


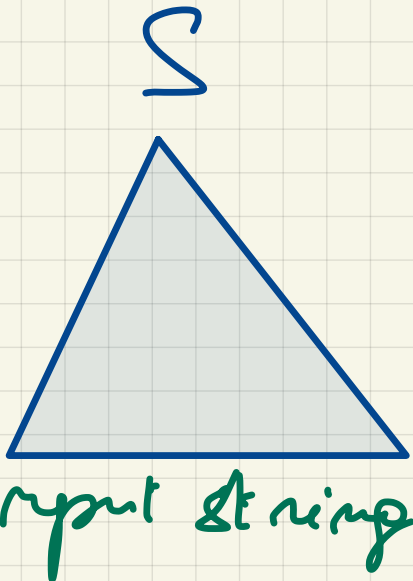
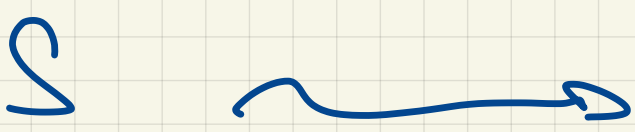
Move rules, 7th

/



Bottom-up parsers

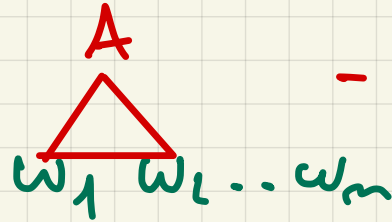
Top-down parser



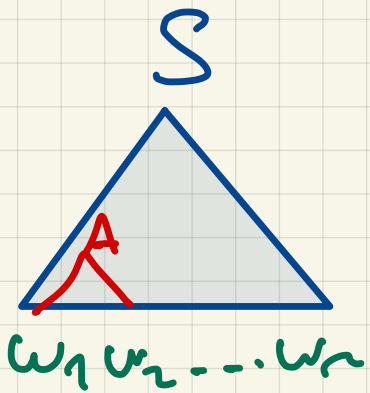
Bottom up

$A \rightarrow w_1 w_2$

input string



...



yacc tool \rightarrow GNU: Bison

\rightarrow yet another compiler compiler.

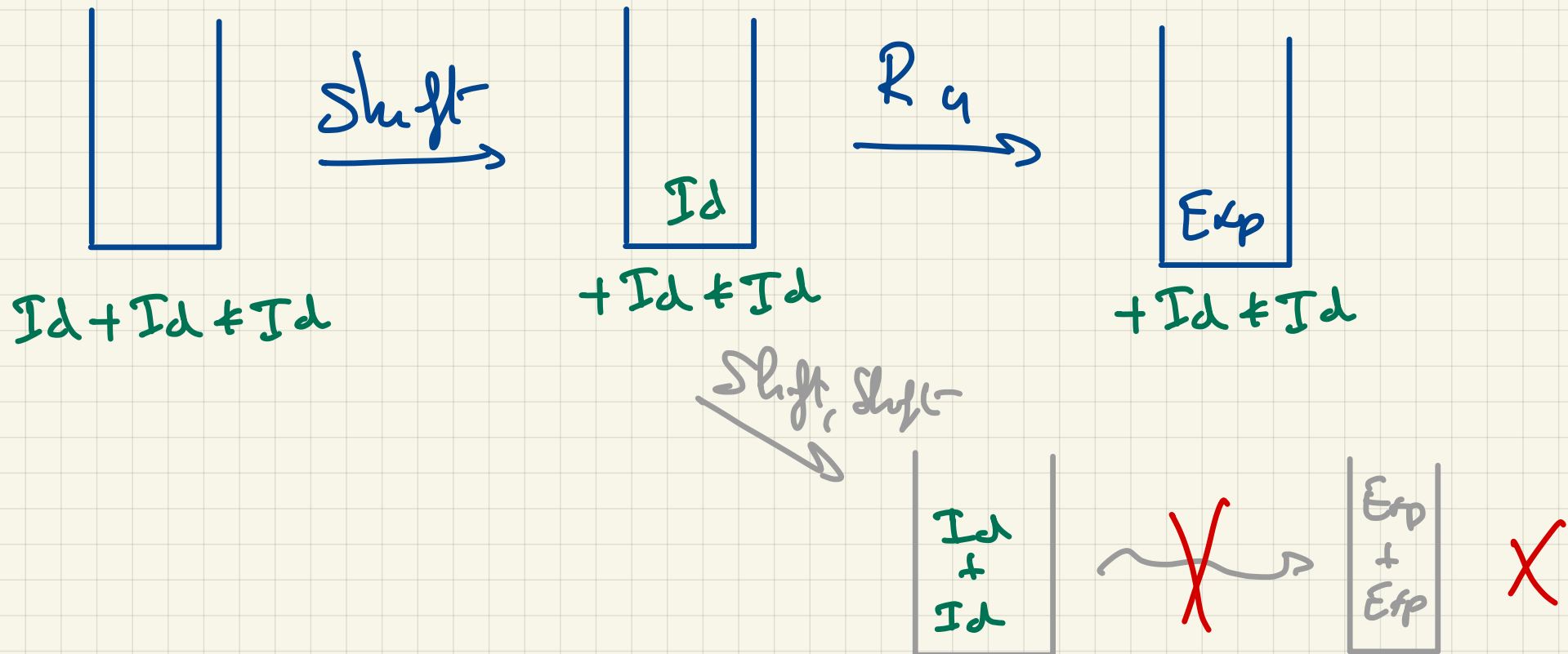
input: grammar + non actions,
(C code)

}
 \rightarrow parser that generates the actions.
LALR(1)

- | | | | |
|-----|-----|---|-----------|
| (1) | Exp | → | Exp + Exp |
| (2) | | → | Exp * Exp |
| (3) | | → | (Exp) |
| (4) | | → | Id |
| (5) | | → | Cst |

$Id + Id * Id$

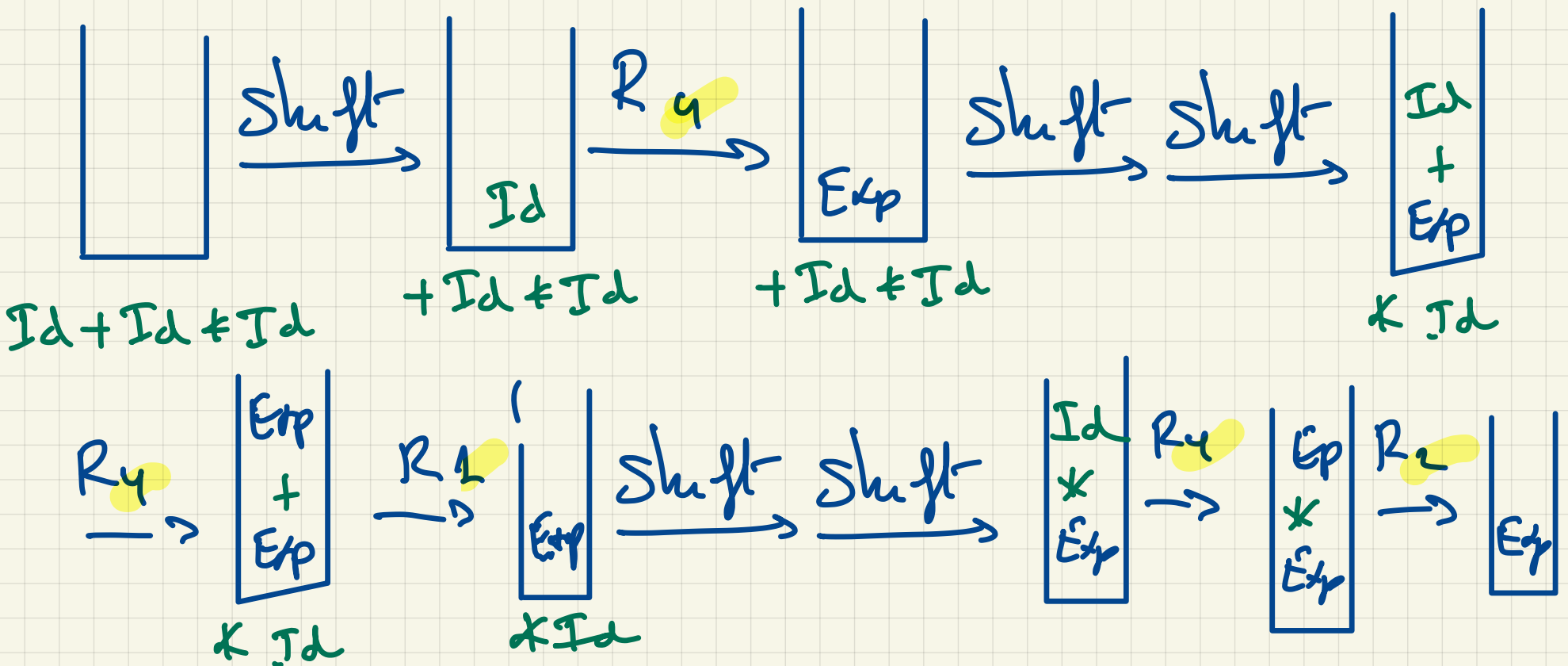
$\underline{\text{Exp}} \xRightarrow{2} \text{Exp} * \underline{\text{Exp}} \xRightarrow{4} \underline{\text{Exp}} * \text{Id} \xRightarrow{1} \text{Exp} + \underline{\text{Exp}} * \text{Id} \xRightarrow{4} \underline{\text{Exp}} + \text{Id} * \text{Id} \xRightarrow{4} \text{Id} + \text{Id} * \text{Id}.$



- | | | | |
|-----|-----|---|-----------|
| (1) | Exp | → | Exp + Exp |
| (2) | | → | Exp * Exp |
| (3) | | → | (Exp) |
| (4) | | → | Id |
| (5) | | → | Cst |

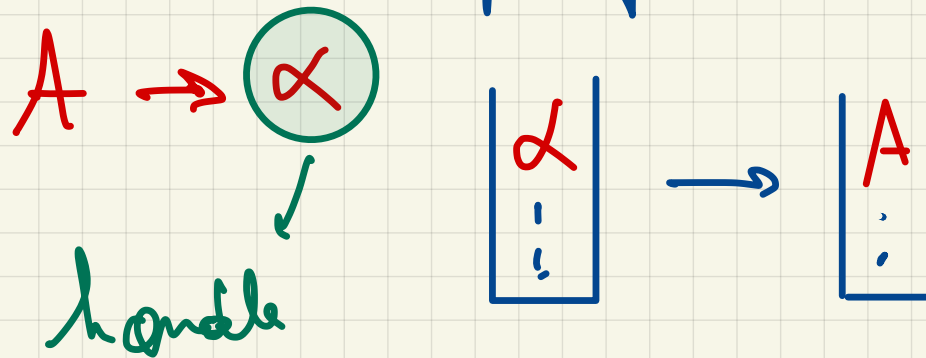
$Id + Id * Id$

$\underline{\text{Exp}} \xRightarrow{2} \text{Exp} * \underline{\text{Exp}} \xRightarrow{4} \underline{\text{Exp}} * \text{Id} \xRightarrow{1} \text{Exp} + \underline{\text{Exp}} * \text{Id} \xRightarrow{4} \underline{\text{Exp}} + \text{Id} * \text{Id} \xRightarrow{4} \text{Id} + \text{Id} * \text{Id}.$



The bottom-up parser can do two actions

- Reduce: replace the right-hand side of a rule by the left-hand side on the top of the stack



- Shift: read the next symbol from input and push it on the stack

Two kinds of conflicts

— Shift / Reduce : Should I reduce now or shift more?

<div style="border-left: 1px solid black; border-bottom: 1px solid black; padding: 5px; display: inline-block; vertical-align: middle;">b a</div>	(1) $A \rightarrow ab$? Reduce (1)
	(2) $B \rightarrow abC$? Shift?
	(3) $C \rightarrow c$	

— Reduce / Reduce : Which rule should I reduce?

Viable prefix

DEFINITION.

Proposition 6.2. Let $G = \langle V, T, P, S \rangle$ be a CFG, and let P'_G be its corresponding bottom-up parser. Let $\langle q_i, w, \gamma Z_0 \rangle$ be a configuration reached along an **accepting run** of P'_G (i.e., a configuration where P'_G is neither in one of the intermediate states introduced for the Reduce, nor in q_a). Then:

$$S \Rightarrow^* \gamma^R \cdot w$$

along a rightmost derivation.

γ : what we have put on the stack by means of Reduce and Shift

w : remaining input

This property will help us to identify the stack contents that can lead to an accepting run.

Definition 6.3 (Viable prefix). A *viable prefix* of a CFG $G = \langle V, T, P, S \rangle$ is a prefix of a right-sentential form that can occur (in reverse) on the stack during an accepting run of the associated bottom-up parser P'_G .

(1)	S	\rightarrow	$A\$$
(2)	A	\rightarrow	aCD
(3)		\rightarrow	ab
(4)	C	\rightarrow	c
(5)	D	\rightarrow	d

$$S \Rightarrow A\$ \Rightarrow aCD\$ \Rightarrow aCd\$ \\ \Rightarrow acd\$$$

$$S \Rightarrow A\$ \Rightarrow a\$$$

Viable prefixes:

$$\{ S, \epsilon, A, A\$, a, \overset{a,}{\checkmark} aC, aCD, \overset{\epsilon \checkmark}{\nearrow} aCd, ab \}$$

↓
We can shift \$
and then reduce (1)

aCD not
viable !!
↓
 $a\tau$

$$\begin{bmatrix} aCD \\ aC \end{bmatrix} \rightarrow X$$

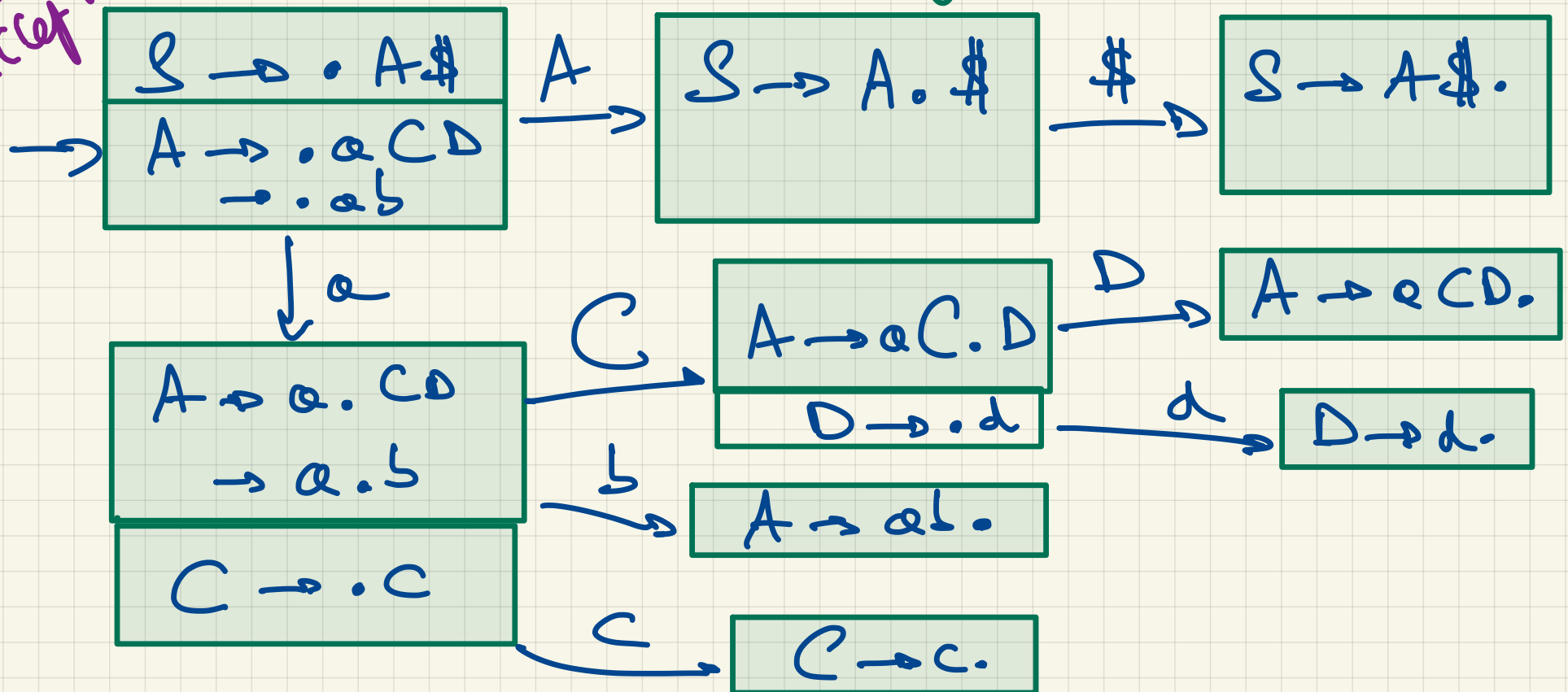
- | | | | |
|-----|-----|---------------|-------|
| (1) | S | \rightarrow | $A\$$ |
| (2) | A | \rightarrow | aCD |
| (3) | | \rightarrow | ab |
| (4) | C | \rightarrow | c |
| (5) | D | \rightarrow | d |

ADFA Not accept the viable prefixes.

$\{ \epsilon, A, A\$, a, aC, aCD, aCd, ab \}$
 \wedge
 $ac,$

CF SM = Canonical
finite state machine

all slots
accepting



Items

An item = grammar rule + •

$A \rightarrow \alpha_1 \cdot \alpha_2$ or $A \Rightarrow \alpha_1 \alpha_2$

↓
advancement of the parser.

→ The parser has already built α_1 on the stack and tries to add α_2 to obtain a handle.

Closure

if you have: $B \rightarrow \alpha_1 \circ A \alpha_2$
 \hookrightarrow Variable

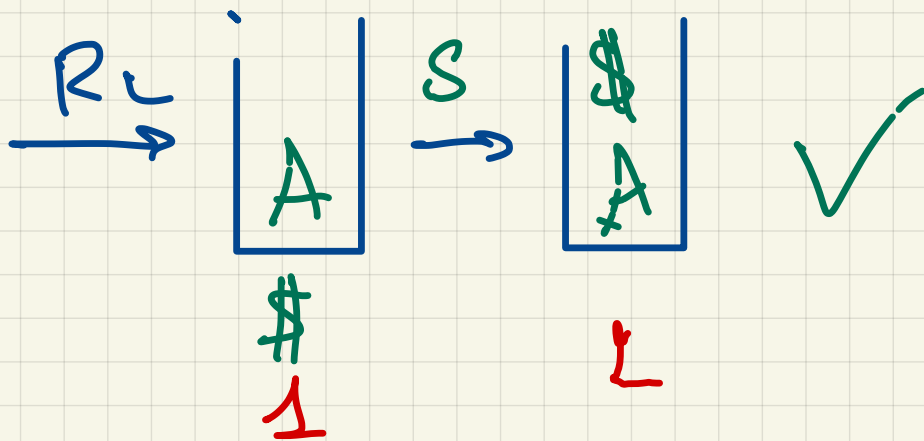
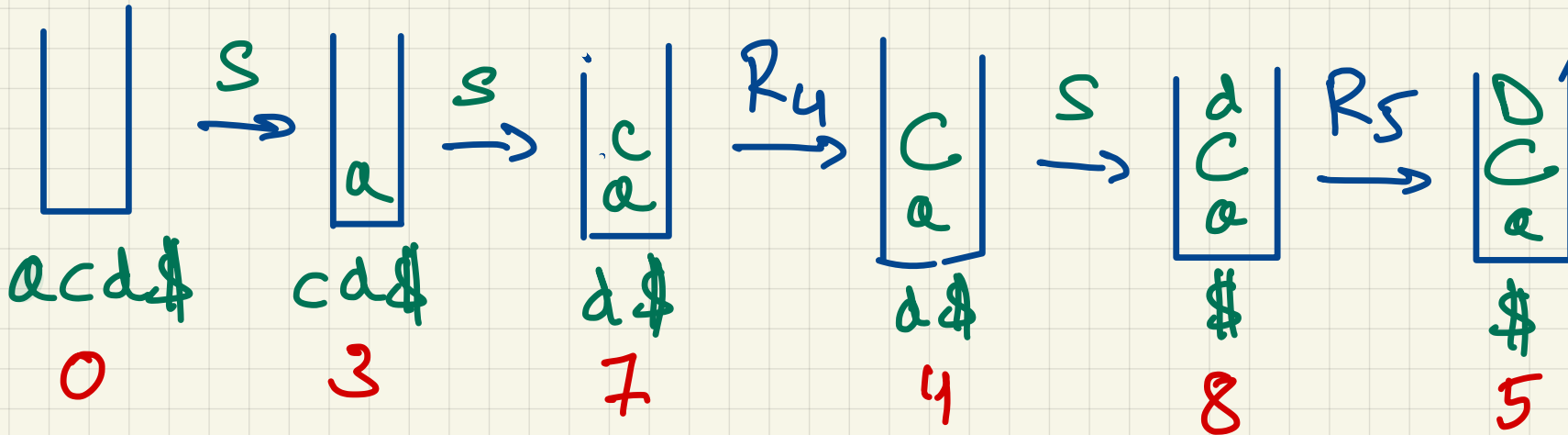
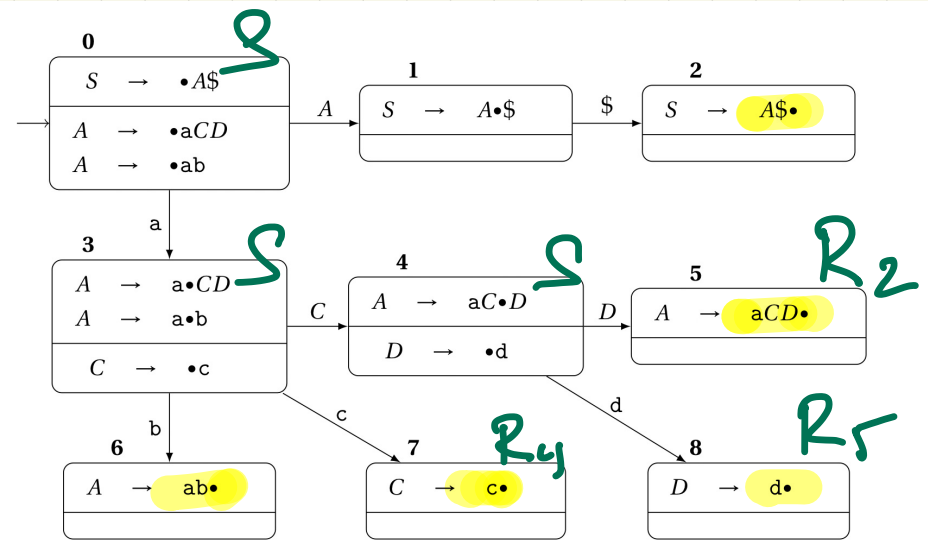
Then, you add to your state all the items of the form

$A \rightarrow \cdot \alpha$ ($A \rightarrow \alpha$ is a rule)
 sub-goal to obtain A on the stack.

Using the CFsn

ecd\$

- | | |
|-----|---------------------|
| (1) | $S \rightarrow A\$$ |
| (2) | $A \rightarrow aCD$ |
| (3) | $\rightarrow ab$ |
| (4) | $C \rightarrow c$ |
| (5) | $D \rightarrow d$ |



item of the form:

$A \rightarrow \alpha \cdot$

handle on the stack
 \rightarrow Reduce.