Relational Logic for Fine-grained Security Policy and Translation Validation

Dave Naumann (Stevens Inst. of Tech.; Princeton U.)

with Anindya Banerjee (NSF; IMDEA)

and Mohammad Nikouei (Stevens)

Dagstuhl Seminar 18201, May 2018

Dependency is a relational property

$$
\begin{array}{lll}
C_1 \;\widehat{=}\; & s := 0; & \\
& \textsf{while } p \neq \textsf{null do} & \\
& \quad \textsf{if } \neg p.del \textsf{ then } s := s + p.val \textsf{ fi}; & \\
& \quad p := p.next \textsf{ od} &
\end{array}
$$

Sum depends only on initial values of non-deleted nodes.

$$(C_1 \mid C_1) : \; \mathbb{A}(listnd(p)) \approx\!> \mathbb{A}(s)$$

- $listnd(p)$ is the list of non-deleted values
- $\mathbb{A}$ means two runs agree

Dependency is a relational property

$$
\begin{array}{lll}
C_1 \;\widehat{=}\; & s := 0; & \\
& \textsf{while } p \neq \textsf{null do} & \\
& \quad \textsf{if } \neg p.del \textsf{ then } s := s + p.val \textsf{ fi}; & \\
& \quad p := p.next \textsf{ od} &
\end{array}
$$

Sum depends only on initial values of non-deleted nodes.

$$(C_1 \mid C_1) : \; \mathbb{A}(listnd(p)) \approx\!> \mathbb{A}(s)$$

- $listnd(p)$ is the list of non-deleted values
- $\mathbb{A}$ means two runs agree

Declassification policy: assume
$numGuesses < 3 \Rightarrow \mathbb{A}(guess = password)$

Relations between programs

Program equivalence, program transformations

$C_2 \mathrel{\widehat{=}} y := x + 1; x := x + 1$ $\qquad$ $C_2' \mathrel{\widehat{=}} x := x + 1; y := x$

$$(C_2 \mid C_2') : \mathit{Agree}(x) \approx\!> \mathit{Agree}(x) \wedge \mathit{Agree}(y)$$

Relations between programs

Program equivalence, program transformations

$C_2 \mathrel{\widehat{=}} y := x + 1; x := x + 1$ $\qquad$ $C_2' \mathrel{\widehat{=}} x := x + 1; y := x$

$$(C_2 \mid C_2') : \mathit{Agree}(x) \approx\!> \mathit{Agree}(x) \wedge \mathit{Agree}(y)$$

Relational Hoare logic

$$\frac{(C \mid C') : \mathcal{Q} \approx\!> \mathcal{R} \qquad (D \mid D') : \mathcal{R} \approx\!> \mathcal{S}}{(C; D \mid C'; D') : \mathcal{Q} \approx\!> \mathcal{S}}$$

Biprograms designate intended alignments for reasoning.

$$(s := s + p.\mathit{val} \mid s := s + p.\mathit{val}) \; ; \; (p := p.\mathit{next} \mid p := p.\mathit{next})$$

Partially synchronized loops, relational procedure specs, . . .

Preserving relational properties

Suppose
$(C \mid C) : \mathcal{P} \approx\!> \mathcal{Q}$ (security policy for source)
$(C \mid C') : \mathcal{R} \approx\!> \mathcal{S}$ (functional equiv. – correct compilation)
then
$(C' \mid C') : \mathcal{R}^{-1}; \mathcal{P}; \mathcal{R} \approx\!> \mathcal{S}^{-1}; \mathcal{Q}; \mathcal{S}$ by composed relations
hence
$(C' \mid C') : \mathcal{P} \approx\!> \mathcal{Q}$ by partial equiv. relations

So transformations preserve security?

Research program $\quad \Phi \vdash CC : \mathcal{Q} \approx\!> \mathcal{R}[\varepsilon]$

Relational logic with explicit heap regions to express agreement and raming, in terms amenable to SMT solvers.

Explicit regions in frame conditions and stateful module encapsulation with cross-module callbacks (J.ACM'13).

Frame conditions including read effects, with extensional semantics (TOPLAS'18).

Relational logic with biprograms—notation to designate intended alignment of intermediate computation steps (FSTTCS'16).

Relational annotations for epistemic security property (CSF'14,'18).

— Security policy as relational assumptions and assertions.
— Translated to policy on target code, in which selected low level observations are made explicit.
— Relational verification for translation validation.

Some related work

- [Benton'04]: idea of relational Hoare logic applied to transformations and NI; simple imperative programs
- [Yang'02–'07]: relational separation logic, incorporating heap
- [Barthe et al.'11,'13,'16]: construction of product programs from program-pairs with similar and dissimilar structures. Handle $\forall\exists$ properties; simple imperative programs
- [Hawblitzel et al.'13, Lahiri et al.'13]: intra- and inter-procedural reasoning about transformations; relational procedure summaries; heap modeled by maps
- [Godlin/Strichmann'13]: sound rules for proving equivalence of programs with similar control structure; implemented in equivalence-checking tools; pointer structures with no sharing
- [Grimm et al.'18] monadic framework for relational verification