truncate(ababba) = ababb

~~If w~~

truncate(L) = {truncate(w) : w ∈ L}

set of strings

We're overloading this function name

result is a set of strings

truncate({ababba, bab, aabab})

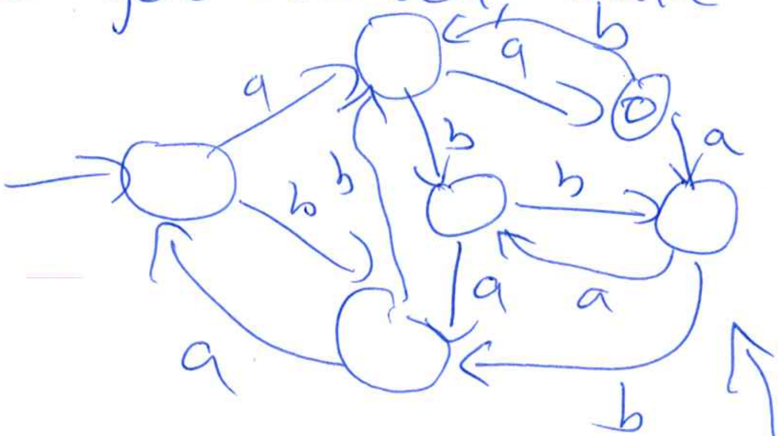
= {ababb, ba, aaba}

Given some unknown DFA for L ~~(you are given)~~ construct a DFA for truncate(L).

This means - describe a computer program that takes as input a DFA M, and outputs a DFA M′, so that if L is the language accepted by M, then M′ accepts truncate(L).

To do this problem, you need to be specific - someone should be able to write the code from your description.

To get started, make a random machine



and think

about what machine would accept truncate(L) (where L is the language accepted by)

Regular grammars

So far, we have been thinking about ~~ta~~ regular languages - by def'n a ~~ta~~ set of strings that is the set accepted by Some DFA.

We have 2 ways so far to describe a regular lang -

- give a DFA or NFA for it

or - give a reg. exp. for it.

Today: Another way to describe languages in general (and specifically regular lang.)

A grammar a set of rules for generating a language that has a specific form (later)

E.g. $T = \{a, b\}$ $V = \{S, C, D\}$

$S \to aCb$

$S \to CbD$

$C \to CD$

$C \to abD$

$C \to b$

$D \to DCSa$

$D \to a$

To generate strings using a grammar, I start with the "start symbol" S, and keep making substitutions until all my symbols are in T.

$$S \to CbD \to CDbD \to bDbD$$
$$\to bDCSabD \to baCSabD$$
$$\to babSabD \to babaCbabD$$
$$\to \boxed{bababbaba}$$

The final result is "a string generated by the grammar"

The language generated by a grammar is the set of a strings that can be generated by the grammar.

Formal definition:

A grammar is a set (V, T, P, S) where

V = finite set of symbols (variables)

T = finite ~~set of~~ alphabet (terminal symbols)

(We assume $V \cap T = \emptyset$)

$S \in V$ the start symbol

$P \subseteq \cancel{V^*} \times (V \cup T)^*$ are my production rules.

$V^* \setminus \{\lambda\}$

not allowed to replace nothing w/something

Def: A string w is produced by a grammar $G=(V,T,P,S)$ if

0) $w \in T^*$

1) There is some sequence of strings $v_0, \ldots, v_k \in (V \cup T)^*$ where

$v_0 = S$

$v_k = w$

and, for each i, "I can get from v_i to v_{i+1} by following a rule in P"

$v_i = xyz$, $v_{i+1} = xy'z$, where $x, y, y', z \in (V \cup T)^*$, and $(y, y') \in P$

This is the formal way to say "replace y by y' to get from v_i to v_{i+1}"

Some boring grammars: (conventions:
capital letters
are in V
lower case in T)

$S \to \cancel{aAb}\ aSb$

$S \to \lambda$

generates a's followed by b's, with same number of each.

~~$S \to \cancel{ab}\ abT$~~

~~$S \to \lambda$~~

$S \to T$

$S \to bT$

$T \to abT$

$T \to \lambda$

bab, babab, bababab...

λ, ab, abab.....

strings that alternated between a's + b's that end with a b.

add $T \to a$ to allow strings to also end with a.

or add $S \to Ta$

Def: A grammar is right-linear if every production in P looks like

$$A \longrightarrow abcdefg\,B$$

single variable

string using the alphabet (no variables) (could be λ)

single variable, at the right end

or

$$A \longrightarrow abcdef$$

single variable

string w/ no variables (could be λ)

Next time - explain the corresp. between right-linear grammars and NFAs.