The Type Base and Undecidability in Type Theory

PRL Seminar

Abhishek Anand

September 21, 2012

The Type Base

- members: all terms of the underlying computation system
- equality : squiggle equality ($\sim$)[1]
- Informally: 2 squiggle equal terms can be substituted for each other in any context. Some of it's key properties are:
  - $t \mapsto t' \Rightarrow t \sim t'$
  - $(t \sim t' \wedge t \in T) \Rightarrow t = t' in T$
  - Any 2 diverging terms are squiggle equal

[1] **Howe89**.

Possibly Simplest Undecidability proof

- Can there be $h : Base \to bool$ such that $h(t)$ is *true* iff t converges?
  - h is total, computable.
- $\bot \stackrel{def}{=} fix(\lambda x.x)$
- $d \stackrel{def}{=} \lambda y.$ if $h(y)$ then $\bot$ else $\lambda x.x$
- What is $h(fix(d))$?
- $h(fix(d)) \mapsto h(f(fix(d))) \mapsto h(\text{if } h(fix(d)) \text{ then } \bot \text{ else } \lambda x.x)$
- So, we have $h(fix(d)) \sim h(\text{if } h(fix(d)) \text{ then } \bot \text{ else } \lambda x.x)$
- $h(fix(d))$ is true iff it is false.

Domain Theoretic Argument

- $d \stackrel{def}{=} \lambda y.$ if $h(y)$ then $\perp$ else $\lambda x.x$
- d is not monotonic
  - $\perp \sqsubseteq \lambda x.x$
  - $d(\perp) \not\sqsubseteq d(\lambda x.x)$
- $\lambda x.x \not\sqsubseteq \perp$
- Monotonocity is implied by congruence of squiggle ordering

Some (Unconventional) Properties of Base

- $A \subseteq B \stackrel{def}{=} (x = y \ in \ A) \Rightarrow (x = y \ in \ B)$
- $Base \rightarrow Base \subseteq Base$
- $Base \rightarrow bool \subseteq Base$

Bar Types

- given a type T which does not equate a converging and diverging term, we can form a type $\bar{T}$
  - members: all terms which are members of T if they converge
  - equality : $t = t'$ *in* $\bar{T} \equiv$
    - ⋆ has-value$(t) \Leftrightarrow$has-value$(t')\wedge$
    - ⋆ has-value$(t) \Rightarrow t = t'$ *in* T
- A type T is admissible[2] if:
  - $f \in \bar{T} \rightarrow \bar{T} \Rightarrow fix(f) \in \bar{T}$
- the type Z is admissible[3]
- $\bar{Z} \subseteq Base$

[2] **Crary98**.
[3] **Crary98**.

Previous Version of the Undecidability Proof

- Can there be $h : \bar{Z} \to bool$ such that h(t) is true iff t converges?
- $\perp \stackrel{def}{=} fix(\lambda x.x)$
- $d \stackrel{def}{=} \lambda y.$ if $h(y)$ then $\perp$ else 0
- $d \in \bar{Z} \to \bar{Z} \Rightarrow fix(f) \in \bar{Z}$
- what is $h(fix(f))$?
- $h(fix(f)) \mapsto h(f(fix(f))) \mapsto h(\text{if } h(fix(f)) \text{ then } fix(\lambda x.x) \text{ else } 0$
- So, we have $h(fix(f)) \sim h(\text{if } h(fix(f)) \text{ then } fix(\lambda x.x) \text{ else } 0)$
- $h(fix(f))$ is true iff it is false.

Corollary: $\neg\forall P : prop.(P \vee \neg P)$

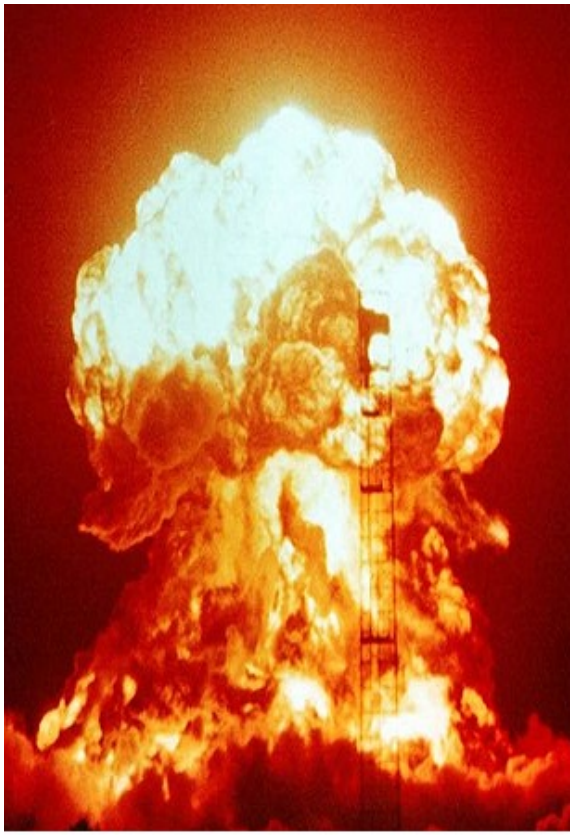
- has-value(t) is a well-formed proposition for t:$\bar{Z}$
- by assuming $\forall P : prop.(P \vee \neg P)$, we get a decider for halting (has-value) on terms of $\bar{Z}$

```
* top 1 1 1

[-]
1. magic : ∀P:ℙ. (P ∨ (¬P))@i'
[-]
⊢ n:bar(ℤ) → {b:𝔹| ↑b ⇐⇒ has-value(n)}

* BY UseWitness ⌜λn.isl(magic has-value(n))⌝·
```

- Contradiction!

How to restore harmony?

- Cannot have both
  - Types which have diverging elements and do not equate a diverging and a converging element. e.g. Base, Bar Types
  - $\forall P : prop.(P \vee \neg P)$
- What to give up?
- We could assume $\forall P : prop.\{(P \vee \neg P)\}$
  - $\{\circ\}$ erases the constructive content(decider)



Base/Bar type+LEM