Register Machines Finitely many registers, program with instructions of form:
L: R^+ □ L'
L: R^- □ L',L"
L: HALT
Computation halts because of HALT or erroneous jump
Specifies partial func. (↑ undef.)
A partial function f : N^n □ N is computable if there is a RM M with n+1 registers such that $f(x_1,\dots,x_n) = y$ □ the computation of M with $R_i = x_i$ halts with $R_0 = y$

Coding Register Machines
$\langle\!\langle x, y \rangle\!\rangle = 2^x(2y+1)$
$\langle x, y \rangle = 2^x(2y+1)-1$

$\langle\!\langle , \rangle\!\rangle$ is a bijection onto N/{0}
$\langle , \rangle$ is a bijection onto N

Lists Nil = 0, Cons x l = $\langle\!\langle x, l \rangle\!\rangle$

Programs
$code(R_i^+ \to L_j) = \langle\!\langle 2i, j \rangle\!\rangle$
$code(R_i^- \to L_j, L_k) = \langle\!\langle 2i+1, \langle j,k \rangle \rangle\!\rangle$
$code(HALT) = 0$
Bijection to N (prog. w/ index e)

Universal Register Machine

1. Copy P to T, copy PCth list item in T to N
2. If N=0 HALT else decode N as $\langle\!\langle y, z \rangle\!\rangle$, assign y to C, z to N
3. Remove register values from list in A up to required one (which is put in R), saving preceding values as list in S (deal with high reg. by zero filling)
4. Execute instruction on R, update PC, restore register values from R, S to A
5. Repeat from step 1

Halting Problem
A RM H decides the halting problem if, loading R_1 with e, R_2 with $[a_1,\dots,a_n]$, computation of H halts with R_0 containing either 0 or 1, and R_0 contains 1 when H halts □ the computation of the RM program $Prog_e$ started with $R_1,\dots,R_n = a_1,\dots,a_n$ does halt
No such H can exist since you can obtain from H an H' that runs the supplied program on the index of the supplied program and does the opposite of the result. Now running H' with the index of H's program reveals a contradiction.

Computable $\varphi_e(x) = y \leftrightarrow$ the computation of

Functions $Prog_e$ starting with R_1=x and other regs. 0 halts with R_0 = y
Not all pfns. computable:
$$f(e) = \begin{cases} 0 & \text{if } \varphi_e(e)\uparrow \\ undefined & \text{if } \varphi_e(e)\downarrow \end{cases}$$

Decidable Sets A subset S of N is decidable □ exists a RM M such that
$\varphi_{i(M)}(x)\downarrow \wedge \varphi_{i(M)}(x) = 1 \leftrightarrow x \in S$

Turing Machines Set Σ of tape symbols, $\_, \triangleright \in \Sigma$
Set K of machine states, $s \in K$
A transition function,
$\delta \in Fun(K \times \Sigma,$
$(K \cup \{acc, rej\}) \times \Sigma \times \{L, R, S\})$
which always moves right when over the initial tape symbol
Configuration specified by (q,l,r)
$q \in K \cup \{acc, rej\}$, l, r left and right finite tape symbol lists
Halts if transition seq. finite

Church-Turing Every algorithm (in the intuitive sense) can be realized as a Turing machine
Extensions to TM, alternative formalizations have been shown to determine same set of computable functions

Kleene Equivalent $e \equiv e'$ if either both are undefined or they are both defined and the values they denote are equal

Primitive Recursion
$proj_i^n(x_1,\dots,x_n) = x_i$
$zero^n(x_1,\dots,x_n) = 0$
$suc(x) = x+1$
$f \circ (g_1,\dots,g_n)(x_1,\dots,x_m) = z \leftrightarrow$
$g_i(x_1,\dots,x_m) = y_i \wedge f(y_1,\dots,y_n) = z$
$[\rho^n(f,g)](x_1,\dots,x_n,x) = y \leftrightarrow$
$f(x_1,\dots,x_n) = y_0 \wedge \forall y \in Z_x$
$g(x_1,\dots,x_n,i,y_i) = y_{i+1} \wedge y = y_x$
Primitive recursive functions are only those built using these rule
All computable and total

Partial Recursion
$[\mu(f)](x_1,\dots,x_n) = x \leftrightarrow$
$\forall i \in Z_{x+1}. f(x_1,\dots,x_n,i) = y_i \wedge$
$\forall i \in Z_x . y_i > 0 \wedge y_x = 0$
Partial recursive functions are those built up using this rule

and those of primitive recursion
There are total recursive functions which are not primitive recursive. Given formal descriptions of primitive recursive functions, say $e(x,y)=f_x(y)$ if x is such a description. Now $e'(x)=e(x,x)+1$ is not primitive recursive since if it was there would be x such that $e'=f_x$, and so $f_x(x) = e'(x) = e(x,x)+1 = f_x(x)+1$
All computable, furthermore all computable functions are in PR

Enumeration

S is recursively enumerable □ it is empty or there is a total recursive function f s.t.:
$S = \{f(n) \mid n \in N\}$
S is co-r.e. iff N\S is r.e.
Decidable set □ recursive set
S is recursive □ it is both r.e. and co-r.e. (i.e. run r.e. and co-r.e. machines together)
Semi decidable □ in_S in PR

$$in_S(x) = \begin{cases} 1 & x \in S \\ undefined & x \notin S \end{cases}$$

S=Im(f) □ S r.e. for f′(x) by decoding x as (a, t) and running f(a) for t steps
S r.e. □ S=Dom(f) for f′=μ(g) g(x,y)=0 only if f(y)=x
S semi decidable □ S=Im(f) by f′(x)=x iff $in_S(x)\downarrow$
Recursive set □ r.e. set
Some sets are not r.e. (e.g. set of codes of total functions: consider tot. $g(x) = \varphi_{f(x)}(x) + 1$)
Some r.e. sets are not recursive (e.g. set of codes of functions that halt w/ input 0: is r.e. because = $Dom(\varphi_x(0))$)
S=Dom(f) □ S semi decidable since $in_s(x) = ifzero(f(x),1,1)$ PR