Deterministic PDA:

A PDA is *deterministic* if we do not have pairs of transitions

① $(q, \ell, s) \to (q', s')$ and $(q, \ell, s) \to (q'', s'')$ (where $q' \neq q''$ or $s' \neq s''$) $\leftarrow$ ($\ell = \lambda$ possible)

or like

② $(q, \lambda, s) \to (q', s')$
$(q, \ell, s) \to (q'', s'')$
(where $\ell \neq \lambda$)

Def: A language is *deterministic context-free* if there exists a deterministic PDA for it.

Note: It is (relatively) easy to show a lang. is det. CF - make a det PDA for it, but it is *hard* to show a lang is *not* det CF.

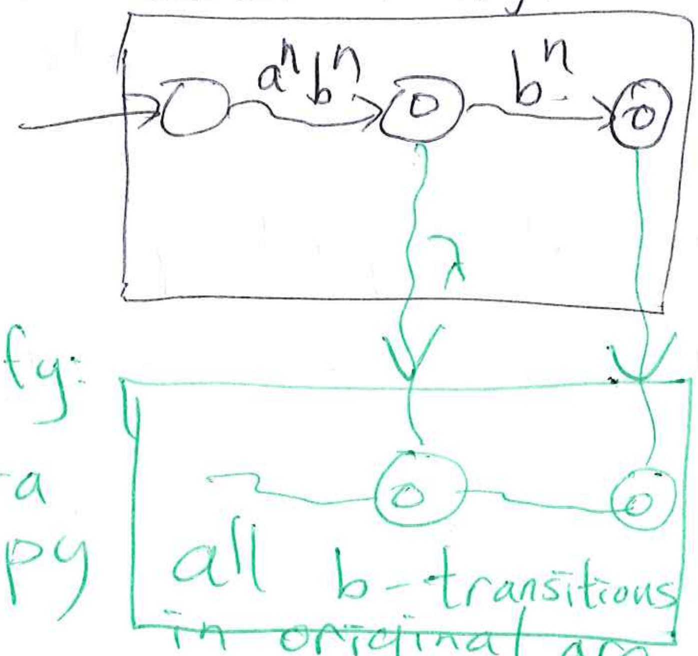
E.g. It turns out

$$\{a^n b^n \mid n \geq 1\} \cup \{a^n b^{2n} \mid n \geq 1\}$$

is CF but not det CF.

Sketch of explanation.

Suppose we had a det. PDA for this lang: Schematically:



Modify: extra copy

all b-transitions in original are c-transitions here.

This modified PDA accepts

$$\{a^n b^n \mid n \geq 1\}$$
$$\cup \{a^n b^{2n} \mid n \geq 1\}$$
$$\cup \boxed{\{a^n b^n c^n \mid n \geq 1\}}$$

we'll prove this is not a context-free language next week.

Homework 4:

Show that, if $L_1 \cup L_2$ is regular and L_1 is finite, then L_2 is regular.

If L_1 wasn't finite but is regular, the statement is not true!

What do all these words mean?

hierachy of objects/types:

1. letters (these are primitive)
2. strings (these are finite lists of letters)
3. languages (these are collections of strings)

Note: A language is allowed to contain infinitely many strings.

Def: A language is finite if it has finitely many strings.

Def: A language is regular if there is a single DFA that accepts every string in the language and rejects every string not in the language.

Note: For a single string, this is always possible - so it makes no sense to talk about strings being regular.

It is certainly possible for $L_1 \cup L_2$ to be regular w/o L_2 being regular.

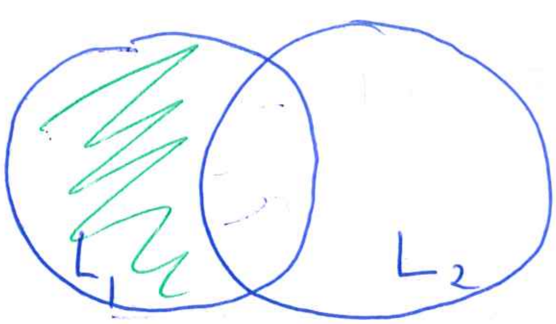
E.g.: Have $L_1 = \{$all strings on a's + b's$\}$

$$= \{a, b\}^*$$

$$L_2 = \{a^n b^n\}$$

Then $L_1 \cup L_2 = L_1 = \{a, b\}^*$

but L_2 is not regular.



We can get L_2 from $L_1 \cup L_2$ as

$L_2 = (L_1 \cup L_2) \setminus$ green stuff green stuff $= L_1 \setminus L_2 = L_1 \cap \overline{L_2}$

$$L_2 = (L_1 \cup L_2) \cap \overline{(L_1 \cap \overline{L_2})}$$

We need some argument that the green stuff $L_1 \cap \overline{L_2}$ is regular.

But, since L_1 is finite, $L_1 \cap \overline{L_2}$ is finite and hence regular.

Then we know $\overline{L_1 \cap \overline{L_2}}$ is regular (since the complement of a reg. lang is reg - swap "finalness" of states in DFA)

Then we know

$$(L_1 \cup L_2) \cap \overline{(L_1 \cap \overline{L_2})}$$

is regular b/c intersection of reg. lang. is reg. (we can construct a DFA that "runs both DFAs simultaneously")

So $L_2 = (L_1 \cup L_2) \cap \overline{(L_1 \cap \overline{L_2})}$ is regular.

$L = \{ a^m b^k c^m \mid m \geq 0, k \geq m \}$

is not regular.

Pumping Lemma usage:

m is an unknown number -
you need to handle every possibility

Given this m, your computer program chooses a string $w \in L$.

Once your program chooses a w, it is given strings x, y, z - you do <u>not</u> choose these where

$w = xyz, \quad |xy| \leq m, \quad y \neq \lambda$.

Once your program is given x, y, z, it chooses a number i.

If $xy^iz \notin L$, your computer program wins. You want to be able to beat a perfect opponent.

For this problem:

We're handed an m.

We pick $w = a^m b^{m+50} c^m$

Our opponent is forced to pick x, y, z where $y = a^j$ for some j, $1 \le j \le m$.

Then we can choose $i = 0$.

Now $x\boxed{y^0}z = a^{m-j} b^{m+50} c^m$

This is deleting y from string - and y is some number of a's (precisely j of them)

$xy^0z \notin L$ since the number of a's $\neq$ number of c's.