Last time: CFG → stack:

$S \to aSb \mid T$

$T \to bT \mid \varepsilon$

read	pop	push
ε	S	aSb
ε	S	T
ε	T	bT
ε	T	ε
a	a	ε
b	b	ε

We can convert stack → grammar

A stack rule:

read	pop	push
x	y	z

becomes: $y \to xz$

$\{x \mid \#a's = \#b's\}$

	read	pop	push
1	a	S	XS
2	a	X	XX
3	a	Y	ε
4	b	S	YS
5	b	Y	YY
6	b	X	ε
7	ε	S	ε

Grammar:

1. $S \to aXS$
2. $X \to aXX$
3. $Y \to a$
4. $S \to bYS$
5. $Y \to bYY$
6. $X \to b$
7. $S \to \varepsilon$

$S \to aXS \mid bYS \mid \varepsilon$

$X \to aXX \mid b$

$Y \to bYY \mid a$

Derive abba

stack: $(abba, S) \mapsto_1 (bba, XS) \mapsto_6 (ba, S) \mapsto_4 (a, YS)$

$\mapsto_3 (\varepsilon, S) \mapsto_7 (\varepsilon, \varepsilon)$

grammar: $S \Rightarrow_1 aXS \Rightarrow_6 abS \Rightarrow_4 abbYS \Rightarrow_3 abbaS \Rightarrow abba$

Our grammar steps always look like

(string read so far / built) (stack.)

action happens here.

What action exactly?

if the stack rule is

read	pop	push
x	y	z,

then (string) y (stack) will become:

(string x) z (stack)

this was a grammar substitution $y \to xz$

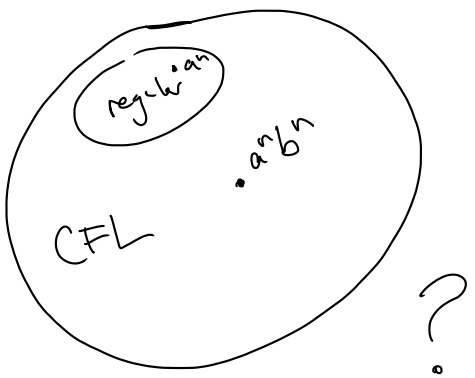
Last time: CFG → stack.

This time: stack → CFG

So CFG & stack machines have the same capabilities.

Thm L is context-free iff

L is the language of some stack machine.



Are there any simple non-CF languages?

Classic example: $a^n b^n c^n$ is not a CFL

Failed attempts: $a^n b^n$ is $S \to aSb \mid \varepsilon$

for $a^n b^n c^n$: ~~$S \to aSbc \mid \varepsilon$~~ makes $a^n (bc)^n$

~~$S \to aSbSc \mid \varepsilon$~~

$S \Rightarrow aSbSc \Rightarrow aaSbScbaSbScc$

$\Rightarrow a^2bcbabc^2$

$S \to aSTR \mid \varepsilon$

$T \to b$

$R \to c$

Later (maybe never)
we'll show why $a^n b^n c^n$ is not CFL.

A grammar for $a^n b^n c^n$:

$S \to aSBc \mid abc \mid \varepsilon$

$cB \to Bc$ ← B can commute with c

$bB \to bb$ ← B can become b, if it's next to b.

$S \Rightarrow aSBc \Rightarrow aabcBc$

$\Rightarrow aabBcc \Rightarrow aabbcc$

Most real-world programming langs are not CFL.

The Python inpreter must analyze & balance more than 2 things at once to parse indents

Python is indented like:

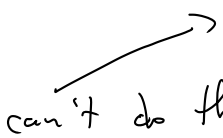
```
XXX:
 XXX
 XXX

XX
XXX:
 XX:
  XXX
  XXX
```

or

```
  XX
  XXX
  XXX
  XXX
```

we have something like

$$T^2(\sim)T^2(\sim)T^2(\sim)T^2(\sim)$$



can't do this in a CFL.

def [name]([vars]): INDENT BLOCK OUTDENT

Turing Machines

In early 1900s math was being rewritten using axioms:

1880s - axioms in algebra
1890s - Hilbert geometry axioms
1900s - axioms for set theory

Axioms for everything?

Can all facts be proved based on axioms?

Is there one single master procedure to prove stuff from axioms?

Turing has the idea to turn all of math into some kind of mechanized procedure.