SAS 2019

Thursday, October 10th 2019

Symposium on Formal Methods, FM'19, Porto, Portugal

Abstract Semantic Dependency

Patrick Cousot

New York University, Courant Institute of Mathematics, Computer Science

pcousot@cs.nyu.edu   cs.nyu.edu/~pcousot

Objective

Objective

- Design a dependency analysis by abstract interpretation of a trace semantics.
- a depends on b iff changing b into a different b' will change a into a different a'
- This involves 2 execution traces $a \rightarrow b$ and $a' \rightarrow b'$ (*i.e.* it is not a trace abstraction)

Objective

- Design a dependency analysis by abstract interpretation of a trace semantics.
- a depends on b iff changing b into a different b' will change a into a different a'
- This involves 2 execution traces $a \to b$ and $a' \to b'$ (*i.e.* it is not a trace abstraction)
- Recent work (Mounir Assaf, David A. Naumann, Julien Signoles, Éric Totel, and Frédéric Tronel and Caterina Urban and Peter Müller) suggests abstract interpretation theory must be revisited

Objective

- Design a dependency analysis by abstract interpretation of a trace semantics.
- a depends on b iff changing b into a different b' will change a into a different a'
- This involves 2 execution traces $a \to b$ and $a' \to b'$ (*i.e.* it is not a trace abstraction)
- Recent work (Mounir Assaf, David A. Naumann, Julien Signoles, Éric Totel, and Frédéric Tronel and Caterina Urban and Peter Müller) suggests abstract interpretation theory must be revisited
- or not?

Syntax and trace semantics

Syntax and trace semantics

- The syntax is a subset of `C` (`while` programs)
- The semantics is a structural prefix (or maximal) trace semantics $\langle \pi\ell, \ell\pi' \rangle \in \mathcal{S}^*[\![\mathtt{S}]\!]$ (where $\ell = \mathsf{at}[\![\mathtt{S}]\!]$) means that an execution reaching the entry point ℓ of program component $\mathtt{S}$ may continue as stated by $\ell\pi'$.
- Example: Assignment $\mathtt{S} ::= {}^{\ell}\ \mathtt{x = A ;}$ (where $\mathsf{at}[\![\mathtt{S}]\!] = \ell$)

$$\mathcal{S}^*[\![\mathtt{S}]\!] \triangleq \{\langle \pi\ell, \ell \rangle, \langle \pi\ell, \ell \xrightarrow{\mathtt{x = A} = v} \mathsf{after}[\![\mathtt{S}]\!] \rangle \mid \pi\ell \in \mathbb{T}^+ \wedge v = \mathcal{A}[\![\mathtt{A}]\!]\varrho(\pi\ell)\} \qquad (0)$$

$$\mathcal{S}^+[\![\mathtt{S}]\!] \triangleq \{\langle \pi\ell, \ell \xrightarrow{\mathtt{x = A} = v} \mathsf{after}[\![\mathtt{S}]\!] \rangle \mid \pi\ell \in \mathbb{T}^+ \wedge v = \mathcal{A}[\![\mathtt{A}]\!]\varrho(\pi\ell)\}$$

$$\mathcal{S}^\infty[\![\mathtt{S}]\!] \triangleq \varnothing \qquad \text{no infinite trace}$$

Informal Requirements for a Semantic Definition of Dependency

Informal Requirements for a Semantic Definition of Dependency

- For simplicity, we consider dependency upon initial states
- The dependency of variables on initial states is local, at each program point (not *global* as in [D. E. Denning and P. J. Denning, 1977] or on *program exit* as in [Assaf, Naumann, Signoles, Totel, and Tronel, 2017; Urban and Müller, 2018])
- We don't want to make a difference between control and data dependency (as in [D. E. Denning and P. J. Denning, 1977] and their followers)
- We ignore timing channels (as usual in compilation)
- We ignore empty observations (observing nothing at a program point is not an observation)

Formal Semantic Definition of Dependency

Sequence of values of a variable at a program point

- $\mathsf{seqval}[\![\mathsf{y}]\!]\ell(\pi_0,\ \pi)$ is the sequence of values of the variable y at program point ℓ along the trace π continuing π_0

$$
\begin{aligned}
\mathsf{seqval}[\![\mathsf{y}]\!]\ell(\pi_0, \ell) &\triangleq \varrho(\pi_0)\mathsf{y} && && (1)\\
\mathsf{seqval}[\![\mathsf{y}]\!]\ell(\pi_0, \ell') &\triangleq \ni && \text{when } \ell' \neq \ell\\
\mathsf{seqval}[\![\mathsf{y}]\!]\ell(\pi_0, \ell \xrightarrow{a} \ell''\pi) &\triangleq \varrho(\pi_0)\mathsf{y} \cdot \mathsf{seqval}[\![\mathsf{y}]\!]\ell(\pi_0 \smallfrown \ell \xrightarrow{a} \ell'', \ell''\pi)\\
\mathsf{seqval}[\![\mathsf{y}]\!]\ell(\pi_0, \ell' \xrightarrow{a} \ell''\pi) &\triangleq \mathsf{seqval}[\![\mathsf{y}]\!]\ell(\pi_0 \smallfrown \ell' \xrightarrow{a} \ell'', \ell''\pi) && \text{when } \ell' \neq \ell
\end{aligned}
$$

- (bi-induction: induction for finite traces, co-induction for infinite ones)

Differences between sequences of values of a variable at a program point

- $\mathsf{diff}(\omega, \omega')$ holds if and only if the sequences of value observations ω and ω' at some program point differ by at least one value

$$\mathsf{diff}(\omega, \omega') \triangleq \exists \omega_0, \omega_1, \omega'_1, \nu, \nu' \,.\, \omega = \omega_0 \cdot \nu \cdot \omega_1 \wedge \omega' = \omega_0 \cdot \nu' \cdot \omega'_1 \wedge \nu \neq \nu' \qquad (2)$$

- $\neg\mathsf{diff}(\omega, \omega')$ implies
  - either that $\omega = \omega'$ (no dependency for same futures)
  - or one is a strict prefix of the other (timing channels are abstracted away).

- Change this definition to get alternative concepts of dependency (*e.g*. timing channels, empty observation, *etc*.)

Definition of value dependency

- $\Pi \in \wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty})$ is a trace semantics
- Properties are represented by sets (of individuals with this property)
- $\Pi \in \mathcal{D}\ell\langle x, y\rangle$ means that y at ℓ depends on the initial value of x

Definition 1 (Dependency $\mathcal{D}$)

$$\begin{aligned}\mathcal{D}\ell\langle x, y\rangle \triangleq \{\Pi \in \wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty}) \mid \exists\langle\pi_0, \pi_1\rangle, \langle\pi'_0, \pi'_1\rangle \in \Pi\ .\\ (\forall z \in \mathbb{V} \setminus \{x\}\ .\ \varrho(\pi_0)z = \varrho(\pi'_0)z) \wedge\\ \mathsf{diff}(\mathsf{seqval}[\![y]\!]\ell(\pi_0, \pi_1), \mathsf{seqval}[\![y]\!]\ell(\pi'_0, \pi'_1))\}\end{aligned} \tag{3}$$

$\square$

Value dependency flow

- $x \rightsquigarrow_{P}^{\ell} y$ iff, at program point ℓ of program P, variable y depends on the initial value of variable x (or the initial value of variable x flows to variable y at program point ℓ)

Definition 2 (Value dependency flow)

$$x \rightsquigarrow_{P}^{\ell} y \triangleq (\mathcal{S}^{+\infty}[\![P]\!] \in \mathcal{D}\ell\langle x, y\rangle). \quad (4)$$

$\square$

- The use of the prefix trace semantics $\mathcal{S}^{*}[\![P]\!]$ is equivalent to that of the maximal trace semantics $\mathcal{S}^{+\infty}[\![P]\!]$

Lemma 1 (Value dependency for finite prefix traces)

$$x \rightsquigarrow_{P}^{\ell} y = (\mathcal{S}^{*}[\![P]\!] \in \mathcal{D}\ell\langle x, y\rangle).$$

$\square$

Value dependency abstraction

- $\alpha^{\natural}(\mathcal{S})$ is the value dependency abstraction of a semantic property $\mathcal{S} \in \wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty}))$ is

Definition (Value dependency abstraction $\alpha^{\natural}$)

$$\alpha^{\natural}(\mathcal{S})\ell \triangleq \{\langle \mathsf{x},\ \mathsf{y}\rangle \mid \mathcal{S} \subseteq \mathcal{D}\ell\langle \mathsf{x},\ \mathsf{y}\rangle\} \tag{5}$$

- This a Galois connection $\langle \wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty})),\ \subseteq\rangle \xrightleftharpoons[\alpha^{\natural}]{\gamma^{\natural}} \langle \mathbb{P}^{\natural},\ \supseteq^{\natural}\rangle$ where $\mathbb{P}^{\natural} \triangleq \mathbb{L} \to \wp(\mathbb{V} \times \mathbb{V})$ is ordered pointwise

Corollary 1 (Value dependency for finite prefix traces)

$$\ell \mapsto \{\langle \mathsf{x},\ \mathsf{y}\rangle \mid \mathsf{x} \rightsquigarrow^{\ell}_{\mathsf{P}} \mathsf{y}\} = \alpha^{\natural}(\{\mathcal{S}^{+\infty}[\![\mathsf{P}]\!]\}) = \alpha^{\natural}(\{\mathcal{S}^{*}[\![\mathsf{P}]\!]\})$$ □

Exact, definite, and potential value dependency semantics

$$
\left[\begin{array}{rcll}
\overline{\mathcal{S}}^{\,\mathrm{diff}}[\![\mathsf{S}]\!] & \triangleq & \alpha^{\mathrm{d}}(\{\mathcal{S}^{+\infty}[\![\mathsf{S}]\!]\}) \;=\; \alpha^{\mathrm{d}}(\{\mathcal{S}^{*}[\![\mathsf{S}]\!]\}) & \text{exact dependency} \\
\widehat{\overline{\mathcal{S}}}{}^{\forall}_{\mathrm{diff}}[\![\mathsf{S}]\!] & \dot{\subseteq} & \alpha^{\mathrm{d}}(\{\mathcal{S}^{+\infty}[\![\mathsf{S}]\!]\}) & \text{definite dependency} \\
\alpha^{\mathrm{d}}(\{\mathcal{S}^{+\infty}[\![\mathsf{S}]\!]\}) & \dot{\subseteq} & \widehat{\overline{\mathcal{S}}}{}^{\exists}_{\mathrm{diff}}[\![\mathsf{S}]\!] & \text{potential dependency}
\end{array}\right. \tag{6}
$$

Calculational design of the structural potential dependency analysis

Calculational design

- Based on the soundness definition

$$\alpha^{\mathrm{d}}(\{\mathcal{S}^*[\![\mathsf{s}]\!]\}) \mathrel{\dot{\subseteq}} \widehat{\mathcal{S}}^{\exists}_{\mathsf{diff}}[\![\mathsf{s}]\!]$$

- The finite abstract domain is $\mathbb{L} \rightarrow \wp(\mathbb{V} \times \mathbb{V})$ ordered pointwise
- Method
  - by structural induction on program components s
  - develop $\alpha^{\mathrm{d}}(\{\mathcal{S}^*[\![\mathsf{s}]\!]\})$ to eliminate the abstraction α^{d}
  - over-approximate to eliminate all concrete computations (*e.g.*value of a test with dead branch)
- A bit more complicated than for DFA since for each program component s, we have to consider any two execution traces of s (only one for DFA)

Structural static potential value dependency analysis

- assignment S ::= x = A ;

$$\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\mathtt{S}]\!]\,\ell \triangleq \big(\!\!\big|\ \ell = \mathrm{at}[\![\mathtt{S}]\!] \ \text{?}\ \mathbb{1}_{\mathbb{V}} \\ \big\|\ \ell = \mathrm{after}[\![\mathtt{S}]\!] \ \text{?}\ \{\langle \mathtt{y},\ \mathtt{x}\rangle \mid \mathtt{y} \in \widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\mathtt{A}]\!]\} \cup \{\langle \mathtt{y},\ \mathtt{y}\rangle \mid \mathtt{y} \neq \mathtt{x}\} \\ \text{:}\ \varnothing\ \big|\!\!\big) \tag{10}$$

$$\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\mathtt{A}]\!] \triangleq \{\mathtt{y} \mid \exists \rho \in \mathbb{E}\mathrm{v}\ .\ \exists \nu \in \mathbb{V}\ .\ \mathcal{A}[\![\mathtt{A}]\!]\rho \neq \mathcal{A}[\![\mathtt{A}]\!]\rho[\mathtt{y} \leftarrow \nu]\} \quad \subseteq \quad \mathbb{vars}[\![\mathtt{A}]\!]$$

Proof of (10) We consider the case $\ell = \mathsf{after}[\![\mathsf{S}]\!]$. (The cases $\ell = \mathsf{at}[\![\mathsf{S}]\!]$ and $\ell \notin \mathsf{labx}[\![\mathsf{S}]\!]$ are simpler.)

$$
\begin{array}{l}
\quad \alpha^{\mathrm{d}}(\{\mathcal{S}^{+\infty}[\![\mathsf{S}]\!]\})\,\mathsf{after}[\![\mathsf{S}]\!] \\
= \alpha^{\mathrm{d}}(\{\mathcal{S}^{*}[\![\mathsf{S}]\!]\})\,\mathsf{after}[\![\mathsf{S}]\!] \hfill \wr\text{Lemma 1}\wr \\
= \{\langle \mathsf{x}',\ \mathsf{y}\rangle \mid \mathcal{S}^{*}[\![\mathsf{S}]\!] \in \mathcal{D}(\mathsf{after}[\![\mathsf{S}]\!])\langle \mathsf{x}',\ \mathsf{y}\rangle\} \hfill \wr\text{def. (5) of } \alpha^{\mathrm{d}} \text{ and def. } \subseteq\wr \\
= \{\langle \mathsf{x}',\ \mathsf{y}\rangle \mid \exists\langle \pi_0,\ \pi_1\rangle, \langle \pi'_0,\ \pi'_1\rangle \in \mathcal{S}^{*}[\![\mathsf{S}]\!] \,.\, \forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \boldsymbol{\varrho}(\pi_0)\mathsf{z} = \boldsymbol{\varrho}(\pi'_0)\mathsf{z} \wedge \\
\quad \mathsf{diff}(\mathsf{seqval}[\![\mathsf{y}]\!](\mathsf{after}[\![\mathsf{S}]\!])(\pi_0, \pi_1), \mathsf{seqval}[\![\mathsf{y}]\!](\mathsf{after}[\![\mathsf{S}]\!])(\pi'_0, \pi'_1))\} \hfill \wr\text{def. } \in \text{ and (3) of } \mathcal{D}\ell\langle \mathsf{x}',\ \mathsf{y}\rangle\wr \\
= \{\langle \mathsf{x}',\ \mathsf{y}\rangle \mid \exists\langle \pi_0,\ \pi_1\rangle, \langle \pi'_0,\ \pi'_1\rangle \in \{\langle \pi\mathsf{at}[\![\mathsf{S}]\!],\ \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle \mid \pi\mathsf{at}[\![\mathsf{S}]\!] \in \mathbb{T}^{+}\} \,.\, \forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,. \\
\quad \boldsymbol{\varrho}(\pi_0)\mathsf{z} = \boldsymbol{\varrho}(\pi'_0)\mathsf{z} \wedge \mathsf{diff}(\mathsf{seqval}[\![\mathsf{y}]\!](\mathsf{after}[\![\mathsf{S}]\!])(\pi_0, \pi_1), \mathsf{seqval}[\![\mathsf{y}]\!](\mathsf{after}[\![\mathsf{S}]\!])(\pi'_0, \pi'_1))\} \\
\hfill \wr\text{def. of the assignment prefix finite trace semantics}\wr \\
= \{\langle \mathsf{x}',\ \mathsf{y}\rangle \mid \exists\langle \pi_0\mathsf{at}[\![\mathsf{S}]\!],\ \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle, \langle \pi'_0\mathsf{at}[\![\mathsf{S}]\!],\ \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle \,.\, \forall \mathsf{z} \in \\
\quad \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])\mathsf{z} = \boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])\mathsf{z} \wedge \mathsf{diff}(\mathsf{seqval}[\![\mathsf{y}]\!](\mathsf{after}[\![\mathsf{S}]\!])(\pi_0\mathsf{at}[\![\mathsf{S}]\!], \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]), \\
\quad \mathsf{seqval}[\![\mathsf{y}]\!](\mathsf{after}[\![\mathsf{S}]\!])(\pi'_0\mathsf{at}[\![\mathsf{S}]\!], \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]))\} \hfill \wr\text{def. } \in\wr \\
= \{\langle \mathsf{x}',\ \mathsf{y}\rangle \mid \exists\langle \pi_0\mathsf{at}[\![\mathsf{S}]\!],\ \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle, \langle \pi'_0\mathsf{at}[\![\mathsf{S}]\!],\ \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle \,. \\
\quad (\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])\mathsf{z} = \boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])\mathsf{z}) \wedge \mathsf{diff}(\boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!])\mathsf{y}, \\
\quad \boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!])\mathsf{y})\} \hfill \wr\text{def. (0) of the future } \mathsf{seqval}[\![\mathsf{y}]\!]\wr
\end{array}
$$

$$= \{\langle \mathsf{x}', \mathsf{y}\rangle \mid \exists \langle \pi_0 \mathsf{at}[\![\mathsf{S}]\!], \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathcal{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle, \langle \pi'_0 \mathsf{at}[\![\mathsf{S}]\!], \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathcal{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle \,.\, (\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \boldsymbol{\varrho}(\pi_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{z} = \boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{z}) \wedge ((\boldsymbol{\varrho}(\pi_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{y} \neq \boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{y}) \vee (\boldsymbol{\varrho}(\pi_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{y} = \boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{y} \wedge \boldsymbol{\varrho}(\pi_0 \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathcal{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!])\mathsf{y} \neq \boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathcal{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!])\mathsf{y})\}$$

≀(2) so that $\mathsf{diff}(a \cdot b,\ c \cdot d)$ if and only if (1) $a \neq c$ or (2) $a = c \wedge b \neq d$.∫

$$= \{\langle \mathsf{x}', \mathsf{y}\rangle \mid \exists \langle \pi_0 \mathsf{at}[\![\mathsf{S}]\!], \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathcal{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle, \langle \pi'_0 \mathsf{at}[\![\mathsf{S}]\!], \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathcal{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle \,.\, (\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \boldsymbol{\varrho}(\pi_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{z} = \boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{z}) \wedge ((\mathsf{y} = \mathsf{x}') \vee (\mathsf{y} = \mathsf{x} \wedge \mathcal{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi_0 \mathsf{at}[\![\mathsf{S}]\!]) \neq \mathcal{A}[\![\mathsf{A}]\!]\boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!])))\}$$

≀def. $\boldsymbol{\varrho}$∫

$$\subseteq \{\langle \mathsf{x}', \mathsf{y}\rangle \mid ((\mathsf{y} = \mathsf{x}') \vee (\mathsf{y} = \mathsf{x} \wedge \exists \rho, \nu \,.\, \mathcal{A}[\![\mathsf{A}]\!]\rho \neq \mathcal{A}[\![\mathsf{A}]\!]\rho[\mathsf{x}' \leftarrow \nu]))\} \qquad (1)$$

≀letting $\rho = \boldsymbol{\varrho}(\pi_0 \mathsf{at}[\![\mathsf{S}]\!])$ and $\nu = \boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!])(\mathsf{x}')$ so that $\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \boldsymbol{\varrho}(\pi_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{z} = \boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!])\mathsf{z}$ implies that $\boldsymbol{\varrho}(\pi'_0 \mathsf{at}[\![\mathsf{S}]\!]) = \rho[\mathsf{x}' \leftarrow \nu]$.∫

$$= \{\langle \mathsf{x}', \mathsf{x}'\rangle \mid \mathsf{x}' \neq \mathsf{x}\} \cup \{\langle \mathsf{x}', \mathsf{x}\rangle \mid \exists \rho, \nu \,.\, \mathcal{A}[\![\mathsf{A}]\!]\rho \neq \mathcal{A}[\![\mathsf{A}]\!]\rho[\mathsf{x}' \leftarrow \nu]\}$$

≀case analysis∫

$$= \{\langle \mathsf{x}', \mathsf{x}'\rangle \mid \mathsf{x}' \neq \mathsf{x}\} \cup \{\langle \mathsf{x}', \mathsf{x}\rangle \mid \mathsf{x}' \in \widehat{\boldsymbol{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\mathsf{A}]\!]\}$$

≀by defining the functional dependency of an expression A as $\widehat{\boldsymbol{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\mathsf{A}]\!] \triangleq \{\mathsf{x}' \mid \exists \rho, \nu \,.\, \mathcal{A}[\![\mathsf{A}]\!]\rho \neq \mathcal{A}[\![\mathsf{A}]\!]\rho[\mathsf{x}' \leftarrow \nu]\}$ in (10)∫ $\square$

Determinacy

- if variables in $\mathtt{x} \in \mathsf{det}(\mathtt{B}_1, \mathtt{B}_2)$ have different values then $\mathtt{B}_1$ and $\mathtt{B}_2$ cannot both be true

i.e. if $\mathtt{B}_1$ and $\mathtt{B}_2$ are both true then the values of variables $\mathtt{x} \in \mathsf{det}(\mathtt{B}_1, \mathtt{B}_2)$ are the same

$$\mathsf{det}(\mathtt{B}_1, \mathtt{B}_2) \subseteq \{\mathtt{x} \mid \forall \rho, \rho' \,.\, (\mathscr{B}[\![\mathtt{B}_1]\!]\rho \wedge \mathscr{B}[\![\mathtt{B}_2]\!]\rho') \Rightarrow (\rho(\mathtt{x}) = \rho'(\mathtt{x}))\} \tag{13}$$

e.g. $\mathsf{det}(\mathtt{x=1}, \mathtt{x=1} \wedge \mathtt{y=42}) = \{\mathtt{x}\}$

- The values of variables in $\mathsf{det}(\mathtt{B}, \mathtt{B})$ are determined by the veracity of $\mathtt{B}$

$$\mathsf{det}(\mathtt{B}, \mathtt{B}) \subseteq \{\mathtt{x} \mid \forall \rho, \rho' \,.\, (\mathscr{B}[\![\mathtt{B}]\!]\rho \wedge \mathscr{B}[\![\mathtt{B}]\!]\rho') \Rightarrow (\rho(\mathtt{x}) = \rho'(\mathtt{x}))\}$$

e.g. $\mathsf{det}(\mathtt{x=y} \wedge \mathtt{z=42}, \mathtt{x=y} \wedge \mathtt{z=42}) = \{\mathtt{z}\}$

Non-determinacy:

- variables in $\mathtt{x} \in \mathsf{nondet}(\mathtt{B}_1, \mathtt{B}_2)$ do not change the veracity of $\mathtt{B}_1$ and $\mathtt{B}_2$

$$\begin{aligned}\mathsf{nondet}(\mathtt{B}_1, \mathtt{B}_2) &\supseteq \mathbb{V} \setminus \mathsf{det}(\mathtt{B}_1, \mathtt{B}_2)\\ &\supseteq \{\mathtt{x} \mid \exists \rho, \rho' \,.\, \mathscr{B}[\![\mathtt{B}_1]\!]\rho \wedge \mathscr{B}[\![\mathtt{B}_2]\!]\rho' \wedge \rho(\mathtt{x}) \neq \rho'(\mathtt{x})\}\end{aligned}$$

e.g. $\mathsf{nondet}(\mathtt{x=1}, \mathtt{x=1} \wedge \mathtt{y=42}) = \{\mathtt{y}\}$

- The values of variables in $\mathtt{x} \in \mathsf{nondet}(\mathtt{B}, \mathtt{B})$ are not determined by the veracity of $\mathtt{B}$

$$\mathsf{nondet}(\mathtt{B}, \mathtt{B}) \supseteq \{\mathtt{x} \mid \exists \rho, \rho' \,.\, \mathscr{B}[\![\mathtt{B}]\!]\rho \wedge \mathscr{B}[\![\mathtt{B}]\!]\rho' \wedge \rho(\mathtt{x}) \neq \rho'(\mathtt{x})\}$$

e.g. $\mathsf{det}(\mathtt{x=y} \wedge \mathtt{z=42}, \mathtt{x=y} \wedge \mathtt{z=42}) = \{\mathtt{x}, \mathtt{y}\}$

Structural static potential value dependency analysis (cont'd)

- conditional $\mathtt{S} ::= \mathtt{if}\ (\mathtt{B})\ \mathtt{S}_t$

$$
\begin{array}{llr}
\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\mathtt{S}]\!]\,\ell \triangleq & (\!\![\ \ell = \text{at}[\![\mathtt{S}]\!]\ \text{?}\ \mathbb{1}_{\mathbb{V}} & \text{(a)} \quad (12) \\
& \|\ \ell \in \text{in}[\![\mathtt{S}_t]\!]\ \text{?}\ \widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\mathtt{S}_t]\!]\,\ell\ \rceil\ \text{nondet}(\mathtt{B},\mathtt{B})^1 & \text{(b)} \\
& \|\ \ell = \text{after}[\![\mathtt{S}]\!]\ \text{?}\ \widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\mathtt{S}_t]\!]\,\text{after}[\![\mathtt{S}_t]\!]\ \rceil\ \text{nondet}(\mathtt{B},\mathtt{B}) & \text{(c.1)} \\
& \qquad\qquad \cup\ \mathbb{1}_{\mathbb{V}}\ \rceil\ \text{nondet}(\neg\mathtt{B},\neg\mathtt{B}) & \text{(c.2)} \\
& \qquad\qquad\quad \cup\ \text{nondet}(\neg\mathtt{B},\neg\mathtt{B}) \times \text{mod}[\![\mathtt{S}_t]\!] & \text{(c.3)} \\
& :\ \emptyset\ ]\!\!) & \text{(d)}
\end{array}
$$

$\text{mod}[\![\mathtt{S}_t]\!]$ is the set of variables that may be modified by $\mathtt{S}_t$

[1] $\rceil$ is left restriction

Example

- $\texttt{S ::=} \ {}^{\ell}\ \texttt{L = H ;}^{\ell'}$

  $\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}]\!]\,\ell = \{\langle \texttt{L},\ \texttt{L}\rangle, \langle \texttt{H},\ \texttt{H}\rangle\}$

  $\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}]\!]\,\ell' = \{\langle \texttt{H},\ \texttt{L}\rangle\} \cup \{\langle \texttt{H},\ \texttt{H}\rangle\}$.

- $\texttt{S}'\ \texttt{::=}\ \texttt{\{ if}\ {}^{\ell_1}\ \texttt{(H)}\ {}^{\ell_2}\ \texttt{L = H ;}^{\ell_3}\ \texttt{else}\ {}^{\ell_4}\ \texttt{L = H ;}^{\ell_5}\ \texttt{\}}^{\ell_6}$

$$
\begin{array}{rcrcl}
 & & \mathsf{nondet}(\texttt{H},\texttt{H}) & = & \mathsf{nondet}(\neg\texttt{H},\neg\texttt{H}) \;=\; \{\texttt{L}\}\\
 & & \widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}']\!]\,\ell_1 & = & \{\langle \texttt{L},\ \texttt{L}\rangle, \langle \texttt{H},\ \texttt{H}\rangle\}\\
\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}']\!]\,\ell_2 & = & \widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}']\!]\,\ell_4 & = & \{\langle \texttt{L},\ \texttt{L}\rangle\}\\
\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}']\!]\,\ell_3 & = & \widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}']\!]\,\ell_5 & = & \{\langle \texttt{H},\ \texttt{H}\rangle\}\\
 & & \widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}']\!]\,\ell_6 & = & \{\langle \texttt{H},\ \texttt{L}\rangle\} \cup \{\langle \texttt{H},\ \texttt{H}\rangle\}
\end{array}
$$

Structural static potential value dependency analysis (cont'd)

- statement list $\mathtt{Sl} ::= \mathtt{Sl}'\ \mathtt{S}$

$$\widehat{\mathcal{S}}^{\exists}_{\mathrm{diff}}[\![\mathtt{Sl}]\!]\,\ell \triangleq \big(\!\!\big[\,\ell \in \mathrm{labx}[\![\mathtt{Sl}']\!]\ \text{?}\ \widehat{\mathcal{S}}^{\exists}_{\mathrm{diff}}[\![\mathtt{Sl}']\!]\,\ell \tag{16.a}$$

$$\big[\!\big]\ \ell \in \mathrm{labx}[\![\mathtt{S}]\!] \setminus \{\mathrm{at}[\![\mathtt{S}]\!]\}\ \text{?}\ \widehat{\mathcal{S}}^{\exists}_{\mathrm{diff}}[\![\mathtt{Sl}']\!]\,\mathrm{at}[\![\mathtt{S}]\!] \mathbin{⨾} \widehat{\mathcal{S}}^{\exists}_{\mathrm{diff}}[\![\mathtt{S}]\!]\,\ell \tag{16.b}$$

$$\text{⸽}\ \varnothing\,\big]\!\!\big)$$

where $r_1 \mathbin{⨾} r_2 \triangleq \{\langle x,\ y\rangle \mid \exists z\ .\ \langle x,\ z\rangle \in r_1 \wedge \langle z,\ y\rangle \in r_2\}$.

Structural static potential value dependency analysis (cont'd)

- iteration $\texttt{S} ::= \texttt{while}\ \ell\ \texttt{(B)}\ \texttt{S}_b$

$$\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}]\!]\,\ell' \quad = \quad (\mathsf{lfp}^{\dot\subseteq}\,\mathcal{F}^{\text{diff}}_{\exists}[\![\texttt{while}\ \ell\ \texttt{(B)}\ \texttt{S}_b]\!])\,\ell' \tag{17}$$

$$\mathcal{F}^{\text{diff}}_{\exists}[\![\texttt{while}\ \ell\ \texttt{(B)}\ \texttt{S}_b]\!]\ X\ \ell' \quad =$$

$$\llparenthesis\ \ell' = \ell\ \text{?}\ \mathbb{1}_{\mathbb{V}} \cup \big(X(\ell)\,\text{⨟}\,(\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}_b]\!]\,\ell\ \rceil\,\mathsf{nondet}(\texttt{B},\texttt{B}))\big) \tag{a}$$

$$\|\ \ell' \in \mathsf{in}[\![\texttt{S}_b]\!]\ \text{?}\ X(\ell)\,\text{⨟}\,(\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}_b]\!]\,\ell'\ \rceil\,\mathsf{nondet}(\texttt{B},\texttt{B})) \tag{b}$$

$$\|\ \ell' = \mathsf{after}[\![\texttt{S}]\!]\ \text{?}\ X(\ell) \cup \big(X(\ell)\,\text{⨟}\,(\mathbb{V} \times \mathsf{mod}[\![\texttt{S}_b]\!])\big) \cup \tag{c}$$

$$X(\ell)\,\text{⨟}\,\Big(\Big(\bigcup_{\ell'' \in \mathsf{breaks\text{-}of}[\![\texttt{S}_b]\!]}\widehat{\mathcal{S}}^{\exists}_{\text{diff}}[\![\texttt{S}_b]\!]\,\ell''\Big)\ \rceil\,\mathsf{nondet}(\texttt{B},\texttt{B})\Big)$$

$$\text{⨟}\ \varnothing\ \rrparenthesis \tag{d}$$

Reduced product with a relational value analysis

Structural compositionality

In the following statement, x and y at ℓ_1 depend on x at ℓ_0

```
ℓ0 y = x ;
ℓ1
```

In the following statement, x and y at ℓ_2 depend on x at ℓ_1

```
ℓ1 y = y-x ;
ℓ2
```

In the sequential composition of the two statements

$$/\star \;\; x = x_0, y = y_0 \;\; \star/$$

```
ℓ0 y = x ;
ℓ1 y = y-x ;
ℓ2
```

y at ℓ_2 depends on x at ℓ_1 which depends on x at ℓ_0

By composition, y at ℓ_2 depends on x at ℓ_0.

However, $y = 0$ at ℓ_2 so y at ℓ_2 does not depend on x at ℓ_0.

Structural compositionality (cont'd)

In the following statement, x and y at ℓ_1 depend on x at ℓ_0

$$/\star\ \ x = x_0, y = y_0\ \ \star/$$

```
ℓ0 y = x ;
ℓ1
```

$$/\star\ \ x = x_0, y = x_0\ \ \star/$$

In the following statement, x and y at ℓ_2 depend on x at ℓ_1

$$/\star\ \ x = x_0, y = y_0\ \ \star/$$

```
ℓ1 y = y-x ;
ℓ2
```

$$/\star\ \ x = x_0, y = y_0 - x_0\ \ \star/$$

In the sequential composition of the two statements

$$/\star\ \ x = x_0, y = y_0\ \ \star/$$

```
ℓ0 y = x ;      /* x = x0, y = x0 */
ℓ1 y = y-x ;    /* x = x0, y = 0 */
ℓ2
```

y at ℓ_2 depends on x at ℓ_1 which depends on x at ℓ_0

By composition, y at ℓ_2 depends on x at ℓ_0.

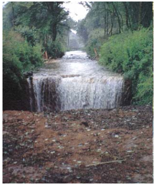
However, $\mathtt{y} = 0$ at ℓ_2 so y at ℓ_2 does not depend on x at ℓ_0.

$\Longrightarrow$ reduced product with a value analysis (here Karr linear equalities)

Dye instrumented semantics

Dye analysis in hydrology

When a river is lost in the ground (*e.g.* la perte du Gour de Champlive in France)



a dye analysis with fluorescein can be used to discover its resurgences

Dye instrumented semantics

- The initial values of the variables are colored with different colors
- The initial color of a variable can be the variable name
- The dye instrumented semantics is sound iff it associates to each variable y and program point ℓ the set of colors/variables x upon which is depends

$$\{x \mid \mathcal{S}^{+\infty}[\![P]\!] \in \mathcal{D}\ell\langle x,\ y\rangle\}$$

- Better approach than postulating the dye instrumented semantics [Cheney, Ahmed, and Acar, 2011] (*e.g*. the mix of colors at tests and assignments can be postulated arbitrarily)

Tracking analysis

- Partition the variables $\mathbb{V}$ into racked $\mathbb{T}$ and untracked $\mathbb{U}$ variables ($\mathbb{V} = \mathbb{T} \cup \mathbb{U}$ and $\mathbb{T} \cap \mathbb{U} = \emptyset$)
- Tracking abstraction $\alpha^{\tau}(\mathbf{D})$ of a dependency property $\mathbf{D} \in \mathbb{L} \to \wp(\mathbb{V} \times \mathbb{V})$

$$\alpha^{\tau}(\mathbf{D})\ell \triangleq \{\mathsf{y} \mid \exists \mathsf{x} \in \mathbb{T} \,.\, \langle \mathsf{x},\ \mathsf{y} \rangle \in \mathbf{D}(\ell)\}$$

- Sound tracking analysis

$$\mathcal{S}^{\tau}[\![\mathsf{s}]\!] \supseteq \alpha^{\tau}(\alpha^{\mathrm{d}}(\{\mathcal{S}^{+\infty}[\![\mathsf{s}]\!]\}))$$

- Examples: taint analysis in privacy/security checks [Ferrara, Olivieri, and Spoto, 2018; Spoto, Burato, Ernst, Ferrara, Lovato, Macedonio, and Spiridon, 2019] (tracked is tainted, untracked is untainted); binding time analysis in offline partial evaluation [Hatcliff, 1998] (tracked is dynamic, untracked is static) and absence of interference [Bowman and Ahmed, 2015; Goguen and Meseguer, 1984; Heinze and Turker, 2018; Lourenço and Caires, 2015; Volpano, Irvine, and Smith, 1996] (tracked is high (private/untrusted), untracked is low (public/trusted)).

Conclusion

Conclusion

- The dependency analysis is not postulated but derived formally by abstract interpretation of the trace semantics.
- No need for extra notions like (hyper)^{n}properties [Assaf, Naumann, Signoles, Totel, and Tronel, 2017], non-standard abstract interpretation [Urban and Müller, 2018], postulated instrumented semantics [Ørbæk, 1995, Sect. 4], multisemantics [Cabon and Schmitt, 2017], monadic reification [Grimm, Maillard, Fournet, Hritcu, Maffei, Protzenko, Ramananandro, Rastogi, Swamy, and Béguelin, 2018], *etc*.

References I

Assaf, Mounir, David A. Naumann, Julien Signoles, Éric Totel, and Frédéric Tronel (2017). "Hypercollecting semantics and its application to static analysis of information flow". In: *POPL*. ACM, pp. 874–887 (9, 36).

Bowman, William J. and Amal Ahmed (2015). "Noninterference for free". In: *ICFP*. ACM, pp. 101–113 (34).

Cabon, Gurvan and Alan Schmitt (2017). "Annotated Multisemantics To Prove Non-Interference Analyses". In: *PLAS@CCS*. ACM, pp. 49–62 (36).

Cheney, James, Amal Ahmed, and Umut A. Acar (2011). "Provenance as dependency analysis". *Mathematical Structures in Computer Science* 21.6, pp. 1301–1337 (33).

Denning, Dorothy E. and Peter J. Denning (1977). "Certification of Programs for Secure Information Flow". *Commun. ACM* 20.7, pp. 504–513 (9).

References II

Ferrara, Pietro, Luca Olivieri, and Fausto Spoto (June 2018). "Tailoring Taint Analysis to GDPR". In: *Privacy Technologies and Policy*. 6th Annual Privacy Forum, APF 2018, Barcelona, Spain, June 13-14, 2018, Revised Selected Papers. DOI: 10.1007/978-3-030-02547-2_4 (34).

Goguen, Joseph A. and José Meseguer (1984). "Unwinding and Inference Control". In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, pp. 75–87 (34).

Grimm, Niklas, Kenji Maillard, Cédric Fournet, Catalin Hritcu, Matteo Maffei, Jonathan Protzenko, Tahina Ramananandro, Aseem Rastogi, Nikhil Swamy, and Santiago Zanella Béguelin (2018). "A monadic framework for relational verification: applied to information security, program equivalence, and optimizations". In: *CPP*. ACM, pp. 130–145 (36).

Hatcliff, John (1998). "An Introduction to Online and Offline Partial Evaluation using a Simple Flowchart Language". In: *Partial Evaluation*. Vol. 1706. Lecture Notes in Computer Science. Springer, pp. 20–82 (34).

References III

Heinze, Thomas S. and Jasmin Turker (2018). "Certified Information Flow Analysis of Service Implementations". In: *SOCA*. IEEE Computer Society, pp. 177–184 (34).

Lourenço, Luísa and Luís Caires (2015). "Dependent Information Flow Types". In: *POPL*. ACM, pp. 317–328 (34).

Ørbæk, Peter (1995). "Can you Trust your Data?" In: *TAPSOFT*. Vol. 915. Lecture Notes in Computer Science. Springer, pp. 575–589 (36).

Spoto, Fausto, Elisa Burato, Michael D. Ernst, Pietro Ferrara, Alberto Lovato, Damiano Macedonio, and Ciprian Spiridon (2019). "Static Identification of Injection Attacks in Java". *ACM Trans. Program. Lang. Syst.* 41.3, 18:1–18:58 (34).

Urban, Caterina and Peter Müller (2018). "An Abstract Interpretation Framework for Input Data Usage". In: *ESOP*. Vol. 10801. Lecture Notes in Computer Science. Springer, pp. 683–710 (9, 36).

Volpano, Dennis M., Cynthia E. Irvine, and Geoffrey Smith (1996). "A Sound Type System for Secure Flow Analysis". *Journal of Computer Security* 4.2/3, pp. 167–188 (34).

The End, Thank you