All the Binaries Together

A Semantic Approach to ABIs

Andrew Wagner, Amal Ahmed

(Secure Interoperability, Languages, and Compilers)

🔮 What Is an ABI?

“Implementation details”

- Data layouts
- Calling conventions

+ *Safety invariants*

+ *Ownership*

…

🤷 Who Cares?

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

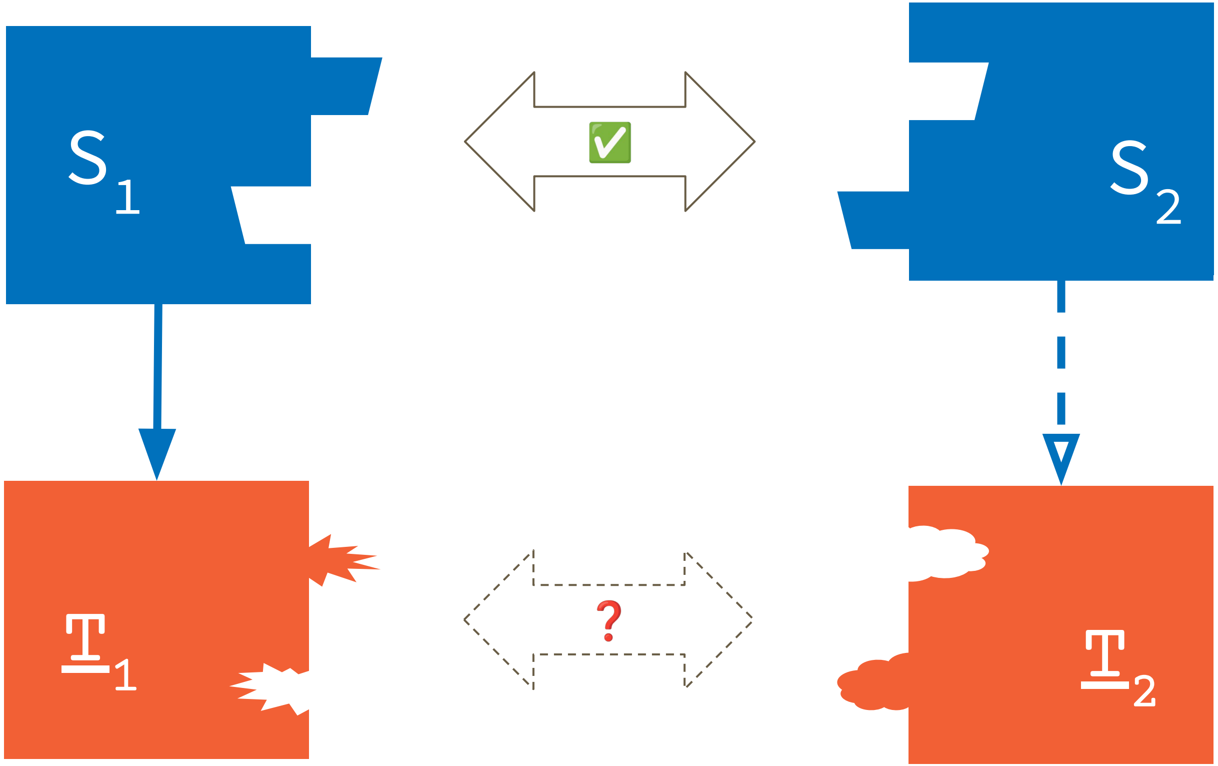
★ **Rust:** *crABI*

★ **C++**: WG21 ARG

★ **WASM:** *Component Model*

★ 🫵 **You!**

All the Compilers Together



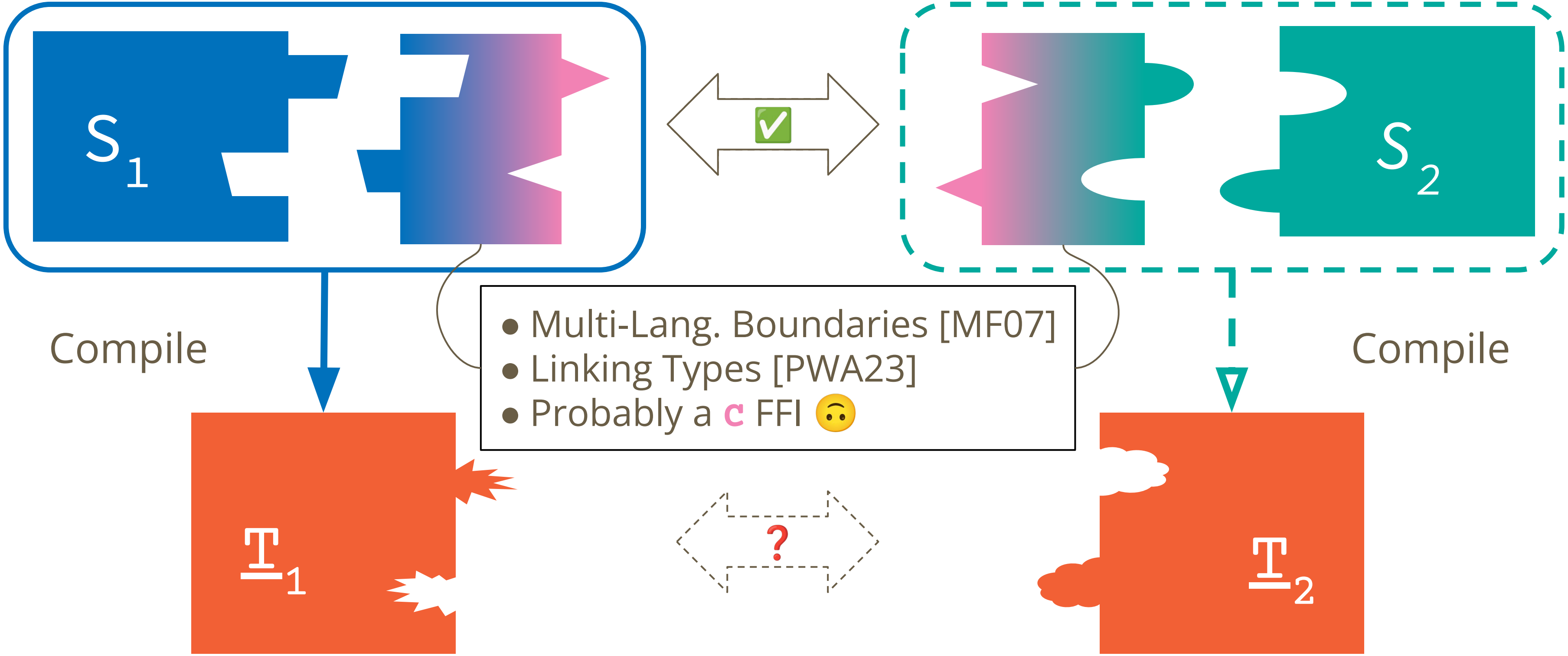
Compiler 1

- **`gcc`**
- *`gcc 9`*

Compiler 2

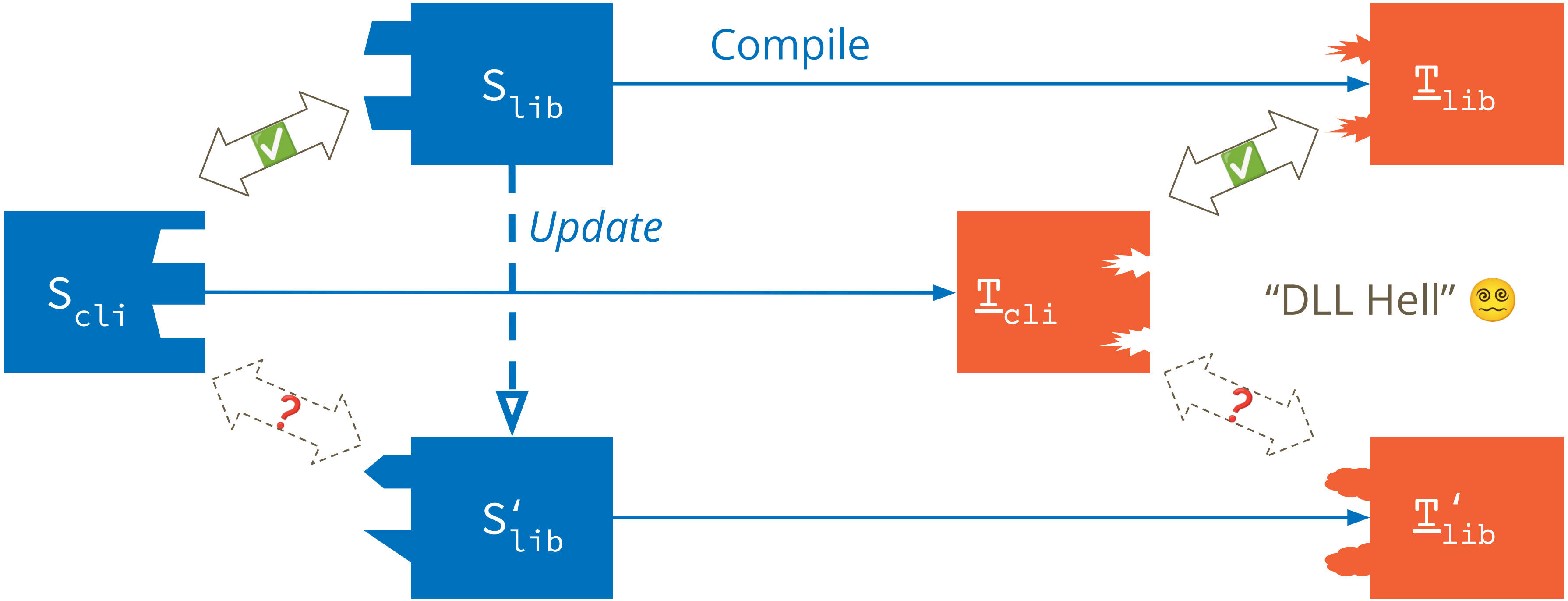
- **`clang`**
- *`gcc 10`*

All the Languages Together



- Multi-Lang. Boundaries [MF07]
- Linking Types [PWA23]
- Probably a C FFI 🙃

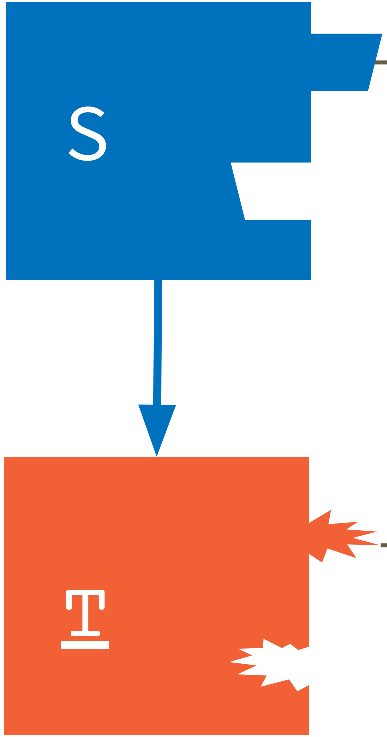
All the Libraries Together



Research Objectives

1. Formalize an ABI
2. Validate real techniques
3. *Recommend improvements!

Formalizing an ABI



“This Interface”

This Type τ

“Has This ABI”

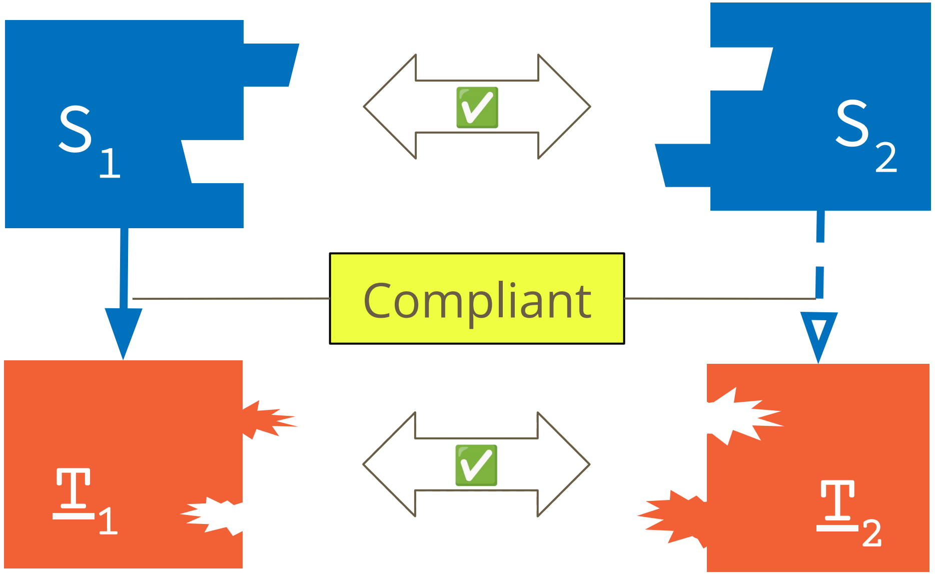
Denotes These Programs

$[\![\tau]\!] = \{\ \underline{\mathtt{T}}\ |\ \dots\ \}$

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

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

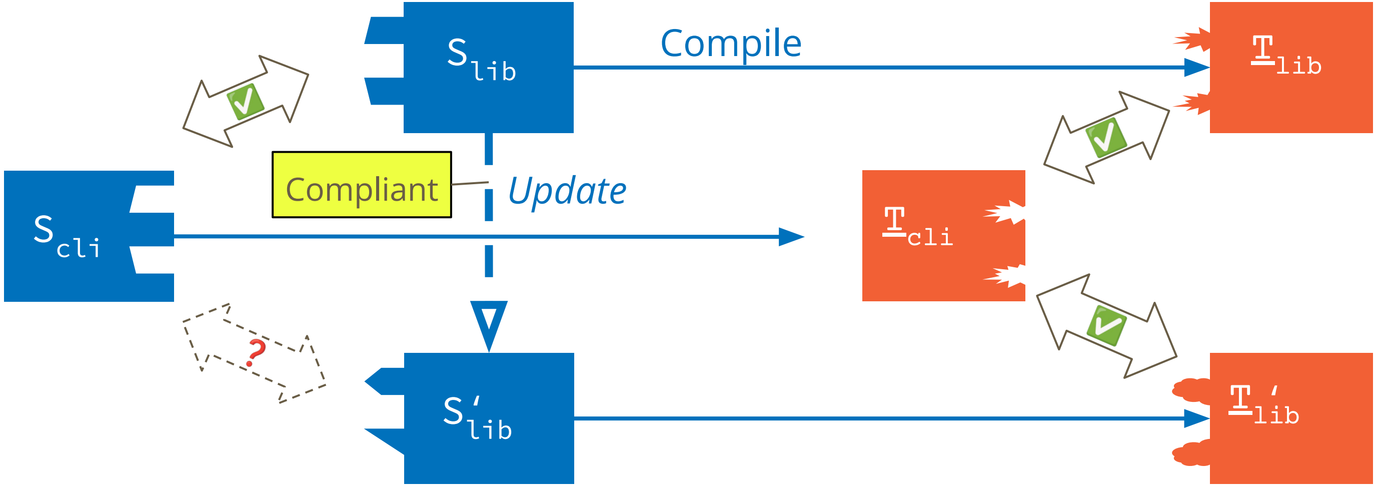
All the Compilers Together, Formally



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

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

All the Libraries Together, Formally



$S_{\mathtt{lib}}$
Compile
$\underline{T}_{\mathtt{lib}}$
Compliant
Update
$S_{\mathtt{cli}}$
$\underline{T}_{\mathtt{cli}}$
?
$S'_{\mathtt{lib}}$
$\underline{T}'_{\mathtt{lib}}$

τ' is an **ABI compatible update** for τ if

$\underline{T} \in [\![\tau']\!]$ *implies* $\underline{T} \in [\![\tau]\!]$

Evaluation

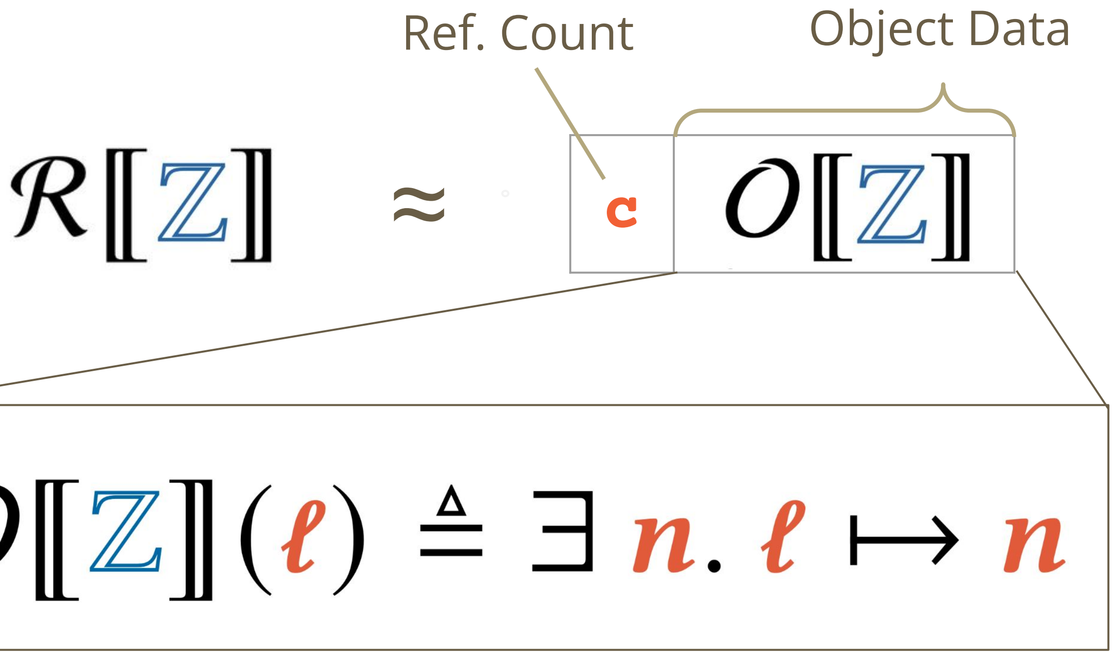
A Case Study

- Breadth over depth

Case Study: Reference Counting

- `Pure, ML-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

A Semantic ABI: Basics



$$\mathcal{R}[\![\mathbb{Z}]\!] \approx \boxed{\mathbf{c}}\boxed{\mathcal{O}[\![\mathbb{Z}]\!]}$$

$$\mathcal{O}[\![\mathbb{Z}]\!](\ell) \triangleq \exists\, \boldsymbol{n}.\, \boldsymbol{\ell} \mapsto \boldsymbol{n}$$

A Semantic ABI: Ownership + Sharing

RC-NEW

$$\frac{\{P \star @_{\ell}\ Q\}\ e\ \{R\}}{\{P \star \ell \mapsto 1 \star Q\}\ e\ \{R\}}$$

RC-INCR

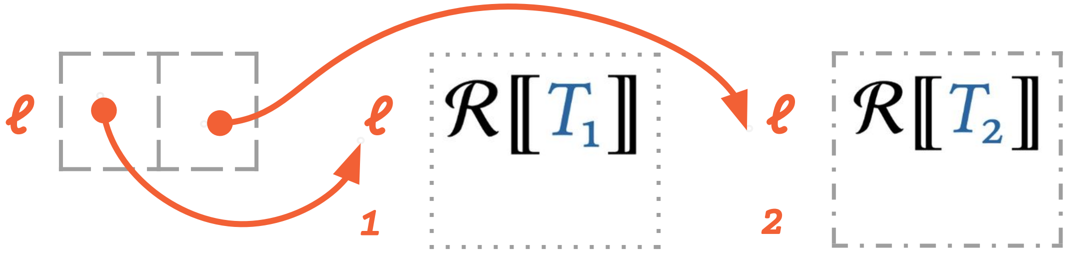
$$\{@_{\ell}\ P\}\ {+}{+}\ell\ \{n.\ \ulcorner n > 1 \urcorner \star @_{\ell}\ P \star @_{\ell}\ P\}$$

RC-DECR

$$\{@_{\ell}\ P\}\ {-}{-}\ell\ \{n.\ \ulcorner n > 0 \urcorner \lor (\ulcorner n = 0 \urcorner \star \ell \mapsto 0 \star P)\}$$

A Semantic ABI: Layout

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



$$\mathcal{O}[\![T_1 \times T_2]\!](\ell) \triangleq \exists\, \ell_1, \ell_2.$$
$$\ell \mapsto \ell_1 \star \ell + 1 \mapsto \ell_2 \star \mathcal{R}[\![T_1]\!](\ell_1) \star \mathcal{R}[\![T_2]\!](\ell_2)$$

A Semantic ABI: Calling Convention

$$\mathcal{O}[\![T_1 \to T_2]\!](\ell) \overset{\triangle}{\approx} \exists\, f.$$

$$\underbrace{\ell \mapsto f}_{\text{Pointer to function}} \land \underbrace{\forall\, \ell_1.\ \{\mathcal{R}[\![T_1]\!](\ell_1)\}\ f(\ell_1)\ \{\ell_2.\ \mathcal{R}[\![T_2]\!](\ell_2)\}}_{\text{Calling convention: caller retain}}$$

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

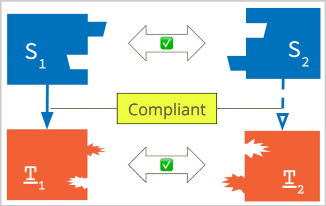
What Can We Do Now?

1. ABI Variations
    a. Unboxed data via pointer tagging
    b. Different calling

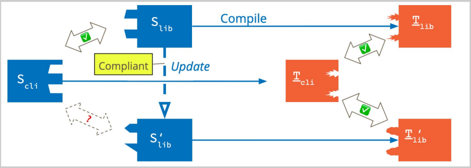
Recap

Use ABI to show:

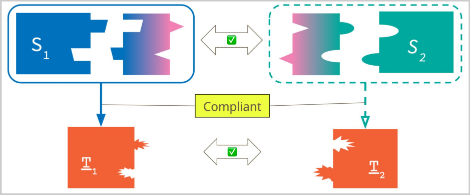
Compiler Compliance



Update Compatibility



FFI Safety



Next Steps

- ★ Wrapping up case study
- ★ Idiosyncrasies of Swift ABI
- ★ Rust ABI over Wasm

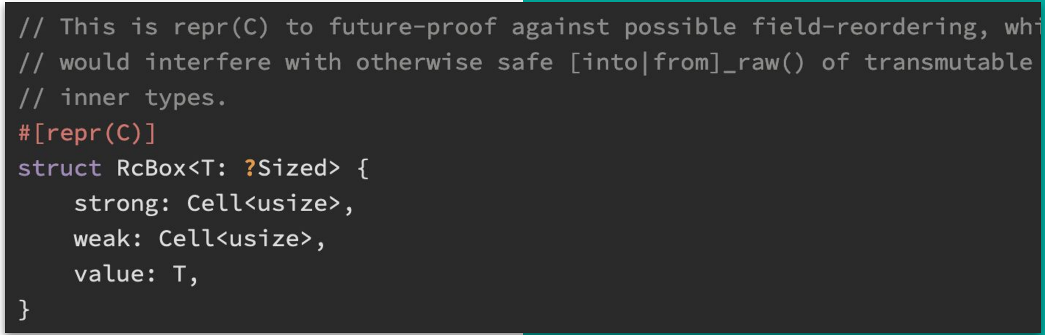
Email: ahwagner@ccs.neu.edu
Web: andrewwagner.io

Why C?

Shallow Answer: Because every language speaks **C**

But *Why* Does Every Language Speak C?

Deeper Answer: Because **C** is committed to ABI stability



```
// This is repr(C) to future-proof against possible field-reordering, whi
// would interfere with otherwise safe [into|from]_raw() of transmutable
// inner types.
#[repr(C)]
struct RcBox<T: ?Sized> {
    strong: Cell<usize>,
    weak: Cell<usize>,
    value: T,
}
```

ABI Stability?

Pros

- ➕ Precise control over interface to other languages
- ➕ Proper support for shared libraries

Cons

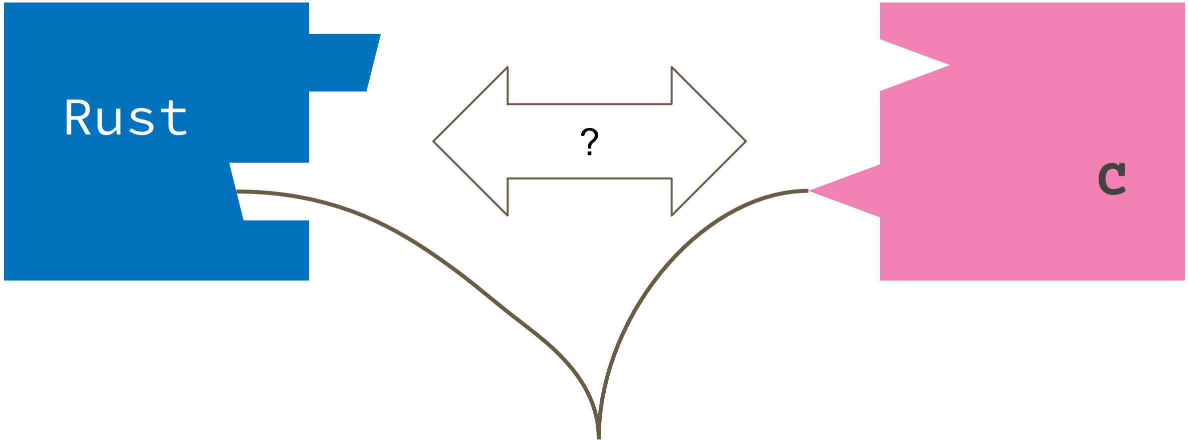
- ➖ Can stunt language growth
- ➖ Limits compiler optimizations
- ➖ Tension between flexibility and performance
- ➖ Pressure on library developers

Interoperability

How can we safely compose diverse programs?

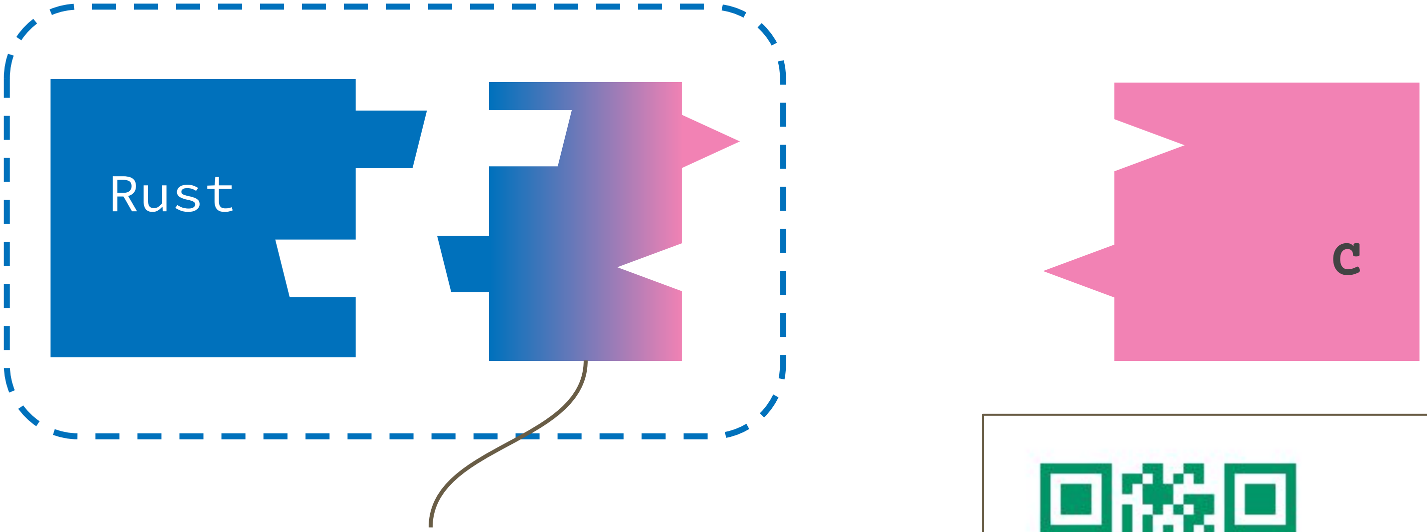
- ★ Most software is multilingual
- ★ Even monolingual software can have diverse components
  - ○ *Different compilers*
  - ○ *Backward/forward compatibility*
  - ○ *“DLL Hell”* 😵‍💫

All the Languages Together



Application Programming Interface (**API**)

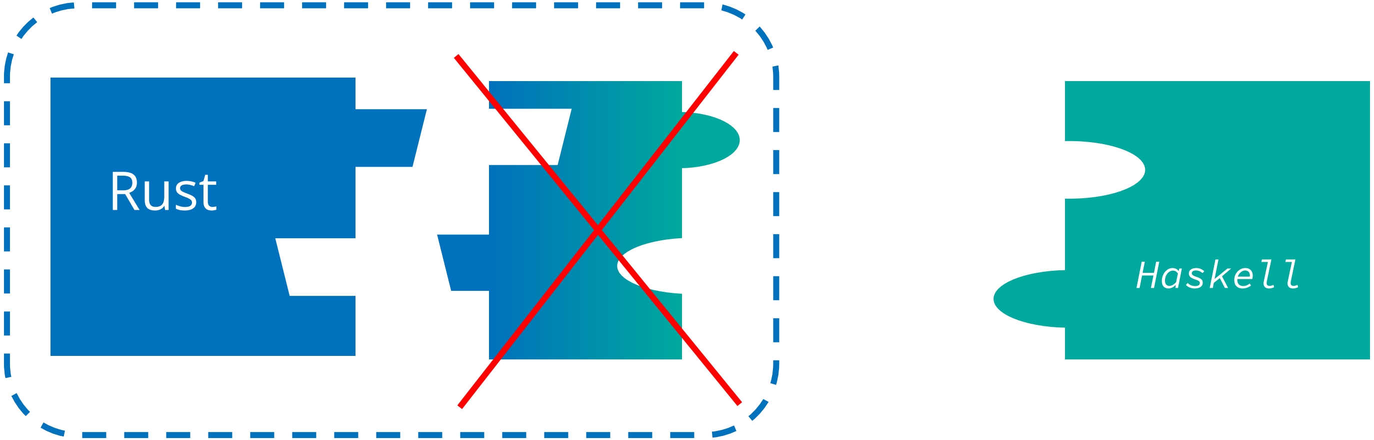
All the Languages Together …



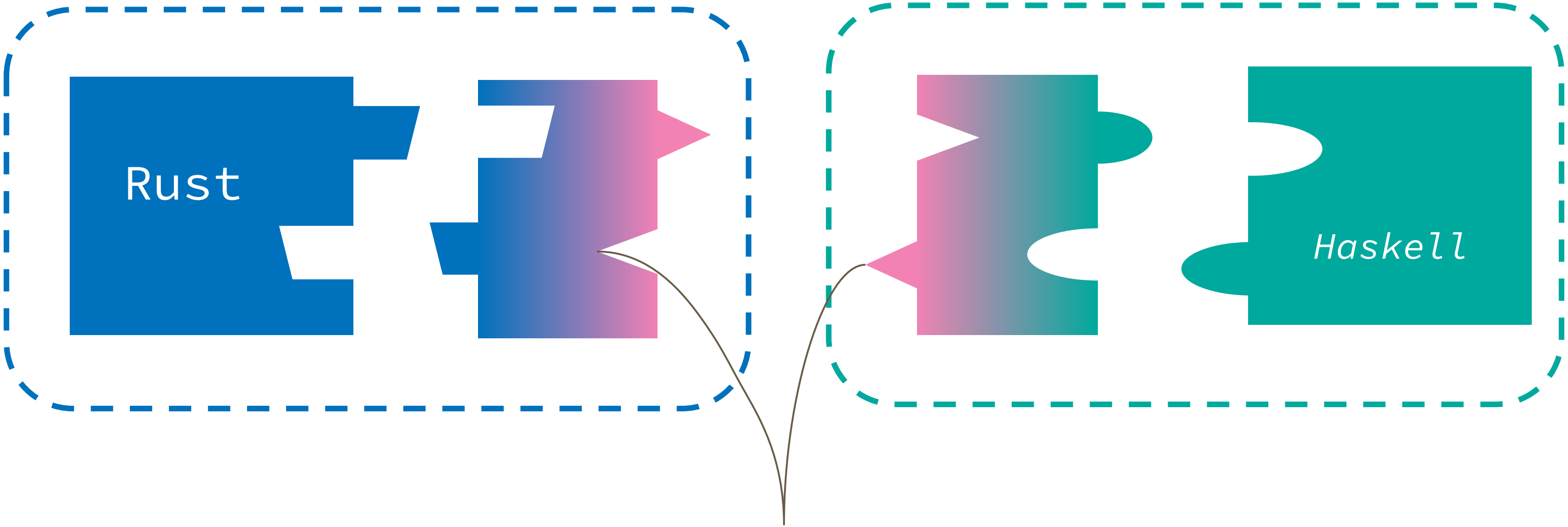
Foreign Function Interface (**FFI**)

Linking Types
Patterson, Wagner, Ahmed
TyDe '23

All the ~~Safe~~ Languages Together

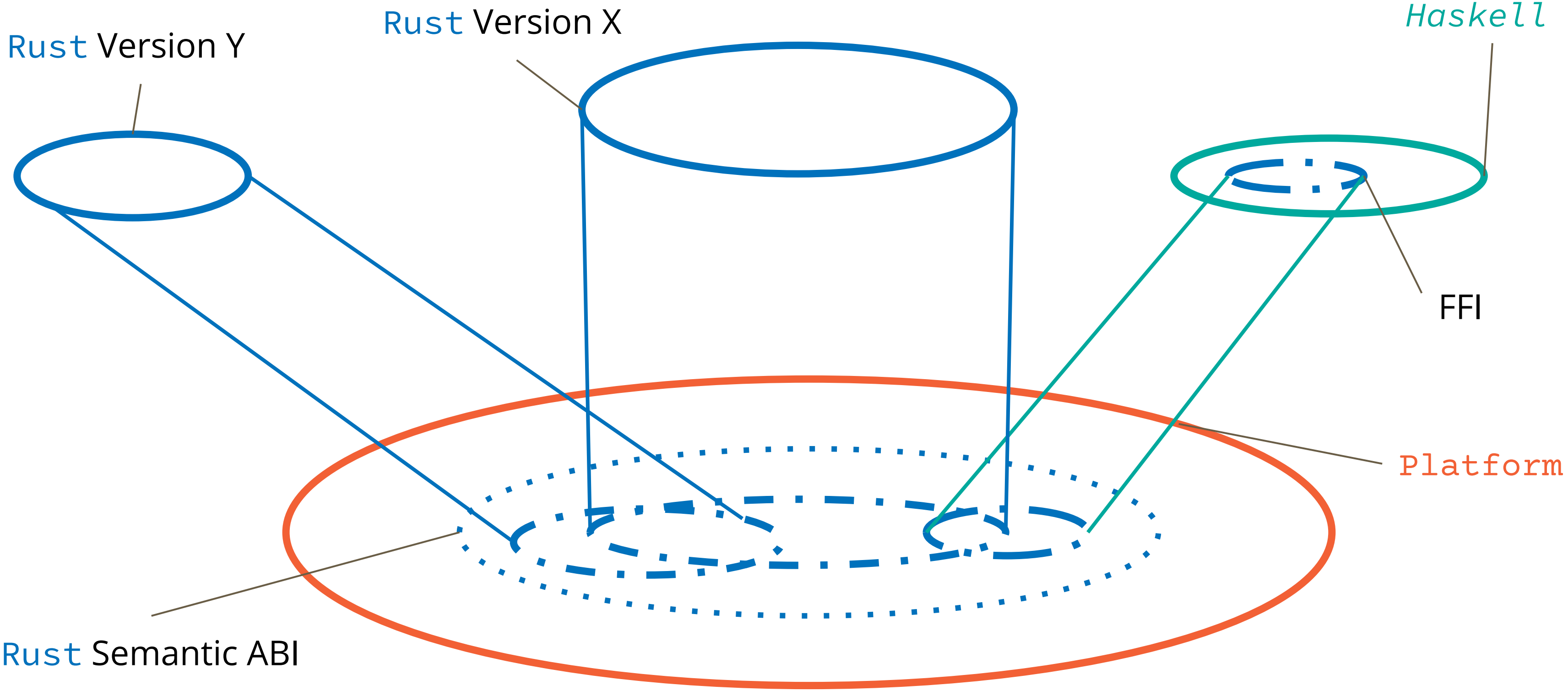


All the Languages Together Again

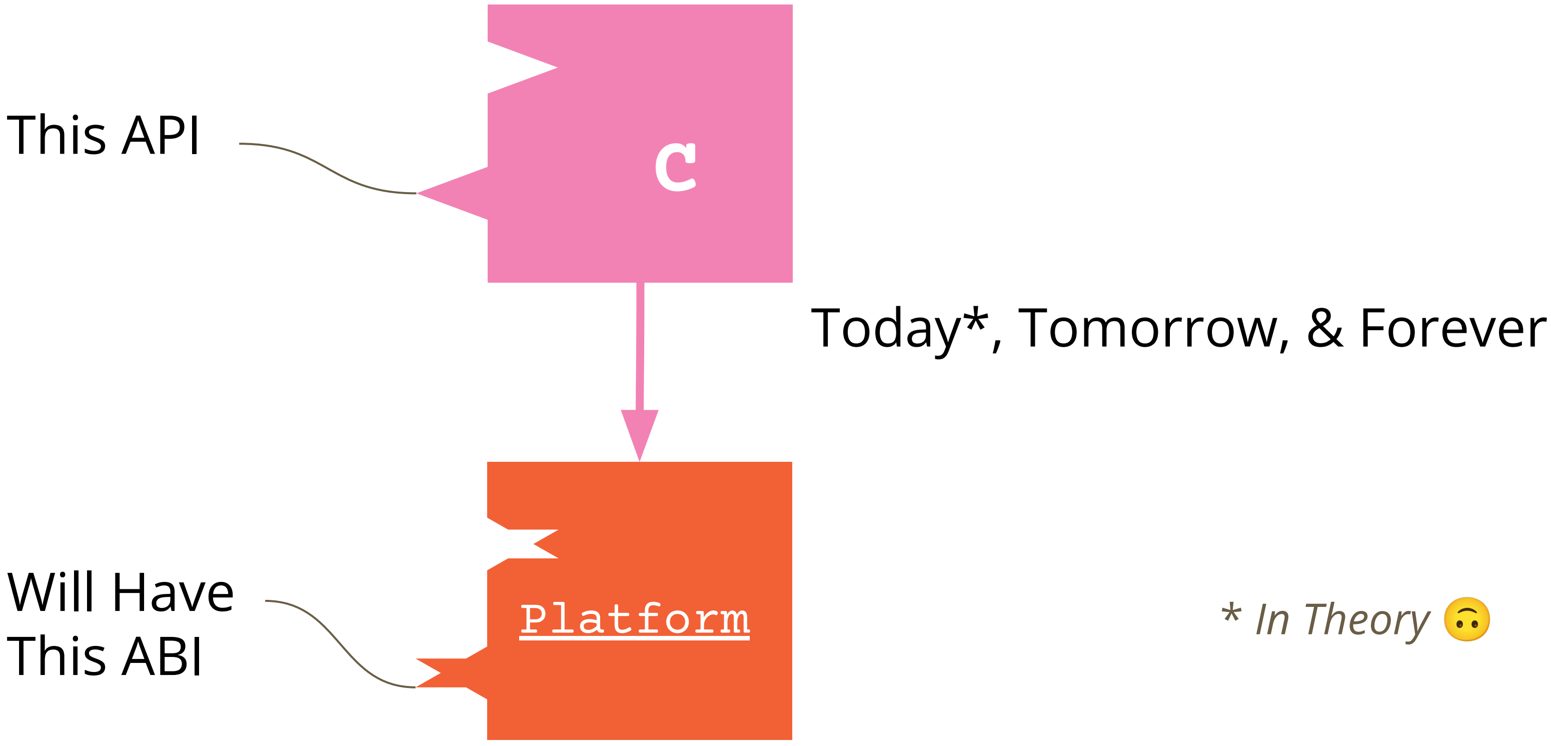


More c Code 🤬

All the Binaries Together!

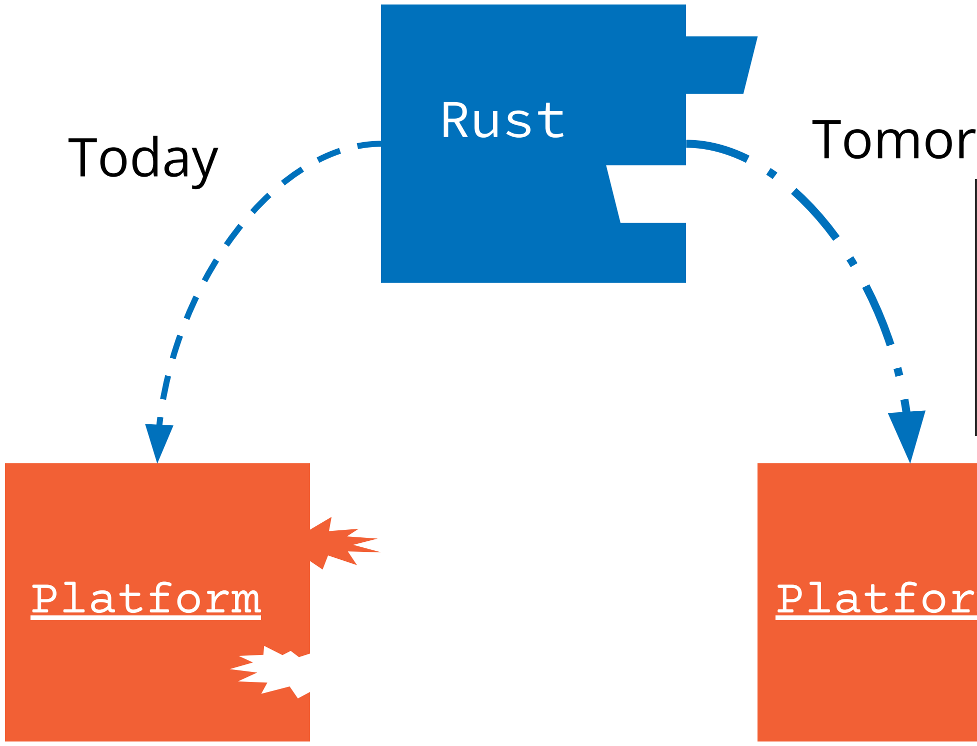


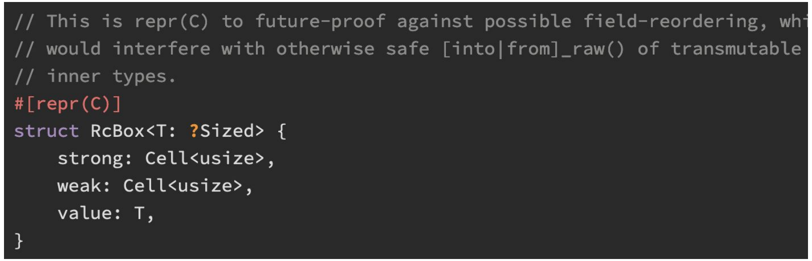
ABI Stability



** In Theory* 🙃

ABI Instability





```
// This is repr(C) to future-proof against possible field-reordering, whi
// would interfere with otherwise safe [into|from]_raw() of transmutable
// inner types.
#[repr(C)]
struct RcBox<T: ?Sized> {
    strong: Cell<usize>,
    weak: Cell<usize>,
    value: T,
}
```

The Times They Are a-Changin'

- ★ **Swift:** *ABI Stability Manifesto*
- ★ **Rust:** RFC#3470 – *crABI*
- ★ **WASM:** *Component Model* Proposal (FKA, *Interface Types*)
- ★ Abundance of libraries, plugins, and tools for low-level interoperability

A Semantic ABI

Realizability Model:

Set of `target terms` indexed by `source types`

$$\mathcal{V}[\![\texttt{u32}]\!] \overset{\text{def}}{=} \{\, \texttt{n} \mid \texttt{n} < 2^{32} \,\}$$

$$\mathcal{V}[\![\texttt{Box<T>}]\!] \overset{\text{def}}{=} \{\, \texttt{l} \mid \texttt{l} \in \mathcal{M}[\![\texttt{T}]\!] \,\}$$

…

$$\mathcal{M}[\![\texttt{u32}]\!] \overset{\text{def}}{=} \{\, \texttt{l} \mid \exists\ \texttt{n} < 2^{32}.\ \texttt{l} \mapsto \texttt{n} \,\}$$

$$\mathcal{M}[\![\,(\texttt{T}_1,\texttt{T}_2)\,]\!] \overset{\text{def}}{=} \{\, \texttt{l} \mid \texttt{l} \in \mathcal{M}[\![\texttt{T}_1]\!] * (\texttt{l} + \mathrm{size}(\texttt{T}_1)) \in \mathcal{M}[\![\texttt{T}_2]\!] \,\}$$

…

$$\mathcal{C}[\![\texttt{T}]\!] \overset{\text{def}}{=} \{\, \texttt{P} \mid \mathrm{wp}(\texttt{P})\{\, v.\ v \in \mathcal{V}[\![\texttt{T}]\!] \,\} \,\}$$

A Semantic ABI: Basics

$$\boxed{\mathcal{O}[\![\mathbb{Z}]\!](\ell) \triangleq \exists\, n.\ \underbrace{\ell \mapsto n}_{\text{Ownership}}}$$

Object Relation:
Repr. of type in memory

A Semantic ABI: Layout

$$\mathcal{O}[\![T_1 \times T_2]\!](\ell) \triangleq \exists\, \ell_1, \ell_2. \\ \ell \mapsto \ell_1 \star \ell + 1 \mapsto \ell_2 \star \mathcal{R}[\![T_1]\!](\ell_1) \star \mathcal{R}[\![T_2]\!](\ell_2)$$

Adjacency

Separation

Reference Relation:
Object + Counter

Records require strategy for field order

You Can't Spell *Interoper<u>ability</u>* Without *ABI!*

Email: ahwagner@ccs.neu.edu

Web: andrewwagner.io

A Semantic ABI: Ownership + Sharing

“Jump” Modality: 1 share of ref. counter

$$\boxed{\mathcal{R}[\![T]\!](\ell) \triangleq @_{\ell}\ \underbrace{\mathcal{O}[\![T]\!](\ell + 1)}_{\text{Resource it counts}}}$$

Location of ref. counter

✅ *Can* read and increment through jump

❌ *Cannot* write or decrement through jump

A Semantic ABI: Ownership + Sharing

$$\mathcal{R}[\![T]\!](\ell) \star \mathcal{R}[\![T]\!](\ell) \approx \ell \; \underbrace{\boxed{1 \; + \; 1}}_{\text{Owned}} \underbrace{\boxed{\mathcal{O}[\![T]\!]}}_{\text{Shared}}$$

$$\boxed{\mathcal{R}[\![T]\!](\ell) \triangleq \underset{\text{Location of ref. count}}{\overset{\text{“Jump” Modality}}{@_{\ell}}} \; \underbrace{\mathcal{O}[\![T]\!](\ell + 1)}_{\text{Resource it counts}}}$$