Using Standard Typing Algorithms Incrementally

with Applications to Security

Matteo Busi
University of Pisa

Overview

1. Motivations
2. The idea
3. An incremental security type system
4. Conclusions

Motivations

Modern software systems:

↙ ↓ ↘

Growing in size and complexity | Subject to C/D | Critical and dependable

We show a methodology to use existing type systems **incrementally** to check absence of or to discover (security) bugs/issues.

The key idea

The idea: retype the new parts and integrate the new information with the existing one, i.e.

- cache typing results during the initial phase.

The key idea

The idea: retype the new parts and integrate the new information with the existing one, i.e.

- cache typing results during the initial phase.
- re-use (and update) results when typing the modified program.

The key idea

The idea: retype the new parts and integrate the new information with the existing one, i.e.

- cache typing results during the initial phase.
- re-use (and update) results when typing the modified program.
- profit ;)

The idea: retype the new parts and integrate the new information with the existing one, i.e.

- cache typing results during the initial phase.
- re-use (and update) results when typing the modified program.
- profit ;)

Most importantly:

- any language, many analyses[a]
- w/o rewriting the algorithm from scratch
- with guarantees on the results

[a] Syntax-based compositional analyses

Recall the `WHILE` language

$$a, b ::= \text{arithmetic and boolean expressions}$$
$$c ::= \texttt{skip} \mid x := a \mid c_1;\, c_2 \mid \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \texttt{while}\ b\ \texttt{do}\ c$$
$$p ::= a \mid b \mid c$$

...equipped with the ***Volpano-Smith-Irvine*** type system:

$$\Gamma \vdash_{\mathcal{S}} p : \varsigma$$

where ς:

$$\tau ::= H \mid L \qquad \varsigma ::= \tau \mid \tau\, var \mid \tau\, cmd$$

Example: a security-related application (I)

Rule for `if-then-else`:

($\mathcal{S}$-If)

$$\frac{\boxed{\Gamma} \vdash_{\mathcal{S}} b : \tau_b \qquad \boxed{\Gamma} \vdash_{\mathcal{S}} c_1 : \tau_1\ cmd \qquad \boxed{\Gamma} \vdash_{\mathcal{S}} c_2 : \tau_2\ cmd \qquad \boxed{\tau_b = \tau_1 = \tau_2 \wedge \varsigma = \tau_b\ cmd}}{\Gamma \vdash_{\mathcal{S}} \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 : \varsigma}$$

Example: a security-related application (I)

Rule for `if-then-else`:

($\mathcal{S}$-If)

$$\frac{\boxed{\Gamma} \vdash_{\mathcal{S}} b : \tau_b \qquad \boxed{\Gamma} \vdash_{\mathcal{S}} c_1 : \tau_1\ cmd \qquad \boxed{\Gamma} \vdash_{\mathcal{S}} c_2 : \tau_2\ cmd \qquad \boxed{\tau_b = \tau_1 = \tau_2 \land \varsigma = \tau_b\ cmd}}{\Gamma \vdash_{\mathcal{S}} \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 : \varsigma}$$

Two ***meta functions*** highlighted:

- $\boxed{tr}$, mapping typing environments of a term into the ones of its subterms
- $\boxed{checkJoin}$, capturing all the other preconditions of the rule and producing the final result

Example: a security-related application (II)

The incremental VSI ($\mathcal{IS}$) is just four steps away:

1. Defining the shape of caches.
2. Building caches.
3. Incremental typing.
4. Type coherence.

$\mathcal{JS}$: Defining the shape of caches.

Caches are simple:

$$C \in Cache = \wp(Phrase \times Env \times PType)$$

i.e. a cache line is a triple that associates programs to typing environments and types.

Starting with the AST annotated with types (***aAST***), visit the tree accumulating typing environment and type information.

`if-then-else` case:

$$
\begin{aligned}
&\mathit{buildCache}\ (\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 : \tau\ \mathit{cmd})\ \Gamma \triangleq \\
&\quad \{(\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \Gamma_{|\,FV(\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2)}, \tau\ \mathit{cmd})\} \\
&\quad \cup (\mathit{buildCache}\ (b : \tau_b)\ \boxed{\Gamma}) \\
&\quad \cup (\mathit{buildCache}\ (c_1 : \tau_1\ \mathit{cmd})\ \boxed{\Gamma}) \\
&\quad \cup (\mathit{buildCache}\ (c_2 : \tau_2\ \mathit{cmd})\ \boxed{\Gamma})
\end{aligned}
$$

Consider

$$\texttt{if}\ y \leq 10\,\texttt{then}\,y := x + y\,\texttt{else}\,x := 42$$

with $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

The cache will be

Expression	**Environment**	**Type**
$\texttt{if}\ y \leq 10\,\texttt{then}\,y := x + y\,\texttt{else}\,x := 42$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$

$\mathcal{JS}$: Building caches.

Consider

$$\texttt{if } y \leq 10 \texttt{ then } y := x + y \texttt{ else } x := 42$$

with $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

The cache will be

Expression	Environment	Type
$\texttt{if } y \leq 10 \texttt{ then } y := x + y \texttt{ else } x := 42$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$y \leq 10$	$[\mathsf{y} \mapsto L\,var]$	L

$\mathcal{JS}$: Building caches.

Consider

$$\texttt{if}\ y \le 10\ \texttt{then}\ y := x + y\ \texttt{else}\ x := 42$$

with $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

The cache will be

Expression	Environment	Type
$\texttt{if}\ y \le 10\ \texttt{then}\ y := x + y\ \texttt{else}\ x := 42$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$y \le 10$	$[\mathsf{y} \mapsto L\,var]$	L
y	$[\mathsf{y} \mapsto L\,var]$	$L\,var$
10	$\emptyset$	L

Consider

$$\texttt{if}\ y \leq 10\ \texttt{then}\ y := x + y\ \texttt{else}\ x := 42$$

with $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

The cache will be

Expression	**Environment**	**Type**
$\texttt{if}\ y \leq 10\ \texttt{then}\ y := x + y\ \texttt{else}\ x := 42$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$y \leq 10$	$[\mathsf{y} \mapsto L\,var]$	L
y	$[\mathsf{y} \mapsto L\,var]$	$L\,var$
10	$\emptyset$	L
$y := x + y$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$

$\mathcal{JS}$: Building caches.

Consider

$$\texttt{if } y \leq 10 \texttt{ then } y := x + y \texttt{ else } x := 42$$

with $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

The cache will be

Expression	**Environment**	**Type**
$\texttt{if } y \leq 10 \texttt{ then } y := x + y \texttt{ else } x := 42$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$y \leq 10$	$[\mathsf{y} \mapsto L\,var]$	L
y	$[\mathsf{y} \mapsto L\,var]$	$L\,var$
10	$\emptyset$	L
$y := x + y$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$x + y$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
x	$[\mathsf{x} \mapsto L\,var]$	$L\,var$

Consider

$$\texttt{if } y \leq 10 \texttt{ then } y := x + y \texttt{ else } x := 42$$

with $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

The cache will be

Expression	**Environment**	**Type**
$\texttt{if } y \leq 10 \texttt{ then } y := x + y \texttt{ else } x := 42$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$y \leq 10$	$[\mathsf{y} \mapsto L\,var]$	L
y	$[\mathsf{y} \mapsto L\,var]$	$L\,var$
10	$\emptyset$	L
$y := x + y$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$x + y$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
x	$[\mathsf{x} \mapsto L\,var]$	$L\,var$
$x := 42$	$[\mathsf{x} \mapsto L\,var]$	$L\,var$

Consider

$$\texttt{if } y \leq 10 \texttt{ then } y := x + y \texttt{ else } x := 42$$

with $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

The cache will be

Expression	**Environment**	**Type**
$\texttt{if } y \leq 10 \texttt{ then } y := x + y \texttt{ else } x := 42$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$y \leq 10$	$[\mathsf{y} \mapsto L\,var]$	L
y	$[\mathsf{y} \mapsto L\,var]$	$L\,var$
10	$\emptyset$	L
$y := x + y$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
$x + y$	$[\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$	$L\,cmd$
x	$[\mathsf{x} \mapsto L\,var]$	$L\,var$
$x := 42$	$[\mathsf{x} \mapsto L\,var]$	$L\,var$
42	$\emptyset$	L

In general, three kind of rules

$$\frac{C(t) = \langle \Gamma', R \rangle \qquad compat_{env}(\Gamma, \Gamma', t)}{\Gamma, C \vdash_{\mathcal{J}} t : R \rhd C}$$

In general, three kind of rules

$$\frac{C(t) = \langle \Gamma', R \rangle \qquad compat_{env}(\Gamma, \Gamma', t)}{\Gamma, C \vdash_{\mathcal{J}} t : R \rhd C}$$

$$\frac{\Gamma \vdash t : R \qquad C' = C \cup \{(t, \Gamma_{|\,FV(t)}, R)\}}{\Gamma, C \vdash_{\mathcal{J}} t : R \rhd C'} \; miss(C, t, \Gamma)$$

$$\frac{\begin{array}{c}\forall i \in \mathbb{I}_t \,.\, \boxed{tr_t^{t_i}(\Gamma, \{R_j\}_{j<i \,\wedge\, j \in \mathbb{I}_t})}, C \vdash_{\mathcal{J}} t_i : R_i \rhd C^i \\ \boxed{checkJoin_t(\Gamma, \{R_i\}_{i \in \mathbb{I}_t}, \texttt{out}\ R)} \qquad C' = \{(t, \Gamma_{|\,FV(t)}, R)\} \cup \bigcup_{i \in \mathbb{I}_t} C^i\end{array}}{\Gamma, C \vdash_{\mathcal{J}} t : R \rhd C'} \; miss(C, t, \Gamma)$$

Note the highlighted parts!

Looks scary? Here's an example for the `if-then-else`:

($\mathcal{JS}$-IF-MISS)

$$\frac{\begin{array}{c}\boxed{\Gamma}, C \vdash_{\mathcal{JS}} b : \tau_b \rhd C'' \qquad \boxed{\Gamma}, C \vdash_{\mathcal{JS}} c_1 : \tau_1\ cmd \rhd C''' \\ \boxed{\Gamma}, C \vdash_{\mathcal{JS}} c_2 : \tau_2\ cmd \rhd C^{iv} \qquad \boxed{\tau_1 = \tau_2 = \tau_b \wedge \varsigma = \tau_1\ cmd} \\ C' = C'' \cup C''' \cup C^{iv} \cup \{(\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2, \Gamma_{|FV(\texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2)}, \varsigma)\}\end{array}}{\Gamma, C \vdash_{\mathcal{JS}} \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 : \varsigma \rhd C'}\ miss(\ldots)$$

Let's type check

$$\texttt{if}\ x \leq 42\ \texttt{then}\ y := x + y\ \texttt{else}\ x := 42$$

in $\Gamma = [\mathsf{x} \mapsto L\, var; \mathsf{y} \mapsto L\, var]$.

$$\frac{}{\Gamma, C \vdash_{\mathcal{JS}} \texttt{if}\ x \leq 42\ \texttt{then}\ y := x + y\ \texttt{else}\ x := 42 : \triangleright \_}$$

Let's type check

$$\texttt{if } x \leq 42 \texttt{ then } y := x + y \texttt{ else } x := 42$$

in $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

$$\frac{\overline{\Gamma, C \vdash_{\mathcal{JS}} x \leq 42 : \triangleright\_}}{\Gamma, C \vdash_{\mathcal{JS}} \texttt{if } x \leq 42 \texttt{ then } y := x + y \texttt{ else } x := 42 : \triangleright\_}$$

Let's type check

$$\texttt{if } x \leq 42 \texttt{ then } y := x + y \texttt{ else } x := 42$$

in $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

$$\dfrac{\dfrac{\boxed{\Gamma, C \vdash_{\mathcal{JS}} x : L\,var \rhd \_}}{\Gamma, C \vdash_{\mathcal{JS}} x \leq 42 : \rhd \_}}{\Gamma, C \vdash_{\mathcal{JS}} \texttt{if } x \leq 42 \texttt{ then } y := x + y \texttt{ else } x := 42 : \rhd \_}$$

$\mathcal{JS}$: Incremental typing.

Let's type check

$$\texttt{if } x \leq 42 \texttt{ then } y := x + y \texttt{ else } x := 42$$

in $\Gamma = [\mathsf{x} \mapsto L\, var; \mathsf{y} \mapsto L\, var]$.

$$\dfrac{\dfrac{\boxed{\Gamma, C \vdash_{\mathcal{JS}} x : L\, var \rhd \_} \qquad \boxed{\Gamma, C \vdash_{\mathcal{JS}} 42 : L \rhd \_}}{\Gamma, C \vdash_{\mathcal{JS}} x \leq 42 : L\, cmd \rhd \_}}{\Gamma, C \vdash_{\mathcal{JS}} \texttt{if } x \leq 42 \texttt{ then } y := x + y \texttt{ else } x := 42 : \rhd \_}$$

Let's type check

$$\texttt{if } x \leq 42 \texttt{ then } y := x + y \texttt{ else } x := 42$$

in $\Gamma = [\mathsf{x} \mapsto L\, var; \mathsf{y} \mapsto L\, var]$.

$$\dfrac{\dfrac{\boxed{\Gamma, C \vdash_{\mathcal{JS}} x : L\, var \rhd \_} \qquad \boxed{\Gamma, C \vdash_{\mathcal{JS}} 42 : L \rhd \_}}{\Gamma, C \vdash_{\mathcal{JS}} x \leq 42 : L\, cmd \rhd \_} \quad \boxed{\Gamma, C \vdash_{\mathcal{JS}} y := x + y : L\, cmd \rhd \_}}{\Gamma, C \vdash_{\mathcal{JS}} \texttt{if } x \leq 42 \texttt{ then } y := x + y \texttt{ else } x := 42 : \rhd \_}$$

$\mathcal{JS}$: Incremental typing.

Let's type check

$$\texttt{if}\ x \leq 42\ \texttt{then}\ y := x + y\ \texttt{else}\ x := 42$$

in $\Gamma = [\mathsf{x} \mapsto L\,var; \mathsf{y} \mapsto L\,var]$.

$$\dfrac{\dfrac{\boxed{\Gamma, C \vdash_{\mathcal{JS}} x : L\,var \triangleright \_} \qquad \boxed{\Gamma, C \vdash_{\mathcal{JS}} 42 : L \triangleright \_}}{\Gamma, C \vdash_{\mathcal{JS}} x \leq 42 : L\,cmd \triangleright \_} \quad \boxed{\Gamma, C \vdash_{\mathcal{JS}} y := x + y : L\,cmd \triangleright \_} \qquad \boxed{\Gamma, C \vdash_{\mathcal{JS}} x := 42 : L \triangleright \_}}{\Gamma, C \vdash_{\mathcal{JS}} \texttt{if}\ x \leq 42\ \texttt{then}\ y := x + y\ \texttt{else}\ x := 42 : \triangleright \_}$$

$\mathcal{JS}$: Incremental typing.

Let's type check

$$\texttt{if } x \le 42 \texttt{ then } y := x + y \texttt{ else } x := 42$$

in $\Gamma = [\mathsf{x} \mapsto L\, var; \mathsf{y} \mapsto L\, var]$.

$$\dfrac{\dfrac{\boxed{\Gamma, C \vdash_{\mathcal{JS}} x : L\, var \rhd \_} \qquad \boxed{\Gamma, C \vdash_{\mathcal{JS}} 42 : L \rhd \_}}{\Gamma, C \vdash_{\mathcal{JS}} x \le 42 : L\, cmd \rhd \_} \quad \boxed{\Gamma, C \vdash_{\mathcal{JS}} y := x + y : L\, cmd \rhd \_} \qquad \boxed{\Gamma, C \vdash_{\mathcal{JS}} x := 42 : L \rhd \_}}{\Gamma, C \vdash_{\mathcal{JS}} \texttt{if } x \le 42 \texttt{ then } y := x + y \texttt{ else } x := 42 : L\, cmd \rhd \_}$$

$\mathcal{JS}$: Type coherence.

Proving type coherence requires

1. showing that that particular $compat_{env}$ expresses a ***compatibility***.

$\mathcal{JS}$: Type coherence.

Proving type coherence requires

1. showing that that particular $compat_{env}$ expresses a ***compatibility***.
2. there's no point two! **;)**

In fact, the following ***general*** theorem holds

Theorem

If $compat_{env}$ expresses a compatibility, then for all terms t, caches C, typing environments Γ, and typing algorithm $\mathcal{A}$

$$\Gamma \vdash_{\mathcal{A}} t : R \iff \Gamma, C \vdash_{\mathcal{JA}} t : R \triangleright C'.$$

Conclusions

What we did:

- We sketched a general process,
- that uses existing typing algorithm as incremental ones,
- which is applicable to any language and many analyses.

Work in progress and future work:

- Prototype at `https://github.com/mcaos/incremental-mincaml`.
- Benchmarking.
- More analyses!
- Applications to secure compilation.

Preprint at: `https://arxiv.org/abs/1808.00225`

THE END

Matteo Busi
matteo.busi@di.unipi.it
University of Pisa