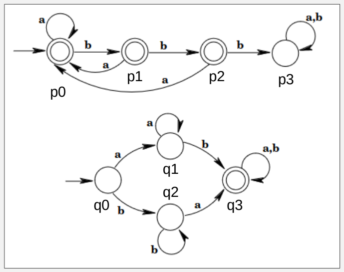
Finite-State Automata and Regular Languages

- for two DFAs M_1, M_2, M accepting $M_1 \cap M_2$ contains pairs of states from M_1 and M_2
  - the idea is to move through M_1, M_2 simultaneously and only accept strings which end in final states in both machines



Non-Regular Languages

- Pumping Lemma
  - contrapositive is used to show that languages are not regular

Theorem 3.6. *If L is a regular language, then there is some number $n > 0$ such that any string w in L whose length is greater than or equal to n can be broken down into three pieces x, y, and z, $w = xyz$, such that*

(i) x and y together contain no more than n symbols;

(ii) y contains at least one symbol;

(iii) xz is accepted by M

(xyz is accepted by M)

$xyyz$ is accepted by M

etc.

  - the key idea is that for a regular language, if a string is long enough, it has to have a certain structure – corresponding to a cycle in *M*
    - if that structure isn't present, the language isn't regular

Non-Regular Languages

Theorem 3.6. *If L is a regular language, then there is some number $n > 0$ such that any string w in L whose length is greater than or equal to n can be broken down into three pieces x, y, and z, $w = xyz$, such that*

(i) x and y together contain no more than n symbols;

(ii) y contains at least one symbol;

(iii) xz is accepted by M

(xyz is accepted by M)

$xyyz$ is accepted by M

etc.

show that $\{ a^n b^n \mid n \geq 0 \}$ is not regular

- let *N* be the threshold length and pick $a^N b^N$ as a string whose length is at least *N*
- show that $a^N b^N$ can't be written in the form *xyz* by showing that any choice for *y* that satisfies (i) and (ii) doesn't satisfy (iii)
  - since *xy* can't contain more than *N* symbols, both *x* and *y* contain only *a*s
  - let *k* be the number of as in *y* – since *y* can't be empty, $1 \leq k \leq N$
  - then $xz = a^{N-k} b^N$ – which is not of the form $a^n b^n$ and thus isn't accepted by *M*

1. Use the Pumping Lemma to show that the following languages over $\{a, b\}$ are not regular.

a) $L_1 = \{x \mid n_a(x) = n_b(x)\}$
b) $L_2 = \{xx \mid x \in \{a, b\}^*\}$
c) $L_3 = \{xx^R \mid x \in \{a, b\}^*\}$
d) $L_4 = \{a^n b^m \mid n < m\}$

Theorem 3.6. *If L is a regular language, then there is some number $n > 0$ such that any string w in L whose length is greater than or equal to n can be broken down into three pieces x, y, and z, $w = xyz$, such that*

(i) x and y together contain no more than n symbols;

(ii) y contains at least one symbol;

(iii) xz is accepted by M

(xyz is accepted by M)

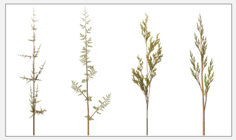
$xyyz$ is accepted by M

etc.

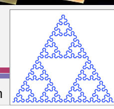
The Big Picture

Why do we care about being able to write computer programs that can recognize or generate languages?

- pattern matching
- L-systems
  - a system for describing fractal shapes
- compilers
  - being able to parse a program file
- …



that DFAs can recognize the languages generated by regular expressions is good news for programs, but there are also languages, like { $a^n b^n$ | n ≥ 0 }, which aren't regular but are still easily recognizable by programs…



https://en.wikipedia.org/wiki/L-system