Trustworthy software

- Various guarantees that we may want a software system to have:
  - reliability (does not crash)
  - safety (cannot be hijacked to do ‘bad’ things)
  - secure (does not divulge private information)
  - obeys certain policies

- How can we guarantee that a software system has these properties?

Trustworthy software

Our approach:
- Write analyses and transformations in a domain specific language called Rhodium
- Check them for correctness automatically

Static analysis for information flow

- “High-security input never affects low-security output”
- Enforceable via sound (incomplete) dataflow analysis
  - \( L \leq H \), assign each variable a level \( \text{sec}(x) \)

\[
\begin{align*}
\text{e1} + \text{e2} & \quad \text{max} (\text{sec}(\text{e1}), \text{sec}(\text{e2})) \\
\text{x} := \text{e} & \quad \text{if} (\text{sec}(\text{x}) \leq \text{sec}(\text{e})) \\
\text{if}(\text{x}) & \quad \text{e}; \\
\end{align*}
\]

- Requires pointer information
- Pointer analyses are hard to get right
Check them for correctness automatically

Write analyses and transformations in a domain specific language called Rhodium

Optimizations in Rhodium

if ( curStmt == [*X := Z] ) and
( mustPointTo(X,Y) before curStmt )
then transform curStmt to [Y := Z]

• We can check Rhodium optimizations for correctness automatically using an automated theorem prover
• Separate profitability from correctness

Current status and future work

• Current status of Rhodium
  – a language for writing analyses and optimizations over a realistic C-like language
  – automated correctness checker
  – implemented and checked a variety of analyses and optimizations in Rhodium

• Future work
  – add support to Rhodium for writing checkers
  – increase expressiveness
  – efficient execution engine
  – infer rules automatically

Leverage domain specific language to allow various kinds of automated processing:
• run Rhodium analyses automatically
• automatically check Rhodium analyses for correctness using an automated theorem prover