Implementing CppInterOp API, Exposing Memory Ownership and Thread Safety

Google Summer of Code 2026

Mentors: Vassil Vasilev, Aaron Jomy, Vipul Cariappa

Author: Kerem Şahin

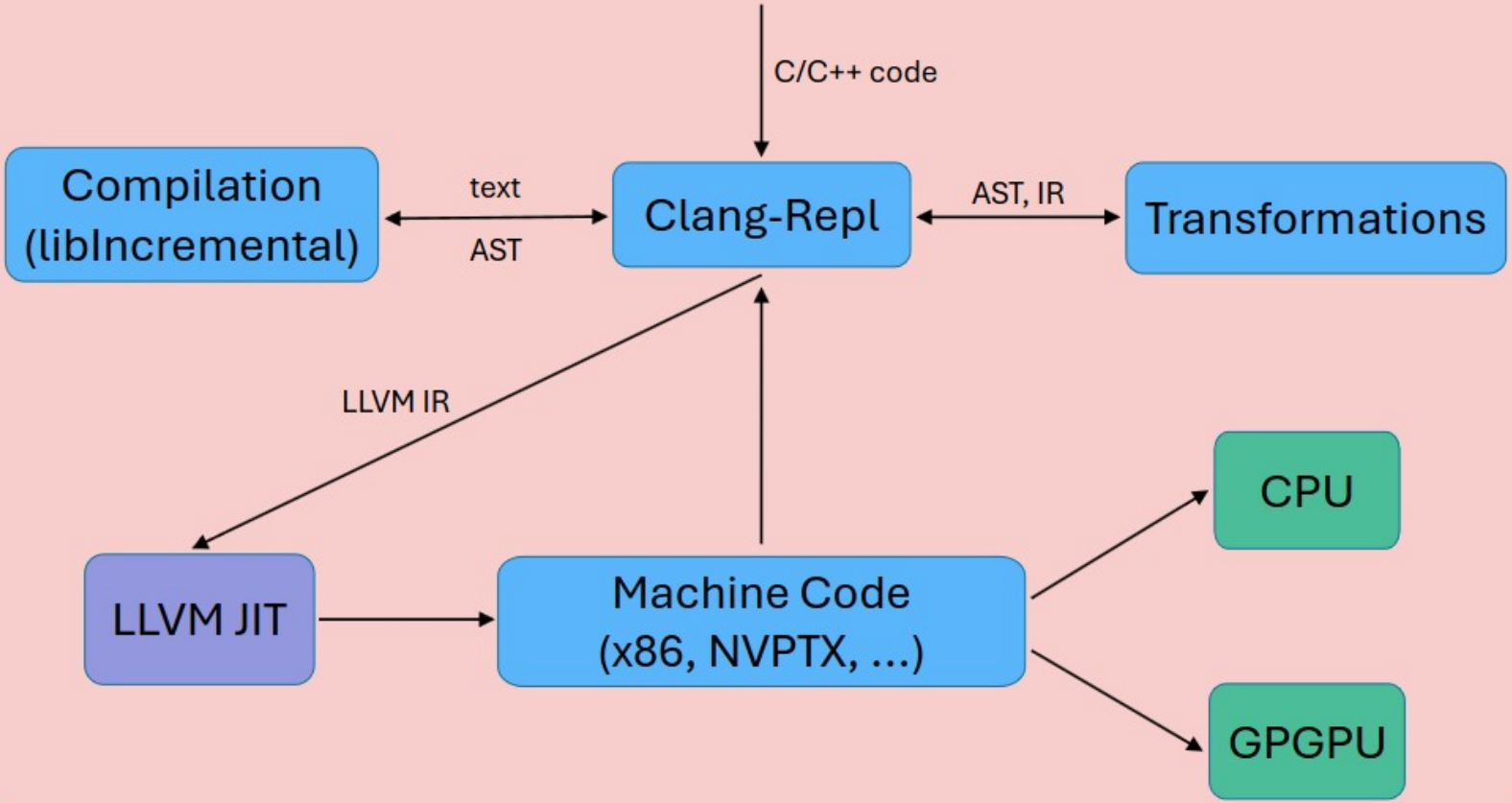
What is CppInterOp?

Compiler C|R Research

- CppInterOp is a project that aims to be a bridge between language binding tools and interactive compilers.

- There are two main purposes of CppInterOp:
  1) Compiling and executing C/C++ code at runtime using the Clang frontend and LLVM ORC JIT (Interpreting)
  2) Introspecting C++ declarations to expose information to language binding tools (Reflection)

CppInterOp Design: *Compiler-as-a-Service*



Compiler C|R Research

What does CppInterOp Currently Lack?

Compiler C|R Research

- CppInterOp's reflection functions currently do not expose any information about memory ownership behaviour or multi-thread compatibility of called functions.

- If this information is extracted from function bodies, the language binding tools could automate memory management and multi-threaded processes.

How to Implement?

Compiler C|R Research

- To detect this information the main way will be traversing the AST of function declarations, using Clang’s internal tools such as RecursiveASTVisitor.

- Other complementary ways are inspecting hints developers left such as function attributes, and IR attributes inferred by the LLVM optimizer.

Challenges

Compiler C|R Research

- The main challenge is that, while it is possible to extract information from the AST, this does not necessarily mean the analysis tool is precise for every kind of compilable function.

- There are some limits for static analysis because of computability boundaries rooted in undecidability, so the goal is building a tool such that it behaves correctly for every computable case, but does not give false positives for undecidable situations.

What is Next?

- The next step is integrating these APIs into Compiler Research's Python-C/C++ binding project cppyy, which powers PyROOT (the Python interface to the ROOT framework).

Thanks for listening :)