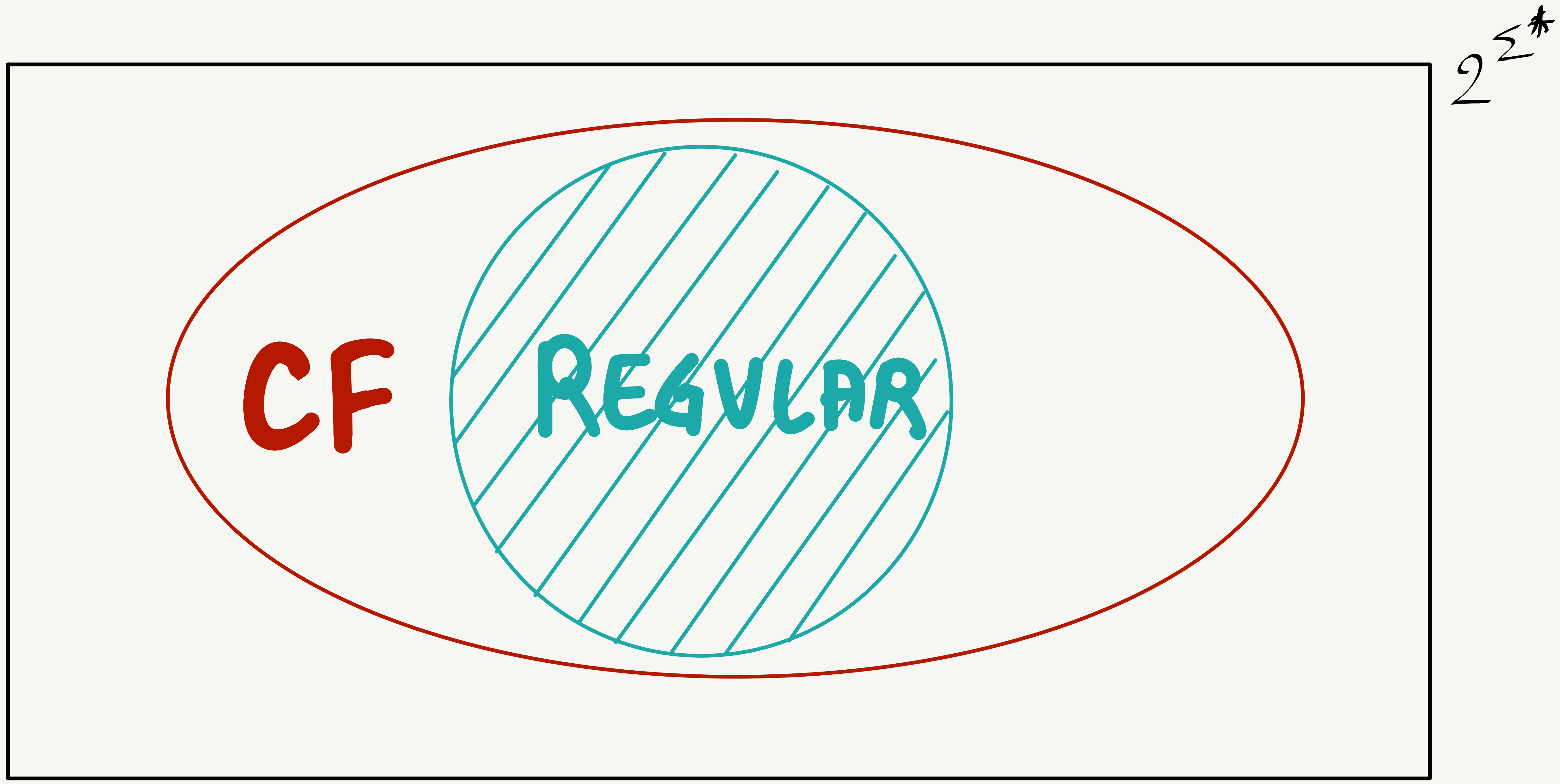
TOWARDS COMPUTABILITY

So far:



We saw operational characterizations of these language classes.

Regular : Deterministic / non-deterministic finite automata

Context-free : Pushdown automata

https://forms.office.com/r/QMf1LnTQ3h

But computation is not just about recognizing languages

What about calculating various functions?

Are all functions computable? there is an effective procedure

Is there always an effective procedure to calculate the value of a function on arbitrary inputs?

$f_1(x, y) = x + y$ $\qquad f(x) = 0$ $\qquad f_3(x,y) = \begin{cases} T, & \text{if } x \% y = 0 \\ F, & \text{otherwise} \end{cases}$

$$f_4(x) = \begin{cases} \text{"Success"}, & \text{if there are } x \text{ consecutive 4s in the decimal expansion of } \pi \\ \text{"Failure"} & \end{cases}$$

https://forms.office.com/r/QMf1LnTQ3h

Any effective procedure must terminate and return expected output in a finite number of steps, each of which takes a finite amount of time.

This is an algorithm.

Are all functions computable?

Not all functions have algorithms; not all functions are computable.

Which functions are computable? Which ones are not?

Needs some uniform notion of a computation of a machine which can perform any such?

https://forms.office.com/r/QMf1LnTQ3h

Register machine:

* Infinite supply of registers $R_1, R_2, \ldots$.

  Number contained in R_n represented by r_n.
* A program is a finite list of instructions
* Instructions are of four types:

Zero: For each $n = 1, 2, 3, \ldots$ $Z(n)$ sets r_n to 0, all others unchanged

Next: For each $n = 1, 2, 3, \ldots$ $S(n)$ increments r_n by 1, all others unchanged

Transfer: For each $m = 1, 2, 3, \ldots$ and each $n = 1, 2, 3, \ldots$

$T(m, n)$ replaces r_n by r_m in R_n, all others (R_m also!) unchanged

Jump: For each $m = 1, 2, 3, \ldots$, each $n = 1, 2, 3, \ldots$, and each $p = 1, 2, 3, \ldots$

$J(m, n, p)$ takes the machine to the p^{th} instruction if $r_m = r_n$, and continues to the next instruction otherwise

if this does not exist, halt!

Start with a program P, and an initial configuration (values of r_i for $i = 1, 2, 3, \dots$)

Example: P is the following program, with the initial configuration

R_1	R_2	R_3	R_4	R_5	
9	7	0	0	0	. . . . . .

$J(1,2,6)$

I_1:	$J(1,2,6)$
I_2:	$S(2)$
I_3:	$S(3)$
I_4:	$J(1,2,6)$
I_5:	$J(1,1,2)$
I_6:	$T(3,1)$

Start with a program P, and an initial configuration (values of r_i for $i = 1, 2, 3, \ldots$)

Example: P is the following program, with the initial configuration

R_1	R_2	R_3	R_4	R_5	
9	7	0	0	0	. . . .
9	7	0	0	0	. . . .

$J(1,2,6)$

$S(2)$

$I_1 : J(1,2,6)$

$I_2 : S(2)$

$I_3 : S(3)$

$I_4 : J(1,2,6)$

$I_5 : J(1,1,2)$

$I_6 : T(3,1)$

Start with a program P, and an initial configuration (values of r_i for $i = 1, 2, 3, \dots$)

Example: P is the following program, with the initial configuration

R_1	R_2	R_3	R_4	R_5	
9	7	0	0	0	. . . . . .
9	7	0	0	0	. . . . . .
9	8	0	0	0	. . . . . .

$J(1,2,6)$
$S(2)$
$S(3)$

$I_1 : J(1,2,6)$
$I_2 : S(2)$
$I_3 : S(3)$
$I_4 : J(1,2,6)$
$I_5 : J(1,1,2)$
$I_6 : T(3,1)$

Start with a program P, and an initial configuration (values of r_i for $i=1,2,3,\ldots$)

Example: P is the following program, with the initial configuration

R_1	R_2	R_3	R_4	R_5		
9	7	0	0	0	. . . .	$J(1,2,6)$
9	7	0	0	0	. . . .	$S(2)$
9	8	0	0	0	. . . .	$S(3)$
9	8	1	0	0	. . . .	$J(1,2,6)$
9	8	1	0	0	. . . .	$J(1,1,2)$
9	8	1	0	0	. . . .	$S(2)$
9	9	1	0	0	. . . .	$S(3)$
9	9	2	0	0	. . . .	$J(1,2,6)$
9	9	2	0	0	. . . .	$T(3,1)$
2	9	2	0	0	. . . .	No further instructions; Halt

I_1: $J(1,2,6)$
I_2: $S(2)$
I_3: $S(3)$
I_4: $J(1,2,6)$
I_5: $J(1,1,2)$
I_6: $T(3,1)$