* Languages as abstractions of problems

Every computational problem can be thought of as a function $f: \{0,1\}^* \to \{0,1\}^*$

$\{0,1\}^*$ = set of all binary strings of finite length

$L_f = \{ (x, i, b) \mid i^{th} \text{ bit of } f(x) \text{ is } b \}$

$b \in \{0, 1, \perp\}$

↳ denotes the end of the string corresponding to $f(x)$

P1: Given x, compute $f(x)$

P2: Given (x, i, b), check if $(x, i, b) \in L_f$

Theorem: $\exists$ an algorithm for P1 iff $\exists$ an algorithm for P2

Encoding L_f:

(	-	000
)	-	001
0	-	010
1	-	011
,	-	100
$\perp$	-	101

This encoding shows that $L_f \subseteq \{0,1\}^*$

Language: Given an alphabet Σ (say $\{0,1\}$)

a language $L \subseteq \Sigma^*$

Decision problems: For $L \subseteq \Sigma^*$, given $x \in \Sigma^*$ decide if $x \in L$

* Can all computational problems solved by writing a C program?

Theorem: The # of C programs is countable

Proof: Every C program can be encoded as a binary string

$\Rightarrow$ Set of C-programs $\subseteq \{0,1\}^*$

$\exists$ bijection $f: \{0,1\}^* \to \mathbb{N}$

$f(0)=1 \quad f(1)=2 \quad f(00)=3 \quad f(01)=4 \quad f(10)=5$
$f(11)=6 \quad f(000)=7 \ldots\ldots$

Exercise: Show that f is a bijection

$\{0,1\}^* = \epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011 \ldots$
$\downarrow \downarrow \downarrow \downarrow \downarrow$
$0 \; 1 \; 2 \; 3 \; 4$

$\overline{b} = \{0,1\}^* - \epsilon$

$f(\overline{b}) = n(\overline{b}) + 2^{|\overline{b}|-1}$