Equivalent Grammars
Parsing table
LR(1)
LR(0)
SLR
LALR(1)
translation scheme
copy rules
syntax-directed
PDA
Bottom up parser
REGEX
Predictive parsing
Useful symbols

Equivalent Grammars
REGEX

EXAM FLC 1

① VERIFY THE EQUIVALENCE OF THE FOLLOWING REGULAR EXPRESSIONS:

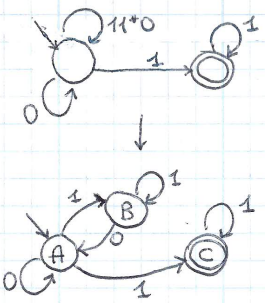
$(0|11*0)*11*$ (a)

$0*1(00*1|1)*$ (b)

CHECK EQUIVALENCE OF REGULAR EXPRESSIONS:
1. OBTAIN FINITE AUTOMATA FROM REGULAR EXPRESSIONS
2. MINIMIZE
⇒ Check if automata represent the same language

NOTE: in general, obtained automata are non deterministic, but in order to apply the minimization process you must derive the corresponding DETERMINISTIC automata.

(a) $(0|11*0)*11*$

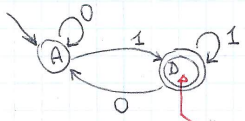


Automaton that represents the regular expression (a).
It is NON DETERMINISTIC because from the start state, e.g., there are two exiting transitions labeled with 1.
⇒ You must transform it in a deterministic automaton.

	0	1
A	A	BC
BC	A	BC

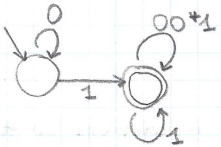
The equivalent deterministic automaton will have just 2 states.

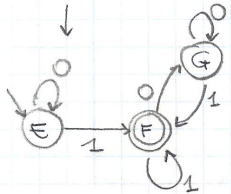
EQUIVALENT DETERMINISTIC AUTOMATON:



It is the final state because it contain a final state (C).

(b) $0^*1\,(00^*1 \mid 1)^*$





In this case there are not many exiting transictions from the same state with the same name
=> It is already a DETERMINISTIC AUTOMATON

Apply the MINIMIZATION procedure to the UNION of states of the two automata:

- Initially you have all final states in a class and all non final states in another class

$0: \{E, G, A\}, \{F, D\}$

- You have to check if pairs of states in the same class are equivalent also for the next step

From F with $0 \to G$, From D with $0 \to A$;
A and G are in the same class ✓
From F with $1 \to F$, From D with $1 \to D$;
F and D are in the same equivalence class ✓
$\Rightarrow$ $1: \{F, D\}$

Consider E and G:

$E \xrightarrow{0} E$; $G \xrightarrow{0} G$; E, G ∈ same equiv class ✓

$E \xrightarrow{1} F$; $G \xrightarrow{1} F$; ✓

$\Rightarrow$ $1: \{E, G\}$

Consider A

$A \xrightarrow{0} A$; A is equivalent to E and G ✓

$A \xrightarrow{1} D$; D is equivalent to F ✓

$\Rightarrow$ $1: \{E, G, A\}, \{F, D\}$

Now, in order to verify the equivalence of the two automata, you must check if the START STATES of the two automaton are in the SAME EQUIVALENCE CLASS.

$\{E, G, A\}, \{F, D\}$ YES!

$\Rightarrow$ The two automata are equivalent, then also the given regular expressions are equivalent.

PDA

(2) FIND A PUSHDOWN AUTOMATON ACCEPTING THE LANGUAGE

$$\{a^m b^{m+n} a^n \mid m >= 0,\ n >= 0\}$$

OBTAIN PUSHDOWN AUTOMATON FROM A GIVEN LANGUAGE
- Obtain a CONTEXT-FREE GRAMMAR generating the string
- Apply general method to obtain a pushdown automaton from a context-free grammar

It's also possible directly generate the pushdown automaton from the language but it's not convenient:
- you don't know if the language is deterministic or not, then you don't know if the language is generated from a deterministic or non deterministic automaton
- it's difficult decide how many states are required and also decide when and what push and pop

GENERATE A GRAMMAR for the string

$$a^m b^{m+n} a^n = a^m b^m b^n a^n$$

Thus, you have a number of "a" followed by the same number of "b", and then a number of "b" followed by the same number of "a".

$\underbrace{a^m b^m}\ \underbrace{b^n a^n}$

$a^m b^m$: STRING OF LENGTH $2m$ (A)

$b^n a^n$: STRING OF LENGTH $2n$ (B)

$$\begin{aligned} S &\to AB \\ A &\to aAb \mid \varepsilon \\ B &\to bBa \mid \varepsilon \end{aligned}$$

CONTEXT-FREE GRAMMAR

FROM GRAMMAR TO PUSHDOWN AUTOMATON

$$\delta(q, \varepsilon, S) = \{(q, AB)\}$$

$$\delta(q, \varepsilon, A) = \{(q, aAb), (q, \varepsilon)\}$$

$$\delta(q, \varepsilon, B) = \{(q, bBa), (q, \varepsilon)\}$$

$$\delta(q, a, a) = \delta(q, b, b) = \{(q, \varepsilon)\}$$

Consider the cases in which the input symbol is the same of the symbol on top of the stack, for EACH TERMINAL SYMBOL IN THE GRAMMAR.

For each terminal symbol in the grammar, introduce a transition that advances one position on the input and deletes top symbol of the stack (that is replaced by ε).

Predictive parsing

3 FIND THE PREDICTIVE PARSING TABLE FOR THE FOLLOWING GRAMMAR:

$S \to AB$
$S \to B$
$A \to aA$
$A \to \varepsilon$
$B \to bB$
$B \to \varepsilon$

PREDICTIVE PARSING TABLE ≡ LL(1) PARSING TABLE

BUILD LL(1) PARSING TABLE
- Evaluate FIRST and FOLLOW functions for the symbols in the grammar

non terminals	nullable	FIRST	FOLLOW
S	YES	a, b	$ (1)
A	YES	a	b (2), $ (3)
B	YES	b	$

(1) S does not apper in the right side of any rule, then there are not more symbols following S

(2) A is followed by B ⇒ FOLLOW(A) = FIRST(B) = b
(3) A is nullable ; A can be the last symbol in the right side ⇒ in FOLLOW(A) you have also FOLLOW(S)

PARSING TABLE:
one ROW for each NON TERMINAL symbol
one COLUMN for each TERMINAL symbol

	a	b	$
S	S→AB	S→AB S→B	S→AB S→B
A	A→aA	A→ε	A→ε
B		B→bB	B→ε

The goal is construct the complete parsing table, then you must go on even if there is a conflict (⇒ the grammar is NOT LL(1))

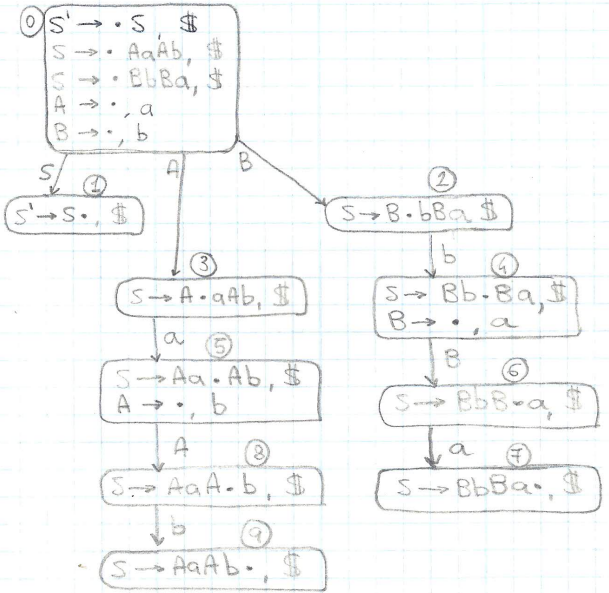
LR(1)
Parsing table

4 FIND THE LR(1) PARSING TABLE FOR THE FOLLOWING GRAMMAR:

1) $S \to AaAb$
2) $S \to BbBa$
3) $A \to \varepsilon$
4) $B \to \varepsilon$

The start symbol of the grammar S is used in more than one rule, then you introduce another rule (and a new non terminal symbol): $S' \to S$

- Develop FINITE STATE AUTOMATON corresponding to LR(1) items



STATE	ACTION			GOTO		
	a	b	$	S	A	B
0	r3	r4		1	3	2
1			accept			
2		s4				
3	s5					
4	r4					6
5		r3			8	
6	s7					
7			r2			
8		s9				
9			r1			

EXAM FLC 2

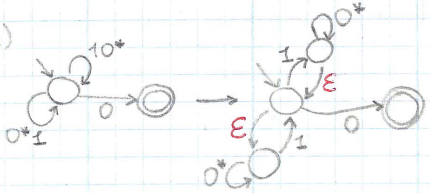
REGEX

1. FIND THE MINIMUM STATE AUTOMATON ACCEPTING THE UNION OF THE LANGUAGES DENOTED BY THE FOLLOWING REGULAR EXPRESSIONS:

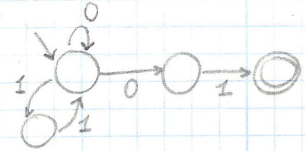
$(0^*1 \mid 10^*)^*0$ (a)

$(0 \mid 11)^*01$ (b)

(a)



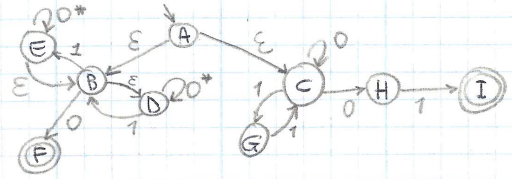
(b)



MAKE THE UNINION:

- INTRODUCE A NEW START STATE
- CONNECT (by means of ε) THE NEW START STATE WITH THE START STATES OF THE TWO AUTOMATA

ε-NFA ↳

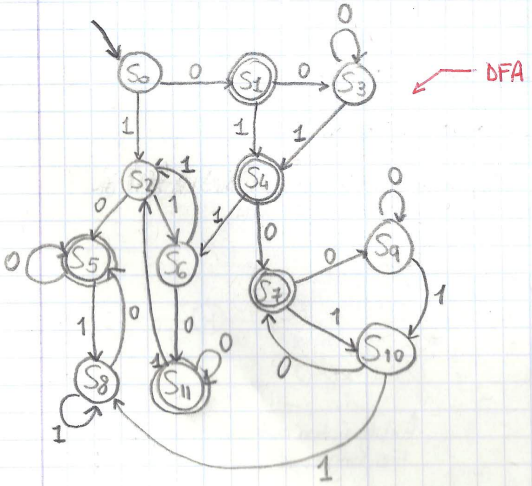


↥ Automaton accepting the union of the two languages.

In order to MINIMIZE, first you must obtain the equivalent DETERMINISTIC automaton.

	0	1
S_0	ABCD ~~CDFH~~	~~BDEG~~
S_1	CDFH ~~CDH~~	~~BDGI~~
S_2	BDEG ~~BDEF~~	~~BCDE~~
S_3	CDH ~~CDH~~	~~BDGI~~
S_4	BDGI ~~DF~~	~~BCDE~~
S_5	BDEF ~~BDEF~~	~~BDE~~
S_6	BCDE ~~BCDEFH~~	~~BDEG~~
S_7	DF ~~D~~	~~BD~~
S_8	BDE ~~BDEF~~	~~BDE~~
S_9	D ~~D~~	~~BD~~
S_{10}	BD ~~DF~~	~~BDE~~
S_{11}	BCDEFH ~~BCDEFH~~	~~B DEG~~

← You must apply the ε-CLOSURE ⇒ when you are in A, you are also in B and C, and from B also in D



DFA

EQUIVALENT STATES:

Π_0: $\{S_0, S_2, S_3, S_6, S_8, S_9, S_{10}\}$, $\{S_1, S_4, S_5, S_7, S_{11}\}$

S_0-S_2 ✓, S_0-S_3 ✗, S_0-S_6 ✓, S_0-S_8 ✓, S_0-S_9 ✗, S_3-S_9 ✗, S_0-S_{10} ✓ → $S_0, S_2, S_6, S_8, S_{10}$

S_1-S_4 ✗, S_1-S_5 ✗, S_4-S_5 ✓, S_1-S_7 ✗, S_4-S_7 ✗, S_1-S_{11} ✗, S_4-S_{11} ✓ → S_1, S_4, S_5, S_{11}, S_7

Π_1: $\{S_0, S_2, S_6, S_8, S_{10}\}$, $\{S_4, S_5, S_{11}\}$, $\{S_3\}$, $\{S_9\}$, $\{S_1\}$, $\{S_7\}$

S_0-S_2: $S_0 \xrightarrow{0} S_1$, $S_2 \xrightarrow{0} S_5$ ✗ → S_0

S_0-S_6: $S_0 \xrightarrow{0} S_1$, $S_6 \xrightarrow{0} S_{11}$ ✗ → S_2, S_6, S_8

S_2-S_6: $S_2 \xrightarrow{1} S_6$, $S_6 \xrightarrow{1} S_2$ ✓ → S_{10}

S_0-S_8: $S_0 \xrightarrow{0} S_1$, $S_8 \xrightarrow{0} S_5$ ✗ → S_4

S_2-S_8: $S_2 \xrightarrow{1} S_6$, $S_8 \xrightarrow{1} S_8$ ✓ → S_5, S_{11}

S_0-S_{10}: $S_0 \xrightarrow{0} S_1$, $S_{10} \xrightarrow{0} S_7$ ✗

S_4-S_5: $S_4 \xrightarrow{0} S_7$, $S_5 \xrightarrow{0} S_5$ ✗

S_4-S_{11}: $S_4 \xrightarrow{0} S_7$, $S_{11} \xrightarrow{0} S_{11}$ ✗

S_5-S_{11}: $S_5 \xrightarrow{0} S_5$, $S_{11} \xrightarrow{0} S_{11}$, $S_5 \xrightarrow{1} S_8$, $S_{11} \xrightarrow{1} S_2$ ✓

Π_2: $\{S_0\}$, $\{S_1\}$, $\{S_2, S_6, S_8\}$, $\{S_3\}$, $\{S_4\}$, $\{S_5, S_{11}\}$, $\{S_7\}$, $\{S_9\}$, $\{S_{10}\}$

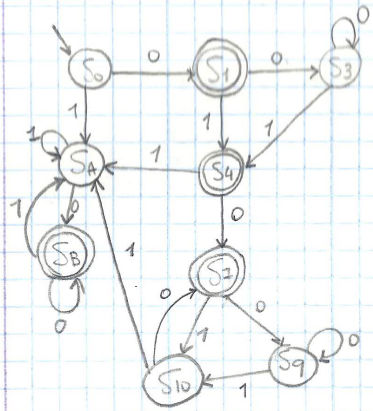
S_2-S_6: $S_2 \xrightarrow{0} S_5$, $S_6 \xrightarrow{0} S_{11}$, $S_2 \xrightarrow{1} S_6$, $S_6 \xrightarrow{1} S_2$ ✓ → S_2, S_6, S_8 (NO CHANGE)

S_2-S_8: $S_2 \xrightarrow{0} S_5$, $S_8 \xrightarrow{0} S_5$, $S_2 \xrightarrow{1} S_6$, $S_8 \xrightarrow{1} S_8$ ✓

S_5-S_{11}: $S_5 \xrightarrow{0} S_5$, $S_{11} \xrightarrow{0} S_{11}$, $S_5 \xrightarrow{1} S_8$, $S_{11} \xrightarrow{1} S_2$ ✓ → S_5, S_{11} (NO CHANGE)

Π_3: $\{S_0\}$, $\{S_1\}$, $\underbrace{\{S_2, S_6, S_8\}}_{S_A}$, $\{S_3\}$, $\{S_4\}$, $\underbrace{\{S_5, S_{11}\}}_{S_B}$, $\{S_7\}$, $\{S_9\}$, $\{S_{10}\}$

MINIMUM-STATE DFA



PDA

② FIND A PUSHDOWN AUTOMATON ACCEPTING THE LANGUAGE

$\{ b^{2m} a^{n+m+1} b^{2n} \mid m>=0,\ n>=0 \}$

$b^{2m} a^n a^m a b^{2n} \rightarrow \underbrace{b^{2m} a^m}_{A}\ a\ \underbrace{a^n b^{2n}}_{B}$

$S \rightarrow A\,a\,B$ ← Context-free grammar
$A \rightarrow bbAa \mid \varepsilon$
$B \rightarrow aBbb \mid \varepsilon$

$\delta = \{$
$\delta(q, \varepsilon, A) = \{(q, \alpha)\ \forall A \rightarrow \alpha \in P\}$
$\delta(q, a, a) = \{(q, \varepsilon)\}\ \forall a \in T$
$\}$

$\delta(q, \varepsilon, S) = \{(q, AaB)\}$
$\delta(q, \varepsilon, A) = \{(q, bbAa), (q, \varepsilon)\}$
$\delta(q, \varepsilon, B) = \{(q, aBbb), (q, \varepsilon)\}$
$\delta(q, a, a) = \{(q, \varepsilon)\}$
$\delta\{q, b, b) = \{(q, \varepsilon)\}$

#Bottom up parser

④ TRANSFORM THE FOLLOWING SDT SO THAT IT CAN BE IMPLEMENTED BY A BOTTOM-UP PARSER.
THEN INDICATE WHICH STRING WILL BE PRINTED WHEN THE INPUT IS "ccaba":

$S \rightarrow c\ \{print\ "x"\}\ S\ S$
$S \rightarrow \{print\ "y"\}\ a$
$S \rightarrow b\ \{print\ "z"\}$

SDT = Syntax-Directed Translation Scheme
is an SDD (Syntax-Directed Definition) with the actions of each semantic rule EMBEDDED at some positions in the RIGHT SIDE of the associated production.

Postfix SDT = SDT having all actions at the right ends of the productions.

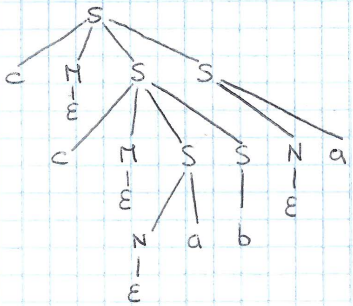
ACTIONS in a postfix SDT can be executed by a BOTTOMUP parser along with REDUCTIONS.

S-attributed SDD's can be converted to postfix SDT's simply by PLACING EACH ACTION at the RIGHT END of the associated production.

$S \rightarrow cMSS$
$M \rightarrow \varepsilon \ \{print\ "x"\}$
$S \rightarrow Na$
$N \rightarrow \varepsilon \ \{print\ "y"\}$
$S \rightarrow b \ \{print\ "z"\}$

← TRANSLATOR THAT CAN BE IMPLEMENTED BY A BOTTOM-UP PARSER.

Build a parse tree that can be constructed by the parser, given the input string "ccaba"



In order to perform the semantic actions you have to follow the order in which the tree would be generated by a BOTTOM-UP PARSER. Thus, visit the tree in a bottom-up way from left to right and execute print actions any time you perform a reduce action according to same rule.

OUTPUT STRING: xxyzy

PDA

EXAM FLC 3

② FIND A PUSHDOWN AUTOMATON ACCEPTING THE LANGUAGE:

$\{ (a^n b^n)^m c^m \mid m >= 0,\ n >= 0 \}$

The indicated language is not CFL, therefore, no PDA accepting it can be defined.

translation scheme # copy rules

④ TRANSFORM THE FOLLOWING TRANSLACTION SCHEME SO THAT THE INHERITED ATTRIBUTES ARE DEFINED BY COPY RULES:

(1) $A \rightarrow aBC \ \{A.a := f(B.b)\}$
(2) $B \rightarrow bC \{A.b := g(C.c)\} A \{B.b := k(b.x)\}$
(3) $C \rightarrow cB \{A.b := h(C.c)\} A \{C.c := k(c.x)\}$

- Move semantic actions inside grammar rules at the end of the right side
- The given L-translaction scheme can be transformed by using marks

(1) Evaluation of a synthesized attribute of the symbol A (A is on the left of the rule) ⇒ rule (1) has not to be changed

(2)(3) B.b and C.c are synthesized attributes, then the corresponding actions are already in place.
You must move and replace the actions inside the rules, taking into account that they cannot be simply eliminated.

because they are not simply copy rules, but evaluate inherited attributes.

(2) $B \to bC\{M.i := C.c\}\ M\ \{A.b := M.s\}\ A\ \{B.b := k(b.x)\}$

↑ copy rule
(can be eliminated by retrieving the position of the symbol in the stack, when the attribute M.i is required)

$M \to \varepsilon\ \{M.s := g(M.i)\}$

(3) $C \to cB\ \{N.i := C.c\}\ N\ \{A.b := N.s\}\ A\ \{C.c := k(c.x)\}$

$N \to \varepsilon\ \{N.s := h(N.i)\}$

#Useful symbols

EXAM FLC 4

(2) FIND A GRAMMAR EQUIVALENT TO THE FOLLOWING ONE, HAVING ONLY USEFUL SYMBOLS:

$S \to aAa \mid bCBb$
$A \to aSa \mid bA \mid b$
$B \to bSBb \mid aBCb$
$C \to aBCa \mid aAS \mid a$

In a CONTEXT-FREE GRAMMAR,
a useful symbol X generates a non-empty language:
$X \Rightarrow^* x \in T^*$
a useful symbol is reachable (from the start symbol)
$S \Rightarrow^* \alpha X \beta$

ELIMINATE USELESS SYMBOLS:
- eliminate symbols generating an empty language
- eliminate unreachable symbols

ALGORITHM TO FIND USEFUL SYMBOLS
- finding symbols generating non-empty languages
  - every symbol of **T** generates a non-empty language
  - if $A \to \alpha$ and all symbols in α generate a non-empty language, then A generates a non-empty language
- finding reachable symbols
  - the start symbol **S** is reachable
  - if $A \to \alpha$ and A is reachable, all symbols in α are reachable

Symbols generating a non-empty language:
$\{b, a\} \cup \{A, C\} \cup \{S\}$
Symbols generating an empty language:
$\{B\}$
↑ All rules having B on the left have also B on the right

$S \rightarrow aAa$
$A \rightarrow aSa \mid bA \mid b$
$C \rightarrow aAS \mid a$

Reachable symbols : $\{S\} \cup \{a, A\} \cup \{b\}$
Unreachable symbols : $\{C\}$

⇒ EQUIVALENT GRAMMAR :

$S \rightarrow aAa$
$A \rightarrow aSa \mid bA \mid b$

Equivalent Grammars

EXAM FLC S

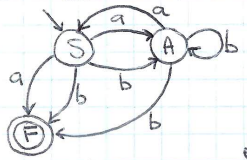
1. VERIFY THE EQUIVALENCE BETWEEN THE FOLLOWING GRAMMARS:

(a) $S \rightarrow Aa \mid Ab \mid Sb \mid a \mid b$ ← LEFT REGULAR GRAMMAR
$A \rightarrow Sa$

(b) $S \rightarrow aA \mid bA \mid a \mid b$ ← RIGHT REGULAR GRAMMAR
$A \rightarrow aS \mid bA \mid b$

Verify EQUIVALENCE:
- obtain AUTOMATA from different language representations
- apply MINIMIZATION technique

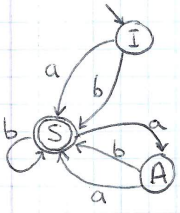
(b) [RIGHT REGULAR GRAMMAR]



Start symbol of the grammar corresponds to start state of automaton

Each non-terminal symbol corresponds to a state in the automaton.

(a) [LEFT REGULAR GRAMMAR]



The start state does not correspond to any symbol in the grammar.
The FINAL STATE in the automaton corresponds to the START of the grammar.

Given the corresponding FA, you must transform them in DETERMINISTIC FA.

The automaton corresponding to the RIGHT regular grammar is **not** deterministic.

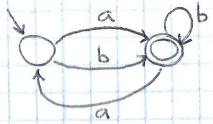
	a	b
S	AF	AF
AF	S	AF

← DETERMINISTIC FINITE AUTOMATON FOR RIGHT REGULAR GRAMMAR

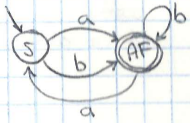
CHECK EQUIVALENCE:

(a) S is final
I and A are non final
You must check if I and A are equivalent:
from I and A you can reach only S state
with the same symbols ⇒ I and A are equivalent

EQUIVALENT MINIMUM-STATE FA:



(b)



⇒ THE TWO GRAMMARS ARE EQUIVALENT

PDA

② FIND A PUSHDOWN AUTOMATON ACCEPTING THE STRINGS IN

$\{a^n b^m \mid 2n <= m \leq 3n\}$

$S \rightarrow \; a\,S\,bb \mid a\,S\,bbb \mid \varepsilon$

$\delta(q, \varepsilon, S) = \{(q, aSbb), (q, aSbbb), (q, \varepsilon)\}$

$\delta(q, a, a) = \delta(q, b, b) = \{(q, \varepsilon)\}$

translation scheme

syntax-directed

③ PRODUCE AN L-ATTRIBUTED TRANSLATION SCHEME EQUIVALENT TO THE FOLLOWING SYNTAX-DIRECTED DEFINITION:

$A \rightarrow BaCD$

$A.x = f(B.y, C.z)$ (1)
$C.z = g(B.x)$ (2)
$B.y = h(A.y)$ (3)
$D.z = K(A, w)$ (4)

SYNTAX-DIRECTED

Syntax-directed definition:
you have SEMANTIC rules and SYNTACTIC rules in TWO DIFFERENT sets.
Each rule has a piece of code that represents the semantic action to be performed when the syntax rule is used by the parser.
In this case, you DON'T know **WHEN** the semantic action must be performed during the parsing.

L-ATTRIBUTE TRANSLATION SCHEME

L-attribute translation scheme:
Distinguish between synthesized and inherited attributes.
Evaluate synthesized attributes AFTER all symbols in the right side have been considered.
Evaluate inherited attributes BEFORE the symbols to which refer.

Insert semantic actions in the body of the right side of the rule (taking into account the kind of action).

(1) A.x is a synthesized attribute, then the corresponding rule can be placed at the end

(2,3,4) C, B and D are symbols in the RIGHT side of the rule, the all of them attributes are inherited attributes.
⇒ Semantic actions evaluating them should be placed BEFORE the symbol corresponding to the associated attribute.

$$A \rightarrow \{B.y = h(A.y)\}\, B\, a\, \{C.z = g(B.x)\}\, C\, \{D.z = k(A.w)\}\, D\, \{A.x = f(B.y, C.z)\}$$

NOTE: in an L-attributed translation scheme all DEPENDENCIES must move from left to right.
⇒ B must depend on the value of an element that is on the left of B ⇒ ok!
(same for C and D)

Predictive parsing

④ ELIMINATE LEFT RECURSION FROM THE FOLLOWING GRAMMAR AND PRODUCE ITS PREDICTIVE PARSING TABLE

$S \rightarrow Aa \mid Bc$
$A \rightarrow aB \mid Bb$
$B \rightarrow Sc \mid a$

NOTE: any topdown parsing strategy requires that the grammar is a NON LEFT RECURSIVE GRAMMAR

The current grammar is a left-recursive grammar because there is at least one indirect recursion (B can generate S, S can generate B).

- ELIMINATE LEFT RECURSION

$S \rightarrow Aa \mid Bc$
$A \rightarrow aB \mid Bb$
$B \rightarrow Aac \mid Bcc \mid a$ ← Also A is before B

⇓

$S \rightarrow Aa \mid Bc$
$A \rightarrow aB \mid Bb$
$B \rightarrow aBac \mid Bbac \mid Bcc \mid a$

⇓

$X \rightarrow X\alpha \mid \beta$ ⇓ $\begin{cases} X \rightarrow \beta X' \\ X' \rightarrow \alpha X' \mid \varepsilon \end{cases}$

In this case:
$X \equiv B$
$\alpha \equiv \{(bac), (cc)\}$
$\beta \equiv$ any rule that does not begin with B $\{(aBac), (a)\}$

$S \rightarrow Aa \mid Bc$
$A \rightarrow aB \mid Bb$
$B \rightarrow aBacB' \mid aB'$
$B' \rightarrow bacB' \mid ccB' \mid \varepsilon$

- PREDICTIVE PARSING TABLE

	NULLABLE	FIRST	FOLLOW
S	FALSE	a	$
A	FALSE	a	a
B	FALSE	a	b, c, a
B'	TRUE	b, c	a, b, c

BECAUSE B IS THE LAST SYMBOL IN $A \rightarrow aB$ ⇒ FOLLOW(B) CONTAINS ALSO FOLLOW(A).
FOR THE SAME REASON, FOLLOW(B') CONTAINS ALSO FOLLOW(B)

	a	b	c	$
S	S → Aa S → Ba			
A	A → aB A → Bb			
B	B → aBacB' B → aB			
B'	B' → ε	B' → bacB' B' → ε	B' → acB' B' → ε	

EACH TIME THERE ARE TWO ENTRIES IN THE SAME CELL ⇒ CONFUCT

B' → ε MUST BE USED WITH ANY INPUT SYMBOLS IN FOLLOW(B')

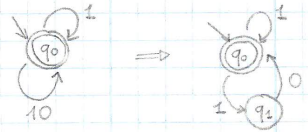
EXAM FLC 6

REGEX

① FIND A REGULAR EXPRESSION DENOTING THE COMPLEMENT OF THE LANGUAGE REPRESENTED BY:

$(1 \mid 10)^*$

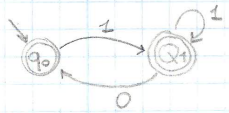
AUTOMATON:



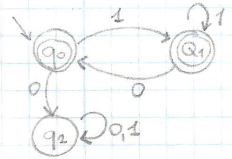
DETERMINISTIC AUTOMATON

	1	0
q0	q0, q1	/
q0, q1	q0, q1	q0

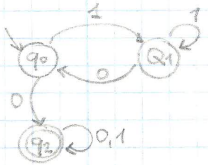
q0, q1 → Q1



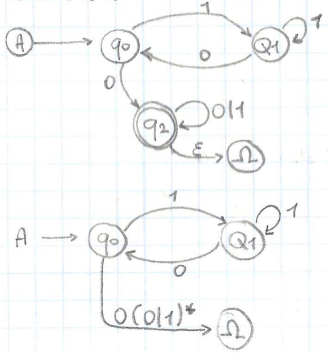
COMPLETELY SPECIFIED DFA



COMPLEMENT OF INITIAL DFA



COMPLEMENT OF THE LANGUAGE :



$A \xrightarrow{(11^*0)^* 0 (011)^*} \Omega$

⇒ REGULAR EXPRESSION OF THE COMPLEMENT OF THE LANGUAGE:

$(11^*0)^* 0 (011)^*$

PDA

② FIND A PUSHDOWN AUTOMATON ACCEPTING THE LANGUAGE

$\{(10)^n 11 (01)^n \mid n \geq 0\}$

$S \rightarrow 10\,S\,01 \mid 11$

$\delta(q, \varepsilon, S) = \{(q, 10S01), (q, 11)\}$

$\delta(q, 0, 0) = \delta(q, 1, 1) = \{(q, \varepsilon)\}$

EXAM FLC 7 # PDA

② FIND A PUSHDOWN AUTOMATON ACCEPTING ALL THE STRINGS IN $\{a,b,c\}^*$ with just one "c" and an equal number of "a" and "b"

$S \to abS \mid aSb \mid Sab \mid$
$baS \mid bSa \mid Sba \mid$
c

$\delta(q,\varepsilon,S) = \{(q,abS),(q,aSb),(q,Sab),$
$(q,baS),(q,bSa),(q,Sba),$
$(q,c)\}$

$\delta(q,a,a) = \delta(q,b,b) = \delta(q,c,c) = \{(q,\varepsilon)\}$

EXAM FLC 9 # PDA

③ FIND A PUSHDOWN AUTOMATON ACCEPTING THE STRINGS IN $\{0,1\}^*$ HAVING THE SAME NUMBER OF 0 AND 1.

$S \to 01S \mid 0S1 \mid S01 \mid$
$10S \mid 1S0 \mid S10 \mid$
ε

$\delta(q,\varepsilon,S) = \{(q,01S),(q,0S1),(q,S01),$
$(q,10S),(q,1S0),(q,S10),$
$(q,\varepsilon)\}$

$\delta(q,1,1) = \delta(q,0,0) = \{(q,\varepsilon)\}$

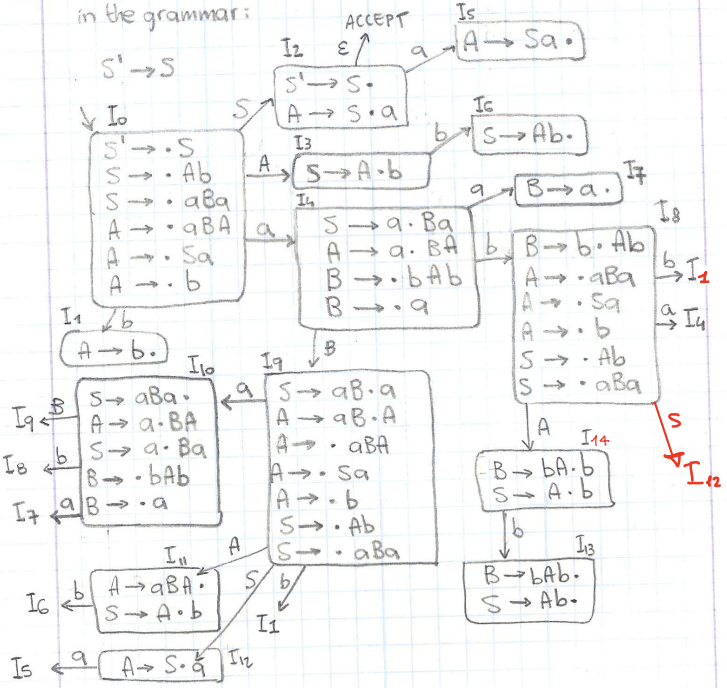
LR(0)
Parsing table

④ FIND THE LR(0) PARSING TABLE FOR THE FOLLOWING GRAMMAR

S → Ab | aBa (1) (2)
A → aBA | Sa | b (3) (4) (5)
B → bAb | a (6) (7)

Introduce a new start symbol, that appears just once in the grammar:

S' → S



LR(0) PARSING TABLE

STATE	ACTION			GOTO		
	a	b	ε	S	A	B
0	s4	s1		2	3	
1	r5	r5	r5			
2	s5		accept			
3		s6				
4	s7	s8				9
5	r4	r4	r4			
6	r1	r1	r1			
7	r7	r7	r7			
8	s4	s1		12	14	
9	s10	s1		12	11	
10	s7, r2	s8, r2	r2			9
11	r3	r3, s6	r3			
12	s5					
13	r1, r6	r1, r6	r1, r6			
14		s13				

EXAM FLC 11

PDA

② FIND A PUSHDOWN AUTOMATON ACCEPTING THE LANGUAGE:

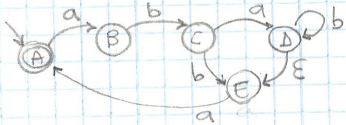
$\{a^n b^m c^k \mid n = 2m \text{ OR } m = 2k\}$

$S \to AC \mid DB$
$A \to aaAb \mid \varepsilon$ ← $n = 2m$
$B \to bbBc \mid \varepsilon$ ← $m = 2k$
$C \to cC \mid \varepsilon$ ← any number of c
$D \to aD \mid \varepsilon$ ← any number of a

$\delta(q, \varepsilon, S) = \{(q, AC), (q, DB)\}$
$\delta(q, \varepsilon, A) = \{(q, aaAb), (q, \varepsilon)\}$
$\delta(q, \varepsilon, B) = \{(q, bbBc), (q, \varepsilon)\}$
$\delta(q, \varepsilon, C) = \{(q, cC), (q, \varepsilon)\}$
$\delta(q, \varepsilon, D) = \{(q, aD), (q, \varepsilon)\}$
$\delta(q, a, a) = \delta(q, b, b) = \delta(q, c, c) = \{(q, \varepsilon)\}$

① FIND A LEFT-REGULAR GRAMMAR THAT GENERATES THE COMPLEMENT OF THE LANGUAGE DENOTED BY THE FOLLOWING EXPRESSION:

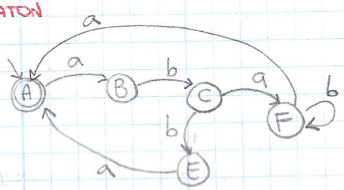
$(ab(ab^* \mid b)a)^*$



FINITE AUTOMATON

DETERMINISTIC FINITE AUTOMATON

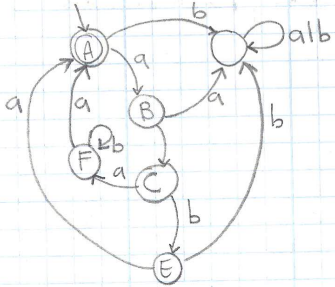
	a	b
A	B	/
B	/	C
C	D, E	E
D, E	A	D, E
E	A	/



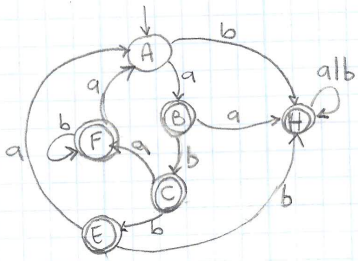
D, E ≡ F

COMPLETELY SPECIFIED DFA

YOU ADD JUST ONE STATE REACHED BY ALL NON COMPLETELY SPECIFIED STATES



COMPLEMENT OF INITIAL DFA



FIND LEFT REGULAR GRAMMAR

FINAL STATE ≡ START SYMBOL OF THE GRAMMAR

(In this case there are many final states, then you add a new final state, reached by all finite states by means of an empy symbol, that will be the start state of the grammar)

$S \to B \mid C \mid E \mid F \mid H$
$B \to Aa \mid a^{(1)}$
$H \to Ha \mid Hb \mid Ab \mid Ba \mid Eb \mid b$
$C \to Bb$
$E \to Cb$
$F \to Fb \mid Ca$
$A \to Fa \mid Ea$

(1) you must add this rule because A is the start state of the AUTOMATON

LALR(1)
Parsing table

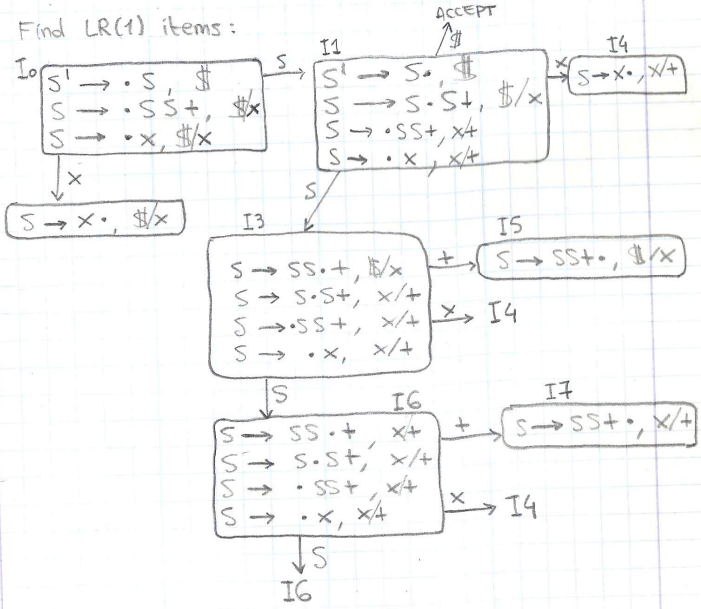
EXAM FLC 2

(3) FIND THE LALR(1) PARSING TABLE FOR THE FOLLOWING GRAMMAR:

0) $S' \to S$
1) $S \to SS+$
2) $S \to x$

- Two sets of LR(1) items have the same CORE if they are identical except for the lookahead symbols
- A set of LALR(1) items is the UNION of sets of LR(1) items having the SAME CORE

Find LR(1) items:



LALR(1) ITEMS:

I0, I1, I2_4, I3_6, I5_7

LALR(1) TABLE

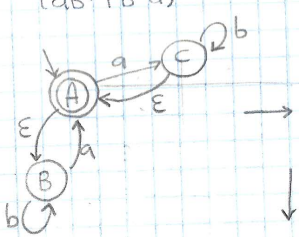
	+	x	#	S'	S
0		s2_4			1
1		s2_4	ACCEPT		3_6
2_4	r2	r2	r2		
3_6	s5_7	s2_4			3_6
5_7	r1	r1	r1		

EXAM FLC 3

REGEX

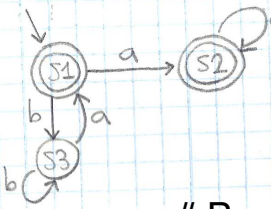
(1) FIND THE MINIMUM STATE AUTOMATON ACCEPTING THE LANGUAGE DENOTED BY THE REGULAR EXPRESSION

$(ab^*|b^*a)^*$



DETERMINISTIC AUTOMATON:

	a	b
S1) AB	ABC	B
S2) ABC	ABC	ABC
S3) B	AB	B



T_0: {S1, S2} {S3}

T_1: {S1} {S2} {S3}

⟸ MINIMUM STATE AUTOMATON

Predictive parsing

3 – Find the LL(1) parsing table for the following grammar:

S → (L) | a
L → L , S | S

TABLE FOR THE FOLLOWING GRAMMAR:

i) Eliminate Left-Recursion L → L,S

S → (L) | a
L → SR
R → ,SR | ε

ii) Build Predictive parsing Table

NT \ input Symbols	(	a	)	,	$
S	S→(L)	S→a			
L	L→SR	L→SR			
R			R→ε	R→,SR	

ii) Build Nullable, First, Follow Table

	NULLABLE	FIRST	FOLLOW
S	F	(, a	$, ,)
L	F	(, a	)
R	T	,	)

EXAM FLC 11 # Predictive parsing

(3) TELL IF THE FOLLOWING GRAMMAR IS LL(1):

$S \to (L) \mid a$
$L \to L,S \mid S$

i) Eliminate Left-Recursion $L \to L,S$

$S \to (L) \mid a$
$L \to SR$
$R \to ,SR \mid \varepsilon$

ii) Build Nullable, First, Follow Table

	NULLABLE	FIRST	FOLLOW
S	F	(, a	$, ',)
L	F	(, a	)
R	T	'	)

iii) Build Predictive parsing Table

NT \ Input Symbols	(	a	)	,	$
S	$S \to (L)$	$S \to a$			
L	$L \to SR$	$L \to SR$			
R			$R \to \varepsilon$	$R \to ,SR$	

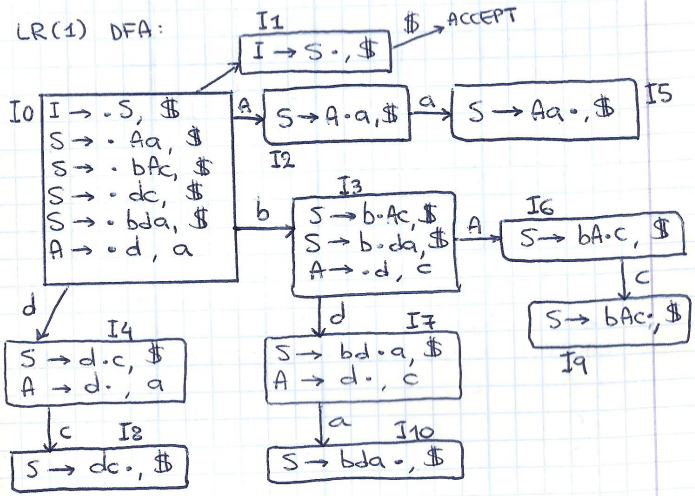
YES, IT IS LL(1) SINCE THE PARSING TABLE HAS NO CONFLICTS

LALR(1)
Parsing table

(4) TELL IF THE FOLLOWING GRAMMAR IS LALR(1):

0) $I \to S$
1) $S \to Aa$
2) $S \to bAc$
3) $S \to dc$
4) $S \to bda$
5) $A \to d$

LR(1) DFA:



LALR(1) DFA ≡ LR(1) DFA
BECAUSE ALL ITEMS HAVE DIFFERENT CORES.

LALR(1) TABLE:

	ACTIONS					GOTO	
STATE	a	b	c	d	#	S	A
0		s3		s4		1	2
1					ACCEPT		
2	s5						6
3				s7			
4	r5		s8				
5					r1		
6			s9				
7	s10		r5				
8					r3		
9					r2		
10					r4		

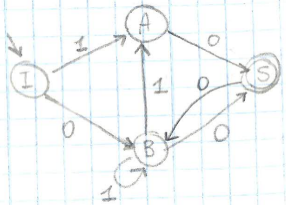
The grammar IS LALR(1) because the LALR(1) ACTION TABLE does not contain conflicts.

REGEX

EXAM FLC 10

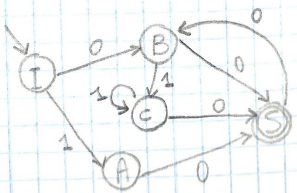
(1) FIND A MINIMUM-STATE DFA EQUIVALENT TO THE FOLLOWING GRAMMAR AND THEN THE REGULAR EXPRESSION FOR THE ACCEPTED LANGUAGE.

$S \to A0 \mid B0$
$A \to B1 \mid 1$
$B \to B1 \mid S0 \mid 0$



• DETERMINISTIC FA:

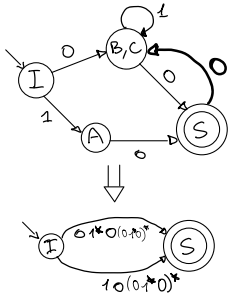
	0	1
I	B	A
A	S	/
B	S	AB ≡ C
C ≡ AB	S	AB ≡ C
S	B	/



• MINIMIZATION

π_0: { I, A, B, C } { S }

π_1: { I } { A } { B, C } { S }



• REGULAR EXPRESSION

$01^*0(0+10)^* \mid 10(0+10)^*$

#Useful symbols

② FROM THE FOLLOWING GRAMMAR ELIMINATE USELESS SYMBOLS AND ε- PRODUCTIONS

S → 0S0 | 1B1 | BB
A → 0C1 | 0
B → 1 B | S0 | ε
C → 0 A | 1

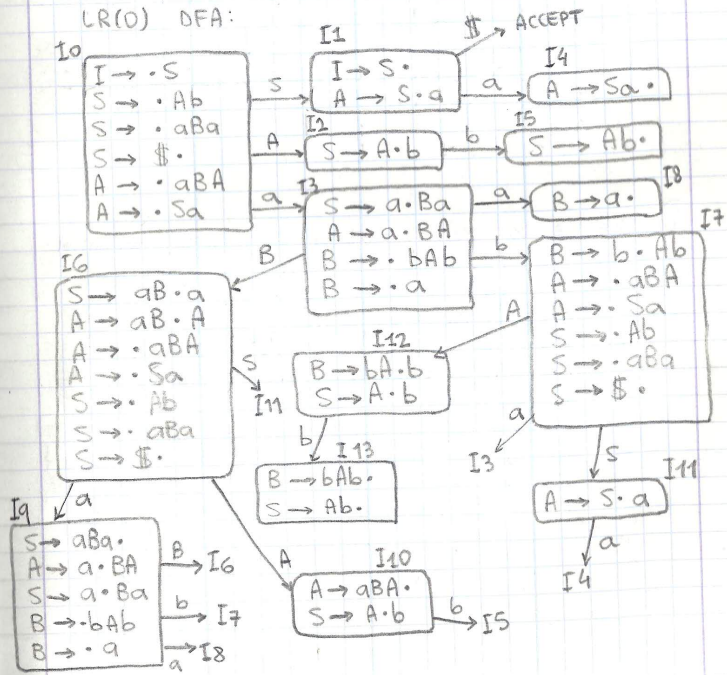
Si può partire con l'eliminazione delle epsilon productions:
Symbols that can produce epsilon: B, S
Equivalent grammar without epsilon productions:
S -> 0S0 | 1B1 | BB | 00 | 11 | B
A -> 0C1 | 0
B -> 1B | S0 | 1 | 0
C -> 0A | 1
Si procede quindi ad eliminare useless symbols:
Symbols generating a non-empty language: {0,1}U{S,A,B,C}
=> All symbols generate a non-empty language, so none of them can be eliminated
Reachable symbols: {S}U{B}
Unreachable symbols: {A,C}
=> A and C can be eliminated
The final equivalent grammar after the elimination of A and C is:
S -> 0S0 | 1B1 | BB | 00 | 11 | B
B -> 1B | S0 | 1 | 0

SLR
Parsing table

③ FIND THE SLR PARSING TABLE FOR THE FOLLOWING GRAMMAR:

I → S (0)
S → Ab | aBa | ε (1) (2) (3)
A → aBA | Sa (4) (5)
B → bAb | a (6) (7)

Non terminals	NULLABLE	FIRST	FOLLOW
S	YES	a, $	$, a
A	NO	a	b
B	NO	a, b	a



SLR PARSING TABLE

	ACTION			GOTO		
STATE	a	b	$	S	A	B
0	s3/r3		r3	1	2	
1	s4		acc			
2		s5				
3	s8	s7				6
4		r5				
5	r1		r1			
6	r3/s9		r3	11	10	
7	r3/s3		r3	11	12	
8	r7					
9	r2/s8	s7	r2			6
10		r4/s5				
11	s4					
12		s13				
13	r6/r1		r1			

④ MAKE THE INHERITED ATTRIBUTES IN THE FOLLOWING TRANSLATION SCHEME BE DEFINED BY COPY RULES:

$A \to aBC \ \{A.a := f(B.b)\}$
$B \to bC \ \{A.b := g(C.c)\} \ A \ \{B.b := k(b.x)\}$
$C \to cB \ \{A.b := h(C.c)\} \ A \ \{C.c := k(c.x)\}$

$A \to aBC \ \{A.a := f(B.b)\}$
$B \to b \ C \ \{M.i := C.c\} \ M \ \{A.b := M.s\} \ A \ \{B.b := k(b.x)\}$
$M \to \varepsilon \ \{M.s = g(M.i)\}$
$C \to cB \ \{N.i = C.c\} \ N \ \{A.b := N.s\} \ A \ \{C.c := k(c.x)\}$
$N \to \varepsilon \ \{N.s = h(N.i)\}$

#translation scheme
#copy rules

EXAM FLC 9

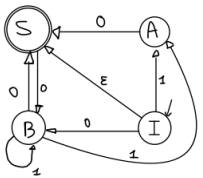
① FIND THE MINIMUM-STATE DFA EQUIVALENT TO THE FOLLOWING GRAMMAR

$S \to A0 \mid B0 \mid \varepsilon$
$A \to B1 \mid 1$
$B \to B1 \mid S0 \mid 0$

1 – Find the minimum-state DFA equivalent to the following grammar

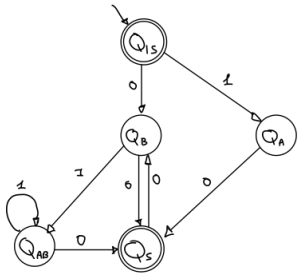
$S \to A0 \mid B0 \mid \varepsilon$
$A \to B1 \mid 1$
$B \to B1 \mid S0 \mid 0$

ε-NFA



Substituting

	0	1
*Q_{IS}	B	A
Q_B	S	AB
Q_A	S	
*Q_S	B	
Q_{AB}	S	AB



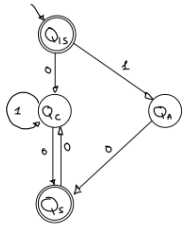
minimization

π_0 : $\{Q_{IS}, Q_S\}, \{Q_A, Q_B, Q_{AB}\}$

π_1 : $\{Q_{IS}\}, \{Q_S\}, \{Q_A\}, \{Q_B, Q_{AB}\}$

π_2 : $\{Q_{IS}\}, \{Q_S\}, \{Q_A\}, \{Q_B, Q_{AB}\}$ = Q_C

minimum-state DFA ⇒



#Useful symbols

(2) FROM THE FOLLOWING GRAMMAR ELIMINATE USELESS SYMBOLS

$S \to 0S0 \mid 1C1 \mid BB$
$A \to 0C1 \mid 0$
$B \to 1B \mid BAC$
$C \to 0B \mid 1S \mid 1$

- ELIMINATE SYMBOLS GENERATING AN EMPTY LANGUAGE:

$S \to 0S0 \mid 1C1$
$A \to 0C1 \mid 0$
$C \to 1S \mid 1$

- ELIMINATE UNREACHABLE SYMBOLS

$S \to 0S0 \mid 1C1$
$C \to 1S \mid 1$

Equivalent Grammars

EXAM FLC 8

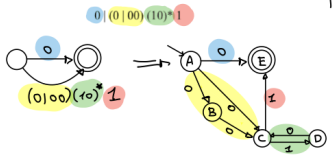
REGEX

(1) VERIFY THE EQUIVALENCE OF THE FOLLOWING REGULAR EXPRESSIONS:

(a) $(0 \mid 01)(01)^*$
(b) $01(0100)(10)^*1$

(1)

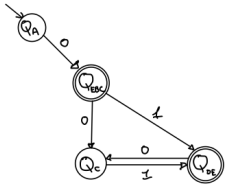
(b) NFA



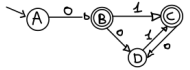
TRANSFORM INTO DFA:

	0	1
Q_A	{E,B,C}	
*Q_{EBC}	{C}	{D,E}
Q_C		{D,E}
*Q_{DE}	{C}	

bDFA



(2) DFA (0|01)(01)*



(2) MINIMIZATION

π_0: $\{Q_{EBC}, Q_{DE}, B, C\}$ $\{A, D, Q_A, Q_C\}$

π_1: $\{Q_{EBC}, B\}, \{Q_{DE}, C\}, \{A, Q_A\}, \{D, Q_C\}$

π_2: $\{Q_{EBC}, B\}, \{Q_{DE}, C\}, \{A, Q_A\}, \{D, Q_C\}$

Starting States of the two automata are in the same EQ CLASS
So DFAs are equivalent $\Leftrightarrow$ Regexes are equivalent!

PDA

(2) FIND A PUSHDOWN AUTOMATON ACCEPTING THE LANGUAGE

$\{(a^n b^n)^m c^m \mid m \geq 0,\ n \geq 0\}$

The indicated language is not CFL, therefore, no PDA accepting it can be defined.

Predictive parsing

(3) PRODUCE THE PREDICTIVE PARSING TABLE FOR THE FOLLOWING GRAMMAR:

$S \to AB \mid B$
$A \to Ba \mid a$
$B \to Sb \mid b$

(1) eliminate Left Recursion

1.1 EXPLODE INDIRECT RECURSION

$B \to Sb \mid b \to B \to ABb \mid Bb \mid b$
$\to B \to BaBb \mid aBb \mid Bb \mid b$

1.2 TRANSFORM RECURSIVE PRODUCTIONS

$B \to aBbB' \mid bB'$
$B' \to BaBbB' \mid BbB' \mid \varepsilon$

(2) COMPUTE NULLABLE/FIRST/FOLLOW AND PREDICTIVE PARSING TABLE

$S \to AB \mid b$
$A \to Ba \mid a$
$B \to aBbB' \mid bB'$
$B' \to aBbB' \mid bB' \mid \varepsilon$

NT	NULLABLE	FIRST	FOLLOW
S	F	a, b	$
A	F	a, b	a, b
B	F	a, b	a, b, $
B'	T	a, b	a, b, $

Predictive Parsing Table

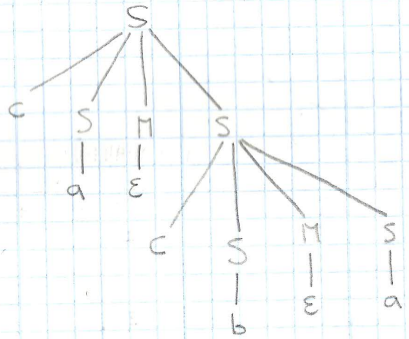
NT	a	b	$
S	S→AB S→B	S→AB S→B	
A	A→Ba A→a	A→Ba	
B	B→aBbB'	B→bB'	
B'	B'→aBbB' B'→ε	B'→bB' B'→ε	B'→ε

Bottom up parser

(4) TRANSFORM THE FOLLOWING SDT SO THAT IT CAN BE IMPLEMENTED BY A BOTTOM-UP PARSER, AND THEN INDICATE WHICH STRING IT WILL PRINT WHEN THE INPUT IS "cacba"

$S \to cS$ { print "x" } S
$S \to a$ { print "y" }
$S \to b$ { print "z" }

$S \to cSMS$
$M \to \varepsilon$ { print "x" }
$S \to a$ { print "y" }
$S \to b$ { print "z" }



OUTPUT: y x z x y

EXAM FLC 7

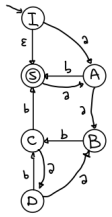
Equivalent Grammars

(1) VERIFY THE EQUIVALENCE OF THE FOLLOWING GRAMMARS:

(a)
$S \to Ab \mid Cb \mid \varepsilon$
$A \to Sa \mid a$
$B \to Aa \mid Da$
$C \to Bb \mid Db$

(b)
$S \to aA \mid \varepsilon$
$A \to aB \mid bS \mid b$
$B \to bA \mid bS \mid b$

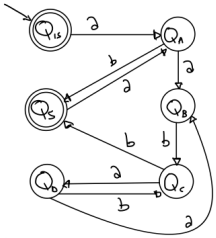
(a) ε-NFA



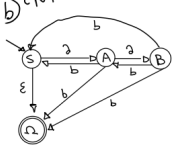
Subsetting

SUBSET	a	b
*Q_{IS}	{A}	
Q_A	{S,B}	{S}
Q_B		{C}
*Q_S	{A}	
Q_C	{D}	{S}
Q_D	{B}	{C}

DFA



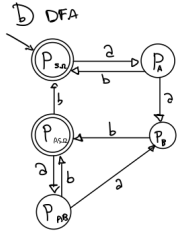
(b) ε-NFA



Subsetting

SUBSET	a	b
*$P_{S\Omega}$	{A}	
P_A	{B}	{S, Ω}
P_B		{A, S, Ω}
*$P_{AS\Omega}$	{A,B}	{S, Ω}
P_{AB}	{B}	{A, S, Ω}

DFA



(2) Minimization of the union of the states

π_0 : $\{Q_{IS}, Q_S, P_{S\Omega}, P_{AS\Omega}\}$, $\{Q_A, Q_B, Q_C, Q_D, P_A, P_B, P_{AB}\}$

π_1 : $\{Q_{IS}, Q_S, P_{S\Omega}\}, \{P_{AS\Omega}\}, \{Q_A, Q_C, P_A, P_{AB}\}, \{Q_B\}, \{Q_D\}, \{P_B\}$

π_2 : $\{Q_{IS}, Q_S\}, \{P_{S\Omega}\}, \{P_{AS\Omega}\}, \{Q_A\}, \{Q_C\}, \{P_A\}, \{P_{AB}\}, \{Q_B\}, \{Q_D\}, \{P_B\}$

Start states in different EQ classes ⇒ AUTOMATON ARE NOT EQUIVALENT

⇓

GRAMMARS NOT EQUIVALENT

Predictive parsing

(3) PRODUCE THE PREDICTIVE PARSIN FOR THE FOLLOWING GRAMMAR:

$S \to AB \mid B$
$A \to aA \mid \varepsilon$
$B \to bB \mid \varepsilon$

NT	NULLABLE	FIRST	FOLLOW
S	YES	a b	$
A	YES	a	$ b
B	YES	b	$

PREDICTIVE PARSING TABLE:

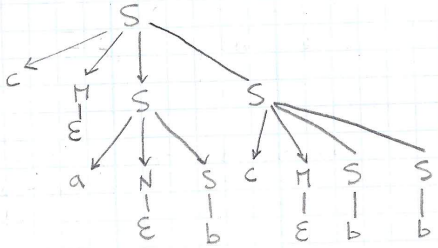
NT	a	b	$
S	S → AB	S → AB | B	S → AB | B
A	A → aA	A → ε	A → ε
B		B → bB	B → ε

Bottom up parser

(4) TRANSFORM THE FOLLOWING SDT SO THAT IT CAN BE IMPLEMENTED BY A BOTTOM-UP PARSER, AND THEN INDICATE WHICH STRING IT WILL PRINT WHEN THE INPUT IS "cabcbb"

$S \to c$ { print "z" } $S\ S$
$S \to a$ { print "x" } S
$S \to b$ { print "y" }

$S \to cMSS$
$M \to \varepsilon$ { print "z" }
$S \to aNS$
$N \to \varepsilon$ { print "x" }
$S \to b$ { print "y" }



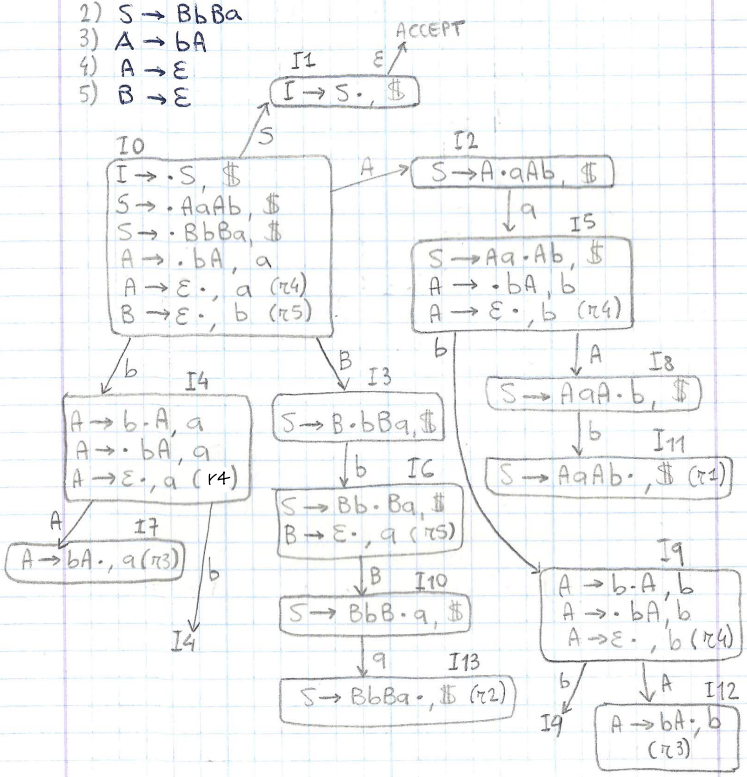
OUTPUT:
zxyzyy

LR(1)
Parsing table

EXAM FLC 6

③ PRODUCE THE LR(1) PARSING TABLE FOR THE FOLLOWING GRAMMAR:

0) I → S
1) S → AaAb
2) S → BbBa
3) A → bA
4) A → ε
5) B → ε



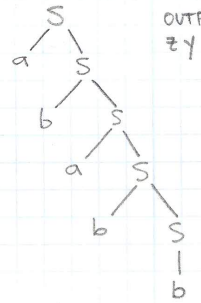
LR(1) PARSING TABLE

STATE	ACTION a	ACTION b	ACTION $	GOTO S	GOTO A	GOTO B
0	r4	s4 | r5		1	2	3
1			ACCEPT			
2	s5					
3		s6				
4	r4	s4			7	
5		r4 | s9			8	
6	r5					10
7	r3					
8		s11				
9		r4 | s9			12	
10	s13					
11			r1			
12		r3				
13			r2			

Bottom up parser

④ ASSUMING THAT THE FOLLOWING SDT IS IMPLEMENTED BY A BOTTOM-UP PARSER, INDICATE THE STRING THAT IT WILL PRINT WHEN THE INPUT IS "abaabb"

S → aS { print "x" }
S → bS { print "y" }
S → a { print "w" }
S → b { print "z" }

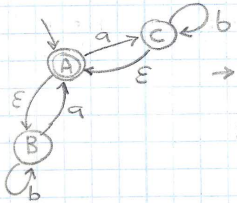


OUTPUT: zyxyx

EXAM FLC 4

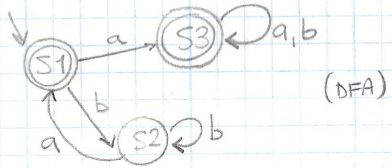
1 FIND THE MINIMUM AUTOMATON ACCEPTING THE LANGUAGE DENOTED BY:

$(ab^* \mid b^*a)^*$



DFA:

	a	b
→ S1) AB	ABC	B
S2) B	AB	B
S3) ABC	ABC	ABC



(DFA)

MINIMIZATION:

Π_0: {S1, S3} {S2}

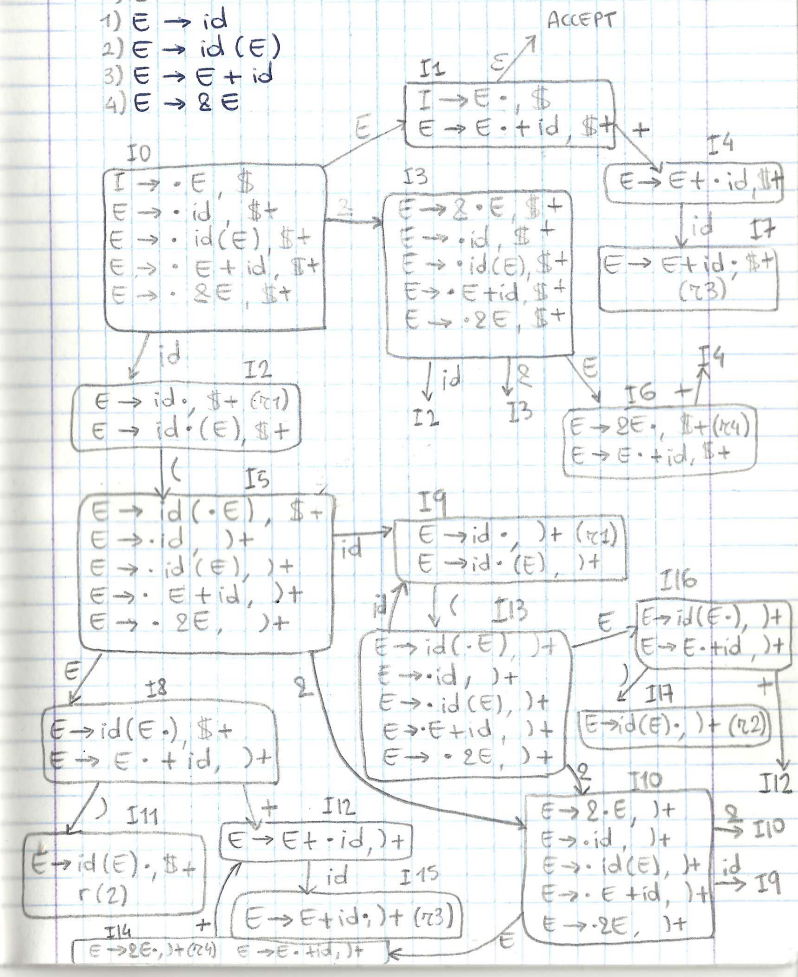
Π_1: {S1} {S2} {S3}

THE DFA IS ALREADY A MINIMUM AUTOMATON

LR(1)
Parsing table

3 PRODUCE THE LR(1) PARSING TABLE FOR THE FOLLOWING GRAMMAR:

0) I → E
1) E → id
2) E → id (E)
3) E → E + id
4) E → E E



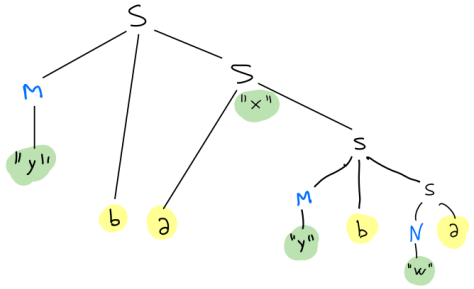
#Bottom up parser

(4) ASSUMING THAT THE FOLLOWING SDT IS IMPLEMENTED BY A BOTTOM-UP PARSER, INDICATE THE STRING THAT IT WILL PRINT WHEN THE INPUT IS "baba"

4 – Assuming that the following SDT is implemented by a bottom-up parser, indicate the string that it will print when the input is "baba" :

S → aS {print "x"}
S → {print "y"} bS
S → {print "w"} a
S → b {print "z"}

S → aS{Print"x"}
S → MbS
M → ε {Print"y"}
S → Na
N → ε {Print"w"}
S → b {Print"z"}



INPUT: baba

OUTPUT: yyWx obtained visiting the parse tree in POST ORDER

- CONTEXT FREE GRAMMAR:
  - The LEFT side of a rule is a SINGLE NON-TERMINAL SYMBOL
  - The RIGHT side CANNOT be an EMPTY set

- RIGHT-LINEAR GRAMMAR
  $P = \{ A \to xB,\ A \to x \mid A,B \in N;\ x \in T^+ \}$
  Left side = a single non terminal symbol
  Right side = one or more terminal symbols FOLLOWED by a single non-terminal symbol

- RIGHT-REGULAR GRAMMAR
  $P = \{ A \to aB,\ A \to a \mid A,B \in N;\ a \in T \}$
  left side = a single non terminal symbol
  Rigth side = (one terminal symbol followed by one non terminal symbol) OR (just one terminal symbol)

- LEFT-LINEAR GRAMMAR
  $P = \{ A \to Bx,\ A \to x \mid A,B \in N;\ x \in T^+ \}$

- LEFT-REGULAR GRAMMAR
  $P = \{ A \to Ba,\ A \to a \mid A,B \in N;\ a \in T \}$

- EQUATION OF REGULAR EXPRESSION:
  $X = \alpha X \mid \beta \quad \Rightarrow \quad X = \alpha^* \beta$
  $X = X\alpha \mid \beta \quad \Rightarrow \quad X = \beta \alpha^*$
  Solve a SYSTEM of equations by substitution.
  You can transform a set of PRODUCTIONS in a set of EQUATIONS ("$\to$" $\Rightarrow$ '=')

- DFA = Deterministic Finite Automata
  $A = (Q, \Sigma, \delta, q_0, F)$
  Q: STATES; Σ: INPUT; δ: TRANSITIONS; q_0: START STATE; F: SET OF FINAL STATES

  TRANSITION TABLE, TRANSITION DIAGRAM
  TRANSITION: $\delta(q_0, 0) = q_2$ (FROM q_0 WITH $0 \to q_2$)

- NFA = Non-deterministic Finite Automata
  - From a state, with the SAME input symbol, you can reach DIFFERENT state
  - Contain EMPTY transitions (ε-NFA)

- FROM FA TO REGULAR EXPRESSION:
  - ADD STATES A (NEW INITIAL STATE) AND Ω (NEW FINAL STATE)
  - ELIMINATE ALL INTERMEDIATE STATES → REG EXP

- FROM RIGHT-REGULAR GRAMMAR TO FA
  Terminal symbols are produced from left to right

- FROM LEFT-REGULAR GRAMMAR TO FA
  $P = \{ S \to A\emptyset \equiv$ S IS REACHABLE FROM A WITH $\emptyset$
  $A \to A\emptyset \mid S1 \mid \emptyset$
  - Add a new initial state I
  - S is the final state accepting the language
  - Terminal symbols are produced from right to left

- MINIMIZATION OF DFA
  PARTITION 0: DISTINCTION BETWEEN FINAL and NON-FINAL STATES

- COMPLEMENT OF A REGULAR LANGUAGE
  - DFA
  - COMPLETELY SPECIFIED DFA
  - EXCHANGE FINAL AND NON-FINAL STATES

- EQUIVALENCE OF REGULAR LANGUAGES
  - WRITE FA FOR BOTH → TRANSFORM IN DFA
  - UNION OF DFAs AND MINIMIZATION OF THE UNION
  - IF BOTH START STATES ARE IN THE SAME EQUIVALENCE CLASS ⇒ EQUIVALENT

CONTEXT-FREE LANGUAGES

- ELIMINATE USELESS SYMBOLS IN CFG:
  - ELIMINATE SYMBOL GENERATING AN EMPTY LANGUAGE
    e.g. $\underline{A} \to a\underline{B}\underline{A} \mid b\underline{A}\underline{S}$
    rules containg A in left side, always contain A also in right side ⇒ you cannot generate strings of terminal symbols ⇒ ELIMINATE A
    NOTE: every TERMINAL symbol generates a NON-EMPTY language (+ INDUCTION)
  - ELIMINATE UNREACHABLE SYMBOLS
    NOTE: START SYMBOL S IS REACHABLE (+ INDUCTION)

- ELIMINATE ε-PRODUCTIONS:
  Write new rules taking in account the non-terminal symbols that produce empty strings.

- LANGUAGES ACCEPTED BY PDA (PUSHDOWN AUTOMATA)
  - LANGUAGE ACCEPTED BY FINAL STATE
  - LANGUAGE ACCEPTED BY EMPTY STACK

- TO WRITE A PDA:
  - WRITE SET OF PRODUCTIONS
    e.g. $S \to 0S0 \mid 1S1 \mid \varepsilon$
  - WRITE TRANSITION FUNCTIONS
    e.g. $\delta(q, \varepsilon, S) = \{(q, 0S0), (q, 1S1), (q, \varepsilon)\}$
    $\delta(q, 0, 0) = \delta(q, 1, 1) = \{(q, \varepsilon)\}$

5A - BOTTOMUP PARSER

- BOTTOM-UP PARSING:
  - PARSE TREE VISITED IN POSTORDER
  - CONSTRUCTION: INPUT STRING ---> START SYMBOL
  - REDUCTION: RIGHT SIDE ---> LEFT SIDE

- SHIFT-REDUCE PARSER:
  - STACK ← GRAMMAR SYMBOLS
  - INPUT BUFFER ← STRING

- ACTIONS of shift-reduce parser
  - SHIFT: next input symbol → TOS
  - REDUCE: right side → left side → STACK
  - ACCEPT
  - ERROR

- LR(0) PARSING TABLE:
  - Starting from a CFG, build a DFA recognizing VIABLE PREFIXES
  - * DFA = SET OF LR(0) ITEMS
  - Build LR(0) PARSING TABLE

	ACTION	GOTO
state	T_1 T_2 ... T_N	NT_1 ... NT_N
	shift/reduce/accept	next state

A grammar is LR(0) if "ACTION" does not include CONFLICTS (shift/reduce or reduce/reduce in the same position).

NOTE: REDUCE action for ALL terminal symbols

- LR (SLR) PARSING TABLE:
  - Build TABLE with NULLABLE, FIRST, FOLLOW
  - DFA = set of LR(0) ITEMS (WITHOUT lookahead symbols)
  - Build LR (SLR) PARSING TABLE:

	ACTION	GOTO
state	T_1 T_2 ... T_N	NT_1 ... NT_N
	shift/reduce/accept	next state

NOTE: REDUCE action just for all a in FOLLOW(A)

- LR (1) PARSING TABLE
  - * LR(1) ITEM: PRODUCTION WITH A DOT IN RIGHT SIDE AND A LOOKAHEAD SYMBOL (terminal or $)
  - Construct a DFA = set of LR(1) ITEMS that recognizes viable prefixes
  - Build LR(1) PARSING TABLE (state, ACTION, GOTO)

  A grammar is LR(1) if the "ACTION" table does not contain conflicts.

- LALR (1)
  - * LR(1) items have the same CORE if they are IDENTICAL EXCEPT for the lookahead symbols
  - * LALR(1) ITEM = UNION of LR(1) ITEMS with the SAME CORE
  - Build DFA with LR(1) items, then obtain DFA with LALR(1) items
  - Build LALR(1) PARSING TABLE (state, ACTION, GOTO)

- LR PARSING OF AMBIGUOUS GRAMMAR:
  - SOLVE CONFLICTS BY MEANS OF
    - ASSOCIATIVITY directives
    - PRECEDENCE directives

- ERROR in LR parsing:
  - BLANKS in LR parsing tables mean ERROR ACTIONS → parser STOP
  - error RULES: CONTAIN error SYMBOL (considered non terminal symbol)
  - error RULES may introduce SHIFT/REDUCE and REDUCE/REDUCE conflicts.

SA - TOP-DOWN PARSER

- TOP-DOWN PARSING:
  - PARSE TREE VISITED IN PREORDER
  - CONSTRUCTION: PREORDER TREE ⟶ INPUT STRING
  - CREATION: LEFT SIDE ⟶ RIGHT SIDE

- LEFT-RECURSIVE PRODUCTION: $A \to Aa$
  ⟹ can cause INFINITE LOOP!

- ELIMINATE LEFT-RECURSION:
  $$A \to A\alpha \mid \beta \quad = \quad \begin{cases} A \to \beta R \\ R \to \alpha R \mid \varepsilon \end{cases}$$

- PREDICTIVE (or LL(1)) PARSING TABLE:
  - Build table with NULLABLE, FIRST and FOLLOW
  - Build PREDICTIVE PARSING TABLE

non terminals	T_1 T_2 ... T_N
NT_1	
⋮	corresponding productions
NT_N	

e.g. $T' \to *FT' \mid \varepsilon$

$T' \to *FT'$ in FIRST(T')
$T' \to \varepsilon$ in FOLLOW(T')

A grammar is LL(1) if its predictive parsing table has NO MULTIPLY-DEFINED ENTRIES (no conflicts)

SYNTAX-DIRECTED TRANSLATION

- SYNTAX-DIRECTED DEFINITION:
  - SYMBOL → SET OF ATTRIBUTES
  - PRODUCTION → SEMANTIC RULES

- TYPES OF ATTRIBUTES:
  - SYNTHESIZED: symbol is on the LEFT side of the production
  - INHERITED: symbol is on the RIGHT side of the production

- SDD is S-ATTRIBUTED if every attribute is SYNTHESIZED (BOTTOM-UP order)

- POSTFIX SDT:
  is an SDT having all actions at the RIGHT ENDS of the productions
  - S-ATTRIBUTED SDD can be converted to POSTFIX SDT simply by placing each ACTION at the RIGHT END of the associated production
  - ACTIONS are executed by a BOTTOM-UP parser along with REDUCTIONS

- SDD is L-ATTRIBUTED if any production has SYNTHESIZED attributes and INHERITED attributes

- Convert L-ATTRIBUTED SDD in SDT:
  - place the actions that compute an INHERITED ATTRIBUTE for a symbol X immediately BEFORE that occurrence of X
  - place the actions that compute a SYNTHESIZED ATTRIBUTE at the END of the production

- Inherited attribute DEFINED BY a COPY RULE
  - The copy rule can be ELIMINATED
  - Use STACK POSITIONS instead of names

- Inherited attribute NOT DEFINED BY a COPY RULE
  - Use MARKERS
  - Example:

$S \to aA \{ C.i = f(A.s) \} C$
$C \to c \{ C.s = g(C.i) \}$

↓

$S \to aA \{ M.i = A.s \} M \{ C.i = M.s \} C$
$M \to \varepsilon \{ M.s = f(M.i) \}$
$C \to c \{ C.s = g(C.i) \}$

↓

$S \to a A M C$
$M \to \varepsilon \{ stack[top].val = f(stack[top-1].val) \}$
$C \to c \{ stack[top].val = g(stack[top-1].val) \}$