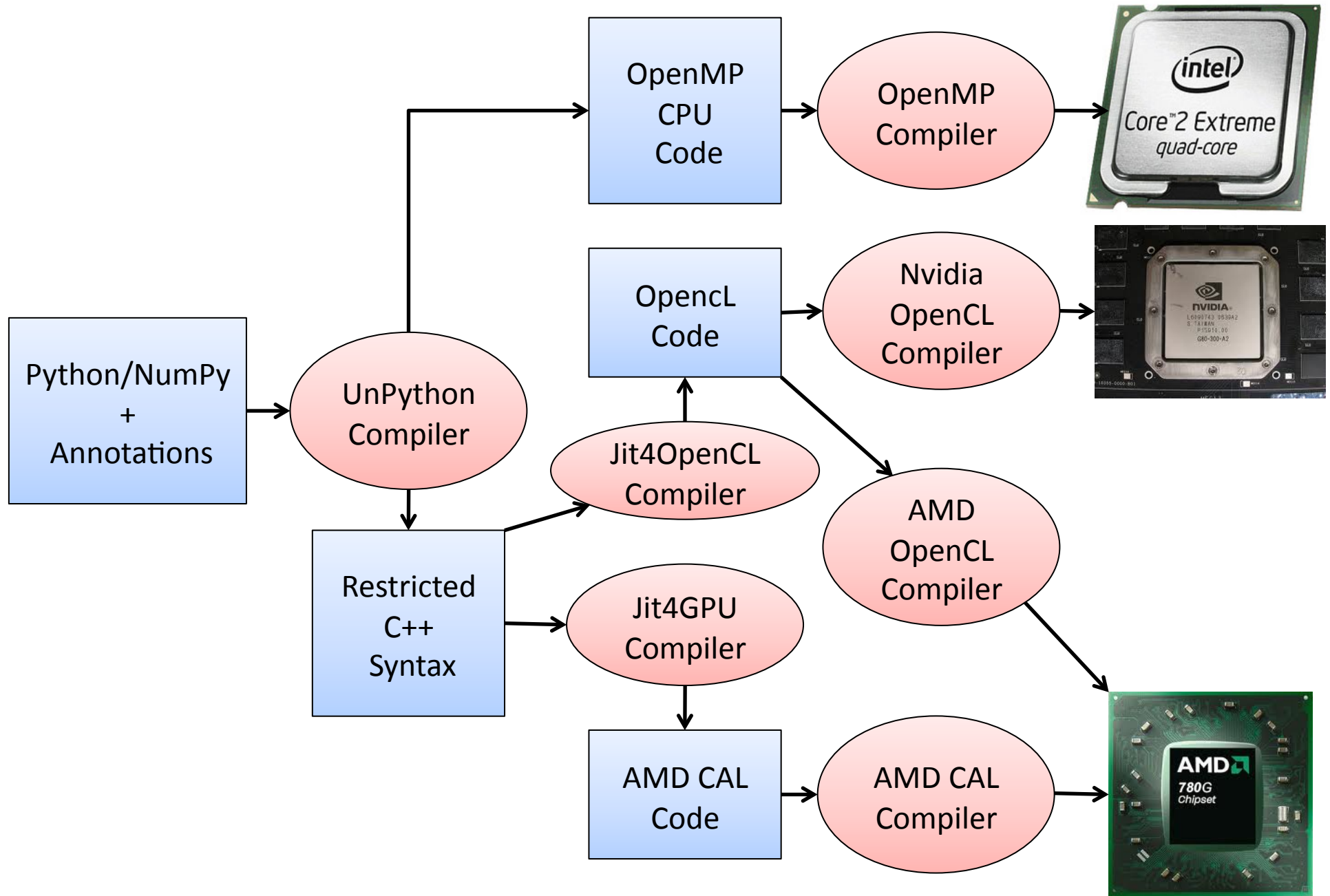# Compilers and VMs for Programming Environments Used by Scientists

José Nelson Amaral – Univ. of Alberta

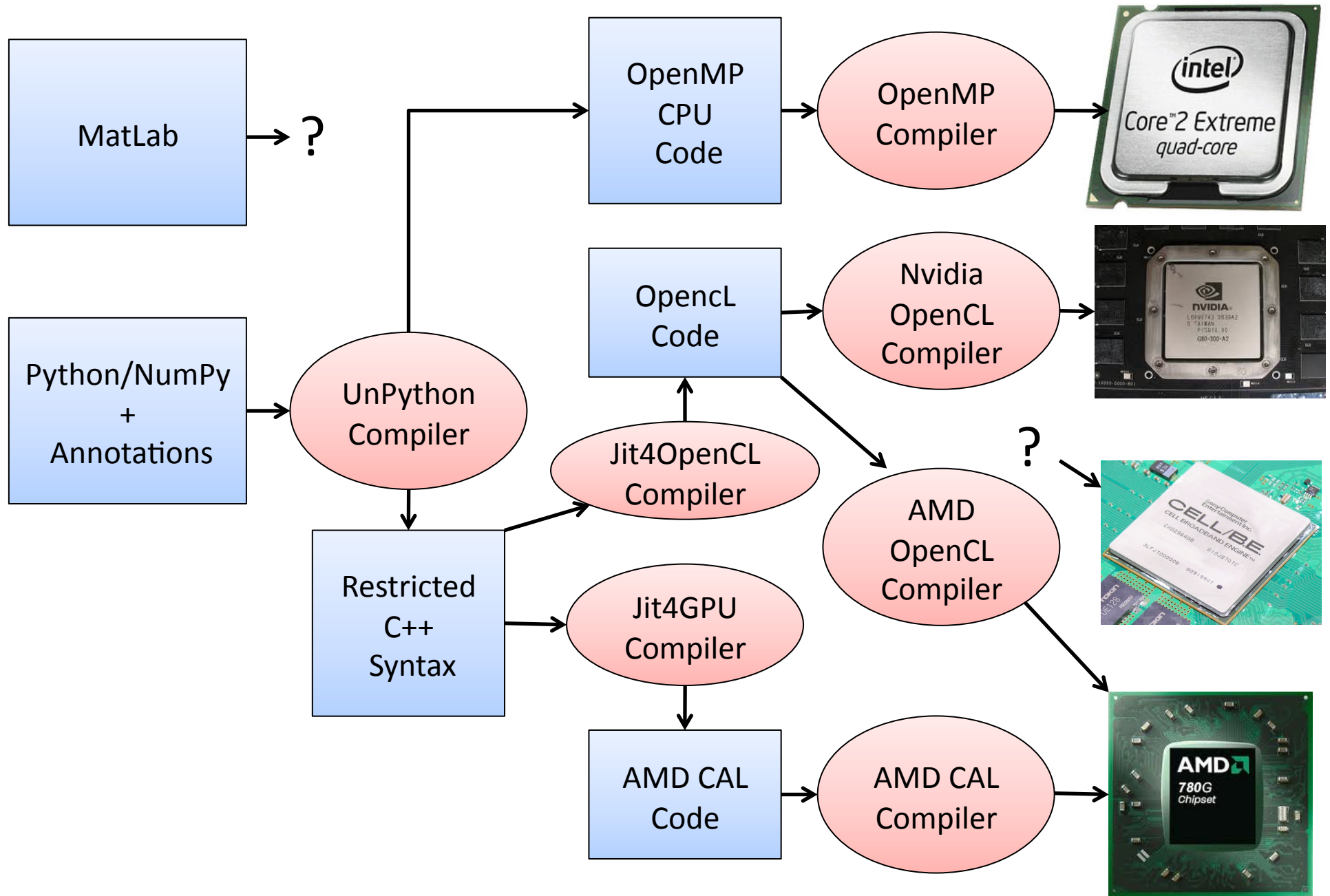(with input from Laurie Hendren and
Rahul Garg – McGill Univ.)

Why Python in Computational Chemistry?

- Easy integration of code written in C/C++;
- Composibility (reuse)
  - New students quickly get results
  - Build from existing modules, instead of building from scratch
- Easy to learn the language
- Visualization (very important)
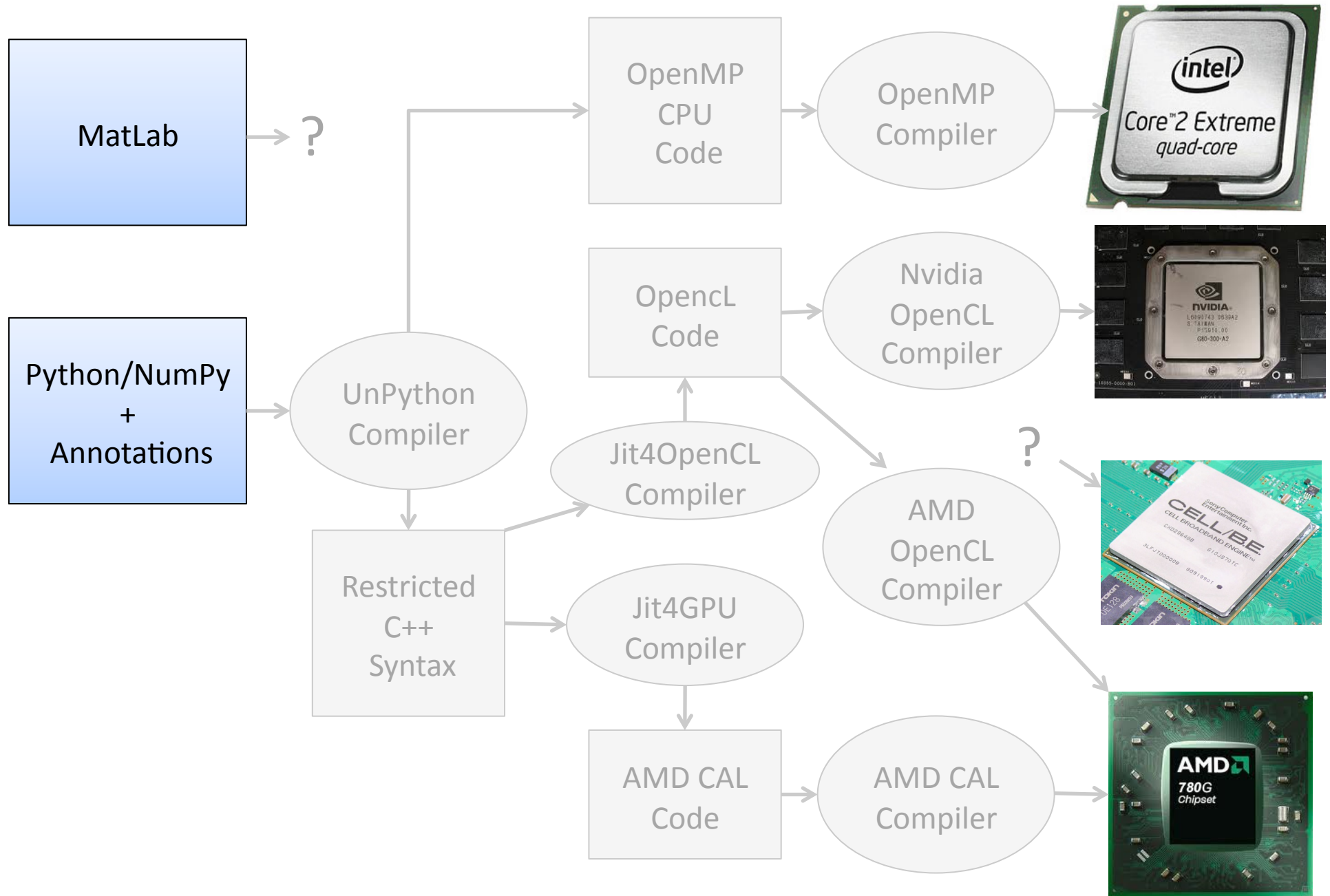- Drawbacks:
  - Performance of Python modules

Completing the Bridge

Completing the Bridge
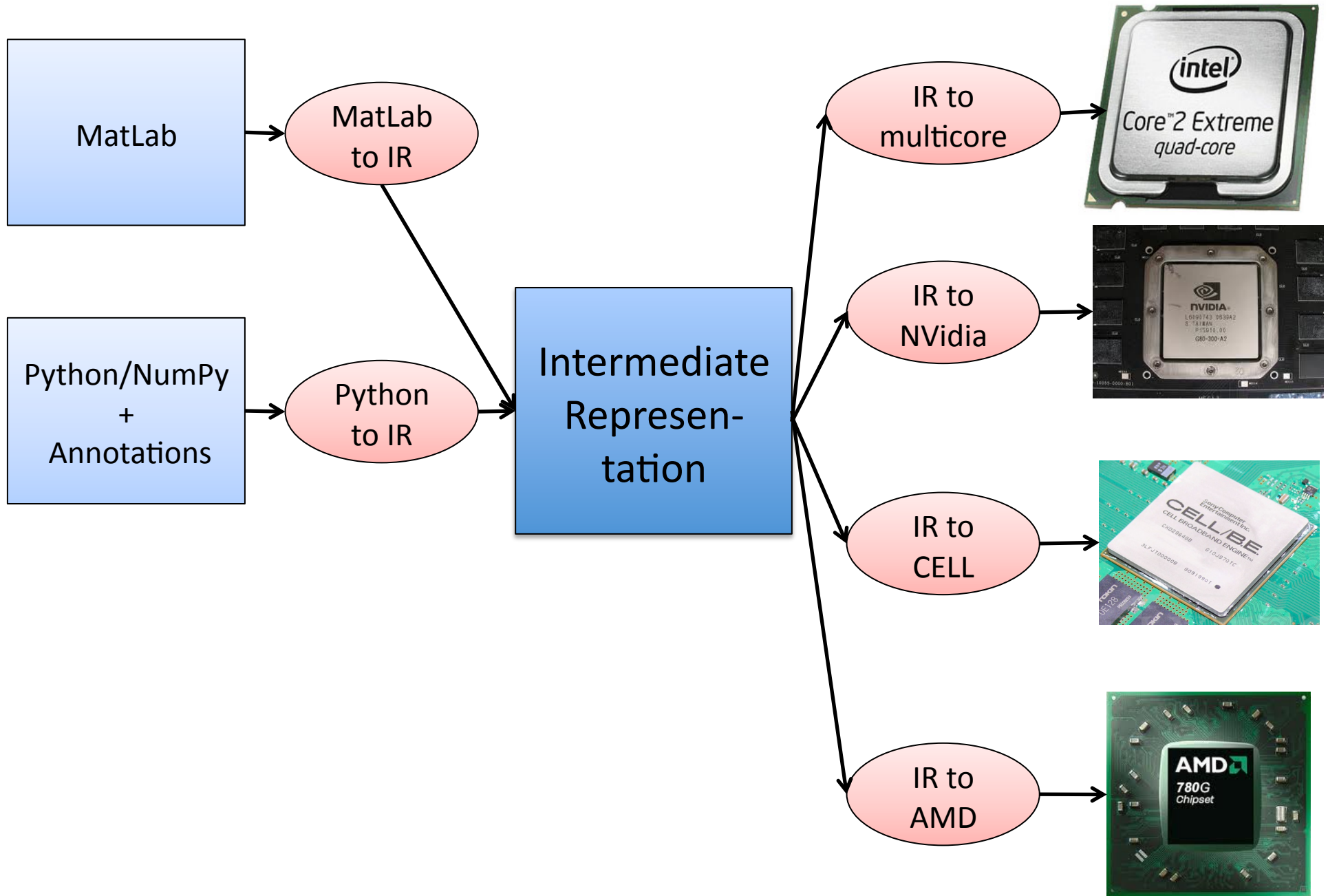
MatLab

Python/NumPy + Annotations

?

UnPython Compiler

Restricted C++ Syntax

Jit4OpenCL Compiler

Jit4GPU Compiler

OpenMP CPU Code

OpenCL Code

AMD CAL Code

OpenMP Compiler

Nvidia OpenCL Compiler

AMD OpenCL Compiler

AMD CAL Compiler

?

What we have

MatLab

? →

OpenMP CPU Code → OpenMP Compiler →

Python/NumPy + Annotations → UnPython Compiler

OpenCL Code → Nvidia OpenCL Compiler →

Jit4OpenCL Compiler

?

AMD OpenCL Compiler

Restricted C++ Syntax → Jit4GPU Compiler

AMD CAL Code → AMD CAL Compiler →

What we want

MatLab → MatLab to IR

Python/NumPy + Annotations → Python to IR

→ Intermediate Representation

→ IR to multicore

→ IR to NVidia

→ IR to CELL

→ IR to AMD

Issues with MATLAB

- No formal specification and no official publicly-available implementation
  - McLab group is creating on a best-effort specification;
- Designed for interpretation not for efficient execution;
- Many dynamic features
  - But many of those are not needed or are misused;

Goals for Hendren's group

- Create a modern and open research framework for MATLAB:

  – Focus on dynamic features that are effectively used

  – Facilitate creation of extensions for domain-specific languages

  – Create a suitable IR
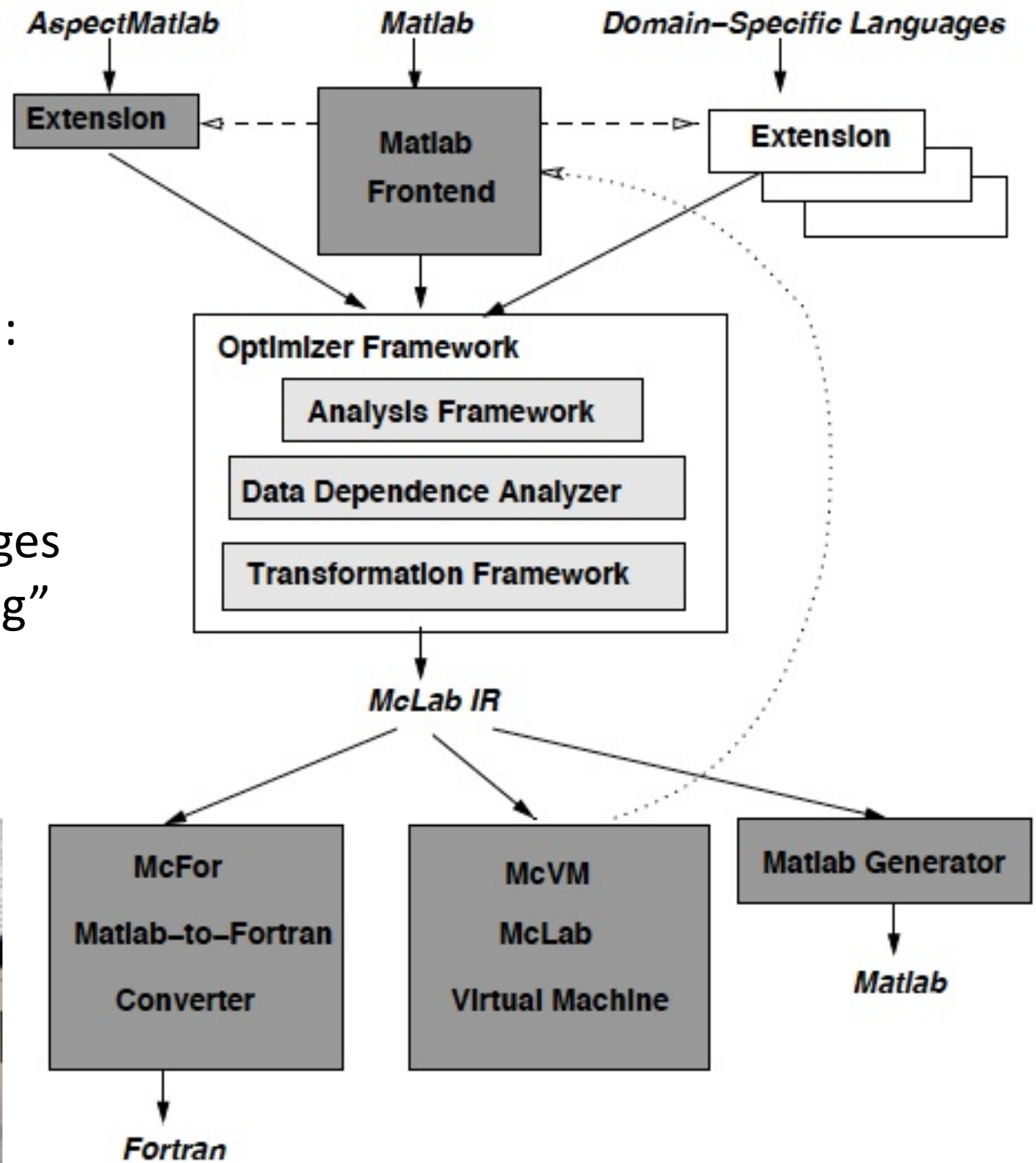
Hendren's McLab Vision:

"start with the languages that scientists are using"

Challenges

- Type inference
- Dynamic features
- Loose Language specifications
- Diverse, complex, and evolving hardware platforms