ON THE COST OF TYPE-TAG SOUNDNESS

Ben Greenman
Zeina Migeed
Northeastern University

RETICULATED
- Gradual typing for Python
- Enforces type-tag soundness

TYPE-TAG SOUNDNESS
If \( \vdash e : \tau \) and \( \tau = K \) then either:
- \( e \rightarrow^{\phi} v \) and \( v \) matches \( K \)
- \( e \rightarrow^{\phi} \Omega \) (runtime error)
- \( e \) diverges

where \( \tau = K \) maps a type to a type-tag for its canonical forms, e.g.:
- \( \lfloor \text{Int} \rfloor = \text{Int} \)
- \( \lfloor \tau \times \tau' \rfloor = \text{Pair} \)
- \( \lfloor \tau \rightarrow \tau' \rfloor = \text{Fun} \)

RUN-TIME ENFORCEMENT
Reticulated performs a run-time tag check on each dynamically-typed value \( v \) that flows into a typed context \( E \) expecting type \( \tau : E[7 : \text{Int}] \rightarrow E[7] \)
\( E[(1,\text{NaN}) : \text{Int} \times \text{Int}] \rightarrow E[(1,\text{NaN})] \)
\( E[\text{snd}((1,\text{NaN})) : \text{Int}] \rightarrow \text{Tag Error} \)

These checks affect performance.

EXPERIMENT
- Evaluated 21 Reticulated programs
- 18 via exhaustive evaluation
- 3 via approximate evaluation
- Ran on the Karst at Indiana University cluster

CONCLUSIONS
- Worst-case overhead: under 10x
- Best-case overhead: 1x -- 4x
- Always slower than Python
- Overhead typically increases linearly with the number of type annotations

REFERENCES
- Vitousek, Swords, Siek. Big Types in Little Runtime:Open World Soundness and Collaborative Blame for Gradual Type Systems. POPL 2017