Meeting 15

Project Ideas

- Make program analysis easy for developers
  - Devs are analyzer creators
  - Interface to show results

Concrete Areas

- Types (looking at things like TypeScript and Gradual Typing)

- Array bounds checks
  - Why does this mean in JavaScript?
- Object-graph construction
  - Points-to analysis (coarse)
  - Shape analysis (precise)

- Property-accesses
  (Reflection things)
\[ \text{Expr} \xrightarrow{F} \text{Val} \]

1. \( \alpha, \beta \text{ monotonic} \)
2. \( \alpha(\hat{V}) \mapsto \exists \)
3. \( \gamma(\alpha(V)) \geq V \)

\[ \begin{array}{c}
\leq 0 \\
\phi \\
T \\
0 \\
1
\end{array} \]

\[ \text{Val} \]
\[ B : \text{Val} \rightarrow \hat{\text{Val}} \]

For signs:
\[ B(n) \overset{\text{def}}{=} \begin{cases} + & \text{if } n > 0 \\ 0 & \text{if } n = 0 \\ - & \text{if } n < 0 \end{cases} \]

\[ \forall v \in \text{Val} \]
\[ \alpha(v) \overset{\text{def}}{=} \bigcup \{ B(v) \mid v \in V \} \]

\[ \forall v \in \hat{\text{Val}} \]
\[ \gamma(v) \overset{\text{def}}{=} \{ v \mid B(v) \subseteq v \} \]
We wrote down the concrete transfer \( F : \text{Expr} \to P(\text{Val}) \)

\( \hat{F}^* \) defined in terms of \( F \). How?

assuming \( \alpha, \delta \) functions

\[
\hat{F}^* : \text{Expr} \to \text{Val}
\]

\[
\hat{F}^* \overset{\text{def}}{=} \alpha \circ \hat{F}
\]

\( \hat{F} : \text{Expr} \to \hat{\text{Val}} \)

show it is sound wrt to \( \hat{F}^* \)

best abstract evaluator
Return to the white language