Type System

C

```
int i;
i + 2;
i + "Hello"
```

C

```
float i;
i + 2;
```

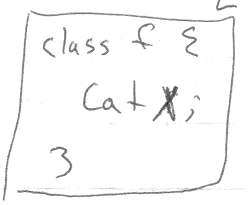
- C annotations do operator overloading
- Determines the size

Java

```
Int i;
fix = i;
if (i == NULL)
...
```

Java

```
Dog;
fix = i;
if (i == nULL)
...
```



- Java types prevent programs from compiling

"Unitype" = There is one type (C, Python, Racket, Javascript, C++)
academic phrase

"Strong/Static/Sound Type" = There are many types
Industry phrase

C number rep = 32 bits

Py num = tag + 32-bits
↳ a special number (67 = number)

Py bool = tag → 18 = true, 19 = false

ML num rep = 32 bits
bool rep = 32 bits

A type system prevents programs from compiling. (aka It separates valid expressions from programs)

$M = 0 \mid 1 \mid \text{true} \mid \text{false} \mid (M + M) \mid (M \oplus M)$

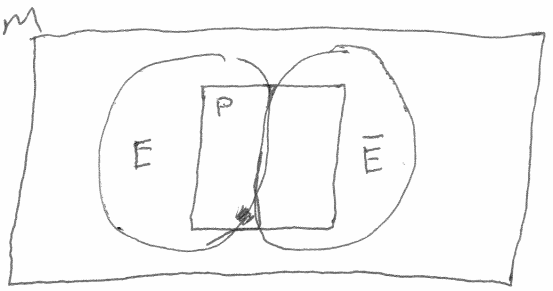
$(0 + \text{true}) \in M$ "$0+1$)" $\notin M$ "q" $\notin M$

Type system define $P \subseteq M$ of valid programs

$(0 + \text{true}) \notin P$

Initially $(M - P)$ = "programs with errors"

P = "program without errors"



"error" $\neq$ "right program"

"error" $\neq$ "throws exception" because we meant that

"error" = "makes no sense" = stuck

Halting Problem / Gödel Incompleteness $\Rightarrow$

$\rightarrow$ good programs will be called bad ($\bar{E} \in \bar{P}$)

bad programs will be called good ($E \subset P$)

$\rightarrow$ means = Type System is an accurate (but pessimistic) predictor,

Before running a program, predict if it will be stuck.

"No" is a valid type system (ie $P = \emptyset$)

"Yes" is valid if $\Rightarrow$ is total (ie $P = M$ because all stuck states were removed)

A "useful" type system has a "big" P

The more precise (ie large) P is, then the more expensive

```
int x; if (cond) { x = A; } else { x = B };
int y; if (cond') { y = C; } else { y = D };
   ⋮
do operation  f(x, y)
```

$\bar{E}$ — [f(A, C) is defined; f(B, D) is defined]

E — [f(A, D) is stuck; f(B, C) is stuck]

cond $\Leftrightarrow$ cond'

Bool ↓ SAT = EXP = 2^n

~~$\Rightarrow$~~ Bool ↓ Gödel $\Rightarrow$ undecidable

```
int x, y;
if (cond) { x = A; y = C; } else { x = B; y = D };
 f(x, y);
```

Relation "$\vdash M : T$" = M has type T

Relation "$\vdash M$" = M does not get stuck (ie $M \in P$)

$M = 0 \mid 1 \mid M+M \mid T \mid F \mid M \wedge M \mid E ? M$

$T = N \mid B$

$\vdash 0 : N \qquad \vdash 1 : N$

$\vdash T : B \qquad \vdash F : B$

$$\frac{\vdash L : N \quad \vdash R : N}{\vdash L + R : N}$$

$$\frac{\vdash M : N}{\vdash E ? M : B}$$

$$\frac{\vdash L : B \quad \vdash R : B}{\vdash L \wedge R : B}$$

$(1, B) \notin \vdash : \qquad (0+0, N) \in \vdash :$

$$\frac{\vdash M : T}{\vdash M}$$

"M won't get stuck if it has some type"

Soundness:

$\forall m. \vdash m \rightarrow eval(m) = v$ for some v

(what if m runs forever?)

1) Progress

$\forall m, T. \quad \vdash M : T \rightarrow (\exists m'. \; M \rightarrow m')$ or $M \in V$

2) Presservation

$\forall m, T, m'. \quad \vdash M : T \wedge M \rightarrow M' \rightarrow \vdash M' : T$

$0 + T \not\rightarrow$

$E ? T \not\rightarrow$

$E ? (1+1) : B$

$\rightarrow E ? 2 : B$

$\rightarrow T : B$