Bi-Abductive Adversarial Program Synthesis

and software security applications

Julien Vanegue

January 14, 2024

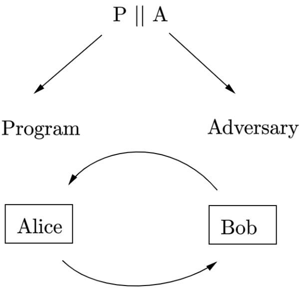
Why Incorrectness Logic Matters

1. The world's software stack becomes even more distributed.
   - There is industry pressure to adopt compositional & incremental program analysis techniques.
2. Large scale review of program analysis alerts requires prioritization.
   - *No False Positive* logic is needed for CI/CD integration.
3. Program errors must be explained to developers.
   - Often requires a proof of vulnerability.

New Research Questions

1. How to prioritize bug investigation and fixing?
   - Help bug triage by using incorrectness reasoning.
2. Which bugs are critical security vulnerabilities?
   - Extend incorrectness logic for *exploitability*
3. Can we generate a bug witness automatically?
   - Guide program synthesis adversarially

Adversarial Logic (SAS'22)



Incorrectness Logic
+ Dolev-Yao Model
= Adversarial Logic

Example: Oscillating Bit Protocol

```
// pre: client socket established
uint secret = rand();
void program(int sock)
{
 uint   err = 0;
 uint   cred = 0;
 while (true) {
    recv(sock, cred);
    if (secret == cred)
     err = 0;
    else if (secret < cred)
      err = 1;
    else if (secret > cred)
      err = 2;
    if (!err) do_serve(sock);
    send(sock, err);
 }
}
```

```
// pre: server socket established
int adversary(int sock)
{
 uint ret = 1;
 uint guess = UINT_MAX;
 uint step = (UINT_MAX/2)+1;
 while (true) {
    send(sock, guess);
    recv(sock, ret);
    if (ret == 1)
      guess = guess - step;
    else if (ret == 2)
      guess = guess + step;
   step = (step / 2) + 1;
   adv_assert(ret == 0);
 }
}
```

Concurrent Adversarial Separation Logic

CASL = CISL + Adversarial Reasoning + Rely-Guarantee

$$
\begin{array}{l} \mathsf{send}(c, 8); \\ \mathsf{recv}(c, y); \end{array}
\quad \Bigg\| \quad
\begin{array}{l}
\mathsf{local}\ \mathit{secret} := *; \\
\mathsf{local}\ w[8] := \{0\}; \\
\mathsf{local}\ z := 0; \\
\mathsf{recv}(c, x); \\
\mathsf{if}\ (x \leq 8) \\
\quad z := w[x]; \\
\quad \mathsf{send}(c, z);
\end{array}
$$

See also:

A marriage of rely/guarantee and separation logic
by Viktor Vafeiadis and Matthew Parkinson (CONCUR'07)

CISL: Concurrent Incorrectness Separation Logic
by Azalea Raad, Josh Berdine, Dereck Dreyer and Peter O'Hearn (POPL'22)

CASL: A General Approach to Under-Approximate Reasoning About Concurrent Programs (CONCUR'23)
by Azalea Raad, Julien Vanegue, Josh Berdine and Peter O'Hearn

What is next?

AL / CASL requires an input adversarial program to reason about.

Can we synthesize the adversarial program and conditions?

Idea: Combine bi-abduction and deductive program synthesis

See also:

Compositional Shape Analysis By means of Bi-Abduction
by Cristiano Calcagno, Dino Distefano, Peter O'Hearn and Hongseok Yang (POPL'09)

Bi-abductive resource invariant synthesis
by Cristiano Calcagno, Dino Distefano, and Viktor Vafeiadis (APLAS'09)

Inductive Invariant Generation Via Abductive Inference
by Isil Dillig, Thomas Dillig, Boyang Li and Ken McMillan (OOPSLA'13)

JaVerT: JavaScript Verification and Testing Framework
by Philippa Gardner (PPDP'18)

Recall: (Bi-)Abduction

Abduction:

Compute the missing part δ of precondition P, such that $P * \delta \vdash Q$

Bi-Abduction:

Compute anti-frame $\mathcal{U}$ and frame $\mathcal{F}$, such that $P * \mathcal{U} \vdash Q * \mathcal{F}$

Chaining:

$$\textbf{BA-seq}\ \frac{\{P_1\}c_1\{Q_1\} \qquad \{P_2\}c_2\{Q_2\}}{\{P_1 * \mathcal{U}\}\ c_1\ ;\ c_2\ \{Q_2 * \mathcal{F}\}}\ Q_1 * \mathcal{U} \vdash P_2 * \mathcal{F}$$

Bi-abductive Adversarial Reasoning

$[\text{AdvPre} : \mathit{Emp}]$
$\mathit{adversary}(\mathit{int}\ x)$
$\{$
$\quad \mathit{int}\ \mathit{ret} = \mathit{program}(x);$
$\}$
$[\text{AdvPost} : \mathit{ret} = v_s]$

$[\text{AdvPre} : P * \mathcal{U}]$
$\mathit{program}(\mathit{int}\ x)$
$\{$
$\quad \text{local}\ \mathit{secret} := v_s;$
$\quad \text{local}\ w[8] := \{0\};$
$\quad \text{local}\ z := 0;$
$\quad \text{if}\ (x \leq 8)$
$\qquad z := w[x];$
$\mathit{return}(z);$
$\}$
$[\text{AdvPost} : Q * \mathcal{F}]$

Adversarial Frame and Anti-frame

$[\text{AdvPre} : P * \mathcal{U}]$

$program(int\ x)$

$\{$

local $secret := v_s$;

local $w[8] := \{0\}$;

local $z := 0$;

if $(x \leq 8)$

$z := w[x]$;

$return(z)$;

$\}$

$[\text{AdvPost} : Q * \mathcal{F}]$

We want $[P * \mathcal{U}]\ c\ [Q * \mathcal{F}]$

That is:
$VCGen(c, P) * \mathcal{U} \vdash Q * \mathcal{F}$

We pick:
$\mathcal{U} : x = 8$
$P : Emp$
$\mathcal{F} : z \mapsto l_z * l_z = v_s$
$Q : \bigast_{i=0}^{7} w_i \mapsto l_i * l_i = 0$
$* \ sec \mapsto l_s * l_s = v_s$

$VCGen(c, P) = Q *$
$((x < 8 \Rightarrow z \mapsto l_z * l_z = 0) \vee$
$(x = 8 \Rightarrow z \mapsto l_z * l_z = v_s))$

Separated Adversarial Synthesis

Program synthesis:

$$\exists c : \{P\}\ c\ \{Q\}$$

Adversarial synthesis:

$$\exists c_a : \{Emp * P_p\}\ c_a \mid\mid c_p\ \{Q_a * Q_p\}$$

Our new separated problem:

$$\exists c_a : \{Emp\}\ c_a\ \{P_p\} \text{ and } \{P_p\}\ c_p\ \{Q_p\} \text{ and } Q_p \implies Q_a$$

We reduced adversarial synthesis to a sequential program synthesis.

Bloomberg

$$\textsc{Emp}\quad \frac{\mathsf{EV}(\Gamma, \mathcal{P}, \mathcal{Q}) = \emptyset \qquad \vdash \phi \Rightarrow \psi}{\Gamma;\ \{\phi; \mathsf{emp}\} \rightsquigarrow \{\psi; \mathsf{emp}\} \,|\, \mathsf{skip}}$$

$$\textsc{Read}\quad \frac{a \in \mathsf{GV}(\Gamma, \mathcal{P}, \mathcal{Q}) \qquad y \notin \mathsf{Vars}(\Gamma, \mathcal{P}, \mathcal{Q}) \qquad \Gamma \cup \{y\};\ [y/a]\{\phi; \langle x, \iota\rangle \mapsto a * P\} \rightsquigarrow [y/a]\{Q\} \,|\, c}{\Gamma;\ \{\phi; \langle x, \iota\rangle \mapsto a * P\} \rightsquigarrow \{Q\} \,|\, \mathbf{let}\ y = *(x + \iota); c}$$

$$\textsc{Write}\quad \frac{\mathsf{Vars}(e) \subseteq \Gamma \qquad e \neq e' \qquad \Gamma;\ \{\phi; \langle x, \iota\rangle \mapsto e * P\} \rightsquigarrow \{\psi; \langle x, \iota\rangle \mapsto e * Q\} \,|\, c}{\Gamma;\ \{\phi; \langle x, \iota\rangle \mapsto e' * P\} \rightsquigarrow \{\psi; \langle x, \iota\rangle \mapsto e * Q\} \,\Big|\, *(x + \iota) = e; c}$$

$$\textsc{Frame}\quad \frac{\mathsf{EV}(\Gamma, \mathcal{P}, \mathcal{Q}) \cap \mathsf{Vars}(R) = \emptyset \qquad \Gamma;\ \{\phi; P\} \rightsquigarrow \{\psi; Q\} \,|\, c}{\Gamma;\ \{\phi; P * R\} \rightsquigarrow \{\psi; Q * R\} \,|\, c}$$

$$\frac{}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \{\mathsf{emp}\} \rightsquigarrow \{\mathsf{emp}\}}\ \textsc{Emp}\text{ with } c_7 = \mathsf{skip}$$

$$c_6 = c_7$$

$$\frac{}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \left\{\boxed{y \mapsto \mathsf{a2}}\right\} \rightsquigarrow \left\{\boxed{y \mapsto \mathsf{a2}}\right\} \Big| c_6}\ \textsc{Frame}$$

$$c_5 = *y = \mathsf{a2}; c_6$$

$$\frac{}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \left\{\boxed{y \mapsto \mathsf{b2}}\right\} \rightsquigarrow \left\{\boxed{y \mapsto \mathsf{a2}}\right\} \Big| c_5}\ \textsc{Write}$$

$$c_4 = c_5$$

$$\frac{}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \left\{\boxed{x \mapsto \mathsf{b2}} * y \mapsto \mathsf{b2}\right\} \rightsquigarrow \left\{\boxed{x \mapsto \mathsf{b2}} * y \mapsto \mathsf{a2}\right\} \Big| c_4}\ \textsc{Frame}$$

$$c_3 = *x = \mathsf{b2}; c_4$$

$$\frac{}{\{x, y, \mathsf{a2}, \mathsf{b2}\};\ \left\{\boxed{x \mapsto \mathsf{a2}} * y \mapsto \mathsf{b2}\right\} \rightsquigarrow \left\{\boxed{x \mapsto \mathsf{b2}} * y \mapsto \mathsf{a2}\right\} \Big| c_3}\ \textsc{Write}$$

$$c_2 = \mathbf{let}\ \mathsf{b2} = *y; c_3$$

$$\frac{}{\{x, y, \mathsf{a2}\};\ \left\{x \mapsto \mathsf{a2} * \boxed{y \mapsto b}\right\} \rightsquigarrow \{x \mapsto b * y \mapsto \mathsf{a2}\} \Big| c_2}\ \textsc{Read}$$

$$c_1 = \mathbf{let}\ \mathsf{a2} = *x; c_2$$

$$\frac{}{\{x, y\};\ \left\{\boxed{x \mapsto a} * y \mapsto b\right\} \rightsquigarrow \{x \mapsto b * y \mapsto a\} \Big| c_1}\ \textsc{Read}$$

From *Structuring the synthesis of heap-manipulating programs*
by Nadia Polikarpova and Ilya Sergey (POPL'19)

Augmenting Program Synthesis

$$\textbf{Seq}\ \frac{\Gamma; \{\phi; P\} \rightsquigarrow \{\psi; Q\} | c_1 \qquad \Sigma; \{\psi'; Q\} \rightsquigarrow \{\kappa; R\} | c_2 \qquad \psi \implies \psi'}{\Gamma, \Sigma; \{\phi; P\} \rightsquigarrow \{\kappa; R\} | c_1; c_2}$$

$$\textbf{Par}\ \frac{\Gamma; \{\phi; P_1\} \rightsquigarrow \{\psi; Q_1\} | c_1 \qquad \Sigma; \{\phi'; P_2\} \rightsquigarrow \{\psi'; Q_2\} | c_2}{\Gamma, \Sigma; \{\phi; P_1 * \phi'; P_2\} \rightsquigarrow \{\psi; Q_1 * \psi'; Q_2\} | c_1 || c_2}$$

Similar rules can be defined for Send, Recv, etc.

To be continued

Thank you Peter!

Thank you

What are your questions?

Contact: julien.vanegue@gmail.com