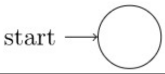
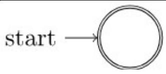
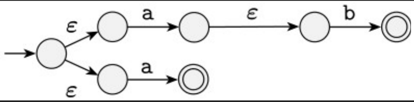
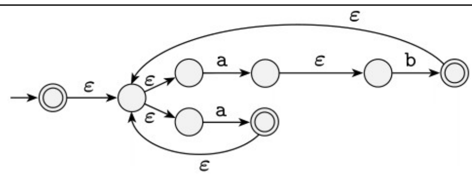
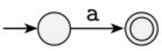
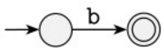
Comp2022 final sheet

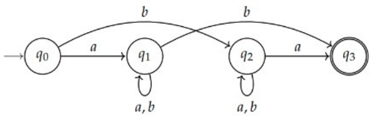
COMP2022 Models of Computation (University of Sydney)

$\{a^ib^j: i > j\}$	Let $x_n = a^n$ for every n. If $i > j$ then x_i, x_j can be distinguished by $z = b^j$. Indeed, $x_iz = a^ib^j \in L$ while $x_jz = a^jb^j \notin L$.
$\{a^nb^m$: n divides m, or m divides $n\}$	Let $p(i)$ be the ith prime number, Let $x_n = a^{p(n)}$ for every n. Then if $i \neq j$ then x_i, x_j are distinguishable by $z = b^{p(i)}$. Indeed, $x_iz = a^{p(i)}b^{p(i)} \in L$ (since every number divides itself), while $x_jz = a^{p(j)}b^{p(i)} \notin L$ (no prime divides any other prime).
$\{a^{n^2}: n \geq 0\}$	Let $x_n = a^{n^2}$ for every n. If $i < j$ then x_i, x_j are distinguishable by $z = a^{2i+1}$. Indeed, $\lvert x_iz\rvert = i^2 + 2i + 1 = (i+1)^2$ and so $x_iz \in L$, while $j^2 < \lvert x_jz\rvert = j^2 + 2i + 1 < j^2 + 2j + 1 = (j+1)^2$ and so $x_jz \notin L$.
All strings a^ib^j such that (a) i is even, or (b) $j < i$ and j is even	Let $x_n = a^{2n+1}$ for every n. If $j < i$ then x_i, x_j are distinguishable by $z = b^{2i}$. Indeed, $x_iz = a^{2i+1}b^{2i} \in L$ while $x_jz = a^{2j+1}b^{2i} \notin L$ since $j < i$ implies $2j+1 < 2i+1$ and in particular that $2j+1 \leq 2i$.

RE to NFA

<table>
<tr><td>∅</td><td></td><td>ab</td><td></td><td colspan="2">(ab|a)*</td></tr>
<tr><td>ε</td><td></td><td>ab|a</td><td></td><td rowspan="2" colspan="2"></td></tr>
<tr><td>a</td><td></td><td>b</td><td></td></tr>
</table>

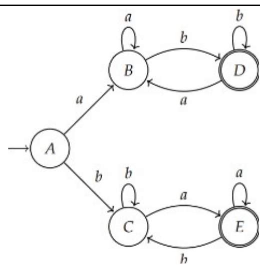
NFA to DFA



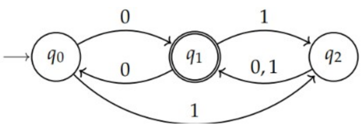
state	a	b
$\{q_0\}$	$\{q_1\}$	$\{q_2\}$
$\{q_1\}$	$\{q_1\}$	$\{q_1, q_3\}$
$\{q_2\}$	$\{q_2, q_3\}$	$\{q_2\}$
$\{q_1, q3\}$	$\{q_1\}$	$\{q_1, q_3\}$
$\{q_2, q_3\}$	$\{q_2, q_3\}$	$\{q_2\}$

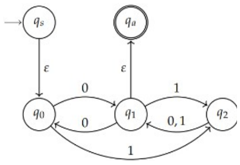
state	a	b
A	B	C
B	B	D
C	E	C
D	B	D
E	E	C

→



→ →



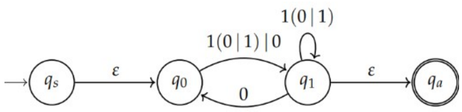


DFA to RE

q(0,1): 0

q(0,2,1): 1→0|1

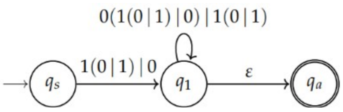
q(1,2,1): 1→0|1



q(s,1): ε → 1(0|1)|0

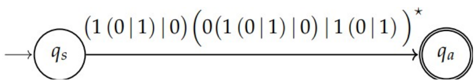
q(1,0,1): 0 → 1(0|1)|0

q(1,1): 1(0|1)



q(s,1,a): 1(0|1)|0 → (0(1(0|1)0)|1(0|1))* →

= (1(0|1)0)(0(1(0|1)0)|1(0|1))*



ε

<table>
<tr><td>5 letters</td><td>X</td><td></td><td></td><td></td><td></td><td>S → AX | AB | ε</td><td rowspan="6">the <eg> cell: all variables to start that arrive a string with 4 letters counting to right: abbb.

The final cell of the CYK table (the top-left cell in the table) needs to contain the start variable S of the grammar to prove that the string belongs to the grammar.</td></tr>
<tr><td>4 letters</td><td>S, T</td><td><eg></td><td></td><td></td><td></td><td>T → AX | AB</td></tr>
<tr><td>3 letters</td><td></td><td>X</td><td></td><td></td><td></td><td>X → TB</td></tr>
<tr><td>2 letters</td><td></td><td>S, T</td><td></td><td></td><td></td><td>A → a</td></tr>
<tr><td>1 letter</td><td>A</td><td>A</td><td>B</td><td>B</td><td>B</td><td>B → b</td></tr>
<tr><td></td><td>a</td><td>a</td><td>b</td><td>b</td><td>b</td><td></td></tr>
</table>

$E \rightarrow E + T \mid E - T \mid T$

$T \rightarrow F \times T \mid T/T \mid F$

$F \rightarrow (E) \mid V \mid C$

$V \rightarrow a \mid b \mid c$

$C \rightarrow 1 \mid 2 \mid 3$

Set of **variables** V = {E, T, F, V, C}

Set of **terminals** T = {a, b, c, 1, 2, 3, (,), ×, /, +, −}

Number of **Production rules** = 5 * 3 = 15 rules

Start variable as E

(if it is not stated, assume it is the first one, or S)

right-most derivation of the string $a + b \times c$

$E \Rightarrow E + T$

$\Rightarrow E + F \times T$

$\Rightarrow E + F \times F$

$\Rightarrow E + F \times V$

$\Rightarrow E + F \times c$

$\Rightarrow E + V \times c$

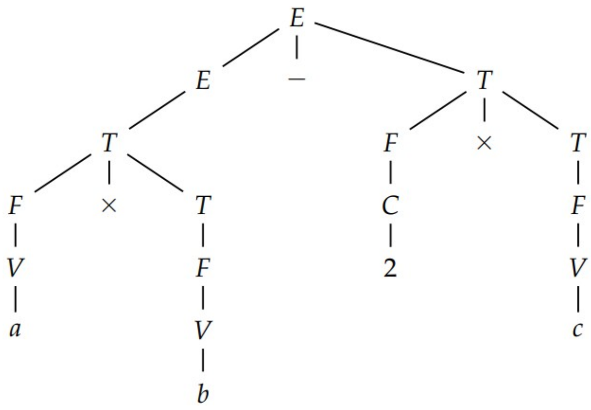
$\Rightarrow E + b \times c$

$\Rightarrow T + b \times c$

$\Rightarrow F + b \times c$

$\Rightarrow V + b \times c$

$\Rightarrow a + b \times c$



parse tree for $a \times b - 2 \times c$

$\{bb, bbbb, bbbbbb, ...\}$ **$\{(bb)^{n+1} \mid n \in \mathbb{N}\}$**	$S \rightarrow bb \mid bbS$ (bSb, Sbb would also work fine)	**$\{a^m b^n \mid n, m \in \mathbb{N}, m > 0\}$**	$S \rightarrow AB$ $A \rightarrow aA \mid a$ (derives at least one a) $B \rightarrow bB \mid \varepsilon$ (derives any number of b's)
$\{a, ba, bba, bbba, ...\}$ **$\{b^n a \mid n \in \mathbb{N}\}$**	$S \rightarrow a \mid bS$	**$\{a^m b^n \mid n, m \in \mathbb{N}, m > 0\}$**	$S \rightarrow AB$ $A \rightarrow aA \mid a$ (derives at least one a) $B \rightarrow bB \mid \varepsilon$ (derives any number of b's)
$\{\varepsilon, ab, abab, ...\}$ **$\{(ab)^n \mid n \in \mathbb{N}\}$**	$S \rightarrow \varepsilon \mid abS$ (Sab would also work fine)	**$\{a^m b^n \mid n, m \in \mathbb{N}, m > 0, n > 0\}$**	$S \rightarrow AB$ $A \rightarrow aA \mid a$ (derives at least one a) $B \rightarrow bB \mid b$ (derives at least one b)
$\{ac, abc, abbbc, ...\}$ **$\{ab^n c \mid n \in \mathbb{N}\}$**	$S \rightarrow aBc$ (puts the a and c on the ends) $B \rightarrow bB \mid \varepsilon$ (derives any number of b's)	**$\{a^n b^n : n > 0\}$**	$S \rightarrow aTb$ $T \rightarrow aTb \mid \epsilon$

The **<u>acceptance problem</u>** for TMs is the language $L_{TM\text{-}acceptance}$ = {M, w | M is a TM that accepts w} where the language is recognisable.
It is shown by building a TM U that recognises it where U on input M, w, simulates M on w, accepts if M enters q_{accept} and rejects if M enters q_{reject} (and diverge otherwise).

The **<u>halting problem</u>** is the language $L_{TM\text{-}Halting}$ = {⟨M, w⟩ | M is a TM that halts on input w} where the language is not decidable.
It is shown by supposing that the problem is decidable and use this to build a decider for the TM acceptance problem, which is known to be undecidable.

Every **regular** language is **context-free**. Every context-free language is **decidable**. Every decidable language is **Turning-recognizable**.

The non-decidable languages are closed under complement. Suppose there exists a non-decidable language L, with its complement L′decidable. Since L′ is decidable, there exists some TM M1 such that M1 decides L′ (M1 always accepts strings in L′, and rejects strings outside L′). Construct another TM M2 that decides L, by swapping all halt-accept with halt-reject in M1 to M2, which suggests that L is decidable, which is false. Hence, the complement of a non-decidable language is also non-decidable.

P (algorithm solving the task that runs in polynomial time) **is closed under union.** the decidable languages are closed under union also shows that that P languages are closed under union. If M1, M2 have polynomial-time complexity, say p1(n) and p2(n), then the machine for their union that runs M1, then runs M2, and accepts if either of the machines accepted, and rejects otherwise, runs in time p1(n) + p2(n), which is also a polynomial.

Formula F is **<u>valid</u>** if every assignment satisfies F. *"if truth-table always has value 1"*	Formula F is **<u>satisfiable</u>** if at least 1 assign satisfies F. *"if truth-table has at least a 1"*

Idempotent Laws	F ≡ (F ∧ F)	F ≡ (F ∨ F)
Commutative Laws	(F ∧ G) ≡ (G ∧ F)	(F ∨ G) ≡ (G ∨ F)
Associative Laws	(F ∧ (G ∧ H)) ≡ ((F ∧ G) ∧ H)	(F ∨ (G ∨ H)) ≡ ((F ∨ G) ∨ H)
Absorption Laws	(F ∧ (F ∨ G)) ≡ F	(F ∨ (F ∧ G)) ≡ F
Distributive Laws	(F ∧ (G ∨ H)) ≡ ((F ∧ G) ∨ (F ∧ H))	(F ∨ (G ∧ H)) ≡ ((F ∨ G) ∧ (F ∨ H))
de Morgan's Laws	¬(F ∧ G) ≡ (¬F ∨ ¬G)	¬(F ∨ G) ≡ (¬F ∧ ¬G)
Double Negation Law	¬¬F ≡ F	
Validity Law	(F ∨ ⊤) ≡ ⊤	(F ∧ ⊤) ≡ F
Unsatisfiability Law	(F ∨ ⊥) ≡ F	(F ∧ ⊥) ≡ ⊥

Constant Laws	$\top \equiv (F \lor \neg F)$	$\bot \equiv (F \land \neg F)$
Negating constants Laws	$\neg\top \equiv \bot$	$\neg\bot \equiv \top$
Conditional Law	$(F \to G) \equiv (\neg F \lor G)$	
Bi-conditional Law	$(F \leftrightarrow G) \equiv ((F \to G) \land (G \to F))$	

Conjunction Introduction	expects two references of the lines to form a conjunction, in any order.
Conjunction Elimination	expects one reference from the original conjunction.
Disjunction Introduction	expects one reference of the line to form a conjunction.
Disjunction Elimination	expects five references, in this order: **1.** Disjunction (A∨B); **2.** Assumption of A; **3.** Conclusion from A; **4.** Assumption of B; **5.** Conclusion from B.
Implication Introduction	expects two references, in this order: **1.** The assumption of A; **2.** The conclusion inferred from A.
Implication Elimination	expects two references, in this order: **1.** Formula A; **2.** Formula (A→B).
Negation Introduction	expects two references, in this order: **1.** The assumption of A; **2.** ⊥ inferred from A.
Negation Elimination	expects two references, in this order: **1.** The assumption of ¬A; **2.** ⊥ inferred from ¬A.
Falsum Introduction	expects two references, in this order: **1.** Formula A; **2.** Formula ¬A.
Falsum Elimination	expects one reference of the line where ⊥ is.

Line	Assumptions	Formula	Justification	References		Line	Assumptions	Formula	Justification	References
1	1	A	Asmp. I		Conjunction Elim. →	1	1	$((A \land B) \land C)$	Asmp. I	
2	2	B	Asmp. I		← Conjunction Intro.	2	1	$(A \land B)$	**∧ E**	1
3	1, 2	$(A \land B)$	**∧ I**	1,2		1	1	$((A \land B) \lor (A \land C))$	Asmp. I	
1	1	$((A \land B) \land C)$	Asmp. I		← Disjunction Intro.	2	2	$(A \land B)$	Asmp. I	
2	1	$(((A \land B) \land C) \lor A)$	**∨ I**	1		3	2	A	∧ E	2
1	1	p	Asmp. I			4	4	$(A \land C)$	Asmp. I	
2	2	$(q \to r)$	Asmp. I		Disjunction Elim. →	5	4	A	∧ E	4
3	3	$(p \to q)$	Asmp. I			6	1	A	**∨ E**	1,2,3,4,5
4	1, 3	q	→ E	1, 3		1	1	$F \to G$	Asmp. I	
5	1, 2, 3	r	→ E	4, 2	Implication Elim. →	3	3	F	Asmp. I	
6	2, 3	$(p \to r)$	**→ I**	1, 5	← Implication Intro.	4	1,3	G	**→ E**	1,3
3	3	A	Asmp. I		← Negation Intro.	2	2	$\neg p$	Asmp. I	
4	4	$\neg A$	Asmp. I		← Negation Elim.	3	3	p	Asmp. I	
5	3, 4	$\bot$	⊥I	3, 4	Falsum Intro. →	4	2, 3	$\bot$	**⊥ I**	3, 2
6	4	$\neg A$	**¬ I**	3, 5	Falsum Elim. →	5	2, 3	$(q \lor r)$	**⊥ E**	4
7	3	A	**¬ E**	4, 5						

Every human loves themself	$\forall x$ loves (x, x)
Everyone loves everyone	$\forall x \forall y$ loves (x, y)
Someone loves everyone	$\exists x \forall y$ loves (x, y) $\exists x \forall y P(x, y)$ means there is a single x such that for all y we have that $P(x, y)$ is true.
Everyone loves someone	$\forall x \exists y$ loves (x, y) $\forall x \exists y P(x, y)$ means for every x there is a y (different y for different x choice) such that $P(x, y)$ is true.
Someone loves someone	$\exists x \exists y$ loves (x, y)
All As are Bs	$\forall x(A(x) \to B(x))$
Some As are Bs	$\exists x(A(x) \land B(x))$
No As are Bs	$\forall x(A(x) \to \neg B(x))$
Some As are not Bs	$\exists x(A(x) \land \neg B(x))$

$\neg \forall x F \equiv \exists x \neg F$

$\neg \exists x F \equiv \forall x \neg F$

$(\forall x F \land \forall x G) \equiv \forall x(F \land G)$

$(\exists x F \lor \exists x G) \equiv \exists x(F \lor G)$

$\forall x \forall y F \equiv \forall y \forall x F$

$\exists x \exists y F \equiv \exists y \exists x F$

if $x \notin \mathrm{Free}(G)$:

$(\forall x F \land G) \equiv \forall x (F \land G)$

$(\forall x F \lor G) \equiv \forall x (F \lor G)$

$(\exists x F \land G) \equiv \exists x (F \land G)$

$(\exists x F \lor G) \equiv \exists x (F \lor G)$