CSE491/596 Lecture Wed. Sep. 9. **Kleene Star and NFA-to-DFA.**

$L(\gamma) = A \cdot B = \{xy : x \in A \wedge y \in B\}.$

Given $A \subseteq \Sigma^*$ (meaning: A is a *language*), define A^2 as shorthand for $A \cdot A$.
Does $A^2 = \{x^2 : x \in A\}$? Where $x^2 = x \cdot x$ for any string x. E.g. if $A = \{0, 01, 10\}$ then $\{x^2 : x \in A\} = \{00, 0101, 1010\}$. But $A^2 = \{00, 001, 010, 0101, 0110, 100, 1001, 1010\}$.
I.e., $A^2 = A \cdot A = \{x \cdot y : x, y \in A\}$, allowing y different from x. Note also $|A^2| = 8$ not 9.

Also A^3 abbreviates $A \cdot A \cdot A$, $A^4 = A \cdot A^3$, etc. Also A^1 just equals A. But what about A^0?
We want languages to obey the same law of powers that numbers do: $a^i \cdot a^j = a^{i+j}$. The special case $i = 0$ needs $a^0 \cdot a^j = a^{0+j} = a^j$ for all j, so a^0 must equal 1. Well, the analogue of 1 for languages is $\{\epsilon\}$, which gives $\{\epsilon\} \cdot B = B \cdot \{\epsilon\} = B$ for all languages B. So we want $A^0 = \{\epsilon\}$.
The tricky thing is that this goes even for $A = \emptyset$: $\emptyset^0 = \{\epsilon\}$, not $\emptyset$! Why should this be?
I wrote one story about it at https://rjlipton.wordpress.com/2015/02/23/the-right-stuff-of-emptiness/
In abstract math, B^A denotes the set of functions $f : A \to B$, and by rule, $|B^A| = |B|^{|A|}$ which is just a power of numbers. So 0^0 equals the cardinality of the set of functions from $\emptyset$ to $\emptyset$. Now:

The empty function $\emptyset$ is a function from $\emptyset$ to $\emptyset$.

And it is the only function from $\emptyset$ to $\emptyset$. This is "Zen" but real. So $0^0 = 1$. Since $\{\epsilon\}$ is like 1, this can be argued to justify $\emptyset^0 = \{\epsilon\}$. But I will try a third way to make it intuitive. First, let's finally get around to defining the (*Kleene*) *star* operation, named for Stephen Kleene (1909--1994):

$$A^* = \bigcup_{i=0}^{\infty} A^i = \{\epsilon\} \cup A \cup A^2 \cup A^3 \cup \cdots$$

It is the set of all strings formed by concatenating zero or more strings from A. Now here is the intuition for why "concatenating zero strings from A" yields ϵ, i.e., why A^* always includes ϵ even when $A = \emptyset$.

Suppose we've designed a security system for a building that periodically runs a status check, say if it detects the possibility of there being an intruder or some other breakdown. The system gets feedback for the check from various cameras and sensors and monitors. Let A be the language of strings representing internal audits of sensory data that pass the status check. Since the check can run multiple times, we can picture it being inside an event-driven `while` loop. Then A^* is the language of inputs that will pass every check, no matter how many times the check is activated. So, finally, what happens if:

- $A = \emptyset$, meaning we are sure to **fail** the check if it is activated; **but**
- the check is never activated---the while loop runs **0** times and falls through!

The upshot is that the system **passes**, with the **empty string** of sensor data. Because it is a *pass*, not a *fail*, the language of inputs that pass "every" check (of 0 checks) is $\{\epsilon\}$, not $\varnothing$. So $\varnothing^0 = \{\epsilon\}$.

Now back to our recursive construction of regular expressions and NFAs corresponding to them. This proves one part of a theorem discovered by Kleene in the 1950s.

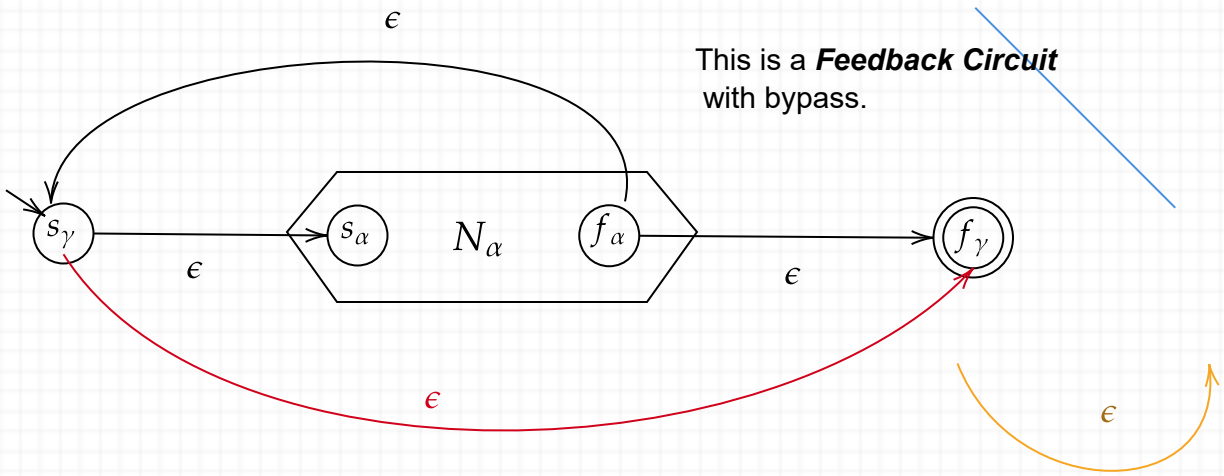
Theorem: For any language A over an alphabet Σ, the following statements are equivalent:

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 $A = L(M)$.

We are in the middle of proving 1 $\implies$ 2. Next will be 2 $\implies$ 3. Then 3 $\implies$ 1 would "complete the cycle of equivalence" but in fact we will use something more general than an NFA to go to 1.

(I3) Given any regexp α, $\gamma = \alpha^*$ is a regexp; $L(\gamma) = L(\alpha)^*$; and we can build:

$N_\gamma =$



Is this good? We want to make $L(N_\gamma) = L(N_\alpha)^*$. Then the IH $L(N_\alpha) = L(\alpha)$ will give $L(N_\gamma) = L(\alpha)^* = L(\alpha^*) = L(\gamma)$ as needed---to finish the whole proof.

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 $M = (\mathbb{Q}, \Sigma, \Delta, S, \mathcal{F})$ 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:

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

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:

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

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:

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

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

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

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.

[Lecture will end by reinforcing how this finishes the proof. Friday will begin by reviewing the proof with 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*.]