Foundational certification of data-flow analyses

Tarmo Uustalu

Joint work with Maria João Frade and Ando Saabas

Theory Days at Voore, 29 Sept.–1 Oct. 2006

MOTIVATION

- In proof-carrying code (PCC), programs come with proofs of functional correctness or safety.
- In particular, this may involve certification of data-flow analyses (to justify optimizations or establish safety).
- But why should one trust a proof, if it checks correctly, assuming it is easy to believe that the checker is correct?
- A proof may be a correct proof in an incorrect proof system, e.g., if the constants of the logic and the inferences rules are specialist.

- *Foundational* PCC sets out to reduce the burden of trusting the certification formalism by preferring special-purpose type systems to program logics, and type systems and program logics to universal-logic formalizations of their underlying semantics.
- The idea: prove everything from first principles or almost so. Less to trust, more to check.
- This results in large certificates: either monolithic (relatively smaller) or modular (bigger).

This talk

- We play with the foundational ideology on data-flow analyses.
- We look at the example of the live variables analysis.
- We show how this is specified formally in a declarative manner as a type system sound wrt. a suitable natural semantics.
- Beyond these two possible levels for reasoning about live variables, we consider three more:
  - a Hoare logic for live variables
  - a natural semantics for def-use futures
  - a Hoare logic for def-use futures

Live variables analysis

- A variable is live at a point on a computation path, if there is a future useful use of it (i.e., a use in the rhs of an assignment to a live variable or in a guard) with no definition before.
- Given the live variables after a run of a statement, the live variable analysis aims to determine which variables may be live before.

A natural semantics for live variables

- Introduce a natural semantics and a type system for live variables.
- States are assignments of values from $\{\mathrm{dd}, \mathrm{ll}\}$, $\mathrm{dd} \sqsubseteq \mathrm{ll}$, to variables, understood as "liveness states".
- We define $\delta \sqsubseteq \delta'$ to mean that $\delta(x) \sqsubseteq \delta'(x)$ for any x.
- Evaluations of a statement are pre-poststate pairs given by the rules

$$\frac{\delta(x) = \mathrm{ll}}{\delta[x \mapsto \mathrm{dd}][\mathrm{FV}(a) \mapsto \mathrm{ll}] \succ x := a \rightarrow \delta} :=^{1}{}_{\mathrm{lvns}} \qquad \frac{\delta(x) = \mathrm{dd}}{\delta \succ x := a \rightarrow \delta} :=^{2}{}_{\mathrm{lvns}}$$

$$\frac{}{\delta \succ \mathsf{skip} \rightarrow \delta} \ \mathrm{skip}_{\mathrm{lvns}} \qquad \frac{\delta \succ s_0 \rightarrow \delta' \quad \delta' \succ s_1 \rightarrow \delta''}{\delta \succ s_0; s_1 \rightarrow \delta''} \ \mathrm{comp}_{\mathrm{lvns}}$$

$$\frac{\delta \succ s_t \rightarrow \delta'}{\delta[\mathrm{FV}(b) \mapsto \mathrm{ll}] \succ \textsf{if } b \textsf{ then } s_t \textsf{ else } s_f \rightarrow \delta'} \ \mathrm{if}^{\mathrm{tt}}_{\mathrm{lvns}}$$

$$\frac{\delta \succ s_f \rightarrow \delta'}{\delta[\mathrm{FV}(b) \mapsto \mathrm{ll}] \succ \textsf{if } b \textsf{ then } s_t \textsf{ else } s_f \rightarrow \delta'} \ \mathrm{if}^{\mathrm{ff}}_{\mathrm{lvns}}$$

$$\frac{\delta \succ s \rightarrow \delta' \quad \delta' \succ \textsf{while } b \textsf{ do } s_t \rightarrow \delta''}{\delta[\mathrm{FV}(b) \mapsto \mathrm{ll}] \succ \textsf{while } b \textsf{ do } s_t \rightarrow \delta''} \ \mathrm{while}^{\mathrm{tt}}_{\mathrm{lvns}}$$

$$\frac{}{\delta[\mathrm{FV}(b) \mapsto \mathrm{ll}] \succ \textsf{while } b \textsf{ do } s_t \rightarrow \delta} \ \mathrm{while}^{\mathrm{tt}}_{\mathrm{lvns}}$$

- E.g., for $s =_{\mathrm{df}} \textsf{if } w = 3 \textsf{ then } \underbrace{x := y}_{s_t} \textsf{ else } \underbrace{x := z}_{s_f}$, we have

$$[w, x \mapsto \mathrm{dd}, y \mapsto \mathrm{ll}, z \mapsto \mathrm{dd}] \succ s_t \rightarrow [w \mapsto \mathrm{dd}, x \mapsto \mathrm{ll}, y, z \mapsto \mathrm{dd}]$$

$$[w, x \mapsto \mathrm{dd}, y \mapsto \mathrm{dd}, z \mapsto \mathrm{ll}] \succ s_f \rightarrow [w \mapsto \mathrm{dd}, x \mapsto \mathrm{ll}, y, z \mapsto \mathrm{dd}]$$

$$[w \mapsto \mathrm{ll}, x \mapsto \mathrm{dd}, y \mapsto \mathrm{ll}, z \mapsto \mathrm{dd}] \succ s \rightarrow [w \mapsto \mathrm{dd}, x \mapsto \mathrm{ll}, y, z \mapsto \mathrm{dd}]$$

$$[w \mapsto \mathrm{ll}, x \mapsto \mathrm{dd}, y \mapsto \mathrm{dd}, z \mapsto \mathrm{ll}] \succ s \rightarrow [w \mapsto \mathrm{dd}, x \mapsto \mathrm{ll}, y, z \mapsto \mathrm{dd}]$$

Backward collection

- The above semantics is non-deterministic, in the backward direction.
- To get a deterministic backward semantics, we define the collecting version

$$[\![s]\!]_{\lll}(\delta') =_{\mathrm{df}} \bigsqcup \{\delta \mid \delta \succ\!\! s \!\!\rightarrow \delta'\}$$

- The collecting semantics calculates the MOP version of the live variables analysis.

A TYPE SYSTEM FOR LIVE VARIABLES

- Also types are assignments of values from $\{\mathrm{dd}, \mathrm{ll}\}$, $\mathrm{dd} \sqsubseteq \mathrm{ll}$ to variables (used as non upper bounds of liveness states).
- Subtyping is a relation on types given by the rule

$$\frac{d' \sqsubseteq d}{d \leq d'}$$

- Typings of a statement are pre-posttype pairs given by the rules

$$\frac{d(x) = \mathrm{ll}}{x := a : d[x \mapsto \mathrm{dd}][\mathrm{FV}(b) \mapsto \mathrm{ll}] \longrightarrow d} :=^1{}_{\mathrm{lvts}} \quad \frac{d(x) = \mathrm{dd}}{x := a : d \longrightarrow d} :=^2{}_{\mathrm{lvts}}$$

$$\frac{}{\mathsf{skip} : d \longrightarrow d} \mathrm{skip}_{\mathrm{lvts}} \quad \frac{s_0 : d \longrightarrow d' \quad s_1 : d' \longrightarrow d''}{s_0; s_1 : d \longrightarrow d''} \mathrm{comp}_{\mathrm{lvts}}$$

$$\frac{s_t : d \longrightarrow d' \quad s_f : d \longrightarrow d'}{\textsf{if } b \textsf{ then } s_t \textsf{ else } s_f : d[\mathrm{FV}(b) \mapsto \mathrm{ll}] \longrightarrow d'} \ \mathrm{if}_{\mathrm{lvts}}$$

$$\frac{s_t : d \longrightarrow d[\mathrm{FV}(b) \mapsto \mathrm{ll}]}{\textsf{while } b \textsf{ do } s_t : d[\mathrm{FV}(b) \mapsto \mathrm{ll}] \longrightarrow d} \ \mathrm{while}_{\mathrm{lvts}}$$

$$\frac{d \leq d_0 \quad s : d_0 \longrightarrow d_0' \quad d_0' \leq d'}{s : d \longrightarrow d'} \ \mathrm{conseq}_{\mathrm{lvts}}$$

- For the example $s =_{\mathrm{df}} \textsf{if } w = 3 \textsf{ then } x := y \textsf{ else } x := z$, one can get

$$s : [w \mapsto \mathrm{ll}, x \mapsto \mathrm{dd}, y, z \mapsto \mathrm{ll}] \longrightarrow [w \mapsto \mathrm{dd}, x \mapsto \mathrm{ll}, y, z \mapsto \mathrm{dd}]$$

but also

$$s : [w, x, y, z \mapsto \mathrm{ll}] \longrightarrow [w \mapsto \mathrm{dd}, x \mapsto \mathrm{ll}, y, z \mapsto \mathrm{dd}]$$

- Define $\delta \models d$ to mean $\delta \not\sqsubseteq d$.
- Subtyping is (trivially) sound and complete:

  $d \leq d'$ iff

  for any δ, $\delta \models d$ implies $\delta \models d'$

  (i.e., $\delta \sqsubseteq d'$ implies $\delta \sqsubseteq d$).
- Typing is sound and complete:

  $s : d \longrightarrow d'$ iff

  for any δ, δ' such that $\delta \succ s \rightarrow \delta'$, $\delta \models d$ implies $\delta' \models d'$

  (i.e., $\delta' \sqsubseteq d'$ implies $\delta \sqsubseteq d$).
- Completeness of typing holds because the transfer functions of live variables (updates) are distributive.

WEAKEST PRETYPES

- Define a syntactic weakest pretype wpt of a type d' by

$$
\begin{aligned}
\mathrm{wpt}(x := a, d') &=_{\mathrm{df}} \begin{cases} d'[x \mapsto \mathrm{dd}][\mathrm{FV}(a) \mapsto \mathrm{ll}] & \text{if } d'(x) = \mathrm{ll} \\ d' & \text{if } d'(x) = \mathrm{dd} \end{cases} \\
\mathrm{wpt}(\mathsf{skip}, d') &=_{\mathrm{df}} d' \\
\mathrm{wpt}(s_0; s_1, d') &=_{\mathrm{df}} \mathrm{wpt}(s_0, \mathrm{wpt}(s_1, d')) \\
\mathrm{wpt}(\mathsf{if}\ b\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f, d') &=_{\mathrm{df}} (\mathrm{wpt}(s_t, d') \cup \mathrm{wpt}(s_f, d'))[\mathrm{FV}(b) \mapsto \mathrm{ll}] \\
\mathrm{wpt}(\mathsf{while}\ b\ \mathsf{do}\ s_t, d') &=_{\mathrm{df}} (\nu(F) \cup d')[\mathrm{FV}(b) \mapsto \mathrm{ll}] \text{ where} \\
&\qquad F(d) =_{\mathrm{df}} (\mathrm{wpt}(s_t, d) \cup d')[\mathrm{FV}(b) \mapsto \mathrm{ll}]
\end{aligned}
$$

- The wpt of a posttype is its principal pretype:

  $$s : d \longrightarrow d' \text{ iff } d \leq \mathrm{wpt}(s, d').$$

- The wpt function calculates the MFP version of the analysis.

- From soundness and completeness, it follows that

  $$\mathrm{wpt}(s, d') = [\![s]\!]_{\lll}(d').$$

  (where the equality holds because of the distributivity of updates).

- Which is more foundational, the semantics or the type system?
- The type system is a particularly styled indirect description of the semantics, for deriving of semantic properties of a certain flavor, so…
- One can formally reason about a program in the type system or in a universal-logic formalization of the semantics.
- In the former case one must trust the type system as a description of the semantics (or check the soundness proof), in the latter case only the semantics.

A HOARE LOGIC FOR LIVE VARIABLES

- Consider an alternative to the type system: a Hoare logic.
- Assertions are logic formulae over a signature with an extralogical constant $ls(x)$ for any program variable x (for the liveness value of x).
- Derivable triples are given by the rules

$$\frac{}{\{P\}\, x := a \left\{ \begin{array}{c} (ls(x) = \mathrm{ll} \supset P[ls(x) \mapsto \mathrm{dd}][ls(\mathrm{FV}(a)) \mapsto \mathrm{ll}]) \\ \wedge (ls(x) = \mathrm{dd} \supset P) \end{array} \right\}} :=_{\mathrm{lvhoa}}$$

$$\frac{}{\{P\}\, \mathsf{skip}\, \{P\}}\ \mathrm{skip_{lvhoa}} \qquad \frac{\{P\}\, s_0\, \{Q\} \quad \{Q\}\, s_1\, \{R\}}{\{P\}\, s_0; s_1\, \{R\}}\ \mathrm{comp_{lvhoa}}$$

$$\frac{\{P[ls(\mathrm{FV}(b)) \mapsto \mathrm{ll}]\}\, s_t\, \{Q\} \quad \{P[ls(\mathrm{FV}(b)) \mapsto \mathrm{ll}]\}\, s_f\, \{Q\}}{\{P\}\, \mathsf{if}\ b\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f\, \{Q\}}\ \mathrm{if}_{\mathrm{lvhoa}}$$

$$\frac{\{P[ls(\mathrm{FV}(b)) \mapsto \mathrm{ll}]\}\, s_t\, \{P\}}{\{P\}\, \mathsf{while}\ b\ \mathsf{do}\ s_t\, \{P[ls(\mathrm{FV}(b)) \mapsto \mathrm{ll}]\}}\ \mathrm{while}_{\mathrm{lvhoa}}$$

$$\frac{P \models P_0 \quad \{P_0\}\, s\, \{Q_0\} \quad Q_0 \models Q}{\{P\}\, s\, \{Q\}}\ \mathrm{conseq}_{\mathrm{lvhoa}}$$

- The logic is sound and complete wrt. the semantics: $\{P\}\, s\, \{Q\}$ iff, for any δ, δ' and α, $\delta \models_\alpha P$ and $\delta \succ s \rightarrow \delta'$ imply $\delta' \models_\alpha Q$.

- A type d can be translated as the assertion $ls \not\sqsubseteq d$.
- Subtypings are preserved:

  If $d \leq d'$, then $ls \not\sqsubseteq d \models ls \not\sqsubseteq d'$.
- ...and so are typings:

  If $s : d \longrightarrow d'$, then $\{ls \not\sqsubseteq d\}\, s\, \{ls \not\sqsubseteq d'\}$.

- Like the type system, the logic derives properties of the semantics, but of a considerably more liberal form.
- For types, the logic is at least as powerful deductively as the type system.
- But in fact the weakest precondition of a posttype can be better than the weakest pretype: e.g., for $s = \textsf{if } w = 3 \textsf{ then } x := y \textsf{ else } x := z$,

$$\begin{aligned}
&\mathrm{wpt}(s, [w \mapsto \mathrm{dd}, x \mapsto \mathrm{ll}, y, z \mapsto \mathrm{dd}]) \\
&\quad = \quad [w \mapsto \mathrm{ll}, x \mapsto \mathrm{dd}, y, z \mapsto \mathrm{ll}] \\
&\mathrm{wpc}(s, \neg(\mathit{ls}(w) = \mathrm{dd} \wedge \mathit{ls}(y) = \mathrm{dd} \wedge \mathit{ls}(z) = \mathrm{dd})) \\
&\quad = \quad \neg(\mathit{ls}(w) = \mathrm{ll} \wedge \mathit{ls}(x) = \mathrm{dd} \\
&\qquad\qquad \wedge((\mathit{ls}(y) = \mathrm{ll} \wedge \mathit{ls}(z) = \mathrm{dd}) \vee (\mathit{ls}(z) = \mathrm{ll} \wedge \mathit{ls}(y) = \mathrm{dd})))
\end{aligned}$$

- Via the translation, the type system is an applied version of the logic, which describes the semantics more directly.

LIVENESS STATES ARE AN ABSTRACTION

- The natural semantics for live variables is in terms of liveness states. The evaluation rules describe some intuitions about the effect of different statement constructions on liveness.
- In reality liveness states are an abstraction of computation paths.
- A more foundational semantics should be based on a more concrete notion of a state.

A NATURAL SEMANTICS FOR DEF-USE FUTURES

- States are lists of tokens D_x, U_x^y, where x is a variable and y is a variable or a special pseudovariable pc, understood as future def-use traces.
- Evaluations are pre-poststate pairs given by the rules

$$\frac{}{\mathrm{U}^x_{\mathrm{FV}(a)} \cdot \mathrm{D}_x \cdot \tau \succ x := a \rightarrow \tau} \;{:=}^1{}_{\mathrm{lvns}}$$

$$\frac{}{\tau \succ \mathsf{skip} \rightarrow \tau}\ \mathrm{skip}_{\mathrm{lvns}} \qquad \frac{\tau \succ s_0 \rightarrow \tau' \quad \tau' \succ s_1 \rightarrow \tau''}{\tau \succ s_0; s_1 \rightarrow \tau''}\ \mathrm{comp}_{\mathrm{lvns}}$$

$$\frac{\tau \succ s_t \rightarrow \tau'}{\mathrm{U}^{pc}_{\mathrm{FV}(b)} \cdot \tau \succ \mathsf{if}\ b\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f \rightarrow \tau'}\ \mathrm{if}^{\mathrm{tt}}_{\mathrm{lvns}} \qquad \frac{\tau \succ s_f \rightarrow \tau'}{\mathrm{U}^{pc}_{\mathrm{FV}(b)} \cdot \tau \succ \mathsf{if}\ b\ \mathsf{then}\ s_t\ \mathsf{else}\ s_f \rightarrow \tau'}\ \mathrm{if}^{\mathrm{ff}}_{\mathrm{lvns}}$$

$$\frac{\tau \succ s \rightarrow \tau' \quad \tau' \succ \mathsf{while}\ b\ \mathsf{do}\ s_t \rightarrow \tau''}{\mathrm{U}^{pc}_{\mathrm{FV}(b)} \cdot \tau \succ \mathsf{while}\ b\ \mathsf{do}\ s_t \rightarrow \tau''}\ \mathrm{while}^{\mathrm{tt}}_{\mathrm{lvns}} \qquad \frac{}{\mathrm{U}^{pc}_{\mathrm{FV}(b)} \cdot \tau \succ \mathsf{while}\ b\ \mathsf{do}\ s_t \rightarrow \tau}\ \mathrm{while}^{\mathrm{tt}}_{\mathrm{lvns}}$$

- Define $\mathsf{LS}(\tau)(z)$ to mean the liveness of z on a future def-use trace τ:

$$
\begin{aligned}
\mathsf{LS}(\varepsilon)(z) &=_{\text{df}} \text{dd} \\
\mathsf{LS}(\text{D}_x \cdot \tau)(z) &=_{\text{df}} \begin{cases} \text{dd} & \text{if } z = x \\ \mathsf{LS}(\tau)(z) & \text{otherwise} \end{cases} \\
\mathsf{LS}(\text{U}_x^y \cdot \tau)(z) &=_{\text{df}} \begin{cases} \text{ll} & \text{if } z = x \wedge (\mathsf{LS}(\tau)(y) = \text{ll} \vee y = pc) \\ \mathsf{LS}(\tau)(z) & \text{otherwise} \end{cases}
\end{aligned}
$$

- The reinterpretation agrees with the natural semantics for live variables:

  If $\tau \succ\!\!s\!\!\rightarrow \tau'$, then

  $\mathsf{LS}(\tau) \succ\!\!s\!\!\rightarrow \mathsf{LS}(\tau')$.

  If $\delta \succ\!\!s\!\!\rightarrow \mathsf{LS}(\tau')$, then there is a trace

  τ such that $\tau \succ\!\!s\!\!\rightarrow \tau'$ and $\mathsf{LS}(\tau) = \delta$.

A HOARE LOGIC FOR DEF-USE FUTURES

- Assertions are logic formulae over a signature with an extralogical constant tr for the current def-use future.

- Derivable triples of a statement are pre-postcondition pairs given by the rules

$$\frac{}{\{P\}\, x := a\, \{P[tr \mapsto \mathrm{U}^x_{\mathrm{FV}(a)} \cdot \mathrm{D}_x \cdot tr]\}} :=_{\mathrm{lvhoa}}$$

$$\frac{}{\{P\}\, \mathsf{skip}\, \{P\}}\ \mathrm{skip}_{\mathrm{lvhoa}} \qquad \frac{\{P\}\, s_0\, \{Q\} \quad \{Q\}\, s_1\, \{R\}}{\{P\}\, s_0; s_1\, \{R\}}\ \mathrm{comp}_{\mathrm{lvhoa}}$$

$$\frac{\{P[tr \mapsto \mathrm{U}^{pc}_{\mathrm{FV}(b)} \cdot tr]\}\, s_t\, \{Q\} \quad \{P[tr \mapsto \mathrm{U}^{pc}_{\mathrm{FV}(b)} \cdot tr]\}\, s_f\, \{Q\}}{\{P\}\ \textsf{if}\ b\ \textsf{then}\ s_t\ \textsf{else}\ s_f\ \{Q\}}\ \mathrm{if}_{\mathrm{lvhoa}}$$

$$\frac{\{P[tr \mapsto \mathrm{U}^{pc}_{\mathrm{FV}(b)} \cdot tr]\}\, s_t\, \{P\}}{\{P\}\ \textsf{while}\ b\ \textsf{do}\ s_t\ \{P[tr \mapsto \mathrm{U}^{pc}_{\mathrm{FV}(b)} \cdot tr]\}}\ \mathrm{while}_{\mathrm{lvhoa}}$$

$$\frac{P \models P_0 \quad \{P_0\}\, s\, \{Q_0\} \quad Q_0 \models Q}{\{P\}\, s\, \{Q\}}\ \mathrm{conseq}_{\mathrm{lvhoa}}$$

- Again, the logic is sound and complete wrt. the semantics: $\{P\}\, s\, \{Q\}$ iff, for any τ, τ' and α, $\tau \models_\alpha P$ and $\tau \succ s \rightarrow \tau'$ imply $\tau' \models_\alpha Q$.

- Define LS to be a syntactic version of LS.
- An assertion P about a liveness state can be translated as the assertion $P[ls \mapsto LS(tr)]$.
- This translation from the Hoare logic of liveness states to the Hoare logic of future def-use traces preserves derivable triples:
  If $\{P\}\, s\, \{Q\}$, then $\{P[ls \mapsto LS(tr)]\}\, s\, \{Q[ls \mapsto LS(tr)]\}$.

APPLICATIONS

- With a transformation component added to the type system for live variables, one can specify dead code elimination.

  A statement is equivalent to its optimized form in a sense determined by the typing.

- A combination of the Hoare logic of live variables with the standard Hoare logic specifies the data-sensitive version of the live variables analysis.

CONCLUSIONS

- Certificates of data-flow analyses are possible on a variety of levels, given by different levels of abstraction of the semantic entities and specificity of the assertion language.
- In modular foundational certification, an initial proof of an interesting property is constructed at a suitable level of abstraction and specificity.

  This proof is then either translated to a proof in a more concrete and universal formalism or supplemented with a once-and-for-all meta-proof that translation of properties preserves their proofs.