CSCE 355
4/2/2025

Intro to the programming project

Pumping Lemma for CFLs

Project Homepage:

https://cse.sc.edu/~fenner/csce355/prog-proj2/sp25/index.html

links are here

Rexes 2 kinds of tasks:

1. Given an input regex r, answer a yes/no question about r

2. Given an input regex r, ~~to~~ output a regex r' such that $L(r')$ is related to $L(r)$ in some way.

Ex: Is $L(r)$ empty? $L(r) = \emptyset$

Rules

r	$L(r) = \emptyset$?
$\emptyset$	yes
$a \in \Sigma$	no
$s+t$	$L(s)=\emptyset$ and $L(t)=\emptyset$
st	$L(s)=\emptyset$ or $L(t)=\emptyset$
s^*	no

s, t regexes

($L(s^*)$ always contains ε.)

Command-line interface on Linux

`$ ./my_program --empty ↵`

regexes typed at the keyboard

^D

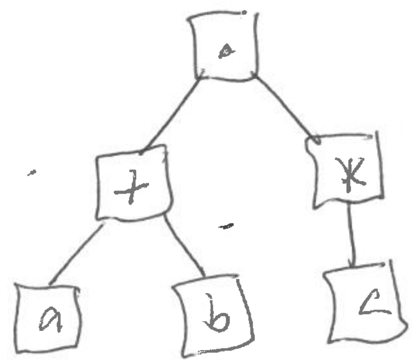
regexes are read from standard input

answers are to std output

Syntax:

• = concatenate

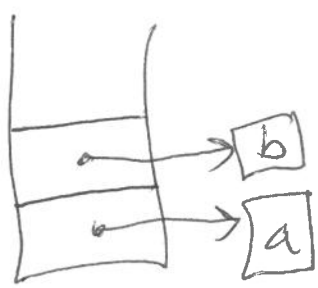
$(a+b)c^*$ ⟹



⇓ infix → postfix (I with provide this)

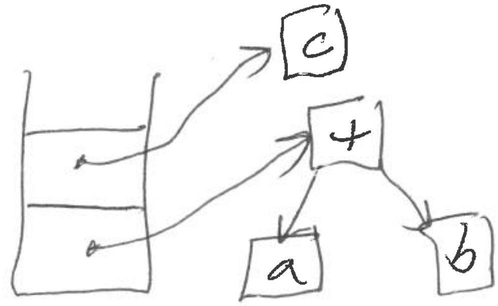
ab+c*•

post-fix

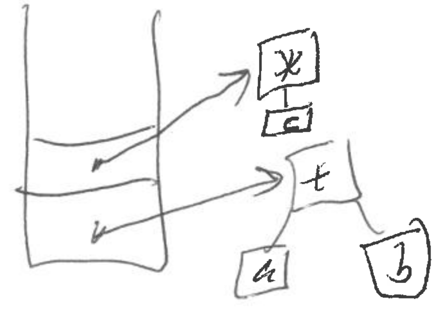


stack of trees

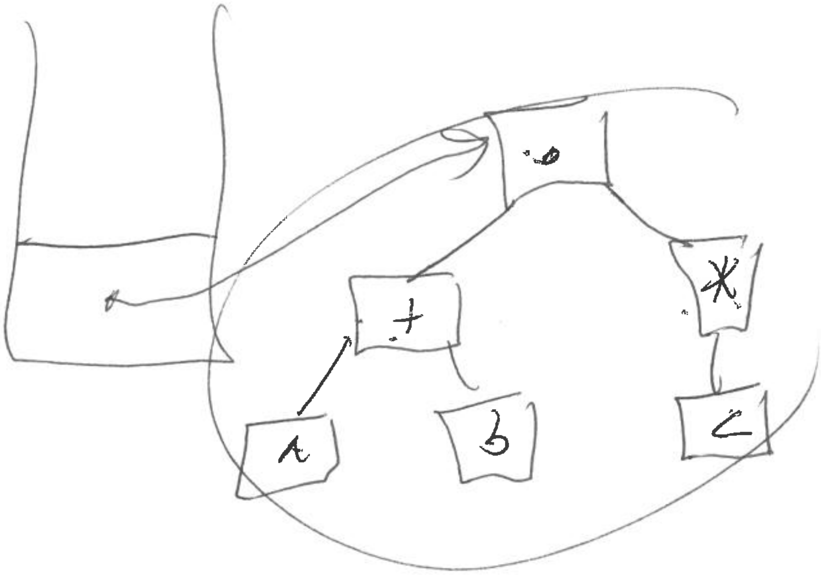
pop 2 items, push on a +-node



pop 1 push new *-node



pop 2 $\Longrightarrow$ push new concat node



Ex: reverse — Given regex r, output r' such that $L(r') = L(r)^R$

Recall:

r	r'
$\emptyset$	$\emptyset$
$a \in \Sigma$	a
$s+t$	$s'+t'$
st	$t's'$
s^*	$(s')^*$

Output is in prefix form (operators precede their operands)

Ex: prefix form ~~of~~ of $(a+b)c^*$ is $\cdot + a b * c$

$\Longrightarrow$ prefix to infix: $(a+b)c^*$ (I provide this)

Pumping Lemma for CFLs.

Lemma (Pumping Lemma for CFLs): Let L be any CFL. ~~There~~

There exists a $p > 0$ ("pumping length") such that

For every $s \in L$ with $|s| \geq p$,

There exist strings u, v, w, x, y such that

1) $s = uvwxy$
2) $|vwx| \leq p$
3) $|vx| > 0$ (v & x are not both ~~emp~~ ε)

and

For every $i \geq 0$,

$uv^iwx^iy \in L$.

L is CFL-pumpable

L is not CFL pumpable iff

For all $p > 0$,

There exists $s \in L$ with $|s| \geq p$ such that

For all u, v, w, x, y where

- $s = uvwxy$
- $|vwx| \leq p$
- $|vx| > 0$,

There exists $i \geq 0$ such that

$uv^iwx^iy \notin L$.

Prop: $\{a^n b^n c^n : n \geq 0\}$ is not CFL-pumpable (thus L is not a CFL).

Proof: Given $p > 0$,

Let $s := a^p b^p c^p$.

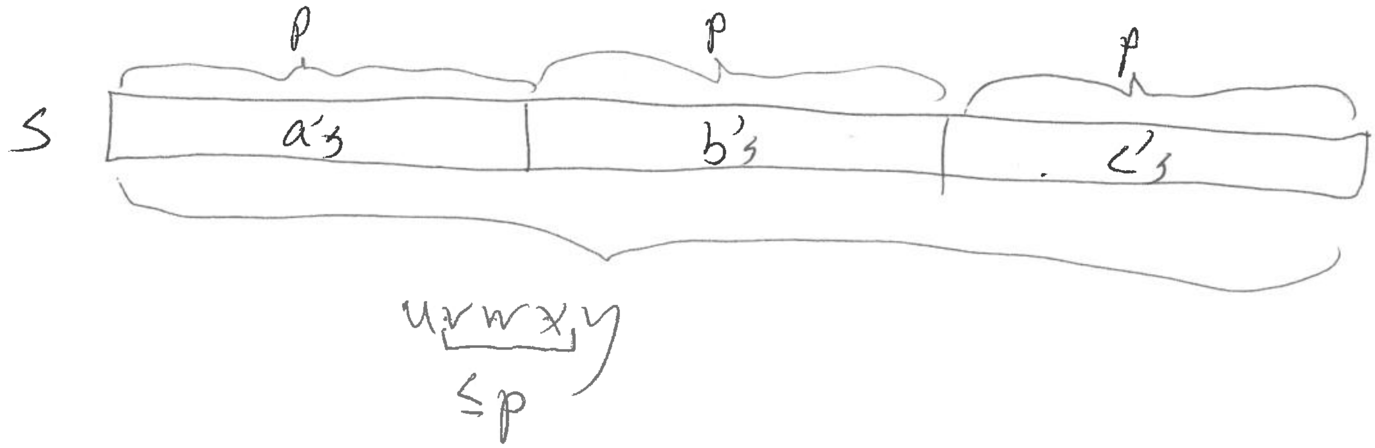
$[s \in L \ \& \ |s| = 3p \geq p]$

Given u, v, w, x, y such that

$s = uvwxy$, $|vwx| \leq p$, $|vx| > 0$,

Let $i := 0$. Then $uv^iwx^iy = uwy \notin L$

because



vwx can't have both a's and c's in it (the a's & c's are too far apart)

So since $vx \neq \varepsilon$, pumping down either:

— removes one or more a's or b's but not c's, or

— removes one or more b's or c's but not a's.

In either case, uwy has #a's, #b's, #c's not all equal, so $uwy \notin L$. //

Ex: $\{a^m b^n c^m d^n : m,n \geq 0\}$ is not CFL-pumpable

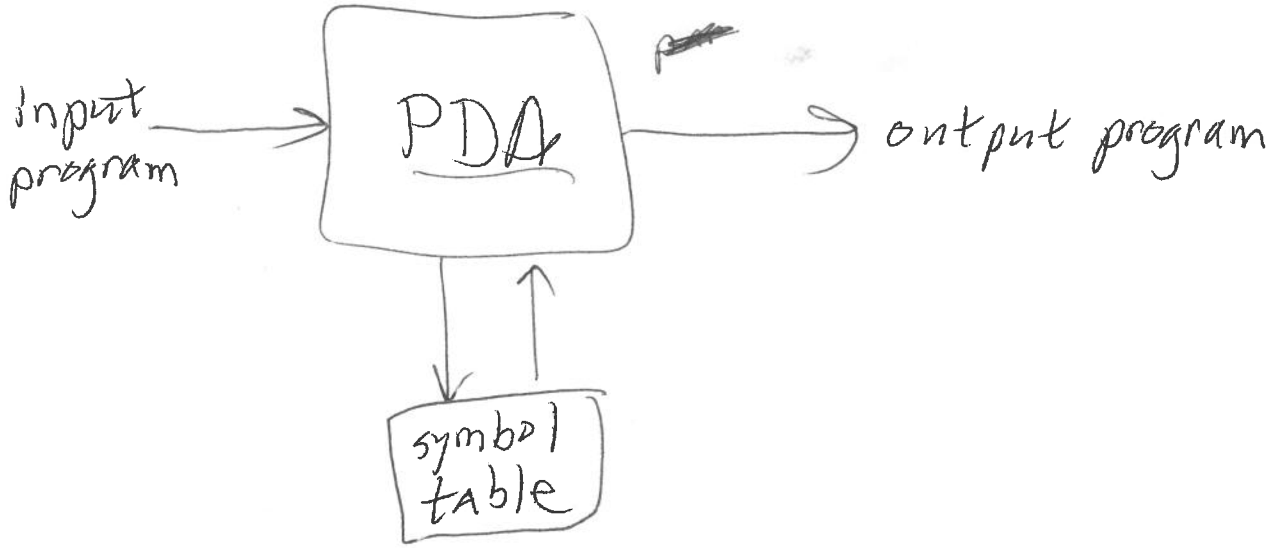
Note: $\{a^m b^m c^n d^n : m,n \geq 0\}$ and $\{a^m b^n c^n d^m : m,n \geq 0\}$ are both CFLs.

In C

```
int f(a,b,c) { ... }
  :
int g(a,b,c,d) { ... }
  :
  :

{



    f(2,1,3)
    g(6,2,0,4)



}
```

Non context-free feature of C, C++, Java

Can't use just a $\underbrace{\text{PDA}}_{\text{parser}}$ to enforce this.


input program
PDA
output program
symbol table