APPENDIX C: SUMMARY

LOGIC – not
$\wedge$ and
$\vee$ (inclusive) or
$\rightarrow$ implies (F$\rightarrow$T/F)
$\leftrightarrow$ same truth values

TAUTOLOGY
A statement which is TRUE in all cases. If there are no quantifiers you can use a Truth Table to test.

QUANTIFIERS
$\forall$x: for all x
$\exists$x: for some x
$\exists x \forall y \rightarrow \forall y \exists x$ but not conversely

NEGATION RULES
negation swaps $\wedge$ with $\vee$ and $\forall$ with $\exists$
– $p \rightarrow q = p \wedge -q$
– $(p \leftrightarrow q) \leftrightarrow -p \leftrightarrow q$

SETS
$\in$ element
$\subseteq$ subset
– complement
$\cap$ intersection
$\cup$ union
$\times$ Cartesian product

EQUIVALENCE RELATIONS
Reflexive: xRx for all x
Symmetric: xR$y \rightarrow y$Rx
Transitive: xR$y \wedge y$R$z \rightarrow x$Rz
Eqvce class $[x]_R = \{y | x$R$y\}$

FUNCTIONS
f:S$\rightarrow$T unique
$f(x) \in$ T,$\forall x \in$S
domain = S;
codomain = T;
image = $\{f(x) | x \in S\} \subseteq$ T
g$\circ$f = f'n of f'n

1-1 & ONTO
***f* is 1-1 if**
f(x)=f(y) $\rightarrow$ x = y
f is onto if
$\forall y \exists x$[f(x) = y]
f–1 exists iff f is 1-1 & onto

LANGUAGES **ALPHABET:** set of symbols **STRING:** sequece of symbols **LANGUAGE:** set of strings	**SUM OF LANGUAGES** + means union Arithmetic of languages differs from numbers: $A + A = A$; $(A + B) - B \neq A$.
MULT'N OF STRINGS $(x_1 \dots x_n)(y_1 \dots y_m)$ $= x_1 \dots x_n y_1 \dots y_m$ **MULT'N OF LANGUAGES** $LM = \{\alpha\beta \mid \alpha \in L, \beta \in M\}$	**ASSOCIATIVE** $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ **NOT COMMUTATIVE** $\alpha\beta \neq \beta\alpha$ in general
LENGTH $\lvert x_1 x_2 \dots x_n \rvert = n$ $\lvert\alpha\beta\rvert = \lvert\alpha\rvert + \lvert\beta\rvert$ **NULL STRING** $\lvert\lambda\rvert = 0$ $\alpha\lambda = \lambda\alpha$ for all α	**POWERS** L^n = all strings which can be cut into n pieces, each coming from L. $L^1 = L$; $L^0 = \{\lambda\}$
KLEENE STAR L^* = all strings which can be cut into zero or more pieces, each from L. L^* always contains λ.	**REGULAR EXPRESSIONS** $\varnothing$, λ, 0, 1 are reg E reg $\rightarrow$ (E), E* reg E, F reg $\rightarrow$ E+F, EF reg

FINITE STATE MACHINES

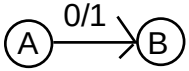
States: A, B, ...

Initial state: the state in which the machine starts.

Transitions: show what state is reached from what in response to an input character.

MEALY MACHINE

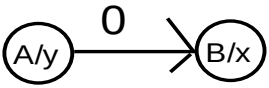
A character is output at the time of the transition.



If in state A and read 0, go to state B and print 1.

MOORE MACHINE

A character is output as the machine enters a state.

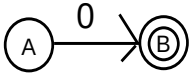


If in state A and read 0, go to state B and print x.

FSA

Accepting states: a chosen set of states.

A string is accepted by the FSA if the machine ends in an accepting state when that string is input.



B here is accepting.

NON-DETERM FSAs

* multiple initial states
* multiple transitions
* null transitions

EASIER TO DESIGN

MULTIPLE TRANSITIONS

More than one arrow out of a state for the same input character.

MULTIPLE INITIAL STATES

More than one initial state.

λ-TRANSITIONS

Transitions which can occur without input.

i.e. “free trips”

NDFSA →DFSA

(1) Set up table. For example in row B, col 0 put all states you can get to by a sequence of λ's followed by 0.

[Ignore any λ's after the 0.]

(2) States which can reach an accepting state by λ's are made accepting.

(3) Set up new table with states 0, 1, ... representing sets of states. State 0 is the set of initials states. If a new set arises encode it with the next available number.

(4) Sets containing accepting states are accepting.

MINIMIZATION	**INACCESSIBLE STATES**
(1)DELETE INACCESSIBLE STATES (2) COMBINE EQUIV STATES (3) PUT IN STANDARD FORM After this process, identical machines become identical.	Start with the initial states and list those which can be reached. Now start with these and continue until all the states reached from those listed are also listed. Any othersare inaccessible and may be deleted.

EQUIVALENT STATES

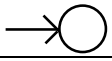
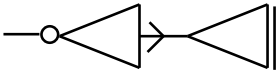
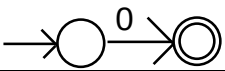
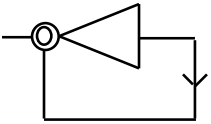
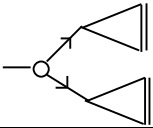
Label the states 0, 1, .. according to their output. For FSA's label accepting states as 1 and others as 0. Convert the transition table using the most recent equivalence col.

Label first state 0. Scan the cols back to and including previous eqvce col. Combinations that have already occurred get same code as before. New combinations get the next code number. Terminate when two partitions are identical (or when each state is by itself).

* Select a leader from each eqvce calss in this final partition.
* Remove all other rows from the FSA table.
* Convert any reference to a deleted state to its leader self.

STANDARD FORM

Code the states 0, 1, 2, ...
with 0 being the initial state and new codes allocated as needed when reading the table row by row.

REGULAR LANGUAGES One which can be expressed as a regular expression.	FSA for ∅ 
They are languages which are accepted by a FSA. The parenthesis language is non-regular.	FSA for LM 
FSA for λ 	FSA for 0 (and 1) 
FSA for L* 	FSA for L + M 

PROOFS * Respond to logical structure of the statement. * Put goal in "goal window" and update. * Declare all variables.	**if p then q** PROOF: Suppose p. GOAL WINDOW: q
not p Suppose p. ..... Contradiction!	**p or q** Suppose not p. ..... ∴ q.

TURING MACHINES

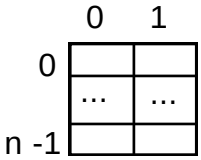
infinite tape +
read/write head

□□□□□□□□□□

↑

n is represented by a string of n 1's with head on leftmost 1.

States 0 to $n-1$
(0 = start).



Halt when sent to state *n*.

Typical instruction:
c D s = print c, move in direction D, go to state s.
c = 0, 1;
D = L(eft), R(ight)
s = 0, 1, 2, ... *n*.
(n = halt)

HALTING PROBLEM (UNSOLVABLE)
Construct a TM which, given the table of a TM, M, plus data, will determine whether M will halt when given that data.

Ω has $a + kb + c$ states.
$\Omega(o) = \beta(2^n)$.
Choose k so:
$2^k > a + kb + c$.
$\therefore \beta(2^k) > \beta(a+kb+c)$
$> \beta(2^k)$.
Contradiction!

If the Halting Problem was solvable we could compute $\beta(n)$ by constructing all n-state TM's, removing the non-halting ones and simulasting the rest in parallel.

INTEGERS MOD *m*	**EULER'S φ-FUNCTION**
$\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$ with addn and multn mod m (remainders after dividing by m)	$\varphi(m)$ = # of numbers from 1 to $m-1$ which are coprime with m.
If p is prime $\varphi(p) = p-1$ and $\varphi(p^n) = p^{n-1}(p-1)$. If p, q are distinct primes $\varphi(pq) = (p-1)(q-1)$.	**EULER'S THEOREM** If GCD(a, n) = 1 $a^{\varphi(n)} = 1 \pmod{n}$.

PUBLIC KEY CODE SETTING UP:	**KNOWLEDGE:**
Issue everyone with numbers e, d, p, q, n such that p, q are distinct primes, $n = pq$ and $ed = 1 \pmod{\varphi(n)}$.	Everyone knows their own n and d and everyone's e. All other info is secret.

ENCODING: work mod *n*	**DECODING: Work mod *n***
Convert message to a number m in range 1 to $n-1$ where GCD(m, n) = 1. Look up receiver's e end send $m' = m^e \pmod{n}$.	Receive m' and use your own d to find $m'' = (m')^d = m^{ed} = m$.

POLYNOMIALS	**PRIME POLYNOMIALS**
$f(x) = a_0 + a_1x + \ldots + a_nx^n$ is a poly in x with coeffs $a_0, a_1, \ldots$ of degree n (provided $a_n \neq 0$).	are those (of deg ≥ 1) which can't be factorized into polys of lower degree.

PRIME POLYS OVER $\mathbb{Z}_2$ x, $x+1$ $x^2 + x + 1$, $x^3 + x^2 + 1$, $x^3 + x + 1$, ...	$\mathbb{Z}_2[t \mid p(t) = 0]$ = polys in t up to deg < deg p(x) with add'n and mult'n mod p(t). It's a field iff p(x) is prime.
POLYNOMIAL CODES SETTING UP: Choose prime $N = 2^n - 1$. Choose prime e(x) $\in \mathbb{Z}_2[x]$ of deg n. Let $M = N - n$. Let t be a zero of e(x) and prepare a table of powers of t up to t^{N-1}, reduced mod e(x).	**BINARY STRING $\leftrightarrow$ $\mathbb{Z}_2$ POLYNOMIAL** Adopt some convention to convert a binary string of length n+1 to and from a poly of deg $\leq n$. e.g first bit becomes const term or the coeff of x^n.
ENCODING: Convert message to poly m(x) of deg $\leq n$ Send s(x) = m(x) e(x). Suppose the received message is r(x) which either equals s(x) or differs from it in one term (an error).	**DECODING:** Substitute x = t, using the table. If the result is 0, there was no error. If the result is $\neq 0$, look up the table backwards to find the corresonding power t^k. Correct the received mesage by adding x^k.