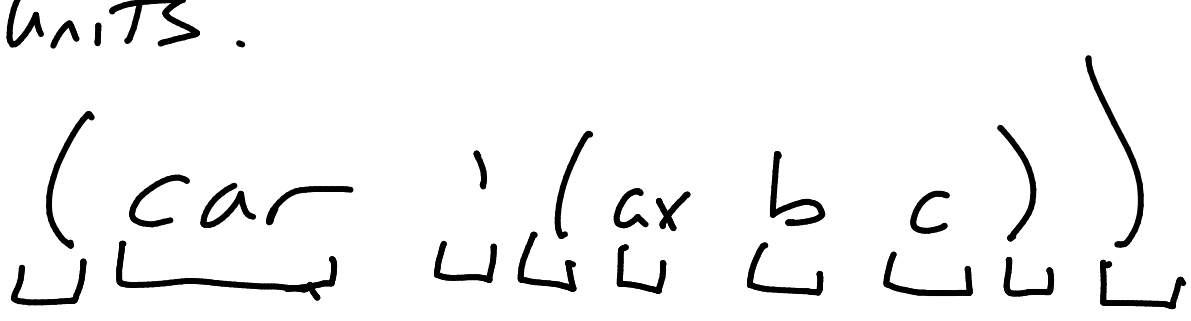
Tokenizing review
Parsing algorithms
Scoping?

| Cok on Fri

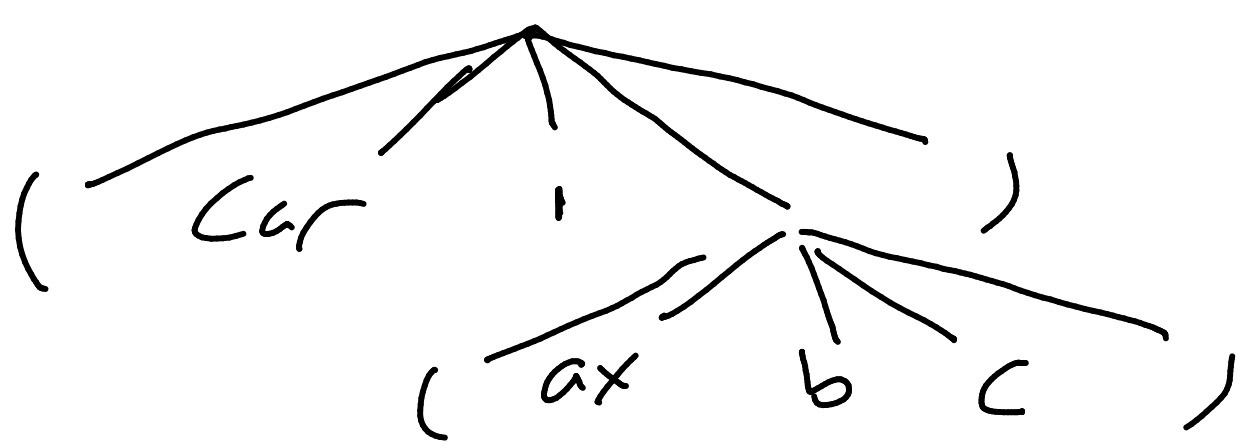
Distinguish between tokenizing and parsing

Tokenizing / Scanning / Lexing -
- splitting code into minimum meaningful units.



- when done, we have a list of tokens

Parsing - builds program structure from that list
- most languages - it builds a tree



Possible errors: are these caught by tokenizer or the poser?

(define 3ab 2)

↑
we don't allow symbols to start w/ digits

should get caught in tokenizing
- tokenizing splits, but also applies labels
- it's not a symbol, it's an int, etc, so error

(a b [missing right paren]

Parsing error. because error is in structure, not in identifying individual tokens

~~(define x 23)~~

(define y z)

This is an error because z is undefined

tokenizing ✓

parsing ✓

executing the code ✗

More work on parsing

stack

(

define

sum

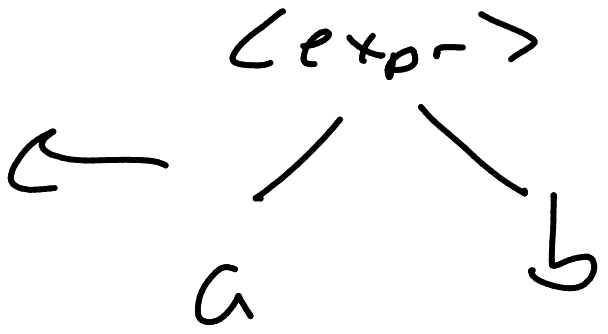
(

lambda

x

y

b



stack

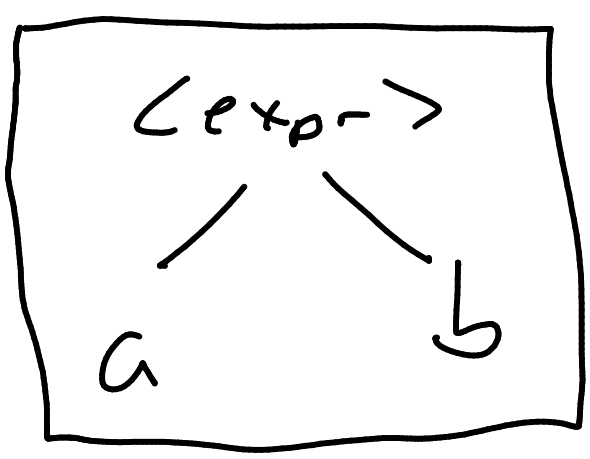
(

define

sum

(

lambda

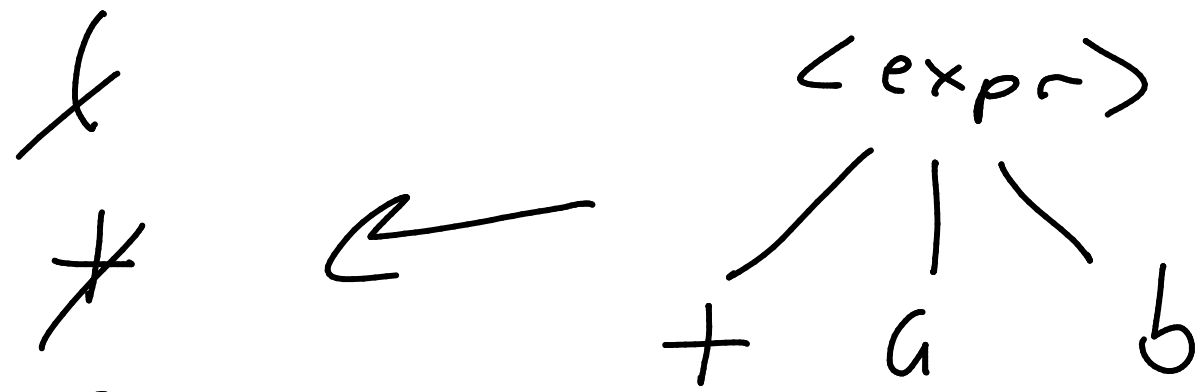


a

b

a

b



stack

(

define

sum

(

lambda

<exp>

a

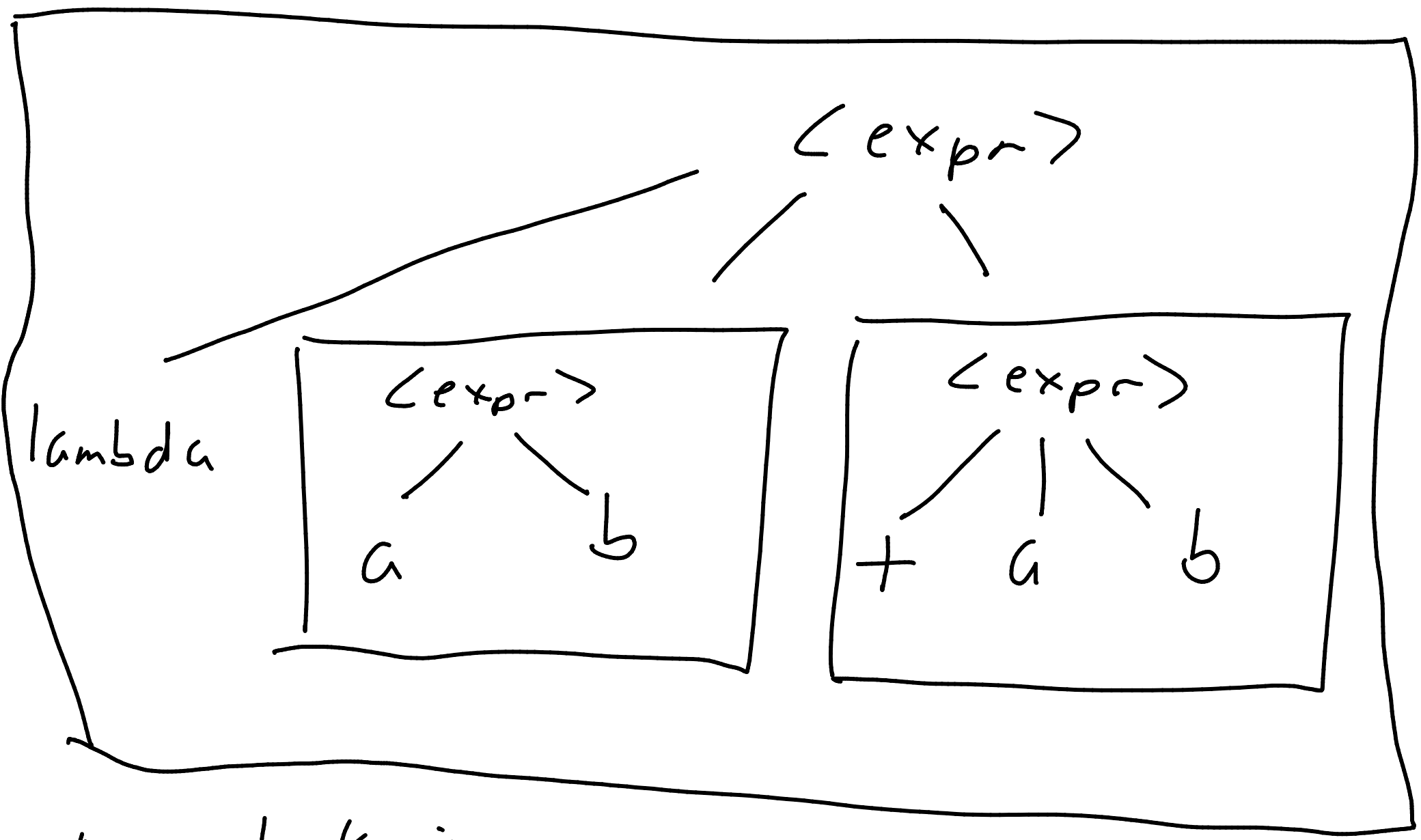
b

<exp>

+

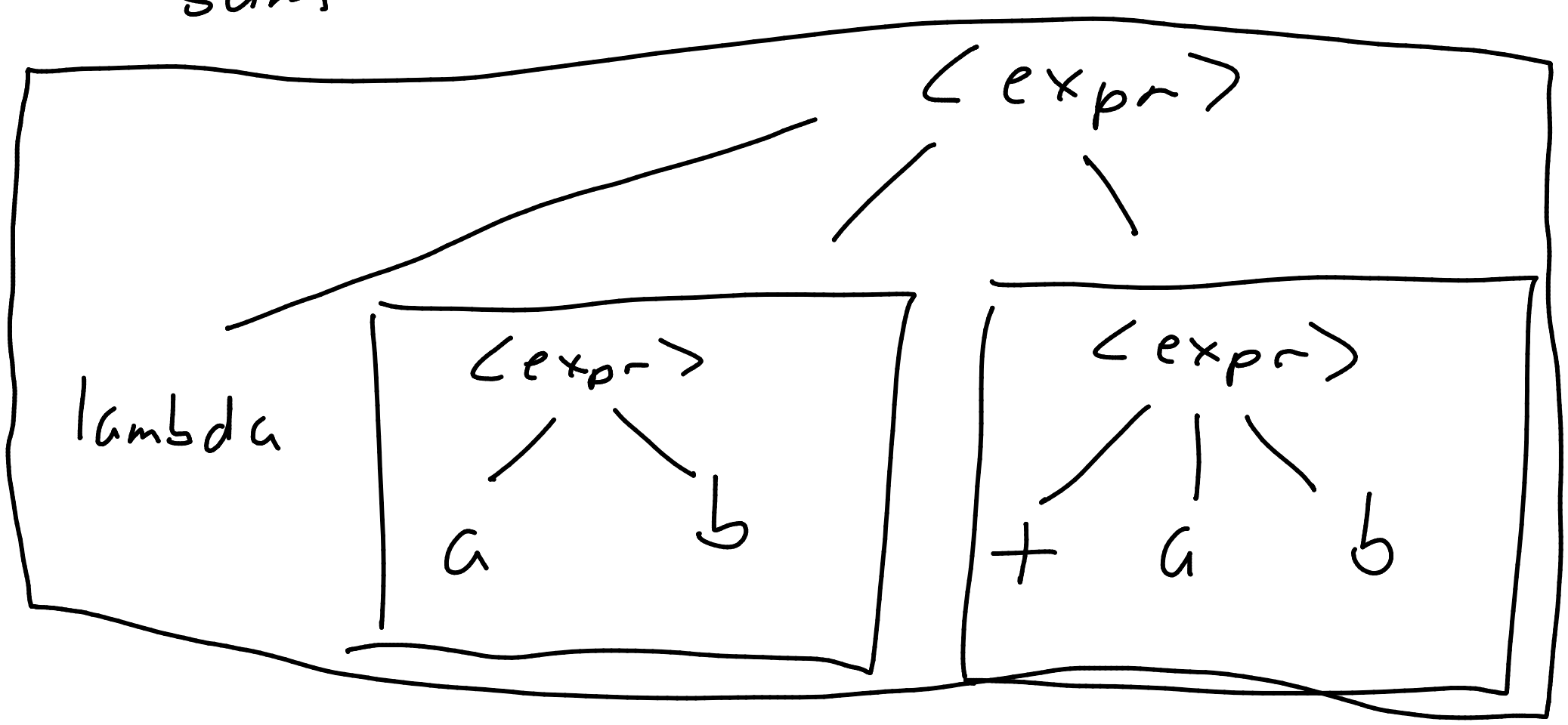
a

b



new stack is

(
define
sum



This process is a bottom-up parsing algorithm

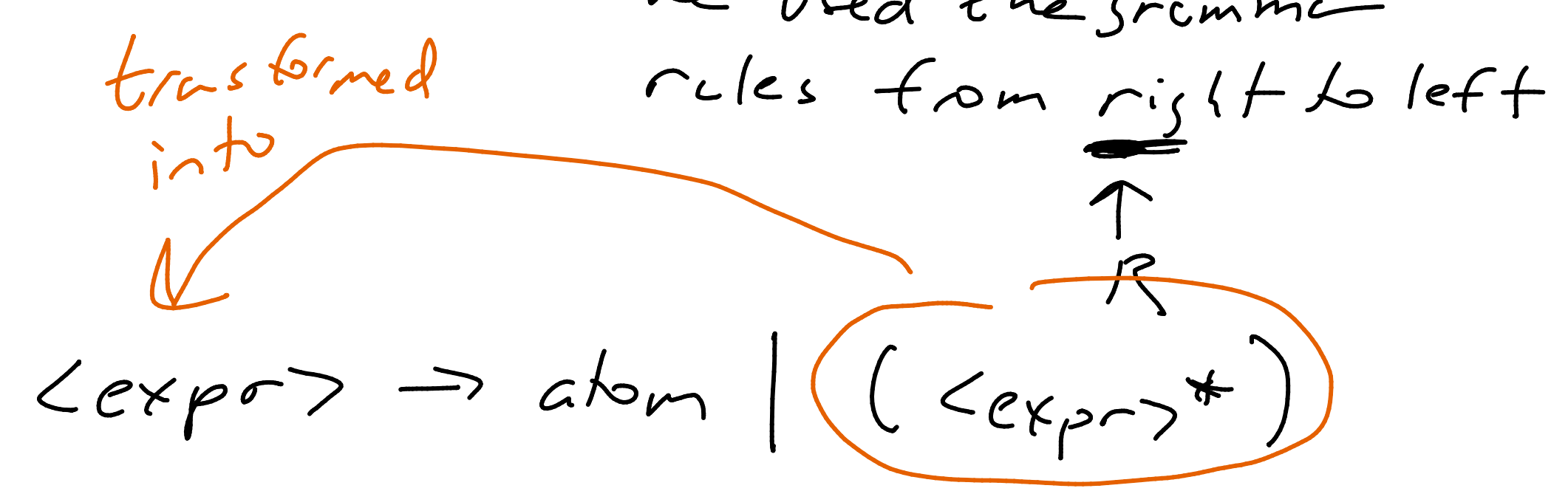
This is referred to as an LR parsing algorithm

we read the code left to right

we used the grammar rules from right to left

transformed into

<expr> → atom | (<expr>*)



where we created a new subtree, we took a list of exprs, and made them the children of another <expr>

(+ a b)