Context free Grammars (recap)

NP → N | NP and NP

NP → ADJ NP | N

N → eggs | ham | spinach | .....

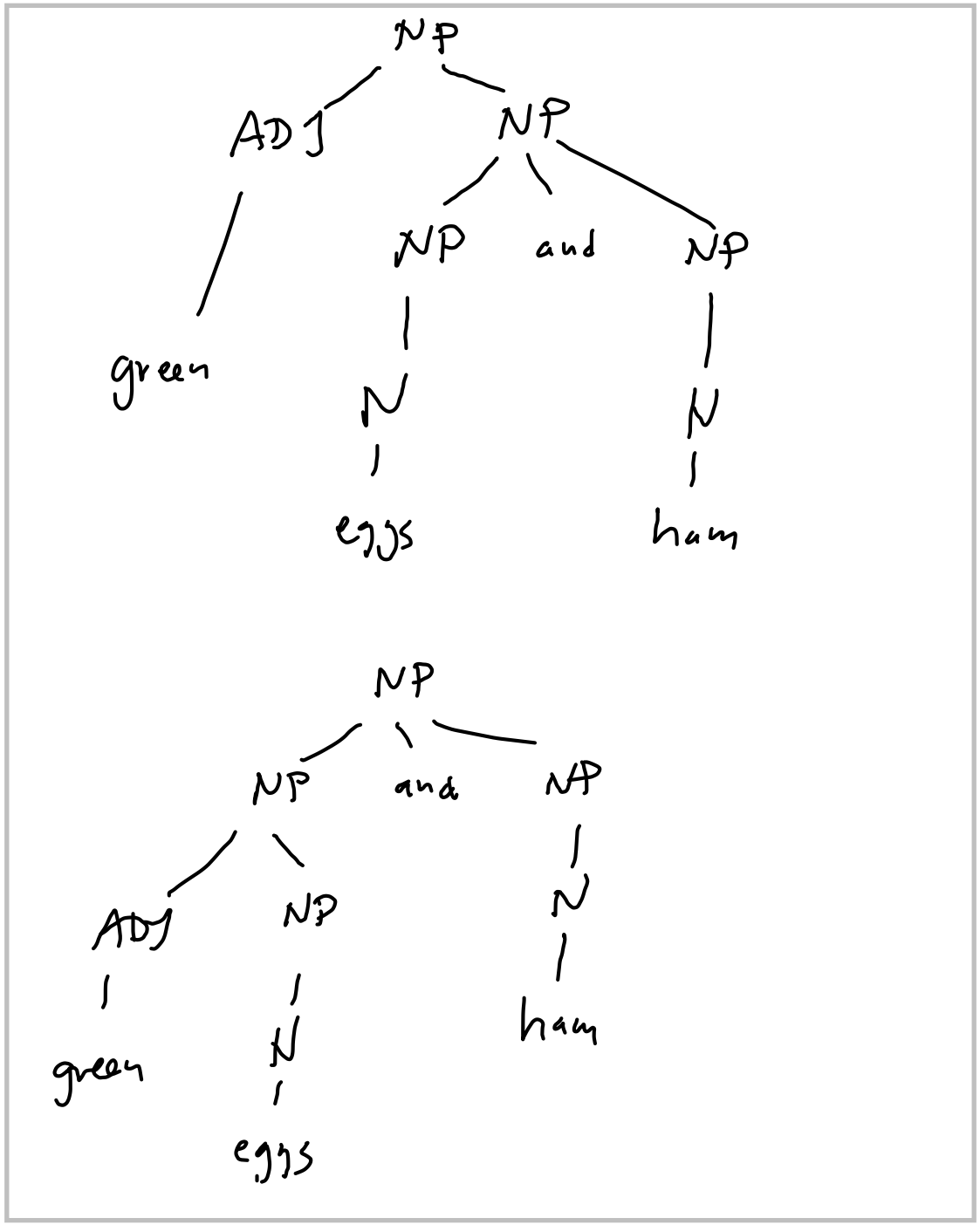
ADJ → green | burnt | .....

$\langle V, \Sigma, R, NP \rangle$ $\langle NP \rangle$

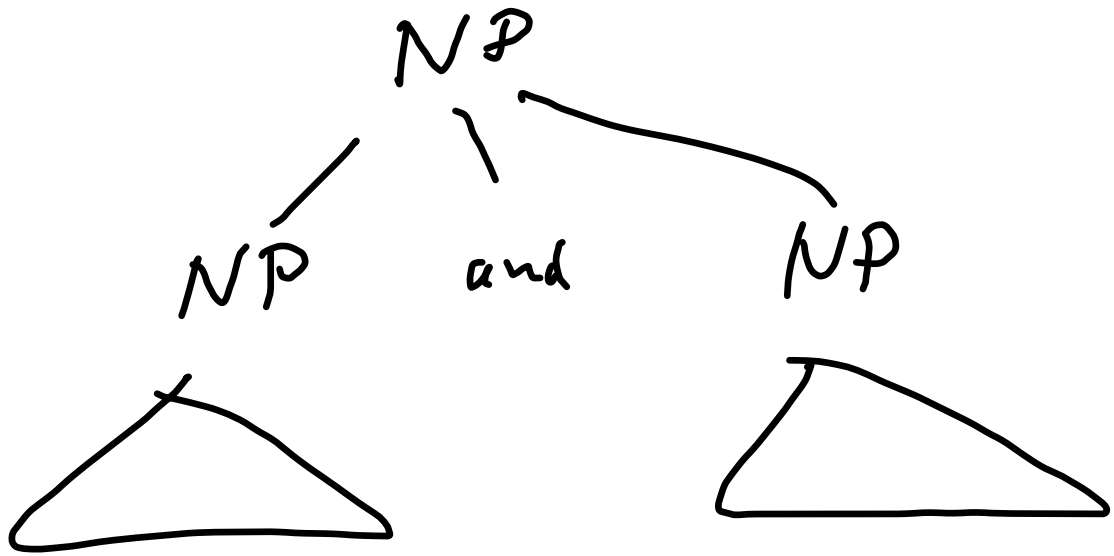
Start symbol is NP

$V = \{NP, ADJ, N\}$

$\Sigma = \{eggs, ham, spinach, green, \ldots\}$


NP
ADJ
NP
green
NP
and
NP
N
N
eggs
ham
NP
NP
and
NP
ADJ
NP
N
green
N
eggs
ham

No common fate



NP → NP and NP

A → ... A ... A ...

S → ... A ... A ...

Chomsky Normal form

Big idea: $A \to BC$ ← variables

$A \to a$ ← terminal

Footnotes:

① What happens with ε?

S can't be on RHS

$S \to \varepsilon$ is allowed

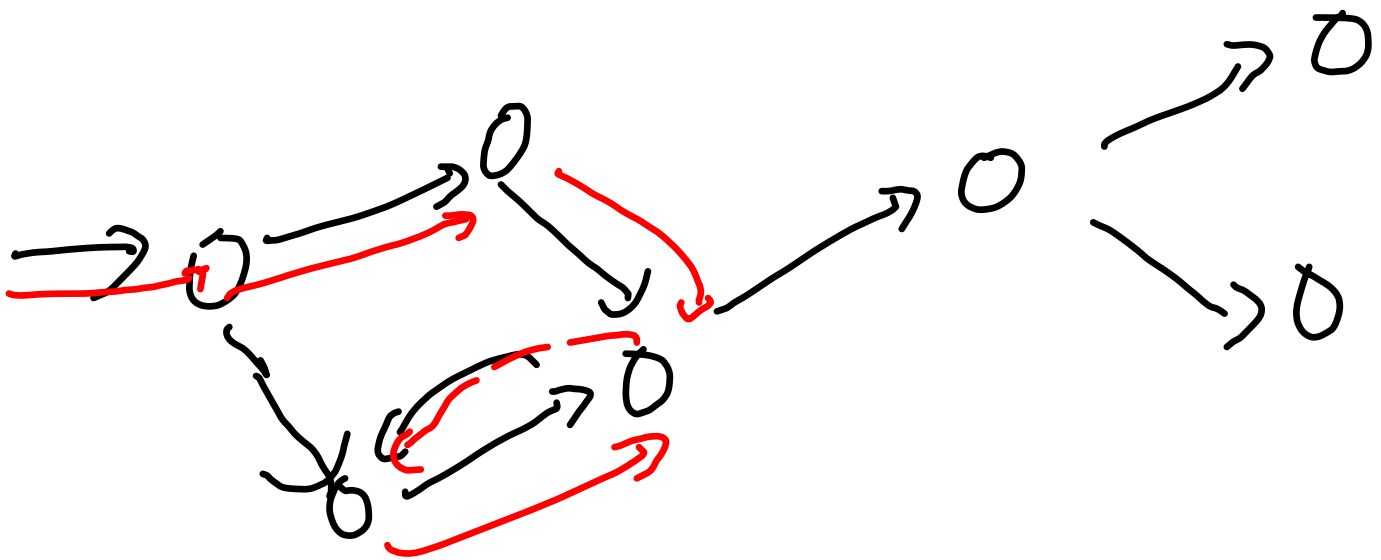
② Some people like to remove useless symbols

CNF good for proofs

relates string length
to height of parse
tree

Not good for practical parsing

Searches and looping



- Table of places visited
- Mark visited nodes

CNF conversion

input : $\langle V, \Sigma, R, S \rangle$

Step ① make start symbol

② remove ε-rules

$A \rightarrow \varepsilon$

③ remove unit-rules

$A \rightarrow B$

④ make RHS all length 2 (variables)

or length 1 terminal

step 1 : Add start Symbol S_0

$S_0 \rightarrow S$

step 2 : remove ε rules

---

to remove $A \to \varepsilon$

find all rules with A on RHS

$B \to \alpha A \beta$

add $B \to \alpha \beta$

if A occurs > 1 times

add all possible ways of deleting A

$B \to \alpha A \beta A \gamma$

add

$B \to \alpha \beta A \gamma$

$B \to \alpha A \beta \gamma$

$B \to \alpha \beta \gamma$

remove $B \to \varepsilon$

exists $A \to B$ add $A \to \varepsilon$

$S \to aB$ $S \to a$

---

remove $A \to \varepsilon$

exists rule

$S \to ASA$

add

$S \to SA$

$S \to AS$

$S \to S$

Exception

Keep a list of ε rules

That have been removed

&

never add These

back

Don't remove $S_0 \to \varepsilon$

Step 3:

Remove unit rules

To remove rule $A \to B$

Find all rules $B \to \alpha$

add $A \to \alpha$

Footnote: once removed

never add back

Step 4 Fixup

no RHS = ε

no RHS = A single variable

$A \rightarrow c_1 c_2 \ldots c_n$

if $n \geq 3$

$A \rightarrow c A'$

$A' \rightarrow c_2 \ldots c_n$

if $n = 2$ but c_1 or c_2 is terminal

$A \rightarrow c_1 B$ ← terminal

becomes

$A \rightarrow A' B$

$A' \rightarrow c_1$

$A \to a$

$S_0 \to SA$

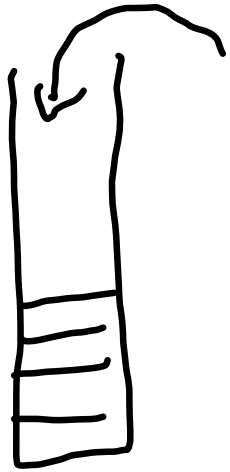
$A_1 \to SA$

$S_0 \to A_1$ not CNF

PDAs

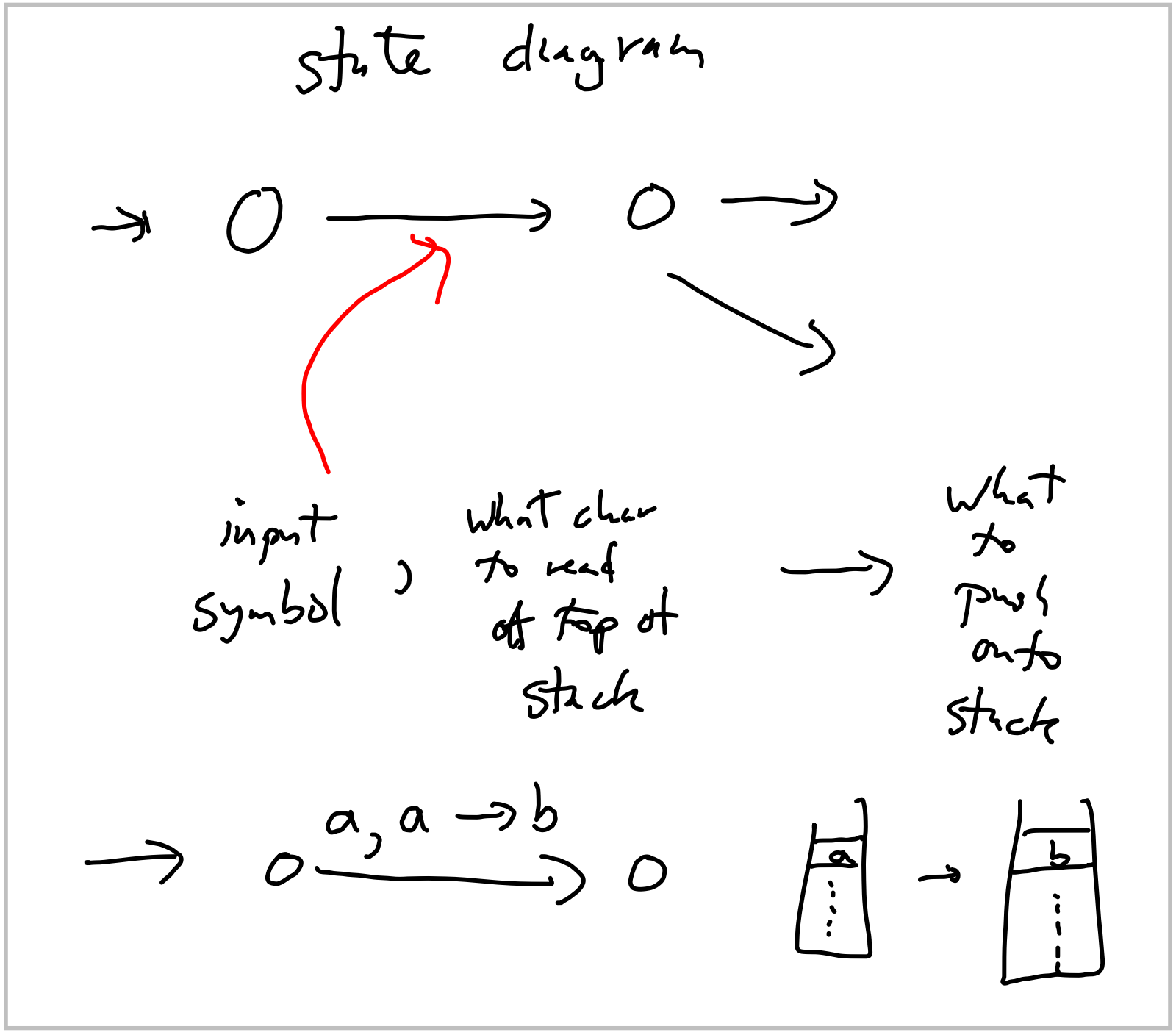
pushdown automata

an NFA w/ a stack



add and remove and read only at the top

pdas recognize languages of CFGs (in theory and a few apps)


state diagram
input symbol
what char to read off top of stack
what to push onto stack
a, a → b
a
b

L = well-bracketed expressions w/ a's

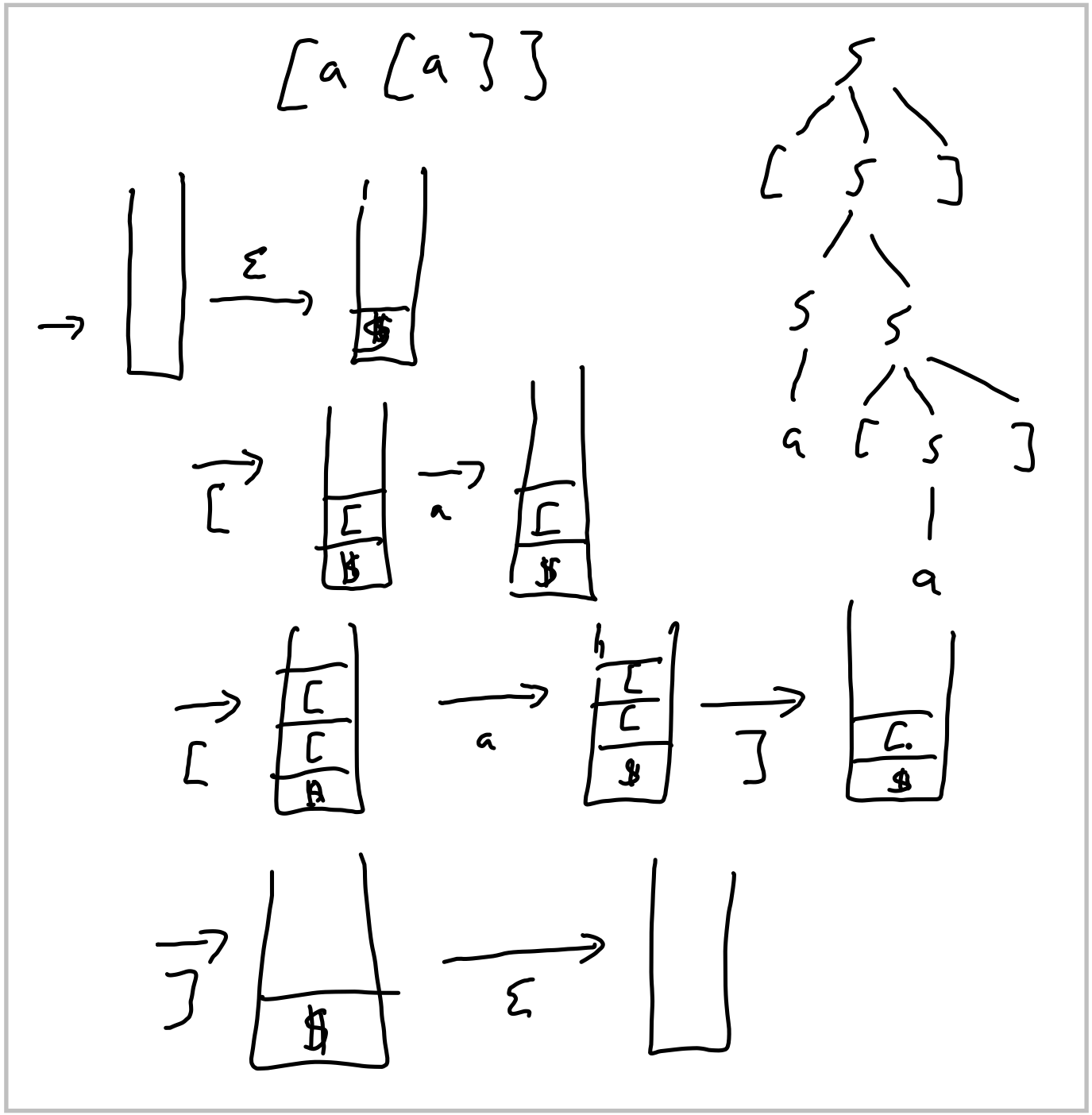
$S \to SS \mid \varepsilon \mid [S] \mid a$

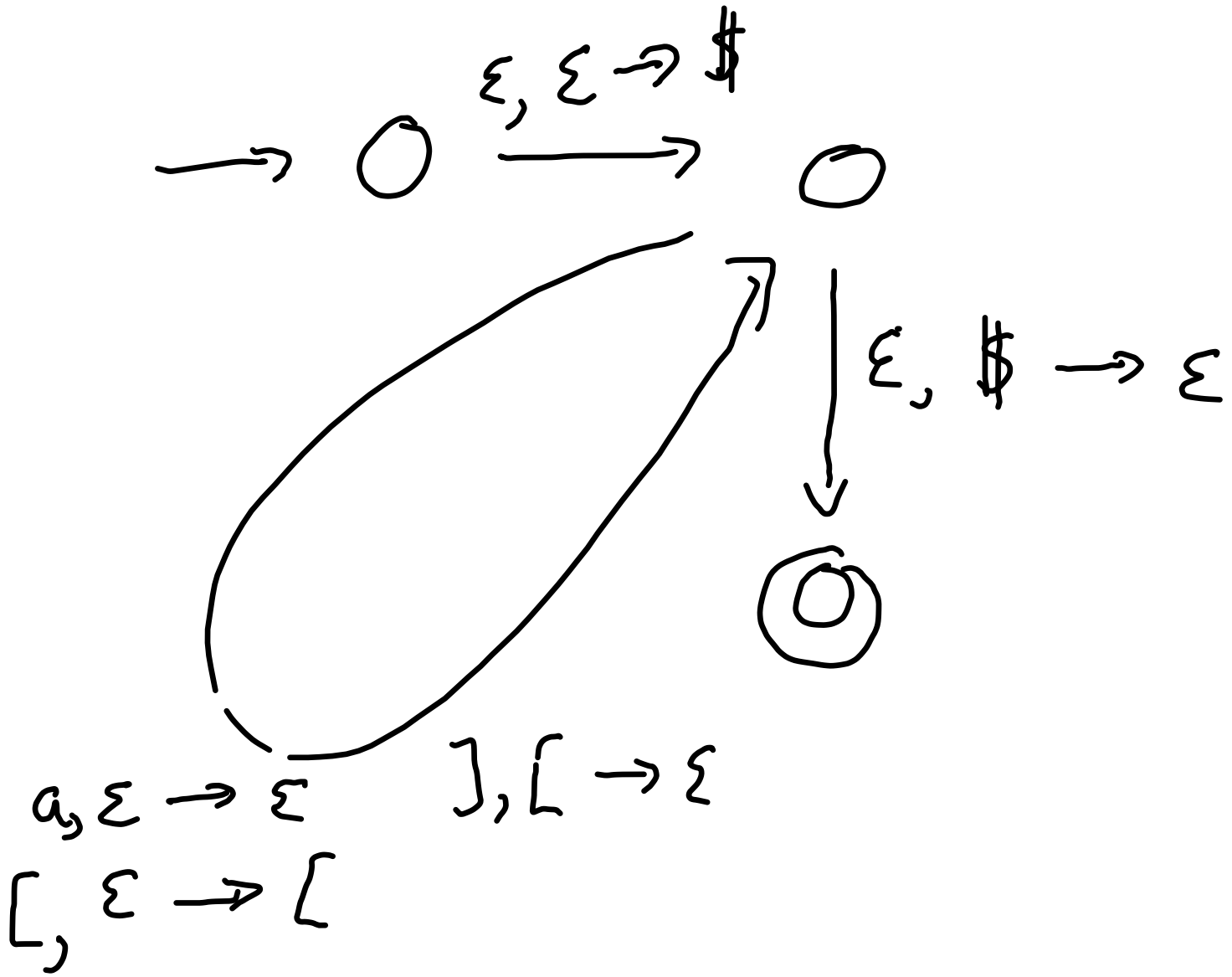
$S \to [S] \to [a]$ (a)

$S \to SS \to \ldots$

$[[\,][\,]]$ ok

$]]][[[$ not


[a [a]]
S
[S]
S S
a [S]
a
ε
$
[
a
]
ε


ε, ε → $
ε, $ → ε
a, ε → ε
], [→ ε
[, ε → [

official PDAs
only push single char
on each transition

$a, t \to s$ ← one char

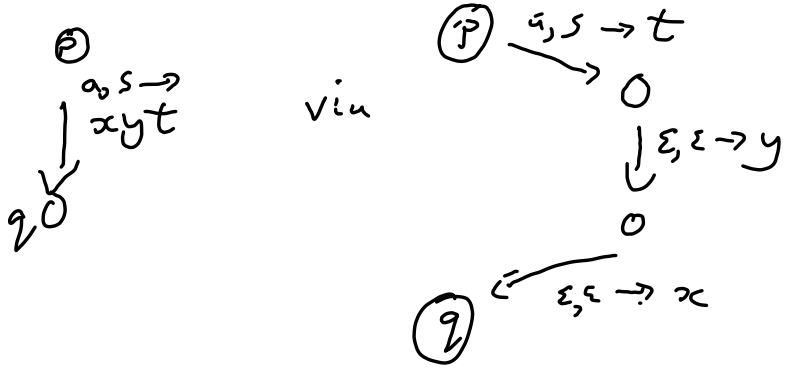
Can extend easily to

$a, t \to \alpha$

e.g. $a, t \to xyt$

top of stack ↑ below xy

officially a shorthand
in state diagrams



via