Principles of Abstract Interpretation
MIT press

Ch. 45, Flow-Insensitive Static Analysis

Patrick Cousot

pcousot.github.io

PrAbsInt@gmail.com   github.com/PrAbsInt/

These slides are available at
http://github.com/PrAbsInt/slides/slides-45--flow-insensitive-analysis-PrAbsInt.pdf

Chapter 45

Ch. 45, Flow-Insensitive Static Analysis

Flow sensitive analysis

Flow sensitive analysis

- Classical flow sensitive analysis attaches information to each program point
- This may not scale up!
- (We have shown that for the structural approach information need only be attached to each loop head)

Flow **in**sensitive analysis

Flow insensitive analysis

- From one extreme to another, flow insensitive static analysis computes to information valid at each program point;
- This is a considerable loss of information that does not even guarantee faster fixpoint computations (the fixpoint iterations may slowly climb much higher);
- So why teach stupid solutions?

- It sometimes works (e.g. types in C)
- To simplify their task, the engineers (and researchers) have a tendency to reuse existing solutions (for parts) of their problems (not worrying on the consequences on the other parts!);
- Example:
  - First built a program graph (so as to immediately loose information on the program semantics, worse using an intermediate language!)
  - If too costly so be flow insensitive (no hope for any kind of precision!)
  - Use only (small) finite domains (even boolean domain may not scale!)
  - etc
- We will show that these flow insensitive analyses are trivial abstractions of the flow sensitive ones, for the same abstract domain;
- So the decision to be flow sensitive/insensitive should not be taken early in the static analysis design;
- And the cost/precision balance can be easily refined by extra abstractions/interpolators

The flow sensitive abstract interpreter

- The generic structural abstract interpreter of section 21.2

$$\widehat{\mathcal{S}}^{\pi}[\![\mathsf{S}]\!] \in \mathbb{P}^{\pi} \longrightarrow (\mathsf{labx}[\![\mathsf{S}]\!] \longrightarrow \mathbb{P}^{\pi})$$

  is parameterized by an abstract domain definition 21.1

$$\mathbb{D}^{\pi} \triangleq \langle \mathbb{P}^{\pi}, \sqsubseteq^{\pi}, \bot^{\pi}, \sqcup^{\pi}, \mathsf{assign}_{\pi}[\![\mathsf{x}, \mathsf{A}]\!], \mathsf{test}^{\pi}[\![\mathsf{B}]\!], \overline{\mathsf{test}}^{\pi}[\![\mathsf{B}]\!]\rangle$$

The flow insensitive abstraction

- The flow insensitive abstraction

$$\langle \mathsf{labx}[\![\mathsf{S}]\!] \longrightarrow \mathbb{P}^{\text{¤}},\ \dot{\sqsubseteq}^{\text{¤}} \rangle \xleftrightarrows[\alpha_{\mathfrak{g}}]{\gamma_{\mathfrak{g}}} \langle \mathbb{P}^{\text{¤}},\ \sqsubseteq^{\text{¤}} \rangle$$

joins all local program properties attached to program points into a single global property holding at any program point.

$$\begin{array}{rcl} \alpha_{\mathfrak{g}} & \in & (\mathsf{labx}[\![\mathsf{S}]\!] \longrightarrow \mathbb{P}^{\text{¤}}) \longrightarrow \mathbb{P}^{\text{¤}} \\ \alpha_{\mathfrak{g}}(P) & \triangleq & \displaystyle\bigsqcup_{\ell \in \mathsf{labx}[\![\mathsf{S}]\!]}^{\text{¤}} P(\ell) \\ \dot{\alpha}_{\mathfrak{g}} & \in & (\mathbb{P}^{\text{¤}} \longrightarrow (\mathsf{labx}[\![\mathsf{S}]\!] \longrightarrow \mathbb{P}^{\text{¤}})) \longrightarrow (\mathbb{P}^{\text{¤}} \longrightarrow \mathbb{P}^{\text{¤}}) \\ \dot{\alpha}_{\mathfrak{g}}(\boldsymbol{\mathcal{S}}) & \triangleq & P \mapsto \alpha_{\mathfrak{g}}(\boldsymbol{\mathcal{S}}(P)) \end{array} \tag{45.1}$$

Calculational design of the flow insensitive abstract interpreter

- By an (easy) calculational design we get a generic structural flow insensitive abstract interpreter

$$\widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathsf{S}]\!] \in \mathbb{P}^{¤} \longrightarrow \mathbb{P}^{¤}$$

- No change is needed in the abstract domain $\mathbb{D}^{¤}$

The flow insensitive abstract interpreter

Flow-insensitive abstract semantics of a program $\mathtt{P} ::= \mathtt{Sl}\ \ell$

$$\widehat{\mathcal{S}}^{\natural}_{\mathrm{g}}[\![\mathtt{P}]\!]\,\overline{P} \quad \triangleq \quad \widehat{\mathcal{S}}^{\natural}_{\mathrm{g}}[\![\mathtt{Sl}]\!]\,\overline{P} \tag{45.2}$$

Flow-insensitive abstract semantics of a statement list $\mathtt{Sl} ::= \mathtt{Sl}'\ \mathtt{S}$

$$\widehat{\mathcal{S}}^{\natural}_{\mathrm{g}}[\![\mathtt{Sl}]\!]\,\overline{P} \quad = \quad \widehat{\mathcal{S}}^{\natural}_{\mathrm{g}}[\![\mathtt{Sl}']\!]\,\overline{P} \sqcup^{\natural} \widehat{\mathcal{S}}^{\natural}_{\mathrm{g}}[\![\mathtt{S}]\!]\,\overline{P} \tag{45.3}$$

Flow-insensitive abstract semantics of an empty statement list $\mathtt{Sl} ::= \epsilon$

$$\widehat{\mathcal{S}}^{\natural}_{\mathrm{g}}[\![\mathtt{Sl}]\!]\,\overline{P} \quad = \quad \overline{P} \tag{45.4}$$

Flow-insensitive abstract semantics of an assignment statement $\mathtt{S} ::= \mathtt{x} = \mathtt{A}\ ;$

$$\widehat{\mathcal{S}}^{\natural}_{\mathrm{g}}[\![\mathtt{S}]\!]\,\overline{P} \quad = \quad \overline{P} \sqcup^{\natural} \mathsf{assign}_{\natural}[\![\mathtt{x}, \mathtt{A}]\!]\,\overline{P} \tag{45.5}$$

Flow-insensitive abstract semantics of a skip statement S ::= ;

$$\widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathsf{S}]\!]\,\overline{P} \quad = \quad \overline{P} \tag{45.6}$$

Flow-insensitive abstract semantics of a conditional statement S ::= **if** (B) S_t

$$\widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathsf{S}]\!]\,\overline{P} \quad = \quad \overline{P} \sqcup^{¤} \widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathsf{S}_t]\!]\,(\mathsf{test}^{¤}[\![\mathsf{B}]\!]\,\overline{P}) \tag{45.7}$$

Flow-insensitive abstract semantics of a conditional statement S ::= **if** (B) S_t **else** S_f

$$\widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathsf{S}]\!]\,\overline{P} \quad = \quad \overline{P} \sqcup^{¤} \widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathsf{S}_t]\!]\,(\mathsf{test}^{¤}[\![\mathsf{B}]\!]\,\overline{P}) \sqcup^{¤} \widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathsf{S}_f]\!]\,\overline{\mathsf{test}}^{¤}[\![\mathsf{B}]\!]\,\overline{P}) \tag{45.8}$$

Immediate consequences of the calculational design

Flow-insensitive abstract semantics of an iteration statement $\mathtt{S} ::= \mathtt{while}\ \ell\ (\mathtt{B})\ \mathtt{S}_b$

$$\widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathtt{S}]\!]\ \overline{P} \quad = \quad \mathsf{lfp}^{\dot{\subseteq}}\,(\mathscr{F}^{¤}_{\mathfrak{g}}[\![\mathtt{while}\ \ell\ (\mathtt{B})\ \mathtt{S}_b]\!]\ \overline{P}) \tag{45.9}$$

$$\mathscr{F}^{¤}_{\mathfrak{g}}[\![\mathtt{while}\ \ell\ (\mathtt{B})\ \mathtt{S}_b]\!]\ \overline{P}\ X \quad = \quad \overline{P} \sqcup^{¤} \widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathtt{S}]\!]\ (\mathsf{test}^{¤}[\![\mathtt{B}]\!]\ X)$$

Flow-insensitive abstract semantics of a break statement $\mathtt{S} ::= \ell\ \mathtt{break}\ ;$

$$\widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathtt{S}]\!]\ \overline{P} \quad = \quad \overline{P} \tag{45.10}$$

Flow-insensitive abstract semantics of a compound statement $\mathtt{S} ::= \{\ \mathtt{Sl}\ \}$

$$\widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathtt{S}]\!]\ \overline{P} \quad = \quad \widehat{\mathcal{S}}^{¤}_{\mathfrak{g}}[\![\mathtt{Sl}]\!]\ \overline{P} \tag{45.11}$$

Well-definedness and completeness I

Theorem (45.14) The flow insensitive abstract semantics $\widehat{\boldsymbol{\mathcal{S}}}^{¤}_{\mathrm{g}}[\![\mathsf{S}]\!]$ on a well-defined abstract domain of definition 21.1 is well-defined.

Theorem (45.13) The flow insensitive abstract semantics $\widehat{\boldsymbol{\mathcal{S}}}^{¤}_{\mathrm{g}}$ of section 45.3 on a domain $\mathbb{D}^{¤}$ is a sound abstraction of the flow sensitive abstract semantics $\widehat{\boldsymbol{\mathcal{S}}}^{¤}$ of section 21.2 on the same domain $\mathbb{D}^{¤}$, $\dot{\alpha}_{\mathrm{g}}(\widehat{\boldsymbol{\mathcal{S}}}^{¤}[\![\mathsf{S}]\!]) \sqsubseteq^{¤} \widehat{\boldsymbol{\mathcal{S}}}^{¤}_{\mathrm{g}}[\![\mathsf{S}]\!]$.

Well-definedness and completeness II

Theorem (45.15, Soundness of the flow insensitive abstract interpreter) Let $\widehat{\boldsymbol{\mathcal{S}}}_{\mathfrak{g}}^{¤}$ and $\widehat{\boldsymbol{\mathcal{S}}}_{\mathfrak{g}}^{\sharp}$ be structural flow insensitive abstract interpreters for well-defined concrete $\mathbb{D}^{¤}$ and abstract domains $\mathbb{D}^{\sharp}$ by definition 21.1 such that $\mathbb{P}^{\sharp}$ is an approximate abstraction of $\mathbb{P}^{¤}$ by definition 27.1-I. Then for all $\overline{P} \in \mathbb{P}^{\sharp}$,

$$\boldsymbol{\mathcal{S}}_{\mathfrak{g}}^{¤}[\![\mathsf{S}]\!](\gamma(\overline{P})) \ \dot{\sqsubseteq}^{¤} \ \dot{\gamma}(\boldsymbol{\mathcal{S}}_{\mathfrak{g}}^{\sharp}[\![\mathsf{S}]\!](\overline{P}))$$

Conclusion

- The abstraction of a sensitive to an insensitive static analysis relies on an abstraction joining cases.
  - *Path insensitivity* join path properties into flow sensitive state properties attached to program points as illustrated in the reachability semantics of chapter 19.
  - *Flow sensitive* static analyses can be abstracted into flow insensitive analyses by the abstraction (45.1) joining local properties attached to program points into a single global property.
  - A *field sensitive* static analysis can be abstracted into a field insensitive static analysis by joining the properties attached to field of a data structure into a single structure property.
  - A *context sensitive* static analysis can be abstracted into a context insensitive static analysis by joining the preconditions of procedure calls into a single precondition.

Conclusion

- Flow insensitive static analysis is not always, if ever, a good solution.
- Flow insensitive static analyses are a drastic abstraction.
- Starting precise and getting imprecise (e.g. with widening) is more flexible and adaptable in the light of experimentations.

Home work

Read Ch. 45 “Flow-Insensitive Static Analysis” of

Principles of Abstract Interpretation
Patrick Cousot
MIT Press

The End, Thank you