Today’s software systems are complex and often made up of parts written in different programming languages with different computational and memory management strategies. This allows programmers to combine different languages and choose the most suitable one for a given problem. It also allows the gradual migration of existing projects from one language to another, or to reuse existing source code.

While this flexibility offers clear advantages, it also introduces significant challenges, as different programming languages may have fundamentally different implementations and may use different runtime environments, which are hard to combine. As a consequence compositing parts written in different languages often results in complex interfaces between languages, insufficient flexibility, poor performance, and hard to diagnose errors. This lack of interoperability support can lead to subtle bugs and security vulnerabilities, especially in large or long-lived systems.

Existing foundations for interoperability often focus on giving guarantees about the compiled code (safety-after-compilation) and they cannot express static typing guarantees (source-level-safety). This view makes it difficult for programmers to diagnose errors in their source programs, as errors are detected only after compilation; furthermore, this approach fails to capture interoperability between languages with different, independent backends, e.g. VMs or different hardwares like GPUs.

We propose a type-theoretic framework for cross-language interoperability where we express and statically reason how languages interact on the type level and we retain the operational semantics of each sub-language. In particular, we leverage the down-shift modality of adjoint logic to model foreign function calls where a program in language B passes control to the runtime engine of language A. To call a function in language A, we must also be able to convert B's data to A and vice versa. Our view allows programmers to implement different data-level conversions, in particular for more complex data-structures, and control their usage. Our system is parametric to a user-defined collection of languages and their accessibility relation, which controls how languages interact with each other. We sketch the statics and an operational semantics of the cross-language interoperability foundation together with properties such as type-safety and accessibility safety, which ensures that languages respect their user-defined boundaries. Thanks to the modular design of our framework, our meta-theoretic results are also compositional.

This is joint work with Junyoung Jang (McGill) and Stefan Monnier (Universite de Montreal)