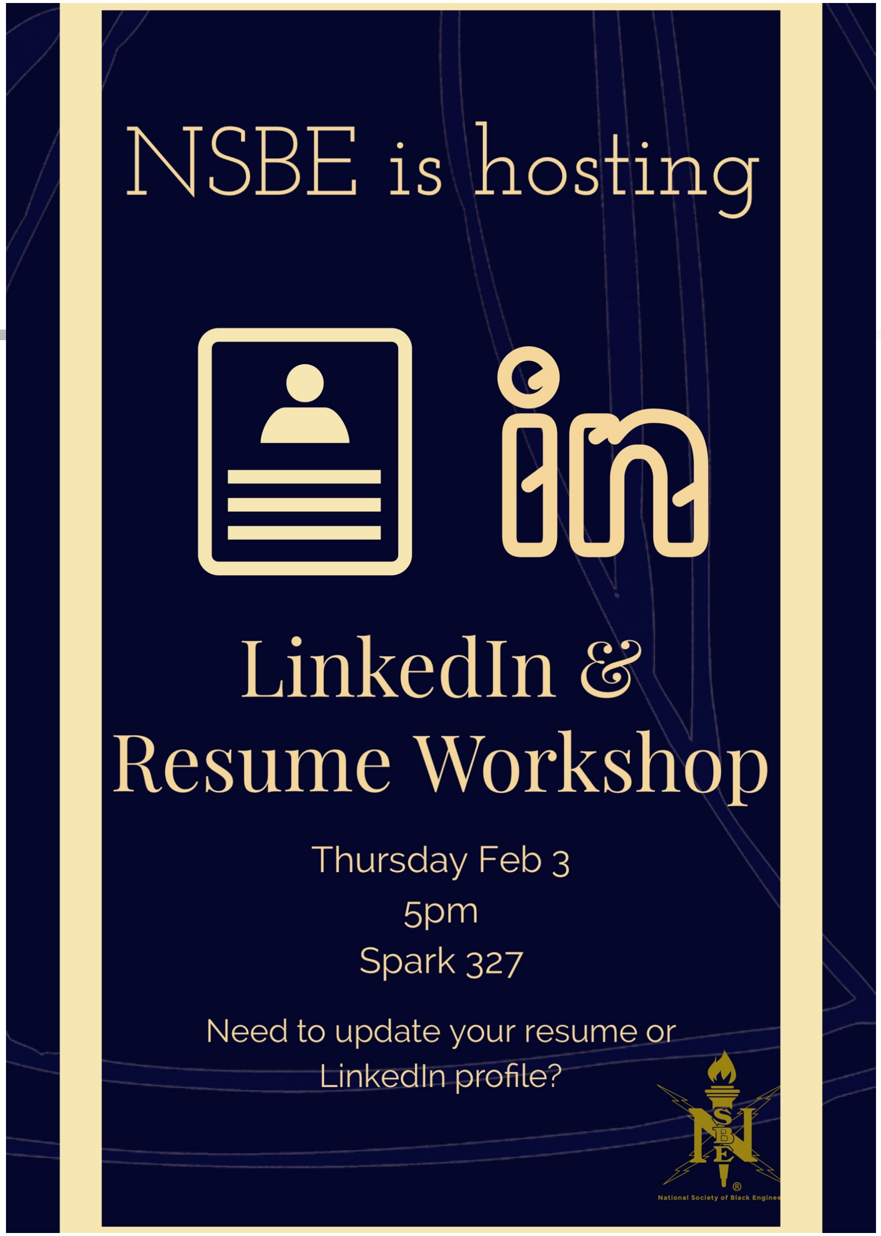
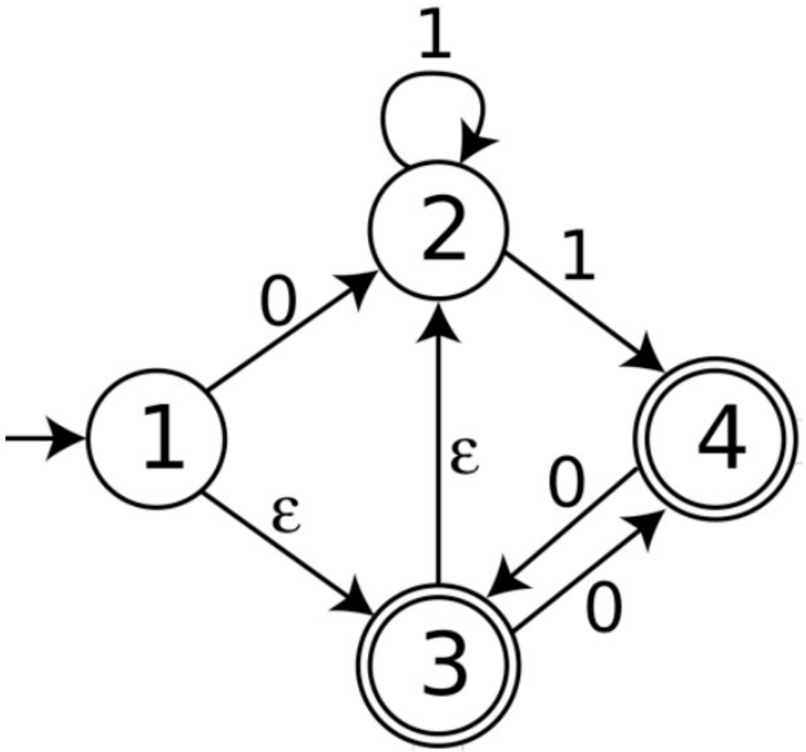
Power Set Construction Recap + Closure Statements Revisit + Regular Expressions Intro

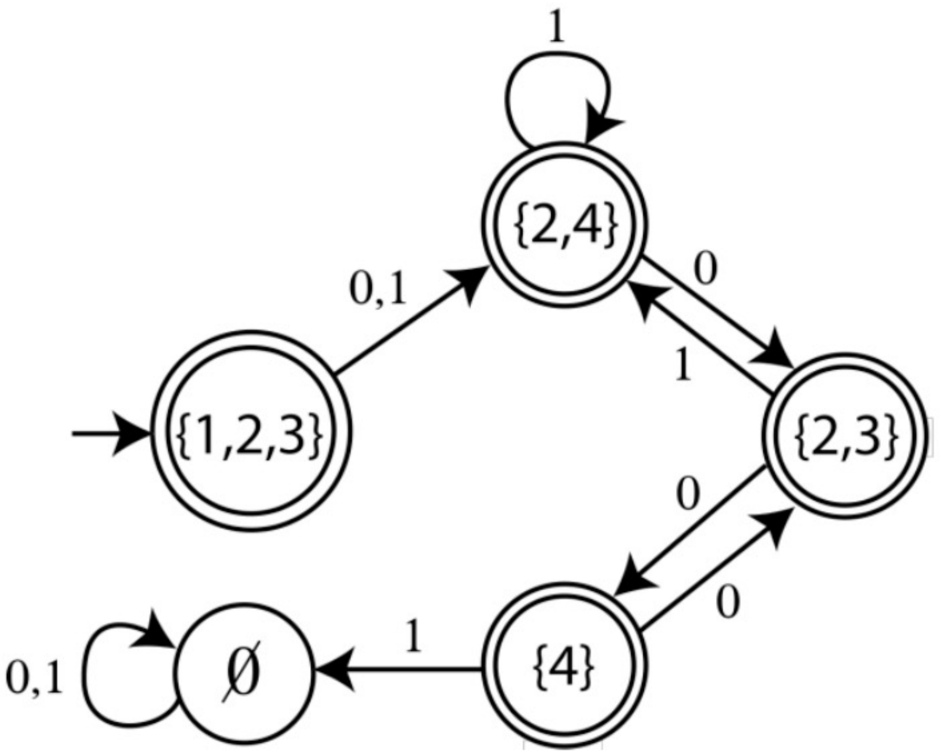
Featured Student Club

Zoom link for the event:
https://wsu.zoom.us/j/91714742835?pwd=eFYwUUNCR1dwTzR4ZUM4VXViUFRsQT09



Power Set Construction (Example, see the PDF posted on Mon 1/31 as further reading for details)





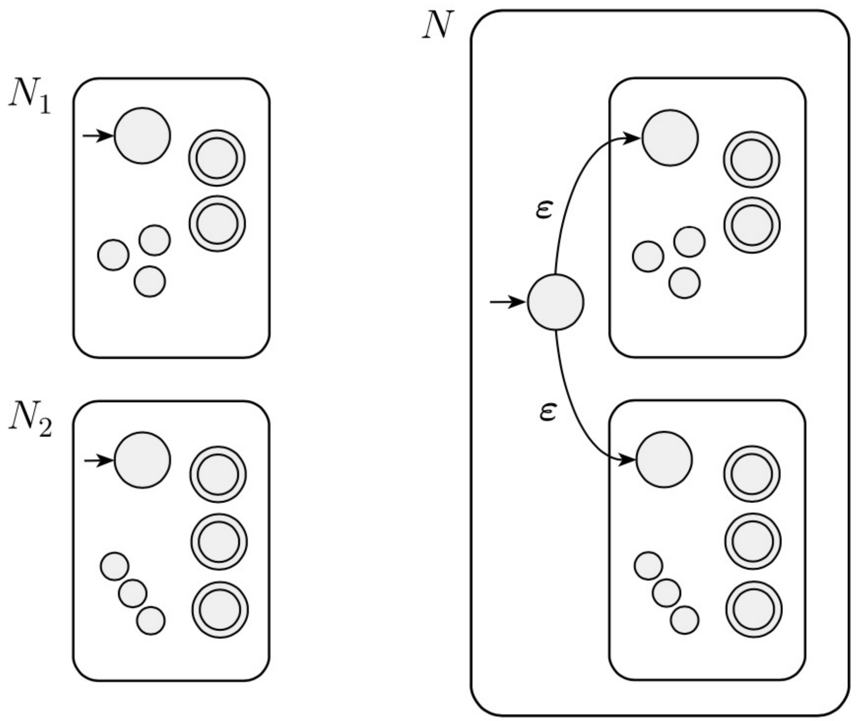
Closures under regular operations

Theorem 1

The class of regular languages is closed under the union operation

IOW: if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof using NFAs



Proof using NFAs (formal)

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
   The states of N are all the states of N_1 and N_2, with the addition of a new start state q_0.
2. The state q_0 is the start state of N.
3. The set of accept states $F = F_1 \cup F_2$.
   The accept states of N are all the accept states of N_1 and N_2. That way, N accepts if either N_1 accepts or N_2 accepts.
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

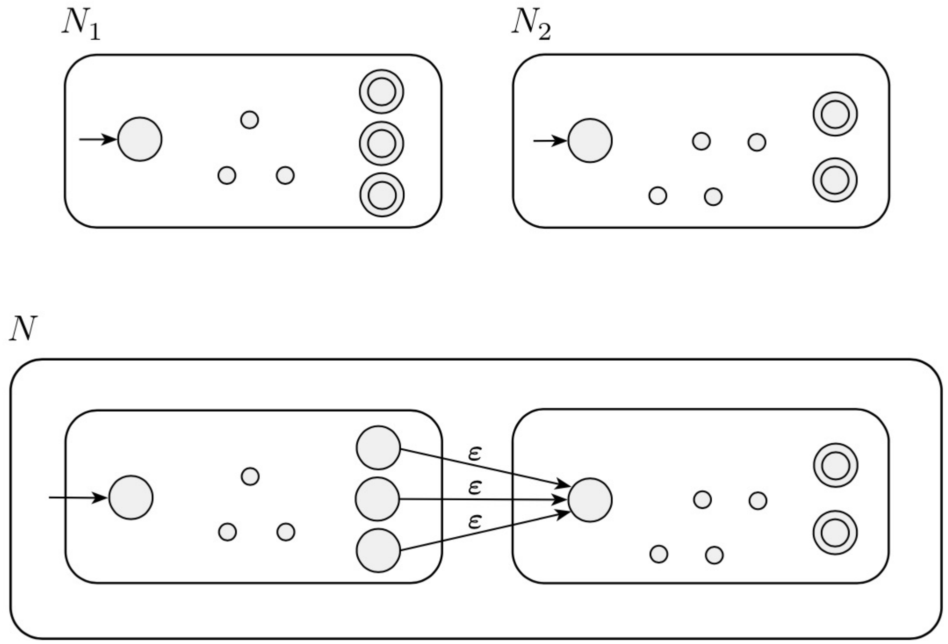
Theorem 2

The class of regular languages is closed under the concatenation operation

IOW: if A_1 and A_2 are regular languages, so is $A_1 \circ A_2$.

Proof Idea

PROOF IDEA We have regular languages A_1 and A_2 and want to prove that $A_1 \circ A_2$ is regular. The idea is to take two NFAs, N_1 and N_2 for A_1 and A_2, and combine them into a new NFA N as we did for the case of union, but this time in a different way, as shown in Figure 1.48.



- Assign N's start state to be the start state of N_1
- The accept states of N_1 have additional ε arrows that nondeterministically allow branching to N_2 whenever N_1 has an accept state, signifying that it has found an initial piece of the input that constitutes a string in A_1.
- The accept states of N are the accept states of N_2 only.
- Therefore, N accepts when the input can be split into two parts, the first accepted by N_1 and the second by N_2.

Formal proof

PROOF

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

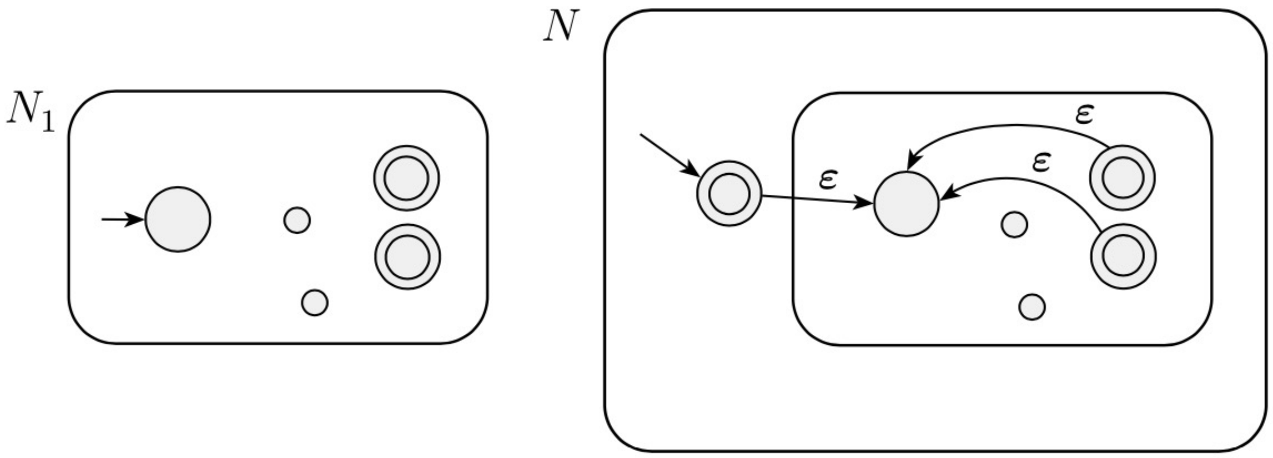
1. $Q = Q_1 \cup Q_2$.
   The states of N are all the states of N_1 and N_2.
2. The state q_1 is the same as the start state of N_1.
3. The accept states F_2 are the same as the accept states of N_2.
4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2. \end{cases}$$

Our third closure theorem

<u>Theorem 3:</u> *The class of regular languages is closed under the star operation.*

PROOF IDEA We have a regular language A_1 and want to prove that A_1^* also is regular. We take an NFA N_1 for A_1 and modify it to recognize A_1^*, as shown in the following figure. The resulting NFA N will accept its input whenever it can be broken into several pieces and N_1 accepts each piece.



Formal proof

PROOF Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$.
   The states of N are the states of N_1 plus a new start state.
2. The state q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$.
   The accept states are the old accept states plus the new start state.

Formal proof (cont'd)

4. Define δ so that for any $q \in Q$ and any $a \in \Sigma_\varepsilon$,

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \boldsymbol{\varepsilon} \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \boldsymbol{\varepsilon} \\ \{q_1\} & q = q_0 \text{ and } a = \boldsymbol{\varepsilon} \\ \emptyset & q = q_0 \text{ and } a \neq \boldsymbol{\varepsilon}. \end{cases}$$

Recaps and revisits completed

Regular Expressions

- Much like we use operations such $+$ and $\times$ in arithmetic to build expressions such as $(2+6) \times 3$, we use regular operations such as **Union**, **Concatenation** and **Star** to build regular expressions.
- Example:
  - $(0 \cup 1)\, 0^*$
- The value of an arithmetic expression is a number; the value of a regular expression is a language.

Regular expressions

- Regular expressions have an important role in computer science applications
  - Unix
  - Programming languages such as Perl
  - Text editors
- These provide mechanisms for description of patterns using regular expressions

Shorthands

- Consider the regular expression $(0 \cup 1)^*$
- The value of this expression is the language consisting of all possible strings of 0s and 1s
- If $\Sigma = \{0, 1\}$, we can write Σ as a shorthand for the regular expression $(0 \cup 1)$
- Generally, if Σ is any alphabet, the regular expression Σ describes the language consisting of *all strings of length 1* over the alphabet, and Σ^* describes the language consisting of *all strings* over that alphabet
- Example
  - The language $(0\Sigma^*) \cup (\Sigma^*1)$ consists of all strings that start with a 0 or end with 1

Precedence

- Precedence order in regular expressions
  1. Star
  2. Concatenation
  3. Union

Unless parentheses change the usual order

Formal definition of a regular expression

DEFINITION 1.52

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

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
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, or
6. (R_1^*), where R_1 is a regular expression.

In items 1 and 2, the regular expressions a and ε represent the languages $\{a\}$ and $\{\varepsilon\}$, respectively. In item 3, the regular expression $\emptyset$ represents the empty language. In items 4, 5, and 6, the expressions represent the languages obtained by taking the union or concatenation of the languages R_1 and R_2, or the star of the language R_1, respectively.

A few points

- The definition we just gave may appear *circular* (that we define a regular expression in terms of itself).
- But this is not the case. We are defining regular expressions in terms of *smaller* regular expressions, which is fine. Such a definition is called *inductive definition.*

- R^+ is usually used as a shorthand for RR^*
- R^k is used as a shorthand for concatenation of k R's with each other

- When we want to distinguish between a regular expression R and the language that it describes, we write $L(R)$ to be the language of R