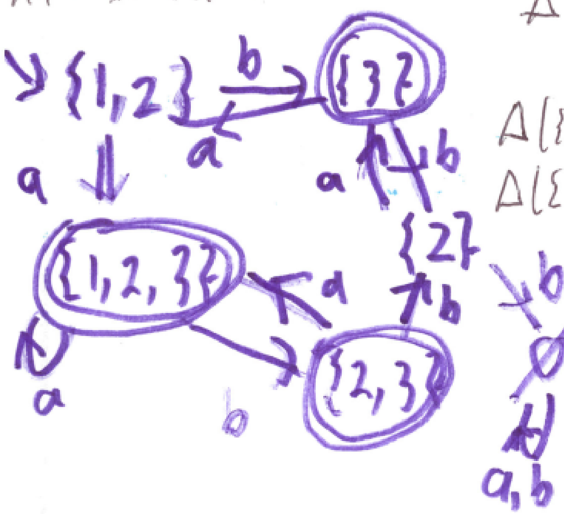
$N =$



$(1, \epsilon, 2)$ means "whenever 1, then also 2."

$S = \{1, \underline{2}\}$, not $\{1\}$

The DFA cannot have the states $\{1\}$ or $\{1,3\}$ because they have 1 but not 2.



In lecture I pointed out:

$\delta(p,c)$: find c then ϵs. (2)

$\delta(1,a) = \{1,2\}$ $\delta(1,b) = \{3\}$

$\delta(2,a) = \{3\}$ $\delta(2,b) = \emptyset$

$\delta(3,a) = \{1,2\}$ $\delta(3,b) = \{2\}$

$\Delta(P,c) = \bigcup_{p \in P} \delta(p,c)$.

Use Breadth First Search from S.

$\Delta(S,a) = \delta(1,a) \cup \delta(2,a)$
$= \{1,2\} \cup \{3\} = \underbrace{\{1,2,3\}}_{\text{new state}}$

$\Delta(S,b) = \delta(1,b) \cup \delta(2,b)$
$= \{3\} \cup \emptyset = \underline{\{3\}}$ also a new state

$\Delta(\{1,2,3\},a) = \{1,2\} \cup \{3\} \cup \{1,2,3\} = \{1,2,3\}$

$\Delta(\{1,2,3\},b) = \{3\} \cup \emptyset \cup \{2\}$ again, not ne[w]
$= \{2,3\}$, which $\underline{\text{is}}$ new.

$\Delta(\{3\},a) = \delta(3,a) = \{1,2\}$

$\Delta(\{3\},b) = \delta(3,b) = \{2\}$ $\underline{\text{new}}$

$\Delta(\{2,3\},a) = \{3\} \cup \{1,2\} = \{1,2,3\}$

$\Delta(\{2,3\},b) = \emptyset \cup \{2\} = \{2\}$

$\Delta(\{2\},a) = \delta(2,a) = \{3\}$ $\Delta(\emptyset,a) = \emptyset$

$\Delta(\{2\},b) = \delta(2,b) = \emptyset$ $\Delta(\emptyset,b) = \emptyset$

No more new states: we say "The BFS has clos[ed]

The extra things pointed out have to do with how the states of the DFA tell what the NFA can and cannot process:

- The NFA cannot process the string bbb from its start state at all. However you try, you come to the NFA state 2 being unable to process a b. Nor can it process bbb from any other state.
- However, N can process a from start to any one of its three states:
  - $(1, a, 1)$
  - $(1, a, 1)(1, \epsilon, 2)$
  - $(1, \epsilon, 2)(2, a, 3)$.

    This is shown in the DFA by the single arc $(S, a, \{1,2,3\})$.

- But in the string $x = abbb$, even though the initial a "turns on all three lightbulbs of N", the final bbb still cannot be processed by N. The DFA M does process it via the computation $(S, a, \{1,2,3\})(\{1,2,3\}, b, \{2,3\})(\{2,3\}, b, \{2\})(\{2\}, b, \emptyset)$, but that computation ends at $\emptyset$, which---when present at all---is always a dead state.