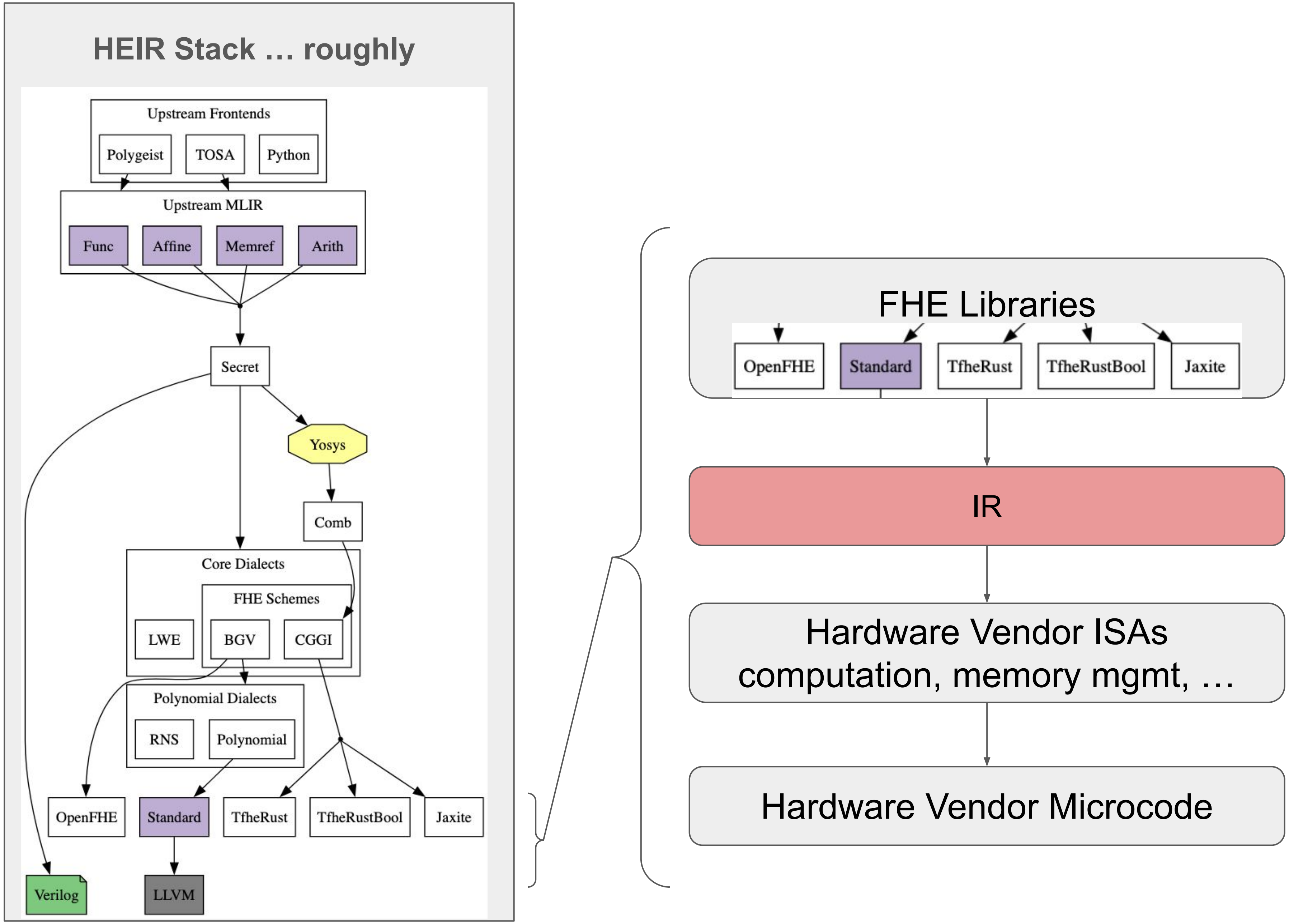
HES 7: Low-Level, Common Hardware Interfaces for FHE

Proposing a Joint **Standardization** Initiative Across the FHE Compiler, Library, and Hardware Accelerator Communities

Context: Below the software stack lurks…*acceleration hardware*



Objectives of the Session

- Introduce FHETCH
- Motivate a common hardware IR
- Sketch a product-oriented, common-sense approach to developing a *de facto* standard hardware IR
- Discuss the IR idea, starting points, and how we can work together
- Call the community to participating in the IR standard effort

Introducing FHETCH

- FHE Technical Consortium for Hardware - inquiries@FHETCH.org
- A growing industry-led community of FHE hardware and software providers, application owners, and end users
  - Founding members: Niobium Microsystems, Optalysys, Chain Reaction
  - Join us!
- Dedicated to advancing commercial availability of FHE products

- Insight: Customers are seeking *flexible* and *open* solutions
  - Especially in a developing market
- Response: Create a market ecosystem for FHE hardware acceleration
  - Hardware interface standards
  - Benchmark suites
  - Optimization toolchains
- “Create the market, and then compete in it”

A Problem for FHE Adoption: the Hardware Abstraction

- Modern FHE compilers target FHE libraries…

  …which target ISAs of hardware (CPUs, GPUs)

- Current lack of commonality makes the latter difficult, costly
  - Waste in development
  - Software obsolescence
  - Unnecessarily complex tools
  - Customer perception of being “locked-in” to a specific vendor
- Emerging FHE co-processors and proprietary ISAs will make this problem worse

Optimizing for Custom HW is *hard…*

especially if you’re not the HW vendor

- Widely different cost models for computation, memory
- Different performance “pinch points”
- Different data types and operation models

Solution Idea: A Common Intermediate Representation

- FHE software ecosystem vendors target a *lingua franca* that HW and SW vendors agree on
- FHE hardware vendors implement that *lingua franca* by
  - Directly matching the IR with a hardware ISA

    or
  - Providing a shim transformer to the IR
- Reduced waste by shared insight
- Optimization and other tools can operate directly on the IR
- Customers can now “plug-n-play” libraries and hardware
- Emerging FHE co-processors naturally fit the IR concept

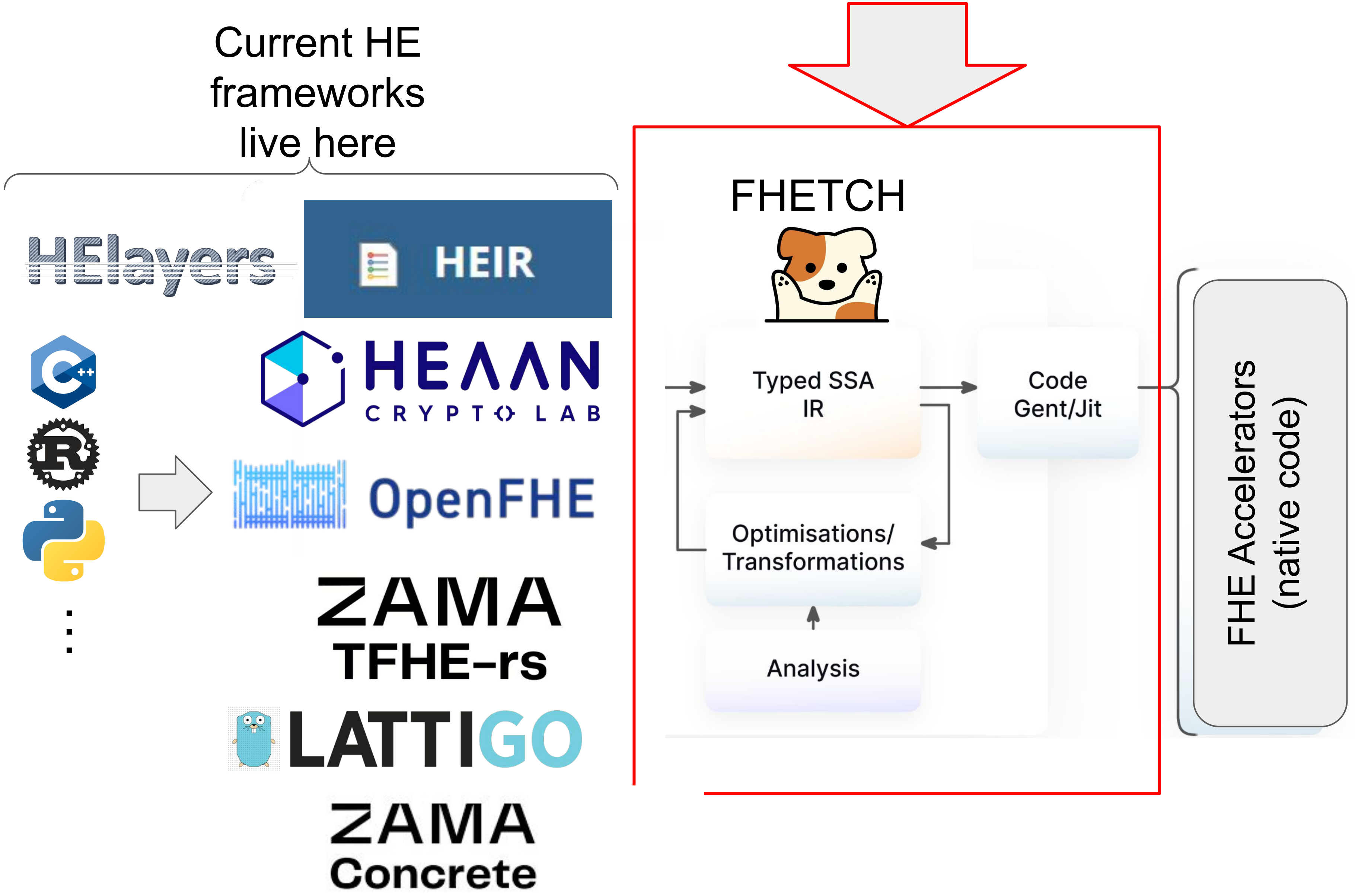
Initiative: the FHETCH IR

- Technical committee with members from software vendors, hardware vendors, application makers, end users
- Standardization practices focused on *productization*
- Leverage ideas from LLVM…
- …to drive a *de facto* industry standard

Agenda, flexible

- Motivate a common hardware IR
- Sketch a product-oriented, common-sense approach to developing a *de facto* standard hardware IR
- Discuss the IR idea, starting points, and how we can work together
- Call the community to participating in the IR standard effort

FHETCH IR and HE Frameworks



Current HE frameworks live here

FHETCH

Typed SSA IR

Code Gent/Jit

Optimisations/ Transformations

Analysis

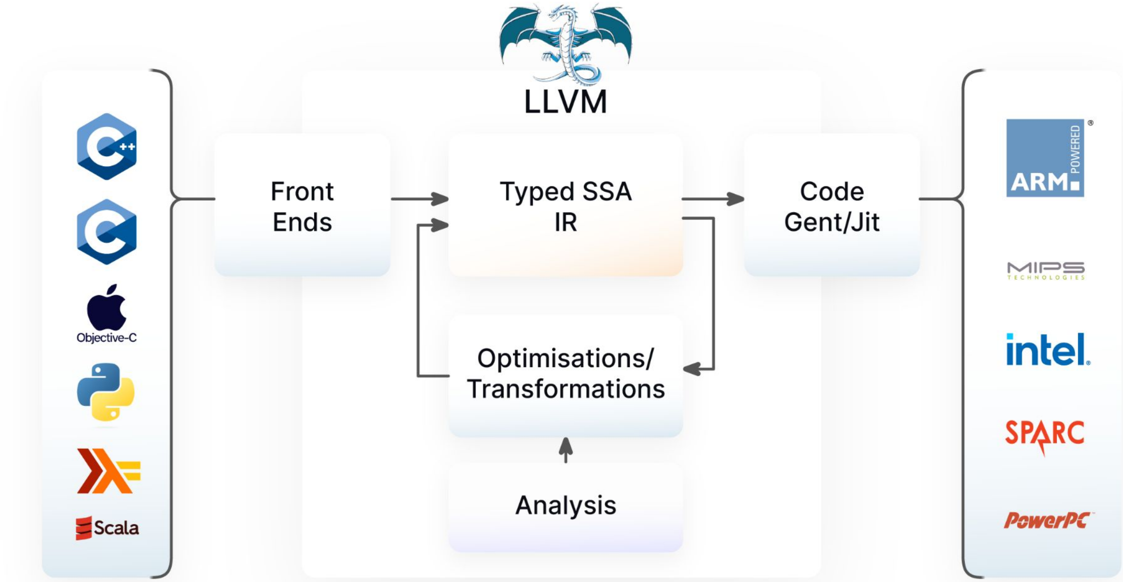
FHE Accelerators (native code)

Desiderata for a common FHE IR

- A minimal viable IR
  - Few operations, few data types - makes code generation tractable
- …plus gadgets
  - Complex operations called out explicitly
  - Each gadget fully emulatable using the base abstraction
  - Also allows HW vendors to implement gadgets directly, as secret sauce
- A natural set of simple data types
  - Explicit, understandable semantics
  - Avoids bias for one scheme over another
- Simple structure to enable downstream optimizations
- Built-in metadata: parameters, modulus chains, instruction-level parallelism guidance, memory pre-fetching hints

Inspiration: LLVM

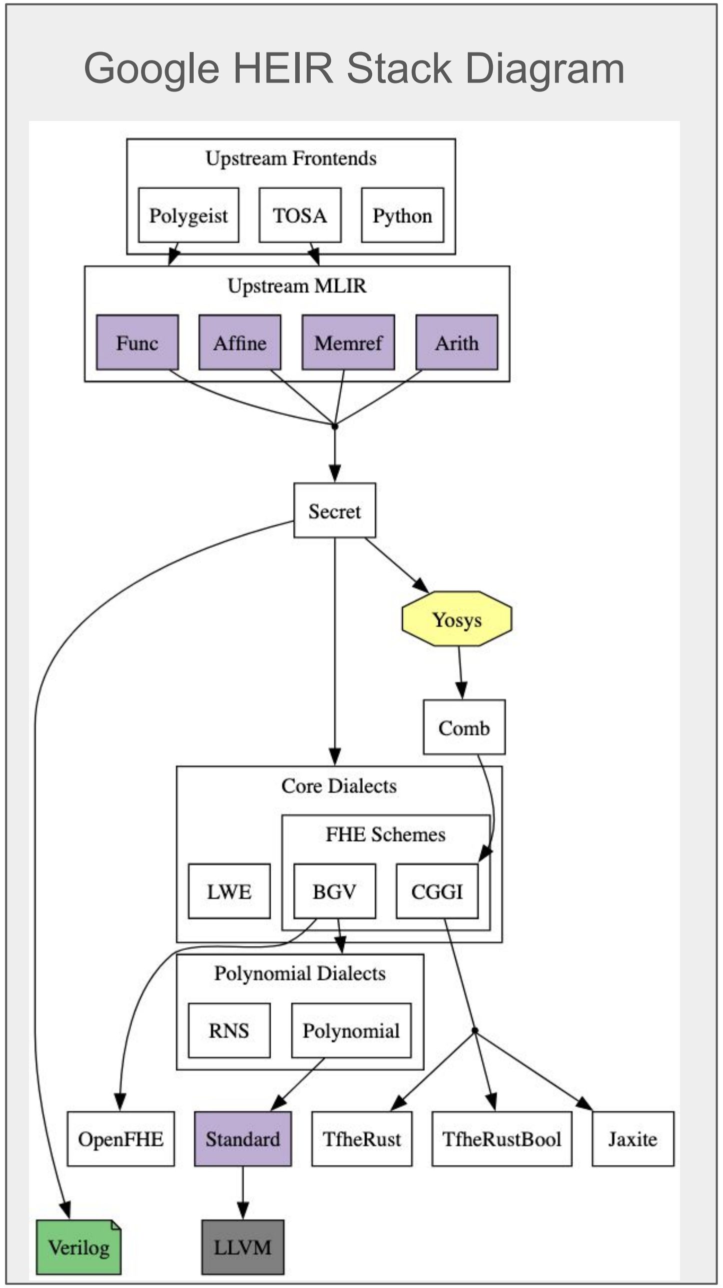
- Defines an IR that can be incrementally processed, refined and optimized
- Language-independent instruction set and type system
  - Basic types such as integer, some derived types
- Target-independent IR, compiled onward to “concrete” code
- Tools for memory locality optimization, parallelization, …



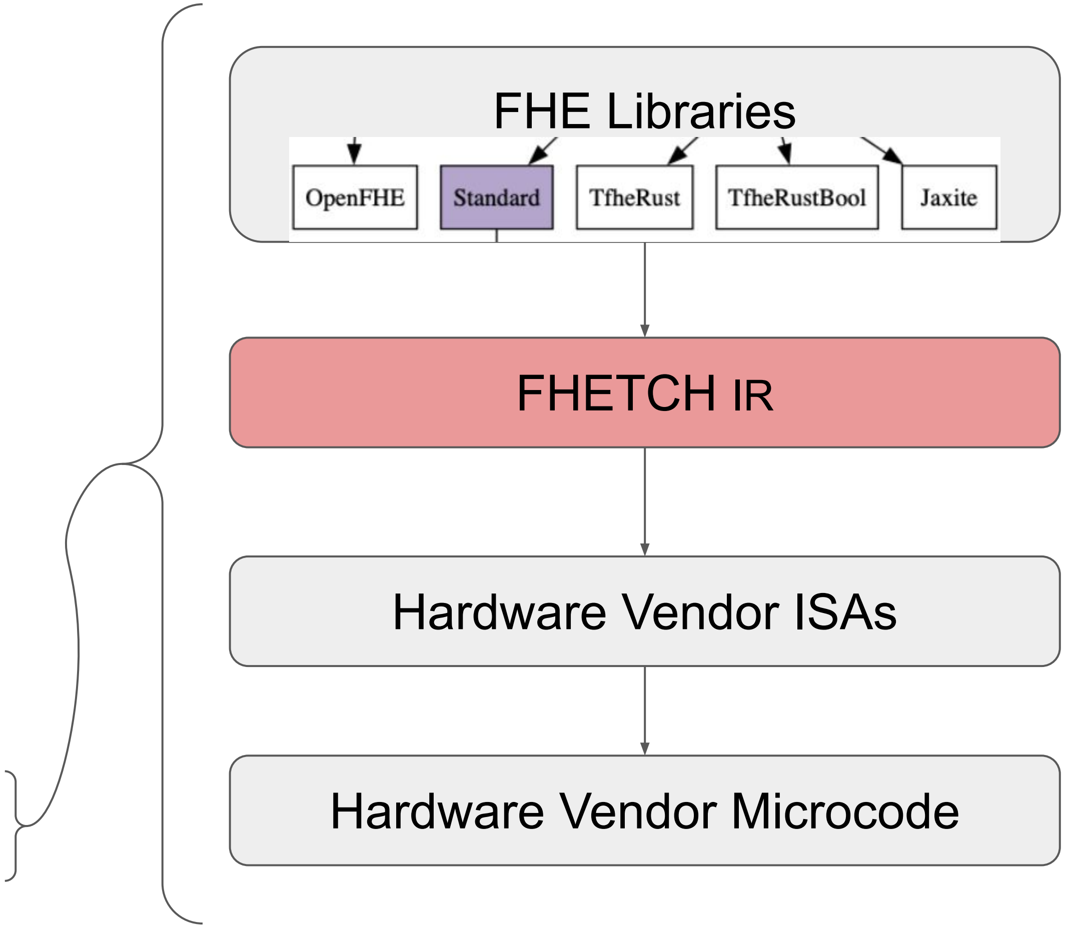
Sketch of a *Potential* FHETCH IR

- Multiple equivalent forms: human-readable, dense bitcode for serialization
- Polynomial as the foundation data type
  - non-RNS ciphertexts represent as Nx2 arrays
  - RNS ciphertexts represent as Nx2xL arrays (L = limbs in RNS)
  - Compact representation of constant values
- Base: minimal operations on polynomials
  - ADD, ADDI, MUL, MULI, Rotation, …
- Extendable Inline Macros/Gadgets
- Infinite register set, single static assignment (SSA)
- Parameters, modulus chains, instruction-level parallelism guidance, memory pre-fetching hints

Where FHETCH IR Fits In



Google HEIR Stack Diagram



FHE Libraries

FHETCH IR

Hardware Vendor ISAs

Hardware Vendor Microcode

Call For Action

- Participate! Contact us at inquiries@FHETCH.org
  - Hardware vendors
  - Library vendors
  - Application makers
  - Compiler ecosystem partners
  - Commercial end users
- Review & Refine
  - Assess draft definitions as they emerge
- Define a common IR before natural diversity prevents it