All the Binaries Together

A Semantic Approach to ABIs

Andrew Wagner, Amal Ahmed



(Secure Interoperability, Languages, and Compilers)

"The standard is haunted … by that **Three Letter Demon**. … a contract was forged in blood."
– JeanHeyd Meneide, WG14 C/C++ Compatibility Chair

🔮 What Is in an ABI?

- Data layouts
- Calling conventions
- Name mangling
+ *Safety invariants*
+ *Ownership*

…

🤷 Who Cares?

★ **Swift:** *ABI Stability Manifesto*

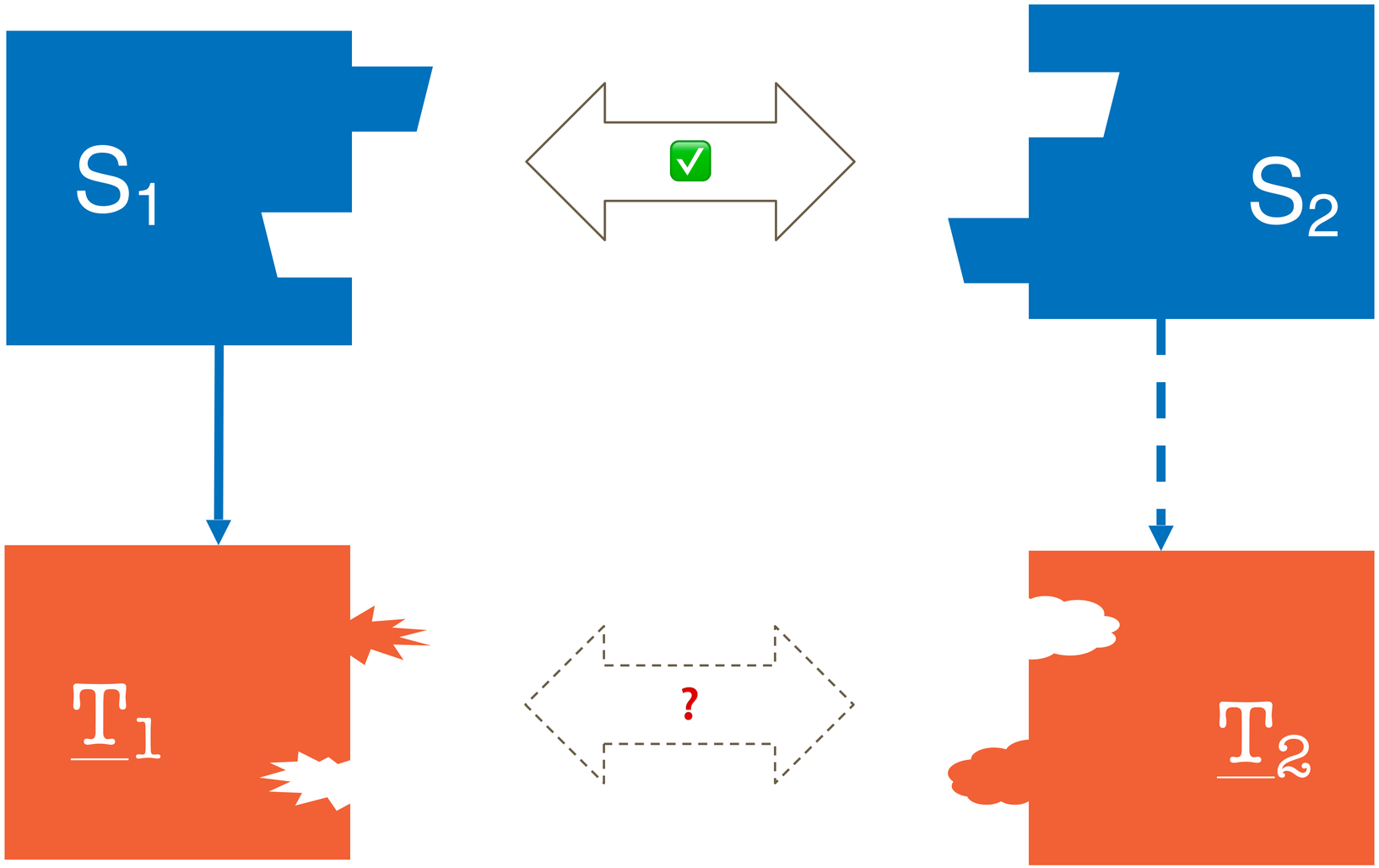
★ **Rust:** *crABI*

★ **C++**: *WG21 ARG*

★ **WASM:** *Component Model*

★ 👊 **You!**

All the Compilers Together



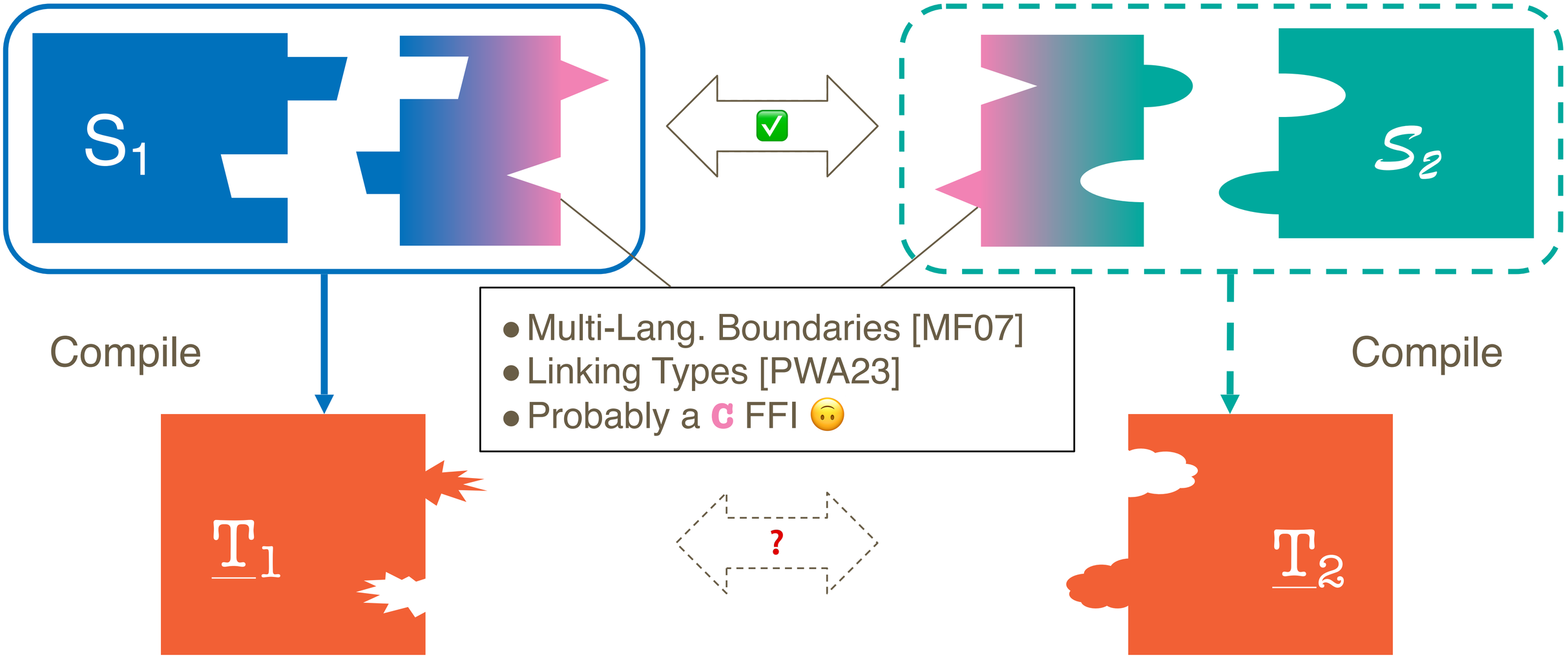
Compiler 1

- **gcc**
- *gcc 9*

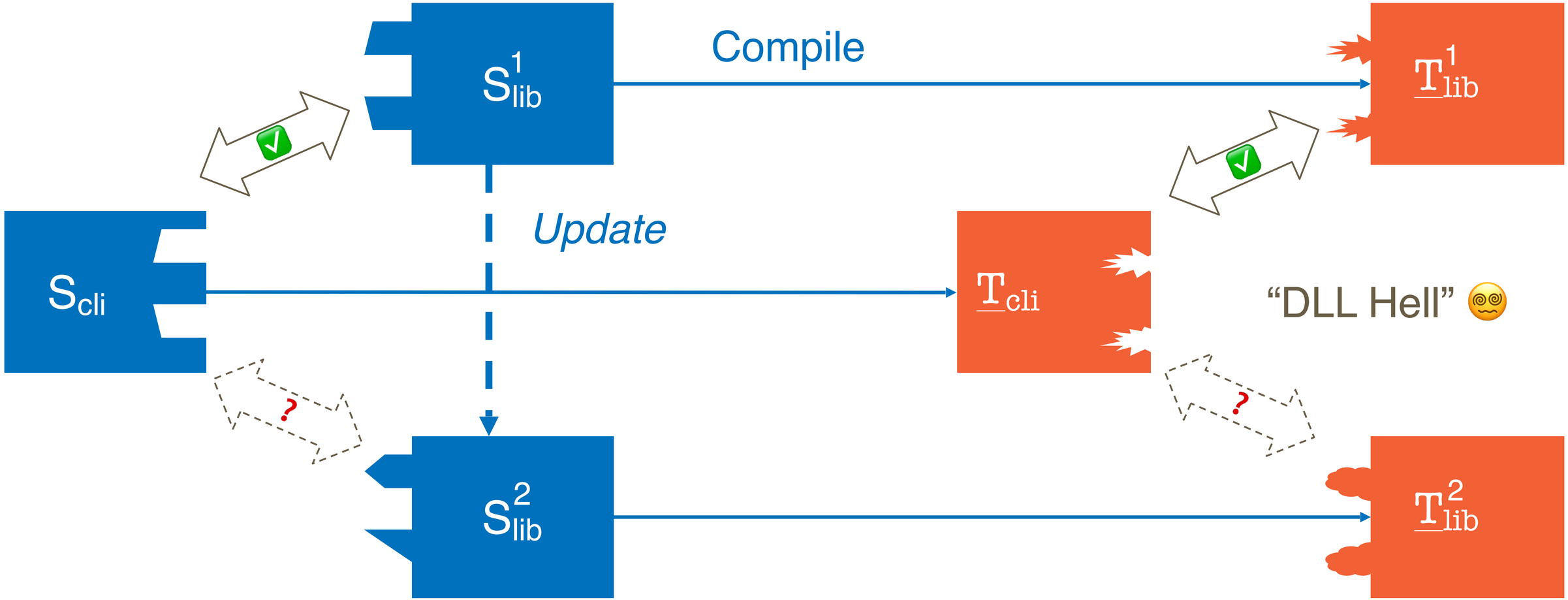
Compiler 2

- **clang**
- *gcc 10*

All the Languages Together



All the Libraries Together

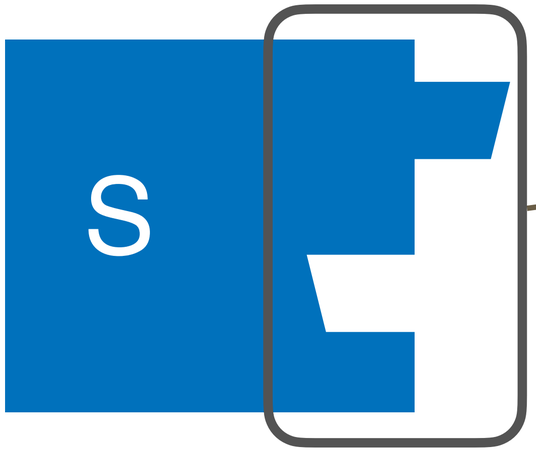


Towards a Formal ABI

- Languages are already grappling with these problems
- Growing dissatisfaction with status quo
- Demand for richer ABIs
- Design decisions, tradeoffs, uncharted territory

Can we provide a semantic foundation?

What Is an ABI, Formally?



“This source interface …”

This Type τ

$\underline{\mathrm{T}}$ is **ABI compliant** with τ if

$$\underline{\mathrm{T}} \in [\![\tau]\!]$$

“… describes target programs like this”

Denotes These Programs

$[\![\tau]\!] = \{ \underline{\mathrm{T}} \mid \ldots \}$ — Semantic Typing via Realizability

$\underline{\mathtt{T}}$ is **ABI compliant** with τ if

$$\underline{\mathtt{T}} \in [\![\tau]\!]$$

Is this a good spec?

1. **Formalization:** Can the spec capture all the pertinent details?
2. **Application:** Can the spec be used in all the relevant scenarios?

Case Study: Reference Counting

- PCF-ish Source
  - Records, variants, higher-order recursive functions
- C-ish Target
  - Block-based memory, pointer arithmetic
- Reference Counting ABI
  - All values are boxed and reference-counted
  - Separation logic specification

Formalization: Semantic Typing via Realizability

$\Gamma \vdash e : T$

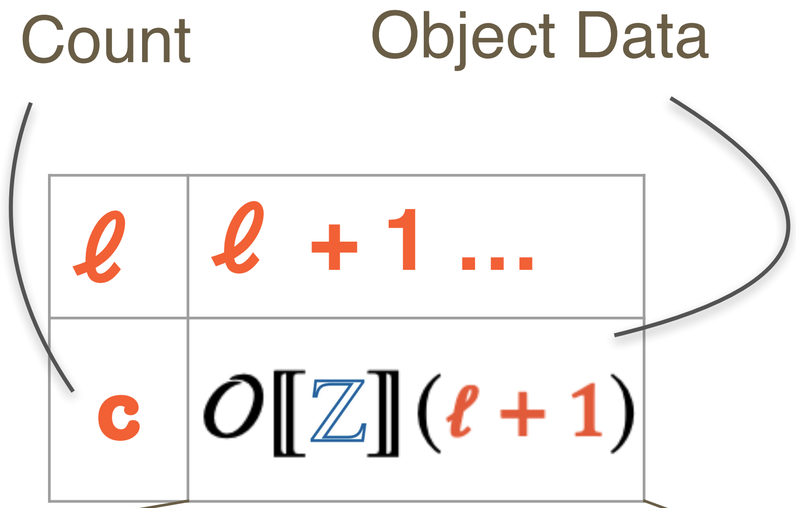
$$\Gamma \vDash e : T$$

$$\approx \left\{ \text{“Prestate like } \Gamma\text{”} \right\} e \left\{ v.\ \text{“}v \text{ like } T\text{”} \right\}$$

$$\approx \left\{ \bigstar \overline{\mathcal{R}[\![T_x]\!](x)} \right\} e \left\{ \ell.\ \mathcal{R}[\![T]\!](\ell) \right\} \quad (\Gamma = \overline{x : T_x})$$

Formalization: Reference Layout

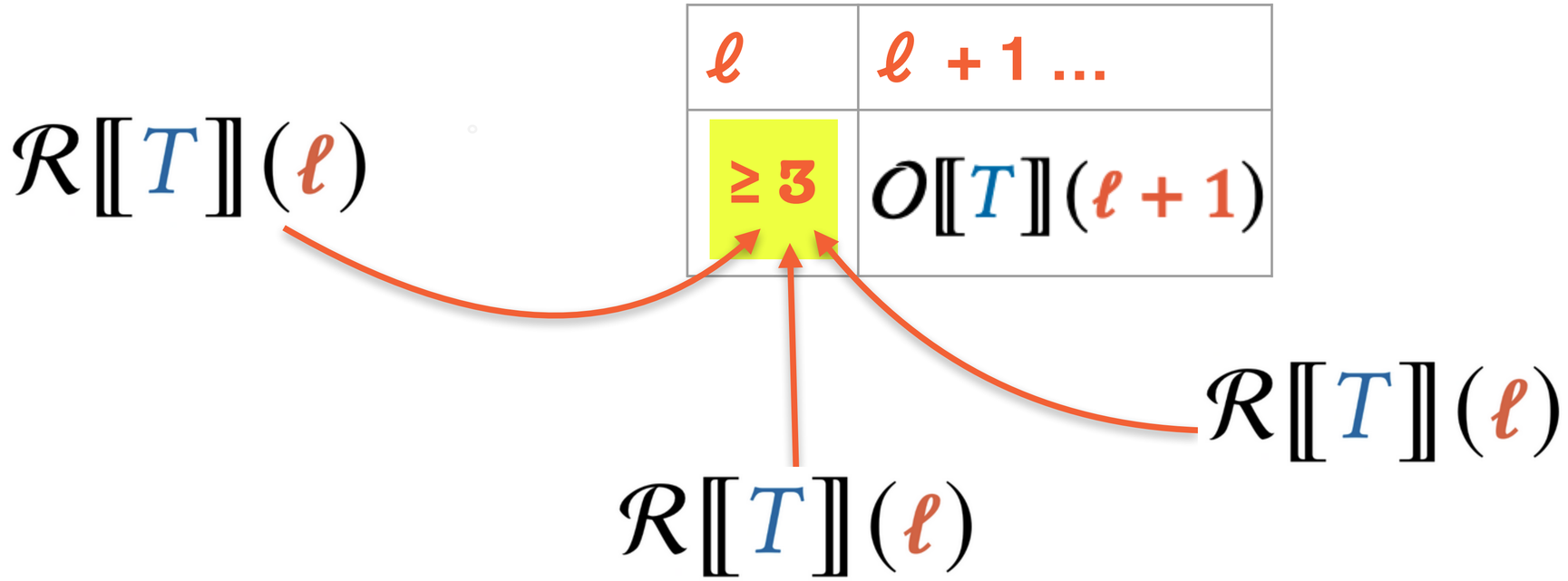
$$\mathcal{R}[\![\mathbb{Z}]\!](\ell) \approx$$



$$O[\![\mathbb{Z}]\!](\ell + 1) = \exists\, n.\, \ell + 1 \mapsto n$$

Also:
Unboxed data via pointer tagging

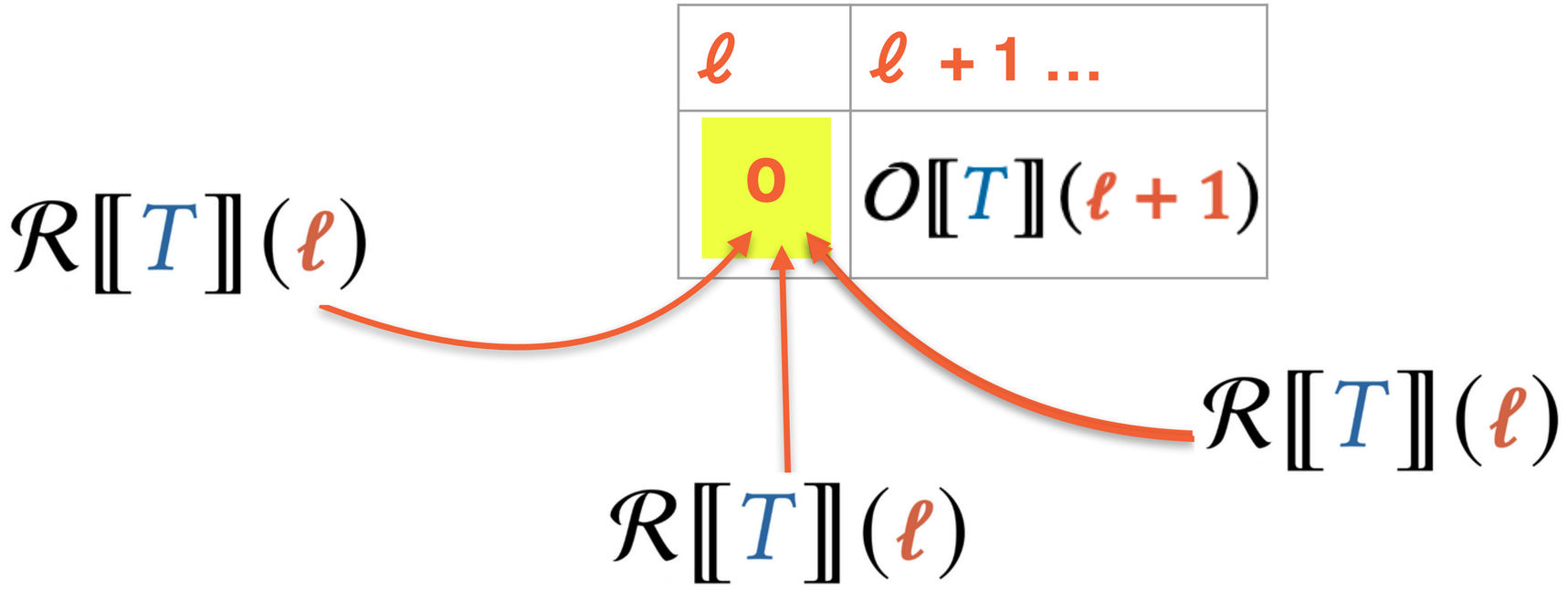
Formalization: Ownership + Sharing



RC-INCR

$$\left\{\mathcal{R}[\![T]\!](\ell)\right\} \; {+}{+}\ell \; \left\{n.\; \ulcorner n > 1 \urcorner \star \mathcal{R}[\![T]\!](\ell) \star \mathcal{R}[\![T]\!](\ell)\right\}$$

Formalization: Ownership + Sharing



RC-DECR

$$\left\{\mathcal{R}[\![T]\!](\ell)\right\} \multimap\!\!-\ell \left\{n.\left(\ulcorner n > 0 \urcorner \wedge \text{emp}\right) \vee \left(\ulcorner n = 0 \urcorner \star \ell \mapsto 0 \star \mathcal{O}[\![T]\!](\ell + 1)\right)\right\}$$

Formalization: Ownership + Sharing

$\mathcal{R}[\![T]\!](\ell)$

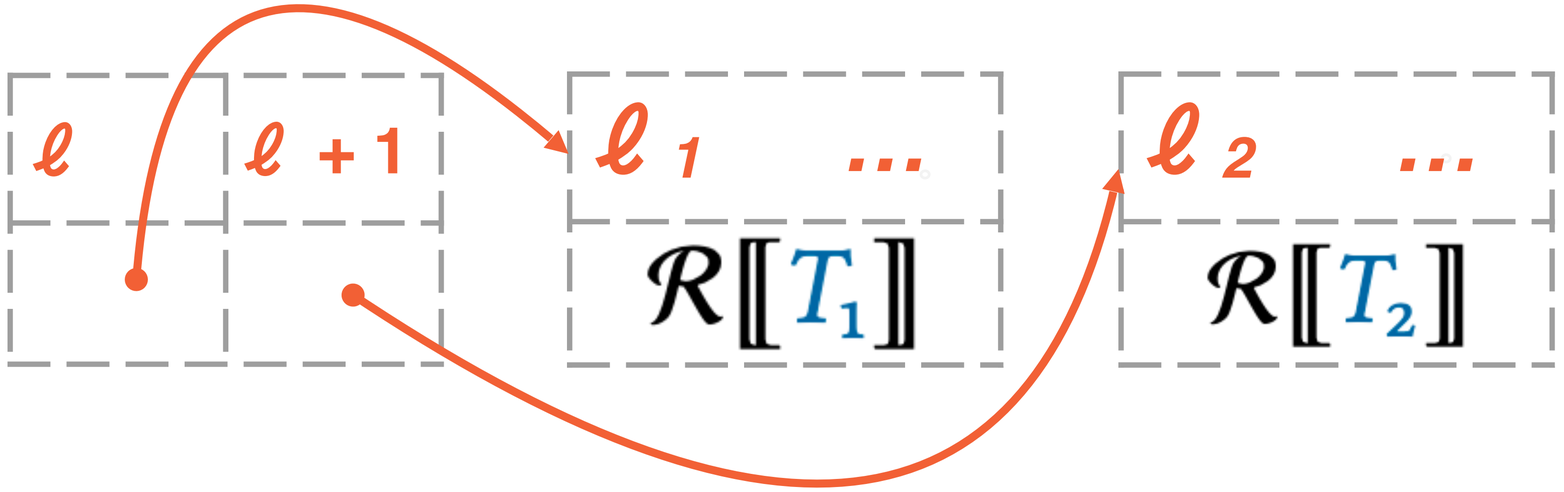
ℓ	$\ell + 1 \ldots$
1	$\mathcal{O}[\![T]\!](\ell + 1)$

RC-NEW

$$\frac{\{\mathcal{R}[\![T]\!](\ell)\}\ e\ \{Q\}}{\{\ell \mapsto 1 \star \mathcal{O}[\![T]\!](\ell + 1)\}\ e\ \{Q\}}$$

Formalization: Compound Layout

$$\mathcal{O}[\![T_1 \times T_2]\!](\ell) \approx$$



$$\mathcal{O}[\![T_1 \times T_2]\!](\ell) \triangleq \exists\, \ell_1, \ell_2.$$

Also:
Records and variants

$$\ell \mapsto \ell_1 \;\star\; \ell + 1 \mapsto \ell_2 \;\star\; \mathcal{R}[\![T_1]\!](\ell_1) \;\star\; \mathcal{R}[\![T_2]\!](\ell_2)$$

Formalization: Calling Convention

$$\mathcal{O}[\![T_1 \to T_2]\!](\ell) \mathrel{\hat{\approx}} \exists f.\, \ell \mapsto f \star$$
$$\forall \ell_1.\, \{\mathcal{R}[\![T_1]\!](\ell_1)\}\ f(\ell_1)\ \{\ell_2.\, \mathcal{R}[\![T_2]\!](\ell_2)\}$$

Pointer to function

Calling convention:
Caller retain

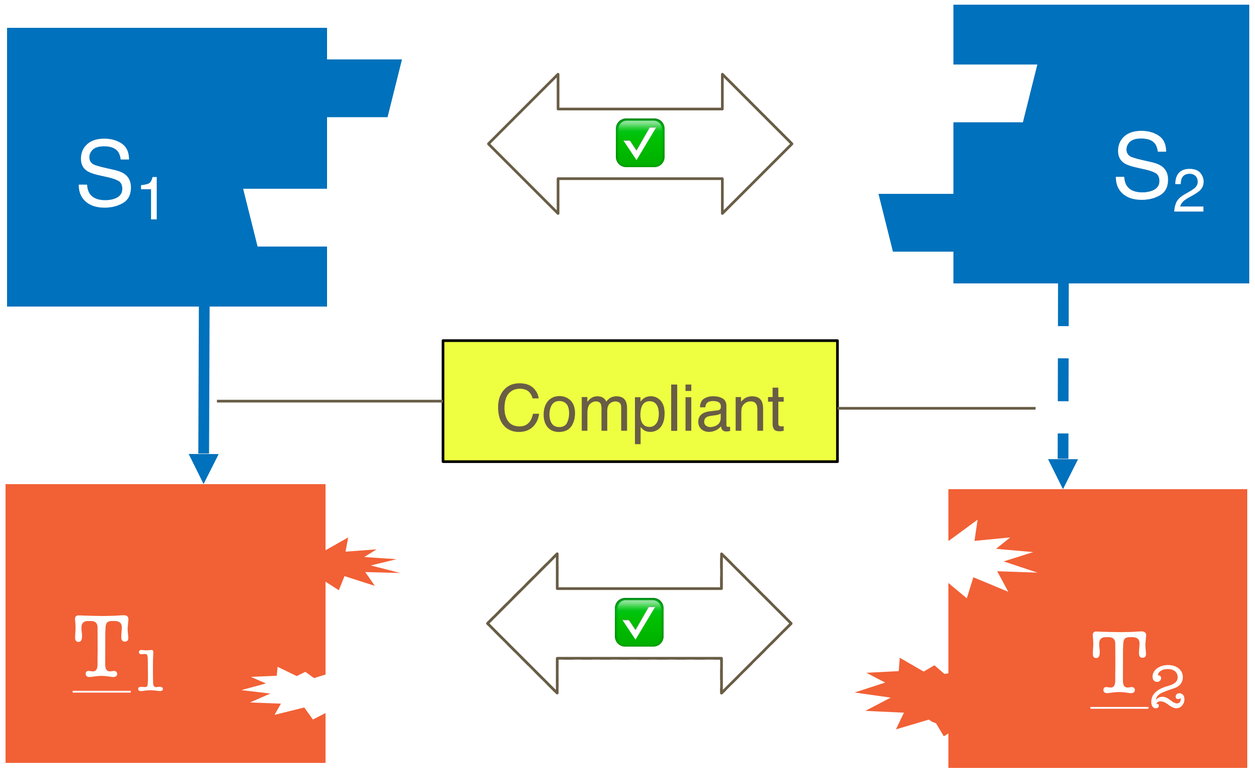
VS.

$$\forall \ell_1.\, \{\mathcal{R}[\![T_1]\!](\ell_1)\}\ f(\ell_1)\ \{\ell_2.\, \mathcal{R}[\![T_2]\!](\ell_2) \star \underbrace{\mathcal{R}[\![T_1]\!](\ell_1)}\}$$

Callee retain

Also:
Closures

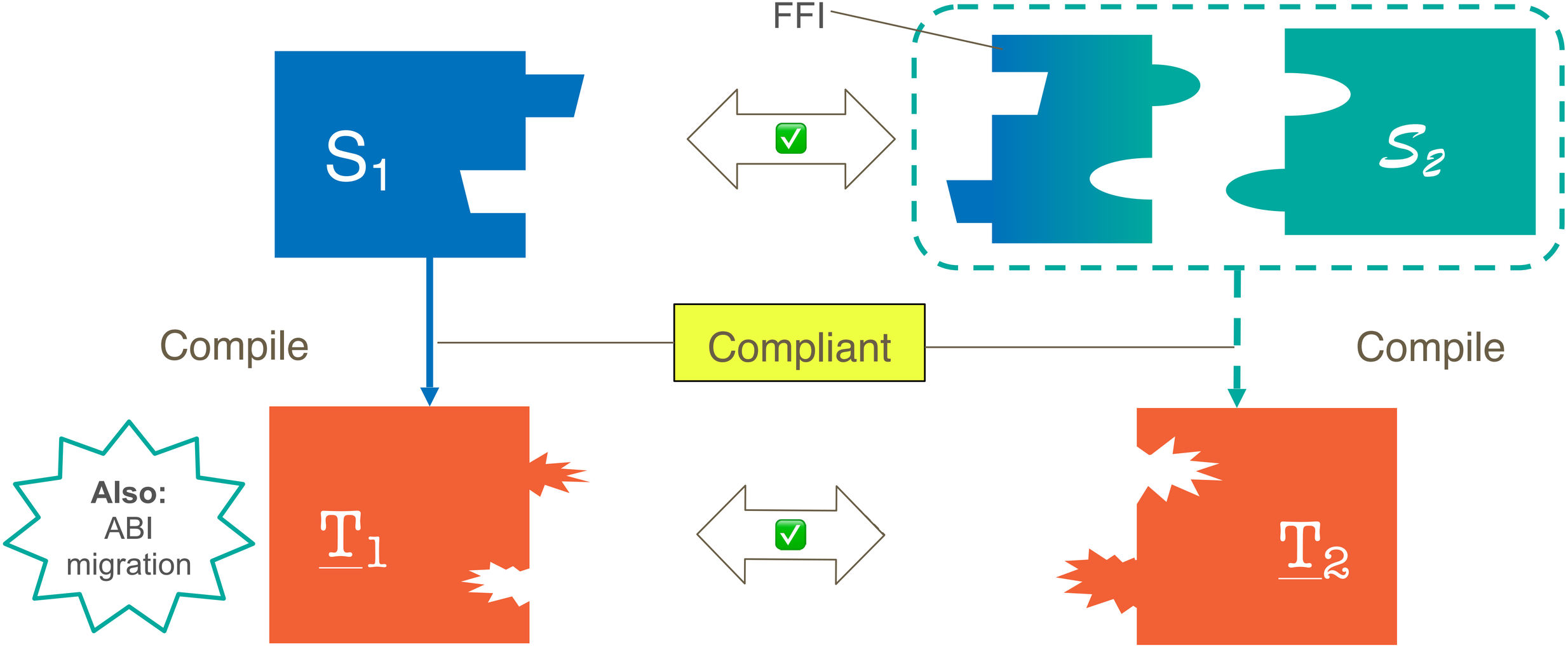
Application: Compiler Compliance



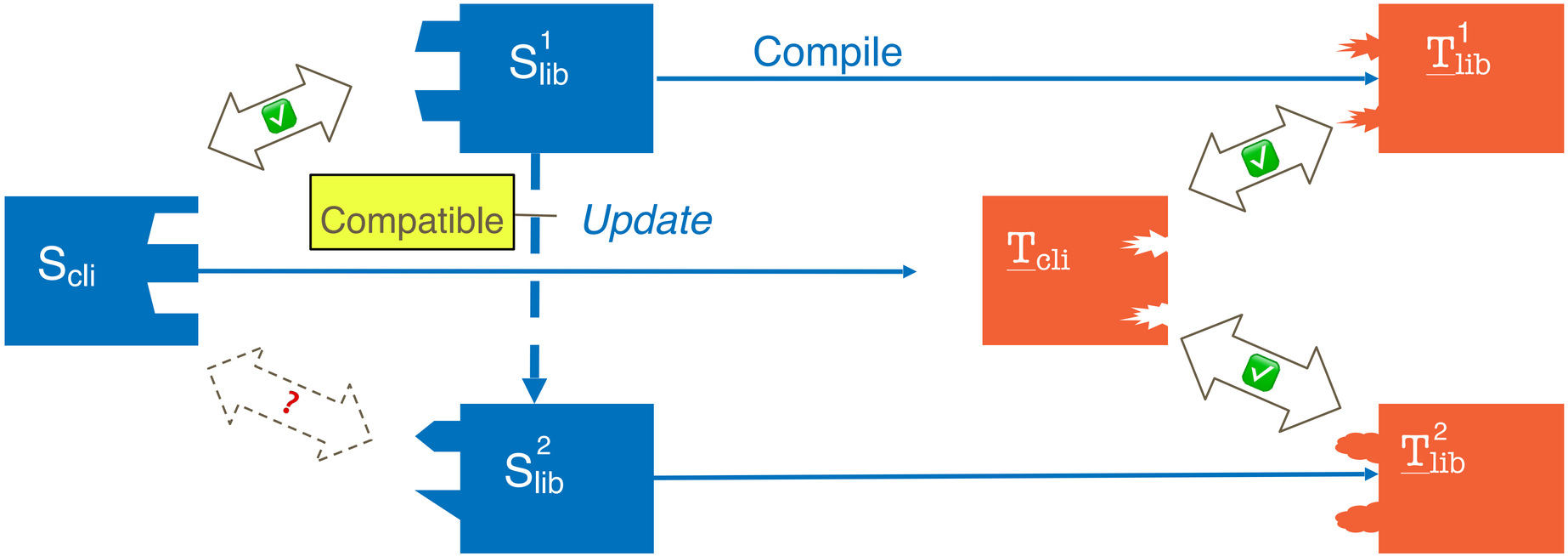
$\rightsquigarrow$ is an **compliant compiler** if

$S : \tau$ *and* $S \rightsquigarrow \underline{T}$ *implies* $\underline{T} \in [\![\tau]\!]$

Application: FFI Safety



Application: Library Compatibility



Swift ? C

Flexible Rigid

τ_2 is an **compatible update** from τ_1 if

$\underline{T} \in [\![\tau_2]\!]$ *implies* $\underline{T} \in [\![\tau_1]\!]$

Next Steps

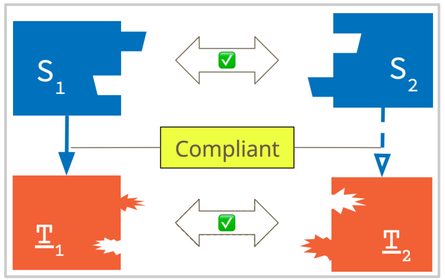
- ★ Wrapping up case study
  - ✦ Variations on design
- ★ Idiosyncrasies of Swift ABI
  - ✦ Resilient type layouts, reabstraction (polymorphism)
- ★ Rust ABI over Wasm
  - ✦ Component Model (prev. Interface Types) building blocks

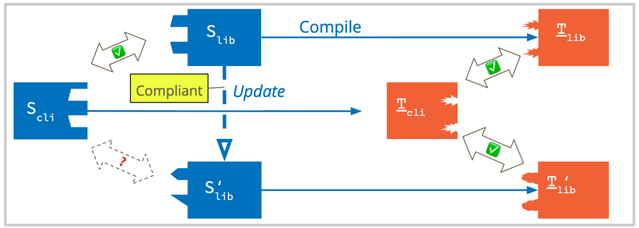
Takeaways

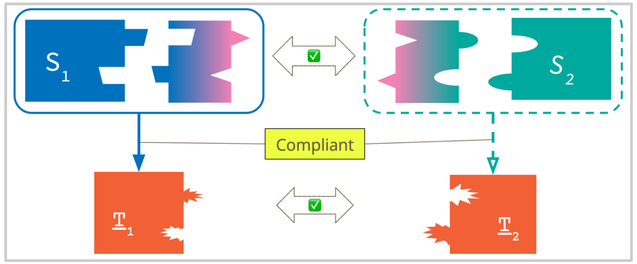
Formalization

$$\underline{\mathtt{T}} \in [\![\tau]\!]$$

Application







Let's Chat!

Email: ahwagner@ccs.neu.edu
Web: andrewwagner.io