Robust is the New Black

new criteria for secure compilation

Marco Patrignani

11th December 2017

CISPA
Center for IT-Security, Privacy
and Accountability

Special Thanks to:

Contents

Robust Compilation Lattice

Robustly-Safe Compilation

Background

Typed Closure Conversion Preserves Observational Equivalence

Amal Ahmed Matthias Blume
Toyota Technological Institute at Chicago
{amal,blume}@tti-c.org

Fully Abstract Compilation to JavaScript

-Chen Pierre-Evariste Dagand Pierre-Yves Strub[1] Benj
-nd MSR-INRIA[1]
-trath.ac.uk pierre-yves@stru

Secure Implementations for Typed Session Abstractions

Ricardo Corin[1,2,3] Pierre-Malo Deniélou[1,2] Cédric Fournet[1,2]
Karthikeyan Bhargavan[1,2] James Leifer[1]
[1] MSR-INRIA Joint Centre [2] Microsoft Research [3] University of T

Fully-Abstract Compilation by Approximate Back-Translation

Dominique Devriese Marco Patrignani Frank Piess-
iMinds-Distrinet, Computer Science dept., KU L-
first.last @ cs.kul-

Authentication primitives and their compilation

Martín Abadi* Cédric Fournet Georges G
Bell Labs Research Microsoft Research INRIA Rocq
Lucent Technologies

On Protection by Layout Randomization

MARTÍN ABADI, Microsoft Research, Silicon Vall-
Santa Cruz; Collège de France
GORDON D. PLOTKIN, M-
University of Edin-

Beyond Good and Evil
Formalizing the Security Guarantees of Compartmentalizing Compilation

Yannis Juglaret[1,2] Cătălin Hriţcu[1] Arthur Azevedo de Amorim[4] Boris Eng[1,3] Benjamin C. Pierce[4]
[1]Inria Paris [2]Université Paris Diderot (Paris 7) [3]Université Paris 8 [4]University of Pennsylvania

Secure Compilation
of Object-Oriented Components
to Protected Module Architectures

Marco Patrignani, Dave Clarke, and Frank Piessens

iMinds-DistriNet, Dept. Computer Sci-
{first.last}@c-

A Secure Compiler for ML Module

-2 and Dave Clar

Local Memory via Layout Randomization

Corin Pitcher Julian Rathke James Riely
University of Southampton -ul University

An Equivalence-Preserving CPS Translation
via Multi-Language Semantics *

Amal Ahmed Matthias Blume
Google
blume@google.com

Secure Compilation to Protected Module Architectures

-ton and Raoul Strackx and Bart Jacobs, i
- and iMinds-D

On Modular and Fully-Abstract Compil

Marco Patr-
MPI-SWS

Dominique D

Fully Abstract Compilation via Universal Embedding *

Marco Patrig
Dept. Comput
and Dave C

Background

Fully abstract compilation (FAC) de-facto standard for compiler security

Typed Closure Conve

Fully Abstract Compilation to JavaScript

Secure Implementations for Typed Session Abstractions

Cédric Fournet[1,2]

Authentication

Martín Abadi*
Bell Labs Research
Lucent Technologies

Secu
of Object-O
o Protected

Marco Patrignani,
iMinds-DistriNet, D

L Module

Dave Clar

Local Memory via Lay

Corin Pitcher

Julian Rathke
University of Southampton

Multi-Language Semantics*

Translation

Amal Ahmed

Secure Compilation to Protected Module Architectures

and Raoul Strackx and Bart Jacobs,

and iMinds-D

On Modular and Fully-Abstract Compil

Marco Pat

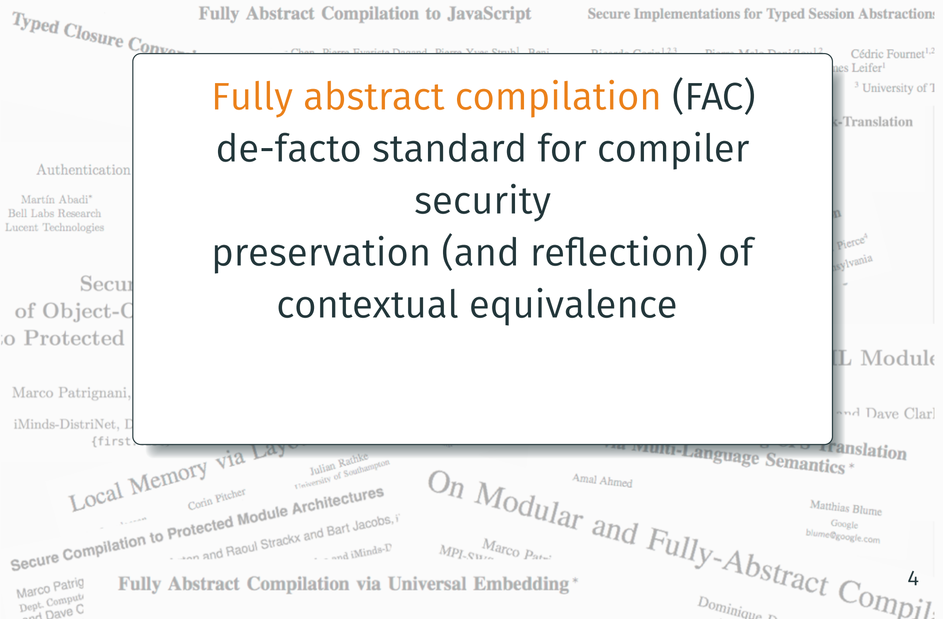
Matthias Blume
Google
blume@google.com

Marco Patrig
Dept. Comput
and Dave C

Fully Abstract Compilation via Universal Embedding*

Dominique

Background



Typed Closure Conve

Fully Abstract Compilation to JavaScript

Secure Implementations for Typed Session Abstraction

Cédric Fournet[1,2]

nes Leifer[1]

[3] University of T

-Translation

Authentication

Martín Abadi*
Bell Labs Research
Lucent Technologies

Secu

of Object-C

o Protected

Marco Patrignani,

iMinds-DistriNet, D

{first

Pierce[4]

sylvania

L Module

and Dave Clar

Local Memory via Laye

Julian Rathke
University of Southampton

Corin Pitcher

ia Multi-Language Semantics *

Translation

Amal Ahmed

Secure Compilation to Protected Module Architectures

on and Raoul Strackx and Bart Jacobs, i

and iMinds-D

On Modular and Fully-Abstract Compil

Marco Pat

MPI-SW

Matthias Blume
Google
blume@google.com

Marco Patrig
Dept. Comput
and Dave C

Fully Abstract Compilation via Universal Embedding *

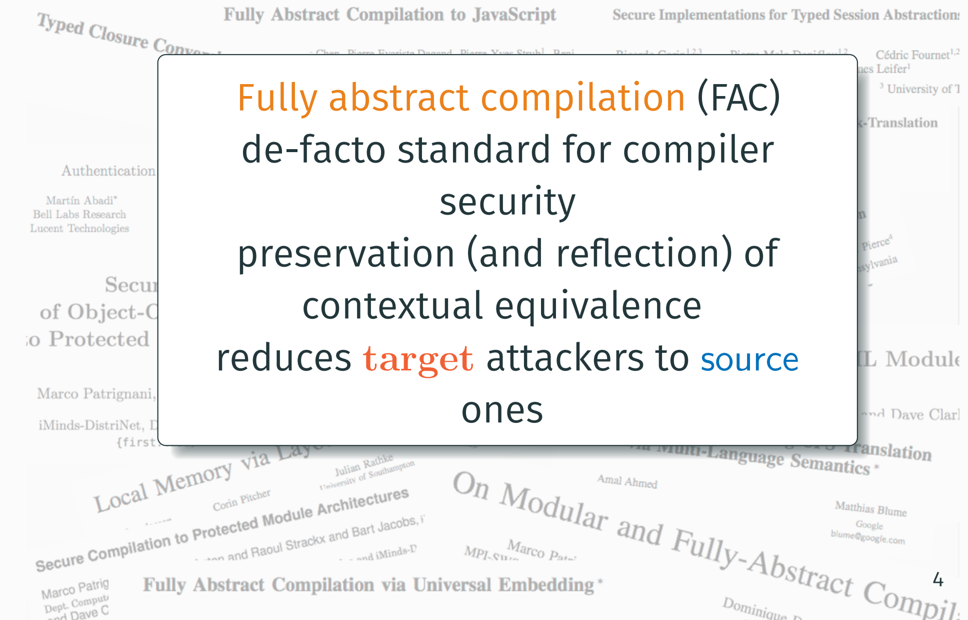
Dominique

Fully abstract compilation (FAC)

de-facto standard for compiler security

preservation (and reflection) of contextual equivalence

Background



Fully abstract compilation (FAC)

de-facto standard for compiler security

preservation (and reflection) of contextual equivalence

reduces target attackers to source ones

Shortcomings for FAC

Shortcomings for FAC

- inefficient code (memory consumption and runtime checks)

Shortcomings for FAC

- inefficient code (memory consumption and runtime checks)
- poor support for multithreaded programs

Shortcomings for FAC

- inefficient code (memory consumption and runtime checks)
- poor support for multithreaded programs
- complex proofs

What do we Want?

- security-aware criteria
- efficient compiled code
- more manageable proofs

Robust Compilation Lattice

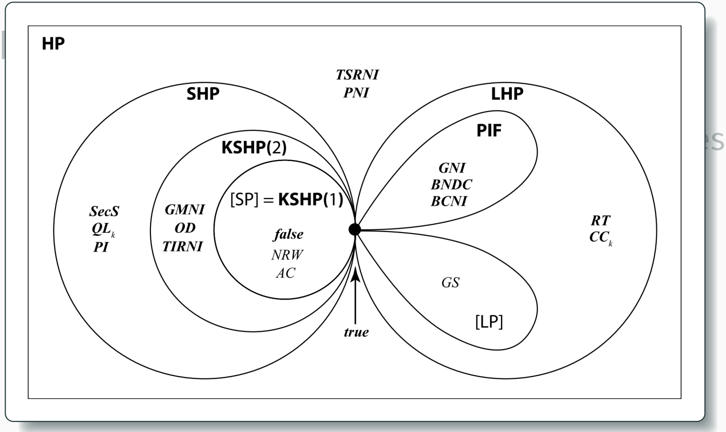
Robust Compilation Lattice (RCL)

- based on hyperproperties (HP)

Robust Compilation Lattice (RCL)

- based on hyperproperties (HP)
  - capture all security properties
  - are organised in subclasses for expressiveness

Robust Compilation Lattice (RCL)



ess

Robust Compilation Lattice (RCL)

- based on hyperproperties (HP)
    - capture all security properties
    - are organised in subclasses for expressiveness

- higher notions are stronger

Robust Compilation Lattice (RCL)

- based on hyperproperties (HP)
  - capture all security properties
  - are organised in subclasses for expressiveness
- higher notions are stronger
  - and trickier to achieve

Robust Compilation Lattice (RCL)

- based on hyperproperties (HP)
  - capture all security properties
  - are organised in subclasses for expressiveness

- higher notions are stronger
  - and trickier to achieve
- each notion comes in two flavours

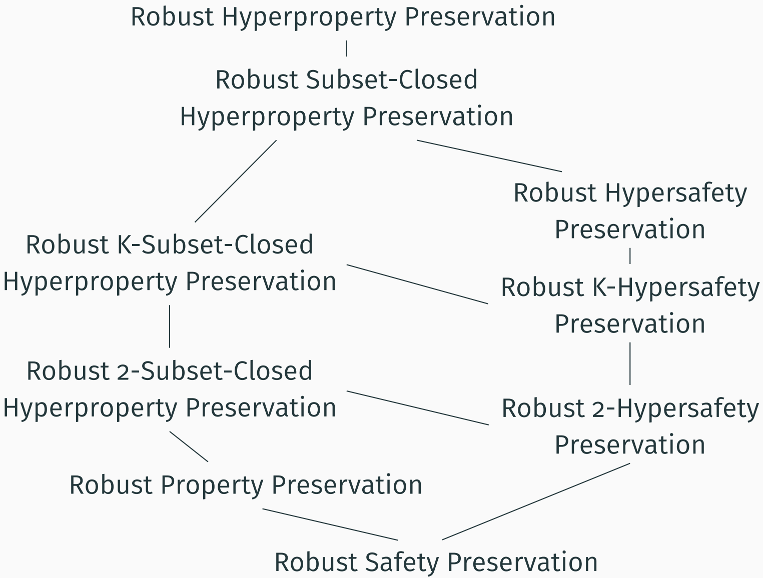
Robust Compilation Lattice (RCL)

- based on hyperproperties (HP)
    - capture all security properties
    - are organised in subclasses for expressiveness
- higher notions are stronger
    - and trickier to achieve
- each notion comes in two flavours
    - one with clear HP correspondence
    - one for simpler proofs

Notation

- $\mathsf{C}, \mathbf{C}$: components of S and $\mathbf{T}$
- $\mathbb{C}\cdot, \mathbb{C}\cdot$: contexts
- $\mathbb{C}\left[\mathsf{C}\right], \mathbb{C}\left[\mathbf{C}\right]$: whole programs
- $[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}} : \mathsf{C} \to \mathbf{C}$: compiler from S to $\mathbf{T}$
- β, β: traces (possibly infinite), I/O with an environment
- $\mathtt{Behav}\left(\mathsf{P}\right)$: set of traces of P
- π, π: prefix (finite)
- $<$: prefixing
- $\approx : \mathsf{sth} \times \mathbf{sth}$: cross-language relation

Robust Compilation Lattice



Robust Hyperproperty Preservation

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

Robust Property Preservation

Robust Safety Preservation

Robust Hyperproperty Preservation

Definition (RHP**)**

$$\begin{aligned}
[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}} \in \mathrm{RHP} \overset{\text{def}}{=} {} & \forall \mathsf{C}, \mathtt{H}, \mathbf{H}. \\
\text{if} \quad & (\forall \mathbb{C}.\mathtt{Behav}\,(\mathbb{C}\,[\mathsf{C}]) \in \mathtt{H}) \\
\text{and} \quad & \mathtt{H} \approx_{\mathtt{H}} \mathbf{H} \\
\text{then} \quad & \left(\forall \mathbb{C}.\mathtt{Behav}\left(\mathbb{C}\left[[\![\mathsf{C}]\!]^{\mathsf{S}}_{\mathbf{T}}\right]\right) \in \mathbf{H}\right)
\end{aligned}$$

Robust Hyperproperty Preservation

Definition (RHP)

$$
\begin{aligned}
[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{RHP} \overset{\mathsf{def}}{=} \forall \mathsf{C}, \mathtt{H}, \mathtt{H}. & \\
\text{if} \quad & (\forall \mathbb{C}.\mathtt{Behav}(\mathbb{C}[\mathsf{C}]) \in \mathtt{H}) \\
\text{and} \quad & \mathtt{H} \approx_{\mathtt{H}} \mathtt{H} \\
\text{then} \quad & \left(\forall \mathbb{C}.\mathtt{Behav}\left(\mathbb{C}\left[[\![\mathsf{C}]\!]^{\mathsf{S}}_{\mathbf{T}}\right]\right) \in \mathtt{H}\right)
\end{aligned}
$$

Robust Hyperproperty Preservation

Definition (RHP)

$$
\begin{aligned}
[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}} \in \mathrm{RHP} \overset{\text{def}}{=} \forall \mathsf{C}, \mathtt{H}, \mathbf{H}.&\\
\text{if}\ \ & (\forall \mathbb{C}.\mathtt{Behav}\,(\mathbb{C}\,[\mathsf{C}]) \in \mathtt{H})\\
\text{and}\ \ & \mathtt{H} \approx_{\mathtt{H}} \mathbf{H}\\
\text{then}\ \ & \left(\forall \mathbb{C}.\mathtt{Behav}\left(\mathbb{C}\left[[\![\mathsf{C}]\!]^{\mathsf{S}}_{\mathbf{T}}\right]\right) \in \mathbf{H}\right)
\end{aligned}
$$

Hyperproperty Robust Compilation

Definition (HRC)

$$
\begin{aligned}
[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{HRC} \overset{\text{def}}{=} \forall \mathsf{C}, \mathbb{C}. \exists \mathbb{C}. \forall \beta, \beta. \beta \approx_{\beta} \beta \\
\beta \in \mathtt{Behav}\left(\mathbb{C}\left[[\![\mathsf{C}]\!]^{\mathsf{S}}_{\mathbf{T}}\right]\right) \\
\iff \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\mathsf{C}\right]\right)
\end{aligned}
$$

Hyperproperty Robust Compilation

Definition (HRC)

$$\llbracket\cdot\rrbracket^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{HRC} \overset{\mathsf{def}}{=} \forall \mathsf{C}, \mathbb{C}. \exists \mathbb{C}. \forall \beta, \beta. \beta \approx_{\beta} \beta$$

$$\beta \in \mathtt{Behav}\left(\mathbb{C}\left[\llbracket\mathsf{C}\rrbracket^{\mathsf{S}}_{\mathbf{T}}\right]\right)$$

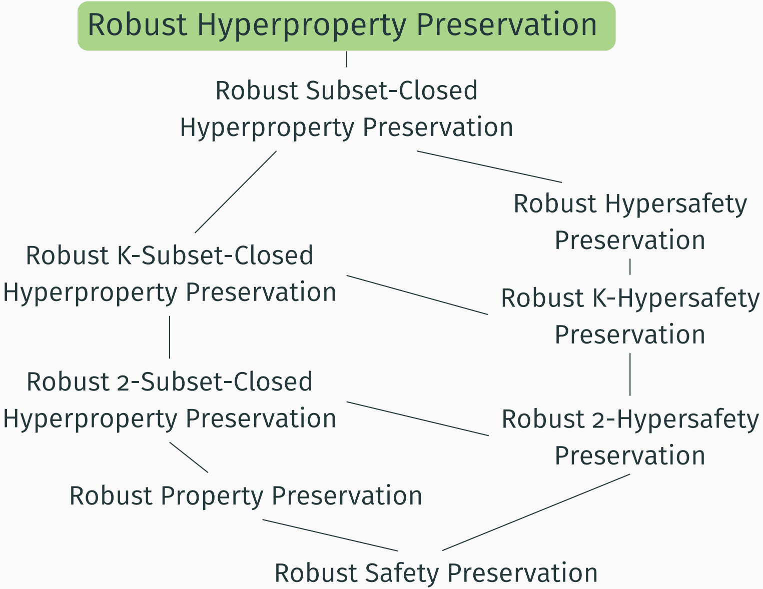
$$\iff \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\mathsf{C}\right]\right)$$

Hyperproperty Robust Compilation

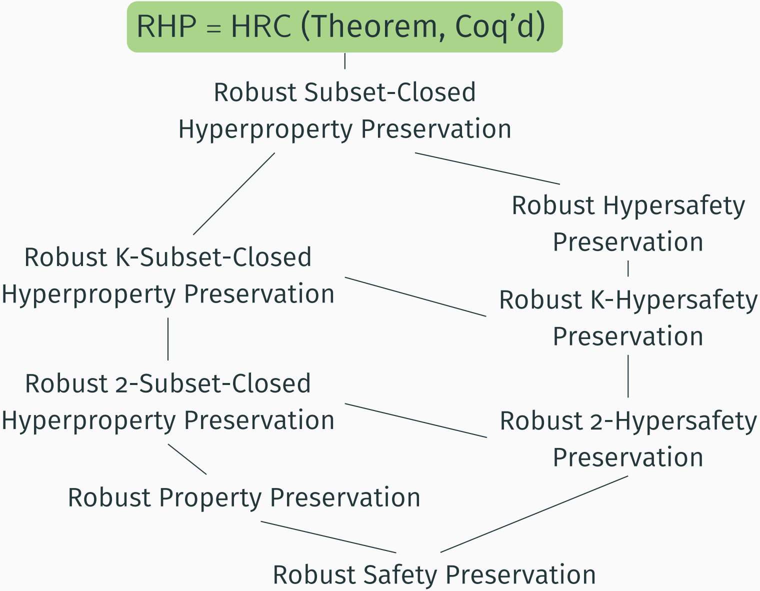
Definition (HRC)

$$
\begin{aligned}
[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{HRC} \overset{\text{def}}{=} \forall \mathsf{C}, \mathbb{C}. \exists \mathbb{C}. \forall \beta, \beta. \beta \approx_{\beta} \beta \\
\beta \in \mathtt{Behav}\left(\mathbb{C}\left[[\![\mathsf{C}]\!]^{\mathsf{S}}_{\mathbf{T}}\right]\right) \\
\iff \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\mathsf{C}\right]\right)
\end{aligned}
$$

Robust Compilation Lattice


Robust Hyperproperty Preservation
Robust Subset-Closed Hyperproperty Preservation
Robust Hypersafety Preservation
Robust K-Subset-Closed Hyperproperty Preservation
Robust K-Hypersafety Preservation
Robust 2-Subset-Closed Hyperproperty Preservation
Robust 2-Hypersafety Preservation
Robust Property Preservation
Robust Safety Preservation

Robust Compilation Lattice



RHP = HRC (Theorem, Coq’d)

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

Robust Property Preservation

Robust Safety Preservation

RPP: Robust Property Preservation

Definition (RPP)

$$
\begin{aligned}
[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}} \in \mathrm{RPP} \overset{\text{def}}{=} {} & \forall \mathsf{C}, \mathsf{P}, \mathbf{P}. \\
\text{if} \quad & \left(\forall \mathbb{C}.\mathtt{Behav}\left(\mathbb{C}\left[\mathsf{C}\right]\right) \subseteq \mathsf{P}\right) \\
\text{and} \quad & \mathsf{P} \approx_{\mathtt{H}} \mathbf{P} \\
\text{then} \quad & \left(\forall \mathbb{C}.\mathtt{Behav}\left(\mathbb{C}\left[[\![\mathsf{C}]\!]^{\mathsf{S}}_{\mathbf{T}}\right]\right) \subseteq \mathbf{P}\right)
\end{aligned}
$$

RC: Robust Compilation

Definition (RC)

$$\llbracket\cdot\rrbracket^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{RC} \overset{\mathsf{def}}{=} \forall \mathbb{C}, \mathsf{C}, \beta. \exists \mathbb{C}, \beta. \beta \approx_{\beta} \beta$$

$$\text{if} \quad \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\llbracket\mathsf{C}\rrbracket^{\mathsf{S}}_{\mathbf{T}}\right]\right)$$

$$\text{then} \quad \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\mathsf{C}\right]\right)$$

RC: Robust Compilation

Definition (RC)

$$\llbracket\cdot\rrbracket^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{RC} \overset{\mathsf{def}}{=} \forall \mathbb{C}, \mathsf{C}, \beta . \exists \mathbb{C}, \beta . \beta \approx_{\beta} \beta$$

$$\text{if} \quad \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\llbracket \mathsf{C} \rrbracket^{\mathsf{S}}_{\mathbf{T}}\right]\right)$$

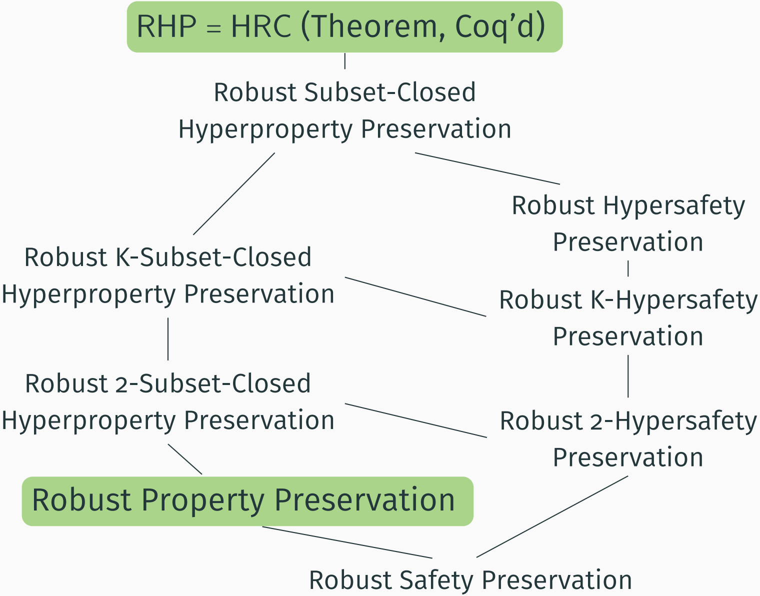
$$\text{then} \quad \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\mathsf{C}\right]\right)$$

RC: Robust Compilation

Definition (RC)

$$
\begin{aligned}
\llbracket\cdot\rrbracket^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{RC} \overset{\text{def}}{=} \forall \mathbb{C}, \mathsf{C}, \beta. \exists \mathbb{C}, \beta. \beta \approx_{\beta} \beta \\
\text{if} \quad \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\llbracket \mathsf{C} \rrbracket^{\mathsf{S}}_{\mathbf{T}}\right]\right) \\
\text{then} \quad \beta \in \mathtt{Behav}\left(\mathbb{C}\left[\mathsf{C}\right]\right)
\end{aligned}
$$

Robust Compilation Lattice



RHP = HRC (Theorem, Coq’d)

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

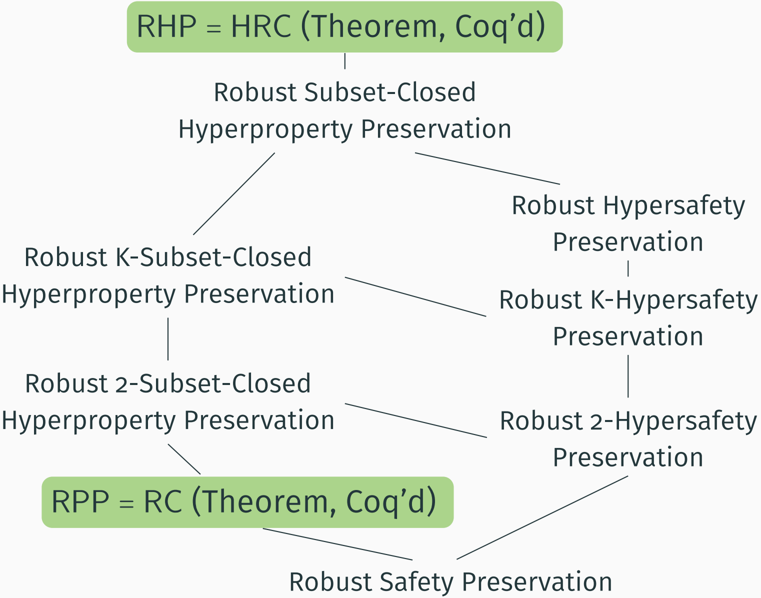
Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

Robust Property Preservation

Robust Safety Preservation

Robust Compilation Lattice



RHP = HRC (Theorem, Coq'd)

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

RPP = RC (Theorem, Coq'd)

Robust Safety Preservation

Robust Safety Property Preservation

Definition (RSPP**)**

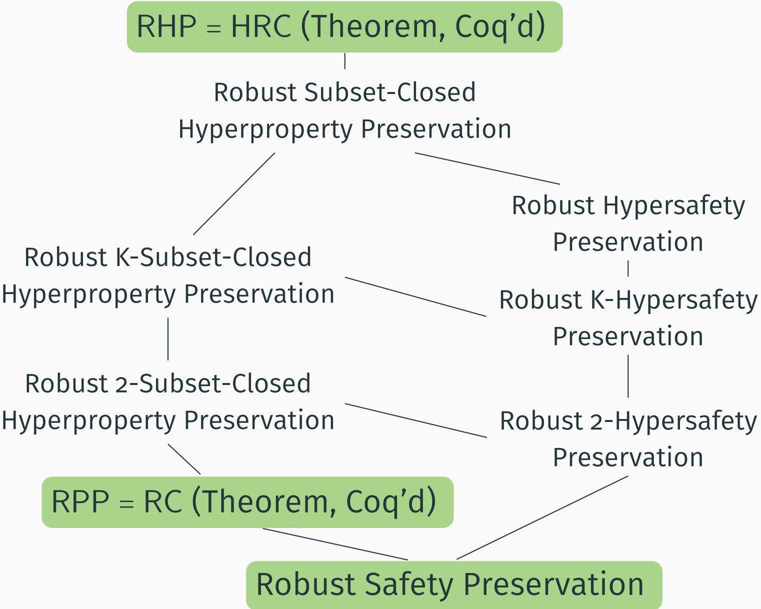
$$
\begin{aligned}
[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{RSPP} \overset{\text{def}}{=} \forall \mathsf{C}, \mathtt{P} \in \mathsf{SP}, \mathtt{P} \in \mathbf{SP}&. \\
\text{if} \quad & (\forall \mathbb{C}.\mathtt{Behav}\,(\mathbb{C}\,[\mathsf{C}]) \subseteq \mathtt{P}) \\
\text{and} \quad & \mathtt{P} \approx_{\mathtt{H}} \mathtt{P} \\
\text{then} \quad & \left(\forall \mathbb{C}.\mathtt{Behav}\left(\mathbb{C}\left[[\![\mathsf{C}]\!]^{\mathsf{S}}_{\mathbf{T}}\right]\right) \subseteq \mathtt{P}\right)
\end{aligned}
$$

Robust Safety Compilation

Definition (SRC)

$$
\begin{aligned}
\llbracket\cdot\rrbracket^{\mathsf{S}}_{\mathbf{T}} \in \mathsf{RC} \overset{\mathsf{def}}{=} \forall \mathbb{C}, \mathsf{C}, \pi. \exists \mathbb{C}, \pi. \pi \approx_{\beta} \pi \\
\text{if } \quad \pi \prec \mathtt{Behav}\left(\mathbb{C}\left[\llbracket \mathsf{C} \rrbracket^{\mathsf{S}}_{\mathbf{T}}\right]\right) \\
\text{then } \quad \pi \prec \mathtt{Behav}\left(\mathbb{C}\left[\mathsf{C}\right]\right)
\end{aligned}
$$

Robust Compilation Lattice



RHP = HRC (Theorem, Coq’d)

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

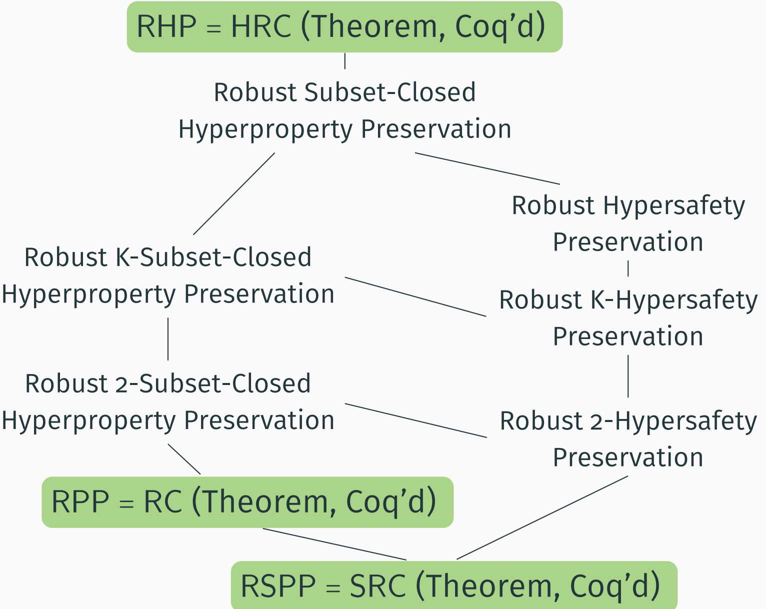
Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

RPP = RC (Theorem, Coq’d)

Robust Safety Preservation

Robust Compilation Lattice



RHP = HRC (Theorem, Coq'd)

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

Robust 2-Subset-Closed Hyperproperty Preservation

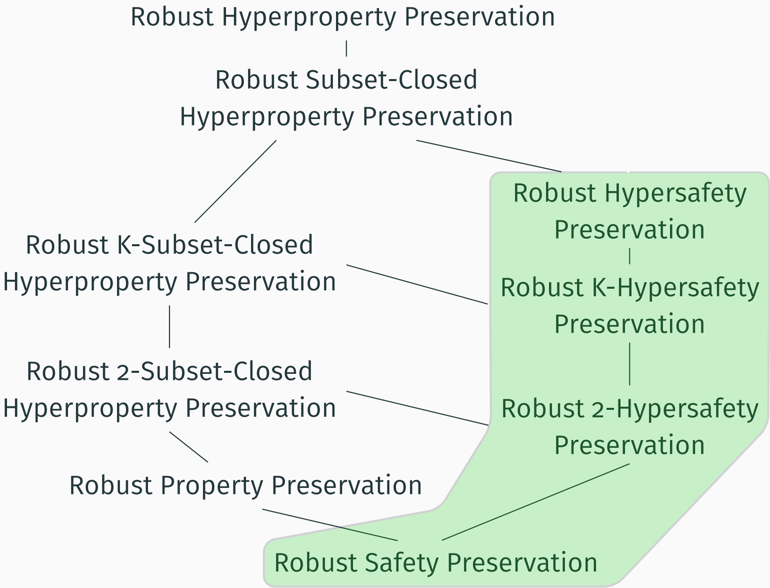
Robust 2-Hypersafety Preservation

RPP = RC (Theorem, Coq'd)

RSPP = SRC (Theorem, Coq'd)

Proof Techniques

Proof Techniques



Robust Hyperproperty Preservation

Robust Subset-Closed Hyperproperty Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust 2-Subset-Closed Hyperproperty Preservation

Robust Property Preservation

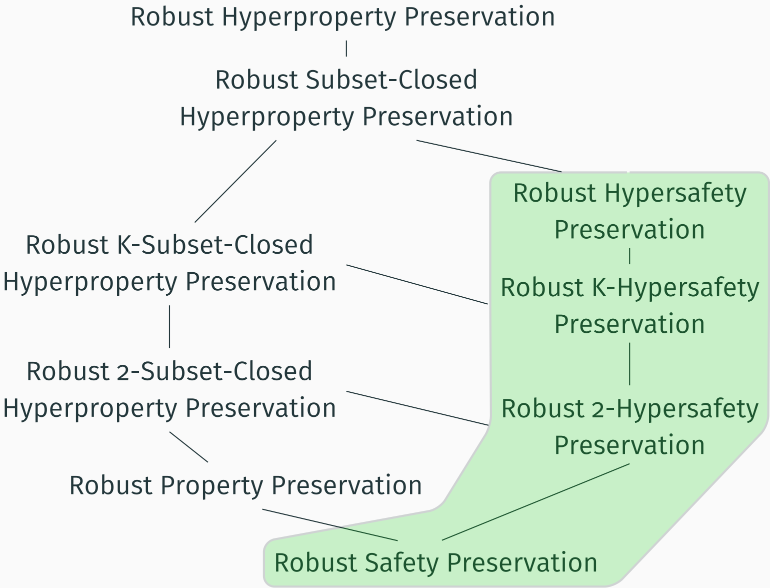
Robust Hypersafety Preservation

Robust K-Hypersafety Preservation

Robust 2-Hypersafety Preservation

Robust Safety Preservation

Proof Techniques



Robust Hyperproperty Preservation

Robust Subset-Closed Hyperproperty Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust 2-Subset-Closed Hyperproperty Preservation

Robust Property Preservation

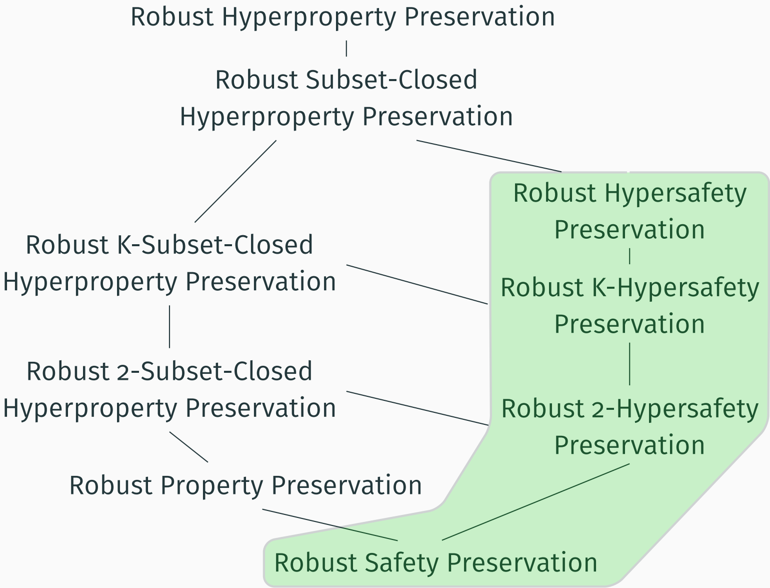
Robust Hypersafety Preservation

Robust K-Hypersafety Preservation

Robust 2-Hypersafety Preservation

Robust Safety Preservation

Proof Techniques



Robust Hyperproperty Preservation

Robust Subset-Closed Hyperproperty Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust 2-Subset-Closed Hyperproperty Preservation

Robust Property Preservation

Robust Hypersafety Preservation

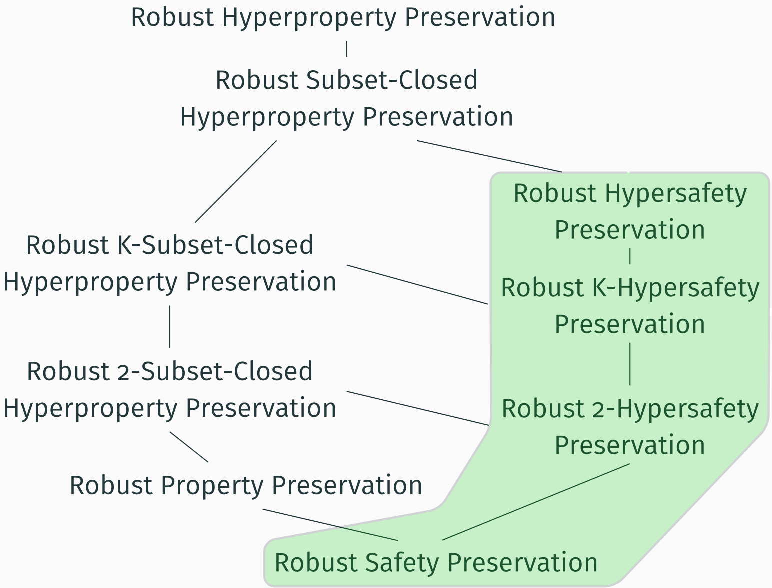
Robust K-Hypersafety Preservation

Robust 2-Hypersafety Preservation

Robust Safety Preservation

Where is Fully Abstract Compilation?

Where is Fully Abstract Compilation?



Robust Hyperproperty Preservation

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

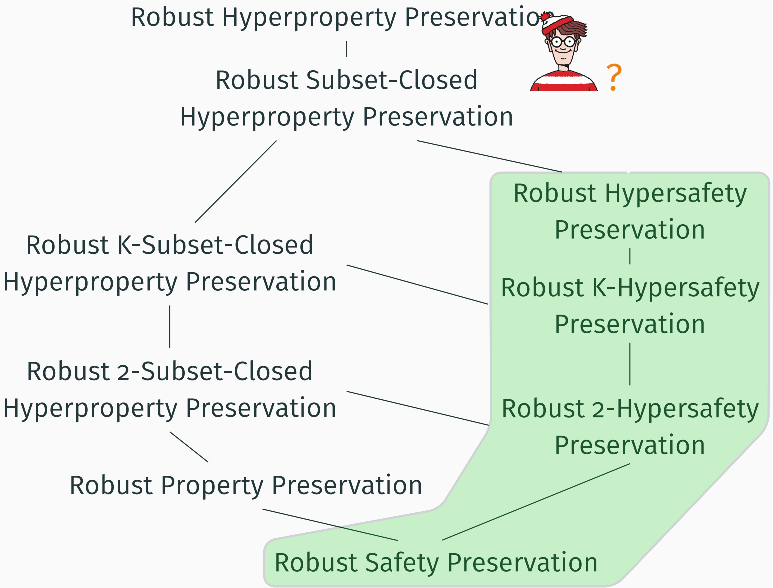
Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

Robust Property Preservation

Robust Safety Preservation

Where is Fully Abstract Compilation?



Robust Hyperproperty Preservation

Robust Subset-Closed Hyperproperty Preservation

?

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

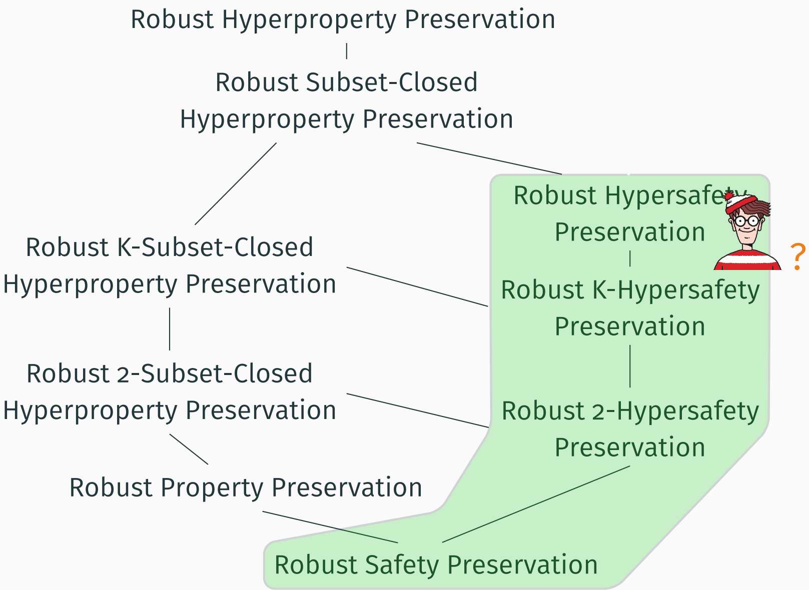
Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

Robust Property Preservation

Robust Safety Preservation

Where is Fully Abstract Compilation?



Robust Hyperproperty Preservation

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

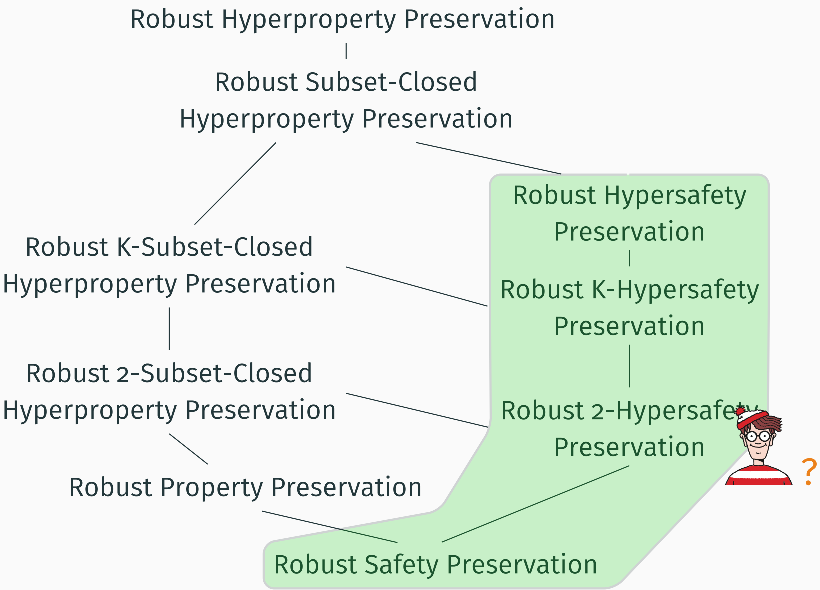
Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

Robust Property Preservation

Robust Safety Preservation

Where is Fully Abstract Compilation?



Robust Hyperproperty Preservation

Robust Subset-Closed Hyperproperty Preservation

Robust Hypersafety Preservation

Robust K-Subset-Closed Hyperproperty Preservation

Robust K-Hypersafety Preservation

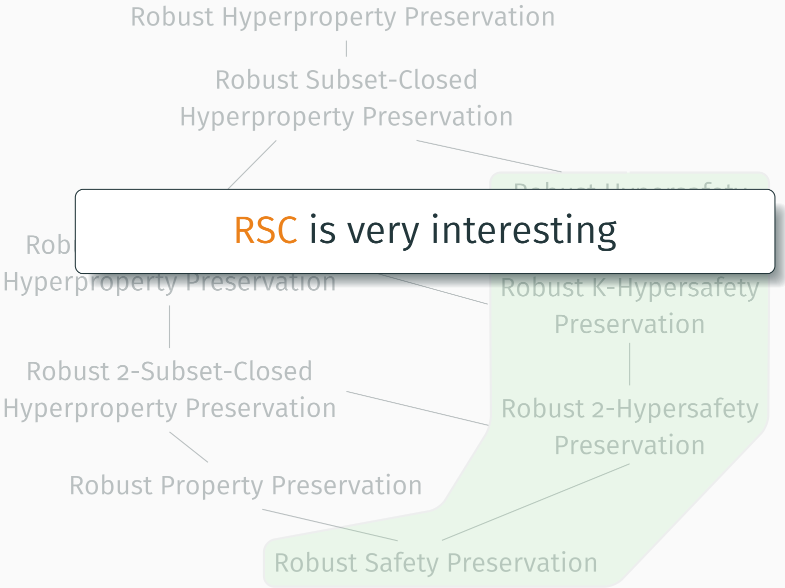
Robust 2-Subset-Closed Hyperproperty Preservation

Robust 2-Hypersafety Preservation

Robust Property Preservation

Robust Safety Preservation

Where is Fully Abstract Compilation?



Robust Hyperproperty Preservation
Robust Subset-Closed Hyperproperty Preservation
Robust Hypersafety
Rob
Hyperproperty Preservation
Robust K-Hypersafety Preservation
Robust 2-Subset-Closed Hyperproperty Preservation
Robust 2-Hypersafety Preservation
Robust Property Preservation
Robust Safety Preservation

RSC is very interesting

Robustly-Safe Compilation

Why RSC?

Why RSC?

- integrity
- weak secrecy
- taint tracking

Why RSC?

- integrity
- weak secrecy
- taint tracking (approx non-interference)

Safety how?

- Monitors M, $\mathbf{M}$ enforce safety

Safety how?

- Monitors M, M enforce safety
- Equivalent to classic $assume/assert$

Safety how?

- Monitors M, M enforce safety
- Equivalent to classic $assume/assert$
- Capture untrusted code scenario

Safety how?

- Monitors M, M enforce safety
- Equivalent to classic $assume/assert$
- Capture untrusted code scenario
- Check heap conditions

Safety how?

(Abstract-monitor)

$$\frac{M = (s, \rightsquigarrow, s_0, l, s_c) \qquad s_c, H\Big|_l \rightsquigarrow s_f \qquad M' = (s, \rightsquigarrow, s_0, l, s_f)}{M, H \triangleright \mathit{monitor} \;\rightarrow\; M', H \triangleright \mathit{skip}}$$

Safety how?

(Abstract-monitor)

$$\frac{M = (s, \rightsquigarrow, s_0, l, s_c) \qquad s_c, H\Big|_l \rightsquigarrow s_f \\ M' = (s, \rightsquigarrow, s_0, l, s_f)}{M, H \triangleright monitor \;\rightarrow\; M', H \triangleright skip}$$

(Abstract-monitor-fail)

$$\frac{M = (s, \rightsquigarrow, s_0, l, s_c) \qquad s_c, H\Big|_l \not\rightsquigarrow \_}{M, H \triangleright monitor \;\rightarrow\; fail}$$

Robust Safety

$$\vdash C, M : \mathit{safe} \overset{\text{def}}{=} \text{if } \vdash C : \mathit{whole}$$

$$\text{then } C_0, M \not\rightarrow^* \mathit{fail}$$

$$\vdash A : \mathit{attacker} \overset{\text{def}}{=} \text{no } \mathit{monitor} \text{ inside } A$$

$$\vdash C, M : \mathit{rs} \overset{\text{def}}{=} \text{ if } \vdash A : \mathit{attacker}$$

$$\text{then } \vdash A\left[C\right], M : \mathit{safe}$$

Robust Safety

Definition (RSC)

$$\vdash [\![\cdot]\!]^{S}_{\mathbf{T}} : \mathsf{SRC} \overset{\text{def}}{=} \text{if } \mathsf{M} \approx \mathbf{M}$$

$$\text{and } \vdash \mathsf{C}, \mathsf{M} : \mathsf{rs}$$

$$\text{then } \vdash [\![\mathsf{C}]\!]^{S}_{\mathbf{T}}, \mathbf{M} : \mathbf{rs}$$

$\text{then} \vdash A[C], M : safe$

Languages

- S and $\mathbf{T}$ are while languages

Languages

- S and $\mathbf{T}$ are while languages
- both are untyped

Languages

- S and $\mathbf{T}$ are while languages
- both are untyped
- both have M, $\mathbf{M}$ and monitor instructions

Languages

- S and **T** are while languages
- both are untyped
- both have M, **M** and monitor instructions
- S has an abstract heap

Languages

- S and T are while languages
- both are untyped
- I s
- S

(E-s-alloc)

$$\frac{H \triangleright e \twoheadrightarrow v \qquad \ell \notin \mathtt{dom}(H)}{C, H \triangleright \text{let } x = \text{new } e \text{ in } s \xrightarrow{\epsilon} C, H; \ell \mapsto v \triangleright s[\ell/x]}$$

Languages

- S and $\mathbf{T}$ are while languages
- both are untyped
- both have M, $\mathbf{M}$ and monitor instructions
- S has an abstract heap

- $\mathbf{T}$ has a concrete heap, abstract capabilities (capabilities / sealing / PMA)

Langu

(E-t-new)

$$\frac{\mathbf{H} = \mathbf{H}_1; \mathbf{n} \mapsto (\mathbf{v}, \eta) \qquad \mathbf{H} \rhd \mathbf{e} \twoheadrightarrow \mathbf{v} \\ \mathbf{H}' = \mathbf{H}; \mathbf{n} + 1 \mapsto \mathbf{v} : \bot}{\mathbf{C}, \mathbf{H} \rhd \mathbf{let}\ \mathbf{x} = \mathbf{new}\ \mathbf{e}\ \mathbf{in}\ \mathbf{s} \\ \xrightarrow{\epsilon} \mathbf{C}, \mathbf{H}' \rhd \mathbf{s}[\mathbf{n} + 1/\mathbf{x}]}$$

Langu

(E-t-new)

$$\frac{\begin{array}{c} H = H_1; n \mapsto (v, \eta) \qquad H \triangleright e \twoheadrightarrow v \\ H' = H; n + 1 \mapsto v : \bot \end{array}}{\begin{array}{c} C, H \triangleright \mathbf{let}\ x = \mathbf{new}\ e\ \mathbf{in}\ s \\ \xrightarrow{\epsilon} C, H' \triangleright s[n + 1/x] \end{array}}$$

(E-t-hide)

$$\frac{\begin{array}{c} H \triangleright e \twoheadrightarrow n \qquad k \notin \mathtt{dom}(H) \\ H = H_1; n \mapsto v : \bot; H_2 \\ H' = H_1; n \mapsto v : k; H_2; k \end{array}}{\begin{array}{c} C, H \triangleright \mathbf{let}\ x = \mathbf{hide}\ e\ \mathbf{in}\ s \\ \xrightarrow{\epsilon} C, H' \triangleright s[k/x] \end{array}}$$

Languages

- S and $\mathbf{T}$ are while languages
- both are untyped
- both have M, $\mathbf{M}$ and monitor instructions
- S has an abstract heap

- $\mathbf{T}$ has a concrete heap, abstract capabilities (capabilities / sealing / PMA)

- $\mathbf{!e\ with\ e}$ $\qquad$ $\mathbf{x := e\ with\ e}$

Compiler

- identity except for

$$\left[\!\left[\begin{array}{l}\mathsf{let\ x = new\ e}\\ \quad\mathsf{in\ s}\end{array}\right]\!\right]^{\mathsf{S}}_{\mathbf{T}} = \begin{array}{l}\mathbf{let\ x_{loc} = new\ } [\![\mathsf{e}]\!]^{\mathsf{S}}_{\mathbf{T}}\ \mathbf{in}\\ \quad\mathbf{let\ x_{cap} = hide\ x_{loc}\ in}\\ \quad\quad\mathbf{let\ x = \langle x_{loc}, x_{cap}\rangle\ in\ } [\![\mathsf{s}]\!]^{\mathsf{S}}_{\mathbf{T}}\end{array}$$

Proof Sketch

- if $\mathsf{M}, \varnothing \triangleright \mathsf{A}\left[\mathsf{C}\right] \overset{\overline{\alpha}}{\Longrightarrow} \mathsf{M}', \mathsf{H} \triangleright \mathsf{A}\left[\mathsf{monitor}\right]$ then $\mathsf{M}', \mathsf{H} \triangleright \mathsf{A}\left[\mathsf{monitor}\right] \xrightarrow{\epsilon} \mathsf{M}', \mathsf{H} \triangleright \mathsf{A}\left[\mathsf{skip}\right]$
- $\mathbf{M}, \varnothing \triangleright \mathbf{A}\left[\llbracket \mathsf{C} \rrbracket^{\mathsf{S}}_{\mathbf{T}}\right] \overset{\overline{\alpha}}{\Longrightarrow} \mathbf{M}', \mathbf{H} \triangleright \mathbf{A}\left[\mathbf{monitor}\right]$

and we need to prove that

- $\mathbf{M}', \mathbf{H} \triangleright \mathbf{A}\left[\mathbf{monitor}\right] \xrightarrow{\epsilon} \mathbf{M}', \mathbf{H} \triangleright \mathbf{A}\left[\mathbf{skip}\right]$

Proof Sketch

- if $\mathsf{M}, \varnothing \triangleright \mathsf{A}\left[\mathsf{C}\right] \overset{\overline{\alpha}}{\Longrightarrow} \mathsf{M}', \mathsf{H} \triangleright \mathsf{A}\left[\mathsf{monitor}\right]$ then $\mathsf{M}', \mathsf{H} \triangleright \mathsf{A}\left[\mathsf{monitor}\right] \xrightarrow{\epsilon} \mathsf{M}', \mathsf{H} \triangleright \mathsf{A}\left[\mathsf{skip}\right]$
- $\mathbf{M}, \varnothing \triangleright \mathbf{A}\left[[\![\mathsf{C}]\!]^{\mathsf{S}}_{\mathbf{T}}\right] \overset{\overline{\alpha}}{\Longrightarrow} \mathbf{M}', \mathbf{H} \triangleright \mathbf{A}\left[\mathbf{monitor}\right]$

and we need to prove that

- $\mathbf{M}', \mathbf{H} \triangleright \mathbf{A}\left[\mathbf{monitor}\right] \xrightarrow{\epsilon} \mathbf{M}', \mathbf{H} \triangleright \mathbf{A}\left[\mathbf{skip}\right]$

$\langle\!\langle \cdot \rangle\!\rangle$ takes $\overline{\alpha}$ and returns A

Proof Sketch

Plus:

- no need of injective relation
- no need of FA traces

and

Proof Sketch

Plus:

- no need of injective relation
- no need of FA traces

Minus:

- complex because of fine granularity
- still requires backtranslation

Extension

- Extend S with a RS type system

Extension

- Extend S with a RS type system
- We can statically know which locations the monitor observes (type $\tau \neq \mathsf{UN}$)

Extension

- Extend S with a RS type system
- We can statically know which locations the monitor observes (type $\tau \neq$ UN) high locations

Extension

- Extend S with a RS type system
- We can statically know which locations the monitor observes (type $\tau \neq \mathsf{UN}$) high locations
- $[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}}$ protects only high locations

Extension

- Extend S with a RS type system
- We can statically know which locations the monitor observes (type $\tau \neq \mathsf{UN}$) high locations
- $[\![\cdot]\!]^{\mathsf{S}}_{\mathbf{T}}$ protects only high locations (efficient!)

Cross-language Bisimulation $\Omega \approx \boldsymbol{\Omega}$

S

$\ell_1 \mapsto \mathsf{v}_1 : \tau_1$ $\ell_4 \mapsto \mathsf{v}_4 : \tau_4$

$\ell_2 \mapsto \mathsf{v}_2 : \tau_2$ $\ell_5 \mapsto \mathsf{v}_5 : \tau_5$

$\ell_3 \mapsto \mathsf{v}_3 : \tau_3$ $\ell_6 \mapsto \mathsf{v}_6 : \tau_6$

$\mathbf{T}$

$\mathbf{n_1 \mapsto v_1 : k_1}$ $\mathbf{n_4 \mapsto v_4 : \perp}$

$\mathbf{k_1}$

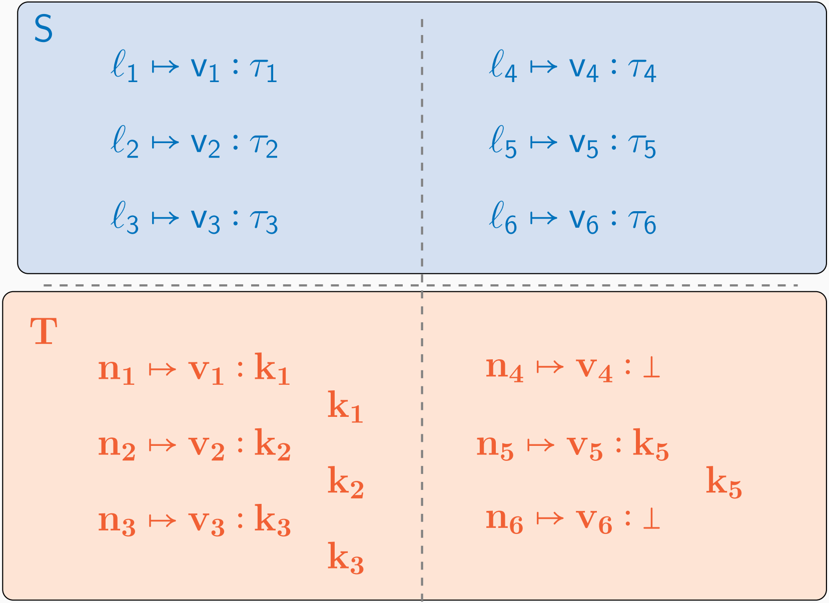
$\mathbf{n_2 \mapsto v_2 : k_2}$ $\mathbf{n_5 \mapsto v_5 : k_5}$

$\mathbf{k_2}$ $\mathbf{k_5}$

$\mathbf{n_3 \mapsto v_3 : k_3}$ $\mathbf{n_6 \mapsto v_6 : \perp}$

$\mathbf{k_3}$

Cross-language Bisimulation $\Omega \approx \Omega$



S

$\ell_1 \mapsto \mathsf{v}_1 : \tau_1$

$\ell_2 \mapsto \mathsf{v}_2 : \tau_2$

$\ell_3 \mapsto \mathsf{v}_3 : \tau_3$

$\ell_4 \mapsto \mathsf{v}_4 : \tau_4$

$\ell_5 \mapsto \mathsf{v}_5 : \tau_5$

$\ell_6 \mapsto \mathsf{v}_6 : \tau_6$

T

$\mathbf{n}_1 \mapsto \mathbf{v}_1 : \mathbf{k}_1$

$\mathbf{k}_1$

$\mathbf{n}_2 \mapsto \mathbf{v}_2 : \mathbf{k}_2$

$\mathbf{k}_2$

$\mathbf{n}_3 \mapsto \mathbf{v}_3 : \mathbf{k}_3$

$\mathbf{k}_3$

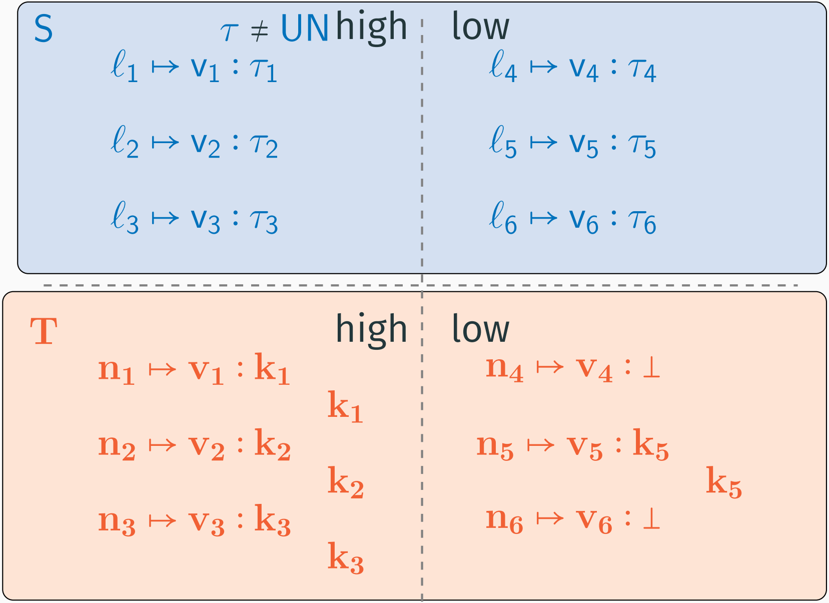
$\mathbf{n}_4 \mapsto \mathbf{v}_4 : \bot$

$\mathbf{n}_5 \mapsto \mathbf{v}_5 : \mathbf{k}_5$

$\mathbf{k}_5$

$\mathbf{n}_6 \mapsto \mathbf{v}_6 : \bot$

Cross-language Bisimulation $\Omega \approx \boldsymbol{\Omega}$



S $\tau \neq \mathsf{UN}$ high | low

$\ell_1 \mapsto \mathsf{v}_1 : \tau_1$

$\ell_2 \mapsto \mathsf{v}_2 : \tau_2$

$\ell_3 \mapsto \mathsf{v}_3 : \tau_3$

$\ell_4 \mapsto \mathsf{v}_4 : \tau_4$

$\ell_5 \mapsto \mathsf{v}_5 : \tau_5$

$\ell_6 \mapsto \mathsf{v}_6 : \tau_6$

T high | low

$\mathbf{n}_1 \mapsto \mathbf{v}_1 : \mathbf{k}_1$

$\mathbf{k}_1$

$\mathbf{n}_2 \mapsto \mathbf{v}_2 : \mathbf{k}_2$

$\mathbf{k}_2$

$\mathbf{n}_3 \mapsto \mathbf{v}_3 : \mathbf{k}_3$

$\mathbf{k}_3$

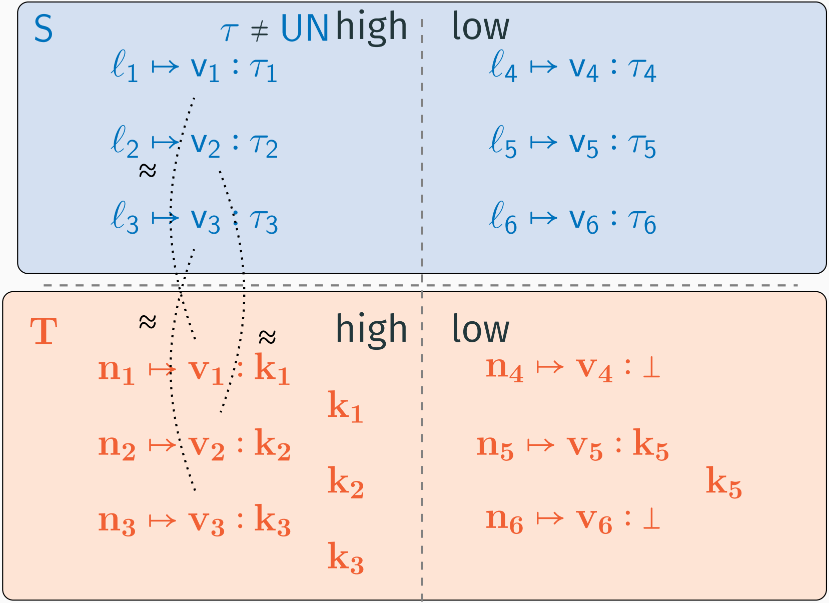
$\mathbf{n}_4 \mapsto \mathbf{v}_4 : \bot$

$\mathbf{n}_5 \mapsto \mathbf{v}_5 : \mathbf{k}_5$

$\mathbf{k}_5$

$\mathbf{n}_6 \mapsto \mathbf{v}_6 : \bot$

Cross-language Bisimulation $\Omega \approx \Omega$



S $\tau \neq$ UN high | low

$\ell_1 \mapsto v_1 : \tau_1$ $\ell_4 \mapsto v_4 : \tau_4$

$\ell_2 \mapsto v_2 : \tau_2$ $\ell_5 \mapsto v_5 : \tau_5$

$\approx$

$\ell_3 \mapsto v_3 : \tau_3$ $\ell_6 \mapsto v_6 : \tau_6$

T $\approx$ $\approx$ high | low

$\mathbf{n_1 \mapsto v_1 : k_1}$ $\mathbf{n_4 \mapsto v_4 : \bot}$

$\mathbf{k_1}$

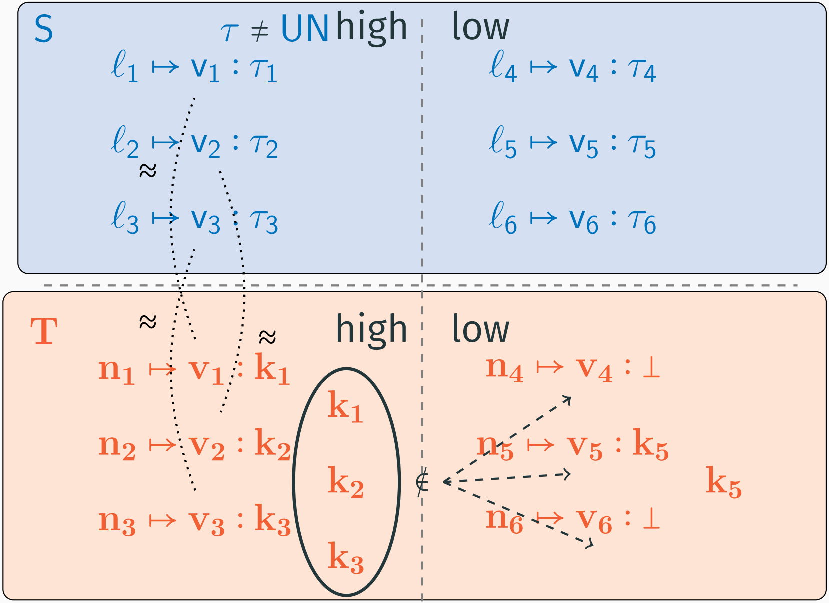
$\mathbf{n_2 \mapsto v_2 : k_2}$ $\mathbf{n_5 \mapsto v_5 : k_5}$

$\mathbf{k_2}$ $\mathbf{k_5}$

$\mathbf{n_3 \mapsto v_3 : k_3}$ $\mathbf{n_6 \mapsto v_6 : \bot}$

$\mathbf{k_3}$

Cross-language Bisimulation $\Omega \approx \Omega$



S $\tau \neq$ UN high | low

$\ell_1 \mapsto v_1 : \tau_1$ $\ell_4 \mapsto v_4 : \tau_4$

$\ell_2 \mapsto v_2 : \tau_2$ $\ell_5 \mapsto v_5 : \tau_5$

$\approx$

$\ell_3 \mapsto v_3 : \tau_3$ $\ell_6 \mapsto v_6 : \tau_6$

T $\approx$ $\approx$ high | low

$n_1 \mapsto v_1 : k_1$ $n_4 \mapsto v_4 : \bot$

k_1

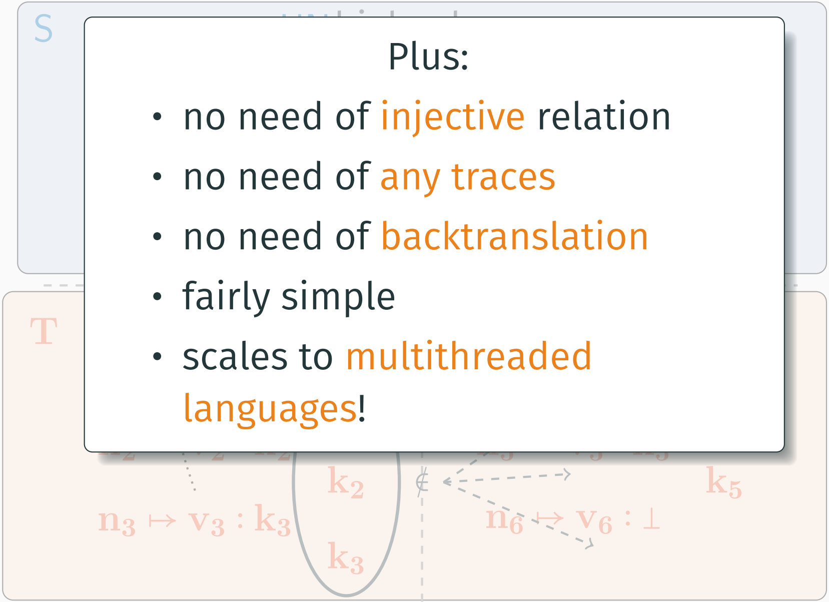
$n_2 \mapsto v_2 : k_2$ $n_5 \mapsto v_5 : k_5$

k_2 $\notin$ k_5

$n_3 \mapsto v_3 : k_3$ $n_6 \mapsto v_6 : \bot$

k_3

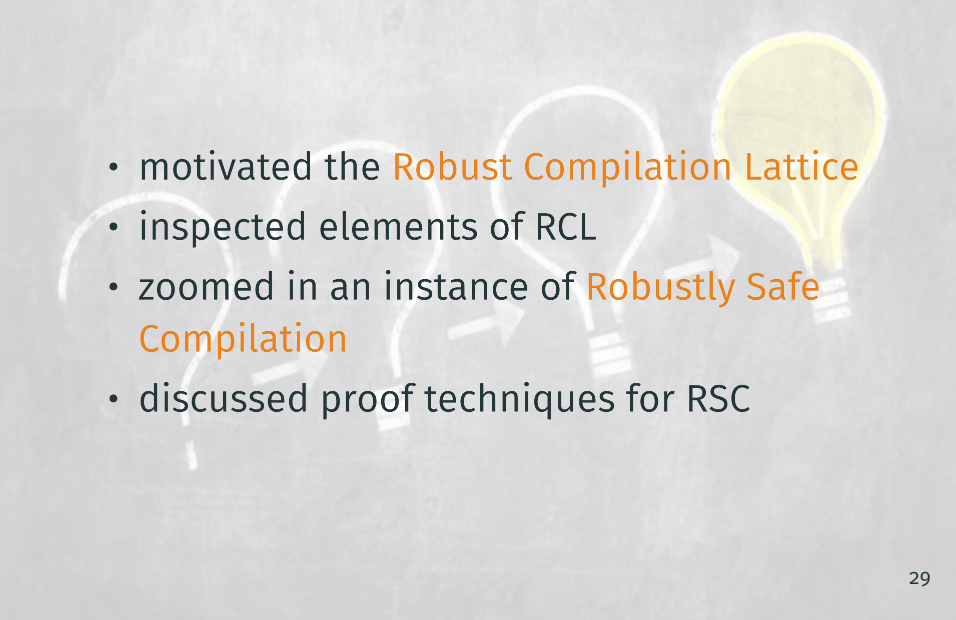
Cross-language Bisimulation $\Omega \approx \Omega$



Plus:

- no need of injective relation
- no need of any traces
- no need of backtranslation
- fairly simple
- scales to multithreaded languages!

Conclusion



- motivated the Robust Compilation Lattice
- inspected elements of RCL
- zoomed in an instance of Robustly Safe Compilation
- discussed proof techniques for RSC

Conclusion