Type Theory

Identify programs w/ errors before running them

"int x;" "*x(17);" → "Yo, did you x is an int? Can't jump to 17!"
"(int(x)(int))x(17)"

"(λx. (+ 17" → no. NOT a program
"17 + (λx.x)" → stuck, Yes! (find all stuck)
"42" → no.

"42 should be 50" → Yes
"42 should be inc(50)" → g is an abstraction / concrete
g(50) = 50 g(50) = positive g(50) = even
g(50) = int

A type system verifies P(M) for all M ∈ the language
P(·) = M does not deref NULL ptrs
P(·) = M acquires lock in alphabetical order

"assert M" "assert False" ≠> stuck
"assert τ" ≡> τ

simplify to => stop / detect stuck programs

A type system will be a relation on programs

NoNullDerefs $\subseteq \mathcal{P}$(Program)

*NULL ∉ NND

1 + 1 ∈ NND

Most typesystems are decidable

good ts. are linear runtime

ideally no (or positive) runtime impact

HasType $\subseteq \mathcal{P}$(Program × Datatype)

(1, int) ∈ HT

(1, bool) ∉ HT

(x,) ∈ HT

Env

HasType $\subseteq \mathcal{P}$(Program × DT)

(any, 1, int) ∈

("intx", x, int) ∈

("intx", x, bool) ∉

(any, 1+1, int) ∈ HT

(any, 1+2, int) ∈ HT

("intx", 1+3, int) ∈ HT

premise

conclusion

$\forall E, L, R.$ (E, L, int) ∈ HT (E, R, int) ∈ HT

(E, L+R, int) ∈ HT

(Env, Program, Type) ∈ HT

Env ⊢ P : T

Γ ⊢ L : int Γ ⊢ R : int

Γ ⊢ L+R : int

Γ (Env) proves that (⊢) has type (:) τ

Γ ⊢ 6 : int

type judgment

type rule

20-3/

$M := x \mid b \mid \lambda x.M \mid o^n M \ldots \mid M\ N$

b N is stuck

$o^n M \ldots (\lambda x.N)\ L \ldots$ is stuck

$T := (T \Rightarrow T) \mid B$ $\quad \Gamma := \cdot \mid \Gamma, x:T$

$\Delta(b) = B$ $\quad$ b is a constant to type fun

$\Gamma \vdash b : B$

$\Delta(5) = Int$

$\Delta(true) = Bool$

$\Gamma \vdash M_i : T_i \quad \Delta(o^n) : T_1 \Rightarrow (T_2 \Rightarrow (T_3 \Rightarrow (\ldots \Rightarrow T_R)))\ldots)$

$\Gamma \vdash o^n M_1 \ldots M_n : T_R$

$\Delta(+) = Int \Rightarrow Int \Rightarrow Int$

$\dfrac{\Gamma \vdash M : T_D \Rightarrow T_R \quad \Gamma \vdash N : T_D}{\Gamma \vdash M\ N : T_R}$ $\qquad$ $\dfrac{\Gamma, x:T_D \vdash M : T_R}{\Gamma \vdash \lambda x.M : T_D \Rightarrow T_R}$ $\qquad$ $\dfrac{\Gamma(x) = T}{\Gamma \vdash x : T}$

TODO: Is the type sys 'correct'?

Is the type sys decidable?

$\forall \Gamma, P, T.\ \{\Gamma \vdash P : T\}$ or $\{\neg\, \Gamma \vdash P : T\}$

$\forall \Gamma, P.\ \{\exists T.\ \Gamma \vdash P : T\}$ or $\{\forall T.\ \neg\, \Gamma \vdash P : T\}$

Is it efficient? run in linear time?

$M := \ldots \mid \lambda x.M \mid \lambda x:T.M$