Specifications for Dynamic Enforcement of Relational Program Properties

Max S. New
Northeastern University

Outline

- High level specification and proof architecture for dynamic enforcement.
  - Not novel, but also not standard, we keep reinventing it.
  - Used to show failures of full abstraction in gradual typing.

Preserving Relations

- Can express security properties as refinement relations in the high level language.
  - Noninterference, Full Abstraction

Preserving Relations

- Can express security properties as refinement relations in the high level language.
    - Noninterference, Full Abstraction
    - Secure compilation: preservation of refinement

$$\frac{s_1 \sqsubseteq_{Src} s_2}{[\![s_1]\!] \sqsubseteq_{Tgt} [\![s_2]\!]}$$

Static vs Dynamic Enforcement

Static	Dynamic
interact with…	
• Verified Code • Compiled Code • Any “good” target language code	• Attackers • Unverified code in the target language

Static vs Dynamic Enforcement

Static	Dynamic
Advantages	
Avoid costly checks when linking verified/compiled code	Securely link with untrusted code

Static **AND** Dynamic Enforcement

Static

Complementary

Dynamic

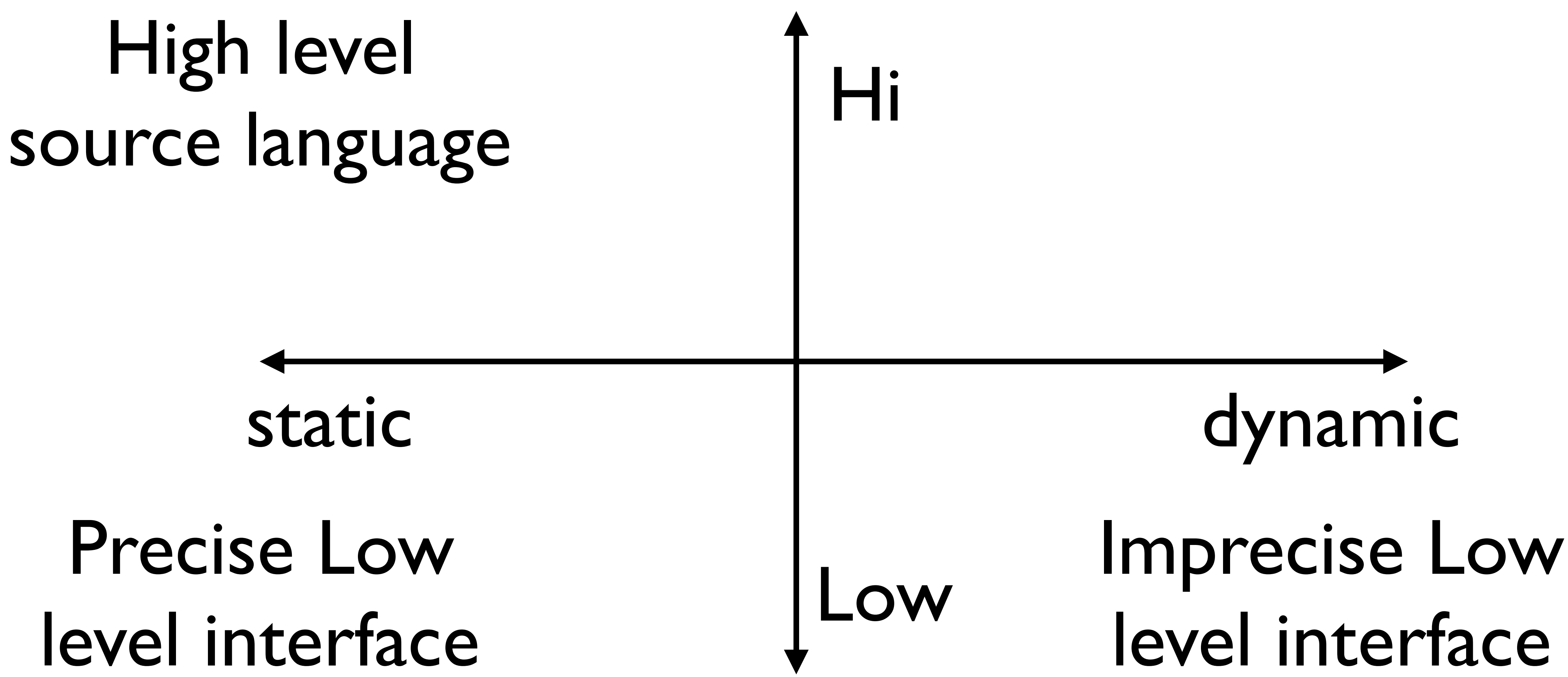
Avoid costly checks when linking verified/ compiled code

Securely link with untrusted code

Static **AND** Dynamic Enforcement

Keep the static vs dynamic typing
flamewars out of secure compilation!

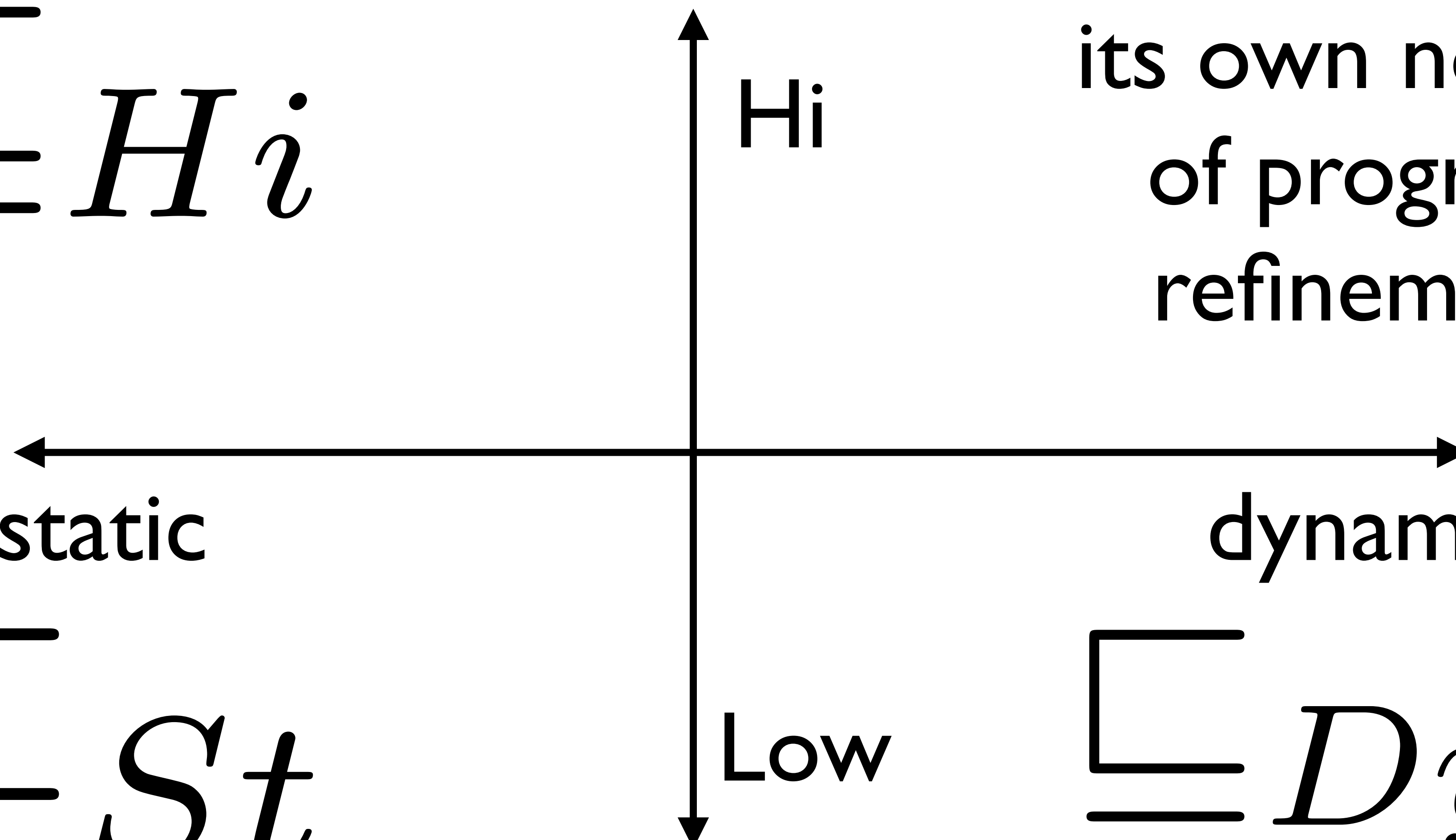
Static **AND** Dynamic Enforcement



Static **AND** Dynamic Enforcement

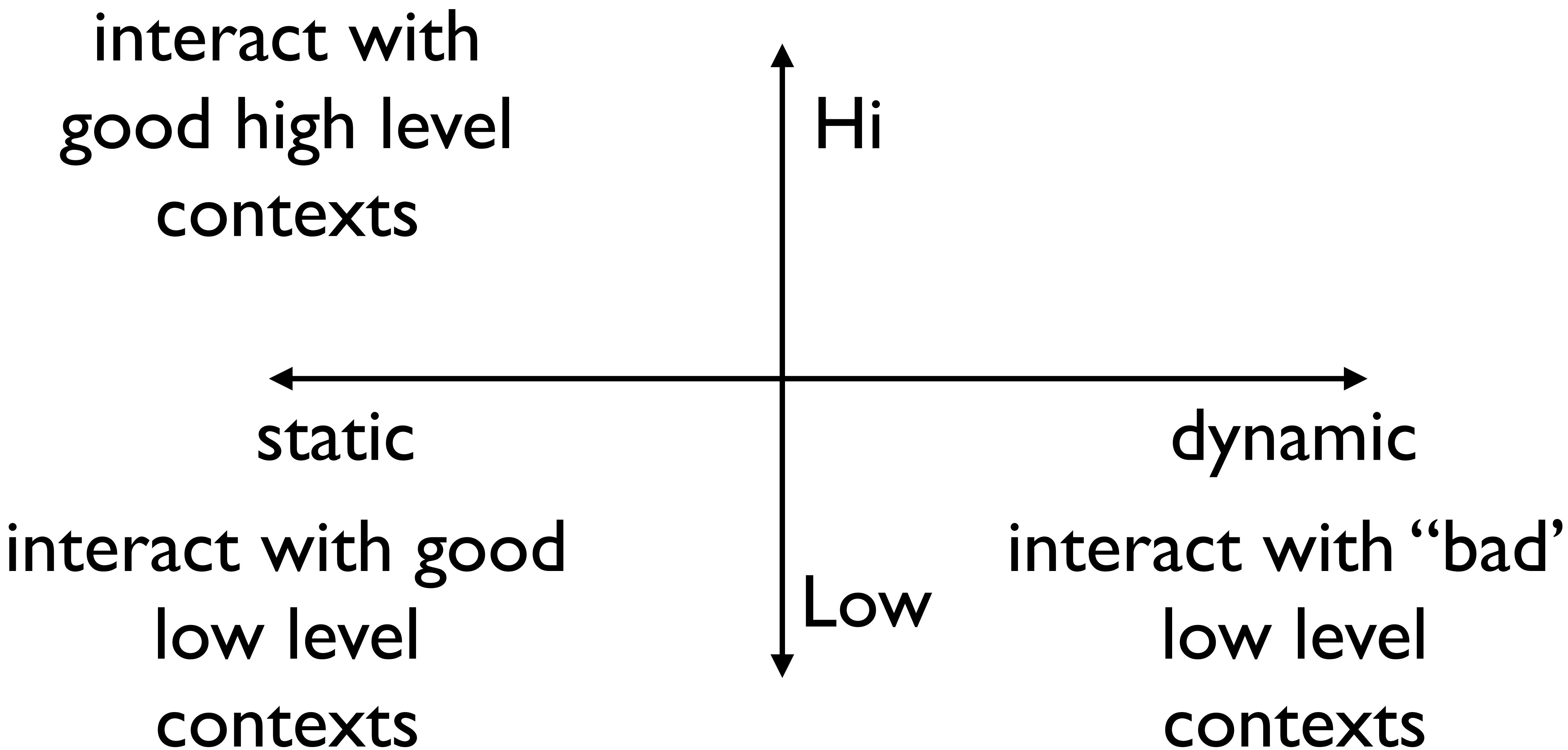
Each comes with its own notion of program refinement

$\sqsubseteq_{Hi}$



$\sqsubseteq_{St}$ $\sqsubseteq_{Dyn}$

Static **AND** Dynamic Enforcement



Static **AND** Dynamic Enforcement

$$h_1 \sqsubseteq_{Src} h_2$$

$$[\![h_1]\!] \sqsubseteq_{St} [\![h_2]\!]$$

$$(\!|h_1|\!) \sqsubseteq_{Dyn} (\!|h_2|\!)$$

Static **AND** Dynamic Enforcement

$$h_1 \sqsubseteq_{Src} h_2$$

Twice the work?

$$\llbracket h_1 \rrbracket \sqsubseteq_{St} \llbracket h_2 \rrbracket \qquad \llparenthesis h_1 \rrparenthesis \sqsubseteq_{Dyn} \llparenthesis h_2 \rrparenthesis$$

Static **AND** Dynamic Enforcement

$$h_1 \sqsubseteq_{Src} h_2$$

Decompose

$$[\![h_1]\!] \sqsubseteq_{St} [\![h_2]\!]$$

$$\mathrm{protect}[\![h_1]\!] \sqsubseteq_{Dyn} \mathrm{protect}[\![h_2]\!]$$

Static **AND** Dynamic Enforcement

What should we
prove about the
protection function?

$$[\![h_1]\!] \sqsubseteq_{St} [\![h_2]\!] \quad \Longrightarrow \quad \mathrm{protect}[\![h_1]\!] \sqsubseteq_{Dyn} \mathrm{protect}[\![h_2]\!]$$

Static $s \sqsubseteq_X d$ Dynamic

First, to specify correctness, define when a precise component refines an imprecise one

Static $s \sqsubseteq_X d$ Dynamic

First, to specify correctness, define when a precise component refines an imprecise one

Reverse of compiler correctness!

Static $s \sqsubseteq_X d$ Dynamic

First, to specify correctness, define when a precise component refines an imprecise one

Reverse of compiler correctness!

$$\frac{s' \sqsubseteq_{St} s \sqsubseteq_X d \sqsubseteq_{Dyn} d'}{s' \sqsubseteq_X d'}$$

Need compatibility with refinement on each side

Specification for Protect

$$\text{protect} : St \to Dyn$$

Specification for Protect

$$\text{protect} : St \to Dyn$$

Correctness

$$s \sqsubseteq_X \text{protect}(s)$$

Specification for Protect

$$\text{protect} : St \to Dyn$$

Correctness

$$s \sqsubseteq_X \text{protect}(s)$$

Minimality

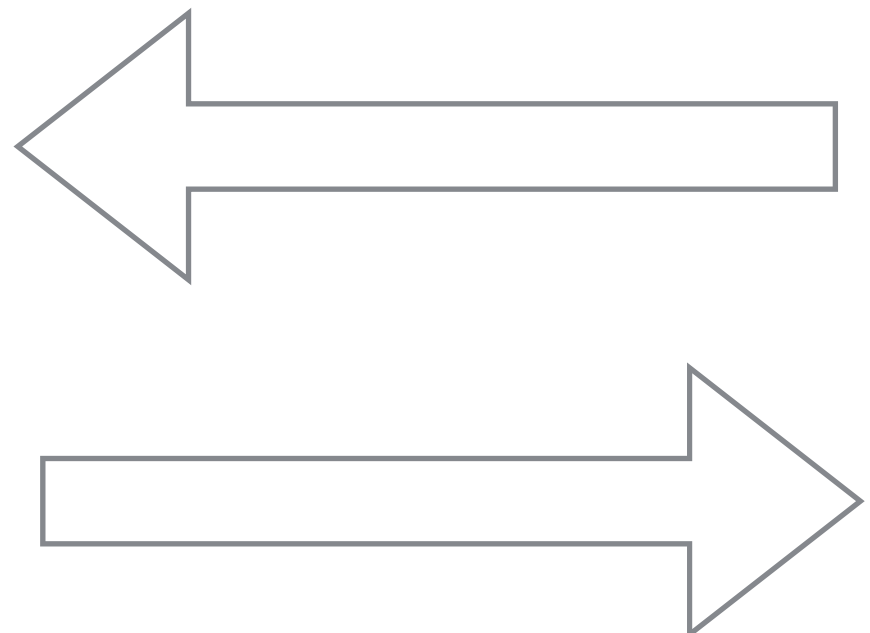
$$\frac{s \sqsubseteq_X d}{\text{protect}(s) \sqsubseteq_{Dyn} d}$$

C+M => Refinement Preservation

$$\dfrac{\dfrac{s \sqsubseteq_{St} s' \qquad s' \sqsubseteq_X \mathrm{protect}(s')}{s \sqsubseteq_X \mathrm{protect}(s')}}{\mathrm{protect}(s) \sqsubseteq_{Dyn} \mathrm{protect}(s')}$$

Precise $\sqsubseteq_X$ Imprecise

Enforce



Protect

when defining protect for higher order interfaces, ***sufficient*** to have enforce and protect

Specification for Enforce

$$\text{enforce} : Dyn \to St$$

Correctness

$$\text{enforce}(d) \sqsubseteq_X d$$

Maximality

$$\frac{s \sqsubseteq_X d}{s \sqsubseteq_{St} \text{enforce}(d)}$$

C+M => Refinement Preservation

$$\dfrac{\dfrac{\mathrm{enforce}(d) \sqsubseteq_X d \qquad d \sqsubseteq_{Dyn} d'}{\mathrm{enforce}(d) \sqsubseteq_X d'}}{\mathrm{enforce}(d) \sqsubseteq_{Dyn} \mathrm{enforce}(d')}$$

Summary

- Protect and Enforce form a Galois Connection
    - But don't have to prove either are monotone directly: comes from the **compatibility** condition on the heterogeneous relation
- Once the heterogeneous relation is fixed, enforce and protect are unique, if they exist
- Used this definition to prove new theorems about Gradual Typing

Application to Gradual Typing

- If the heterogeneous relation is built inductively on the type structure, we can derive the implementation.
- Any **other** implementation **must** break correctness or minimality
- Shows some gradually typed languages break common optimizations (eta reduction)
- Secure compilation analogue? Necessity of back-translation?

$$\mathrm{protect}_{A \to B}(f) \cong \mathrm{protect}_{B} \circ f \circ \mathrm{enforce}_{A}$$