Compiler Research in the Open: Connecting People, Projects, and Progress

Vassil Vassilev
{ Princeton, compiler-research.org }

This work is partially supported by National Science Foundation under Grant OAC-2311471

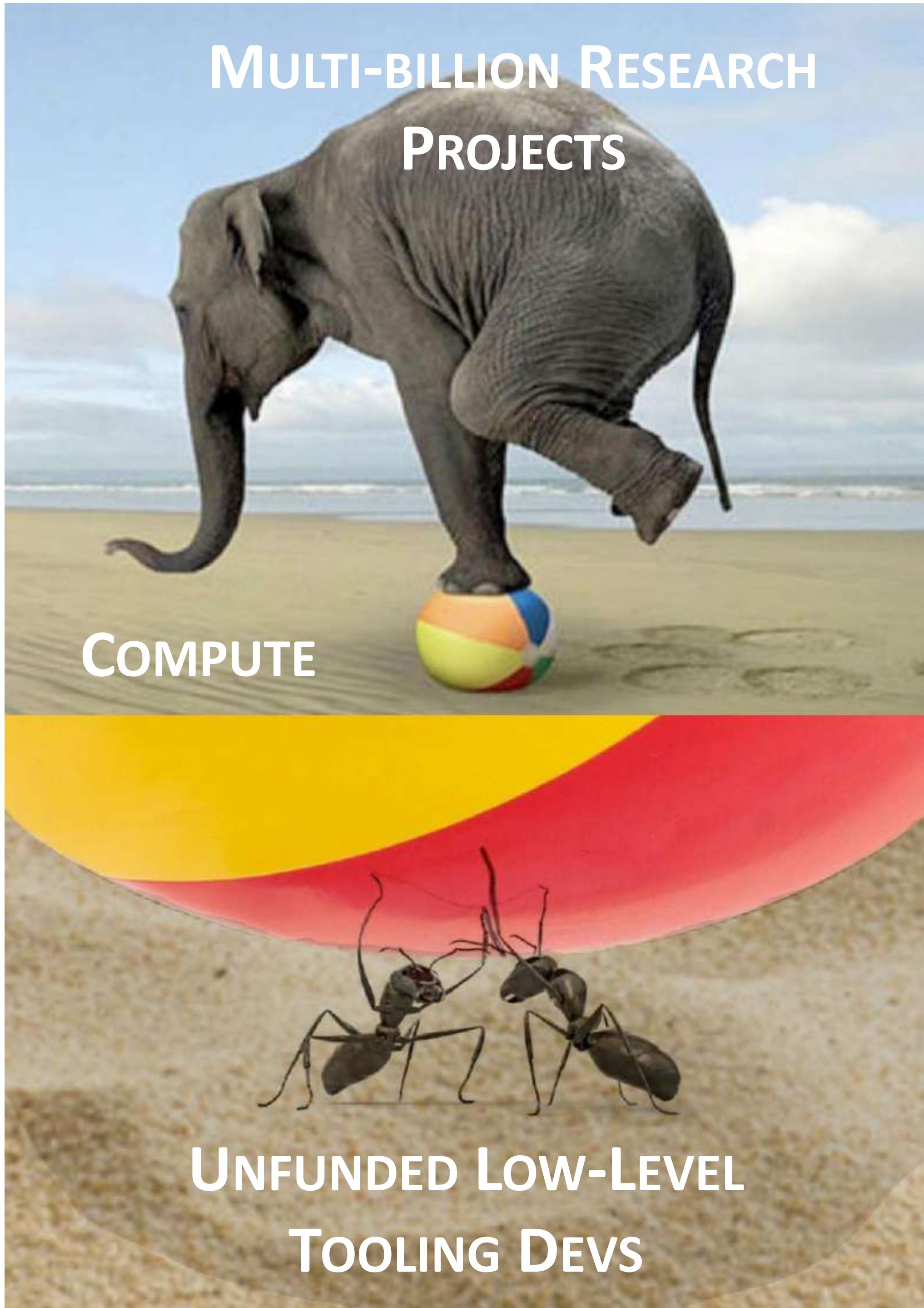
Why Low-Level Tooling Is Foundational

- Modern scientific discovery depends on large-scale computation
- Systems tooling mediate C++, CUDA, and heterogeneous hardware
- Control at the compiler/runtime layer determines efficiency, portability, and sustainability

These tools don’t just run science: they increasingly shape how science is done

The Problem

- Expertise in compilers and systems engineering is rare and disconnected
- Only a small number of individuals can navigate and evolve core tooling
- Early-career researchers lack mentorship to enter the field
- Entire scientific communities rely on toolchains they do not control and cannot adapt
- This leads to duplicated effort, technical debt, and limited strategic independence of critical infrastructure


Multi-billion Research Projects
Compute
Unfunded Low-Level Tooling Devs

Our Initiative

compiler-research.org aims to:

- Make compiler research visible and connected
- Support impactful open-source R&D
- Train and mentor the next generation of compiler engineers
- Build bridges between academia, industry, and scientific domains

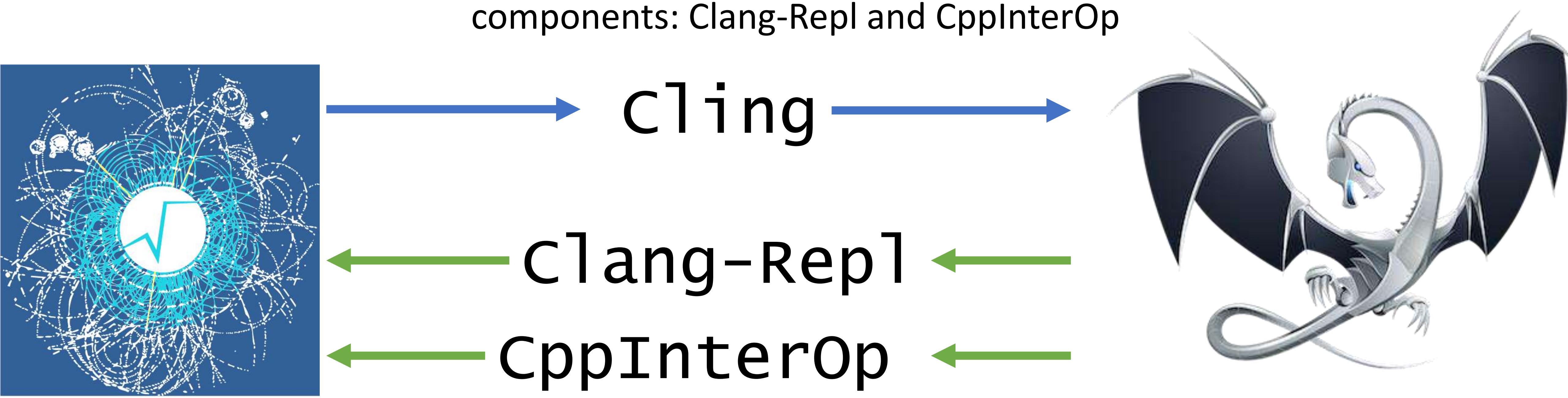
Focus Areas

- Transform domain-specific tools into upstream, shared infrastructure
- Enable hybrid language ecosystems without performance penalties
- Make interactive, heterogeneous, and differentiable programming practical
- Build foundational compiler technology for scientific workflows

System Programming

From Domain-Specific to Shared Infrastructure and Back

Moved parts of Cling upstream as Clang-Repl and they come back to ROOT as two components: Clang-Repl and CppInterOp



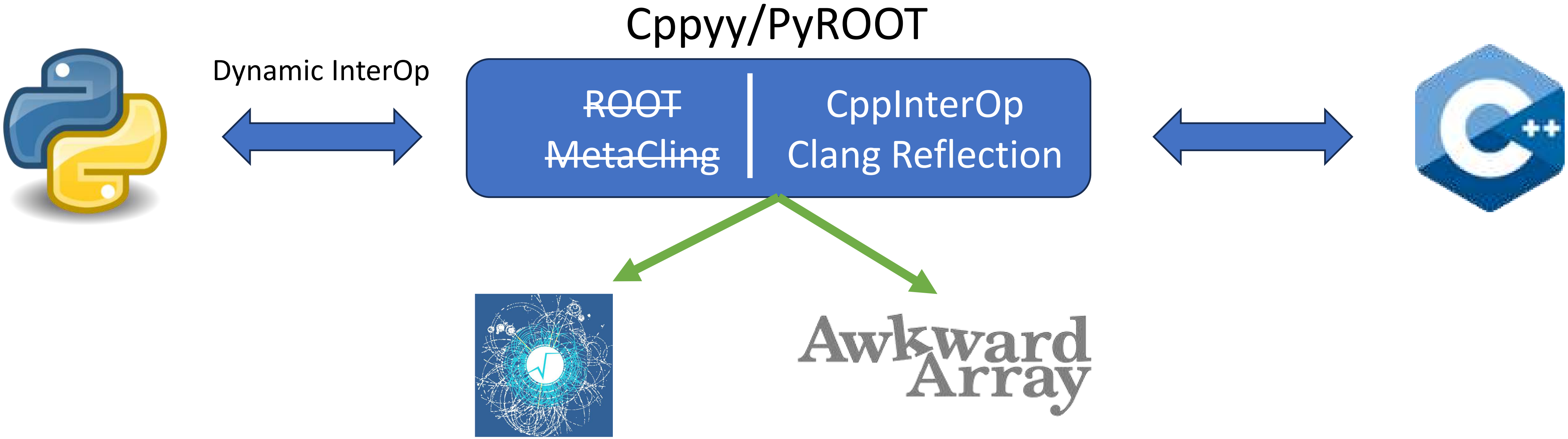
Cling

Clang-Repl

CppInterOp

https://root.cern/blog/cling-in-llvm/

Enable hybrid language ecosystems **horizontally**



Cppyy/PyROOT

Dynamic InterOp

~~ROOT~~

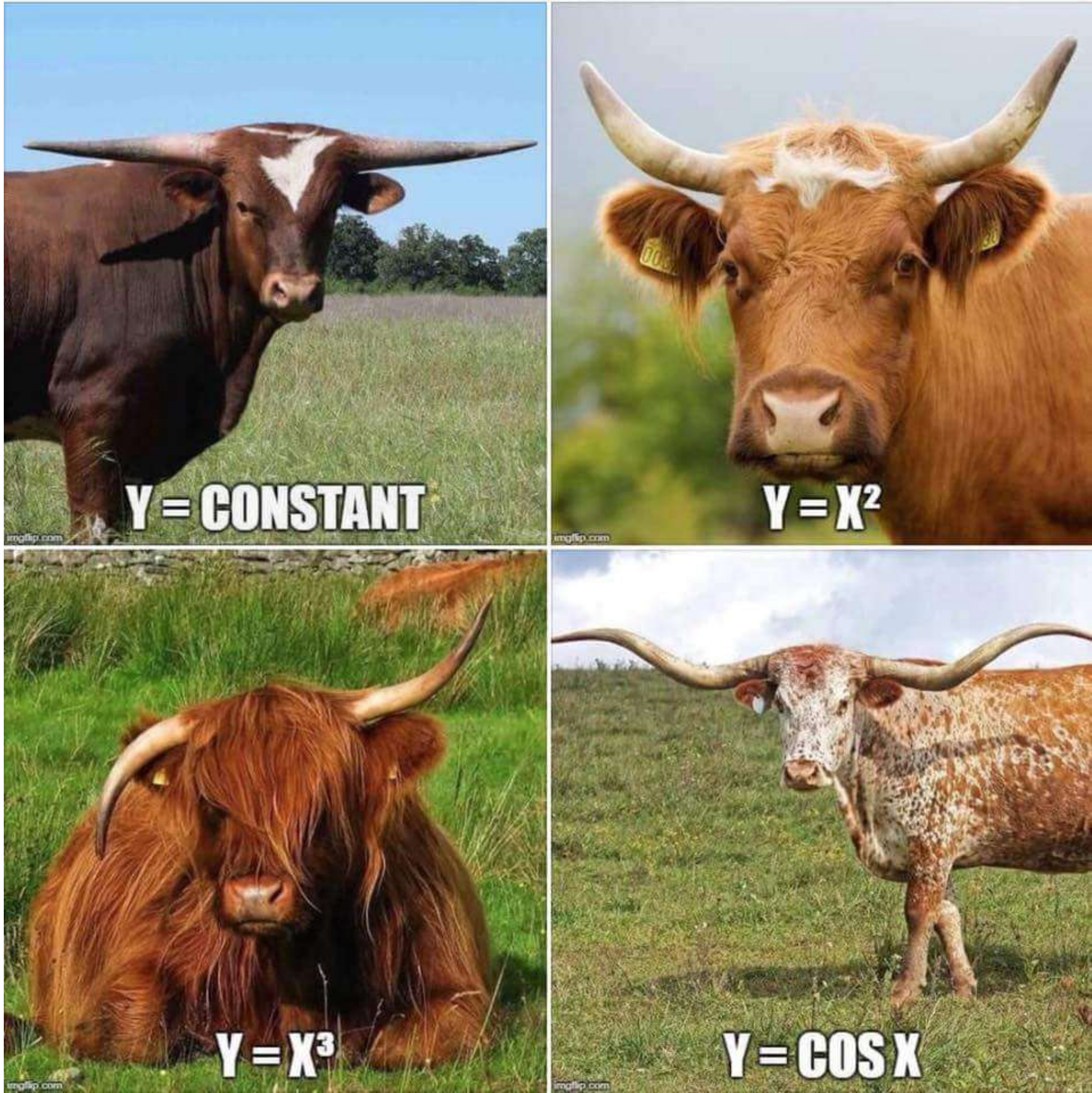
~~MetaCling~~

CppInterOp

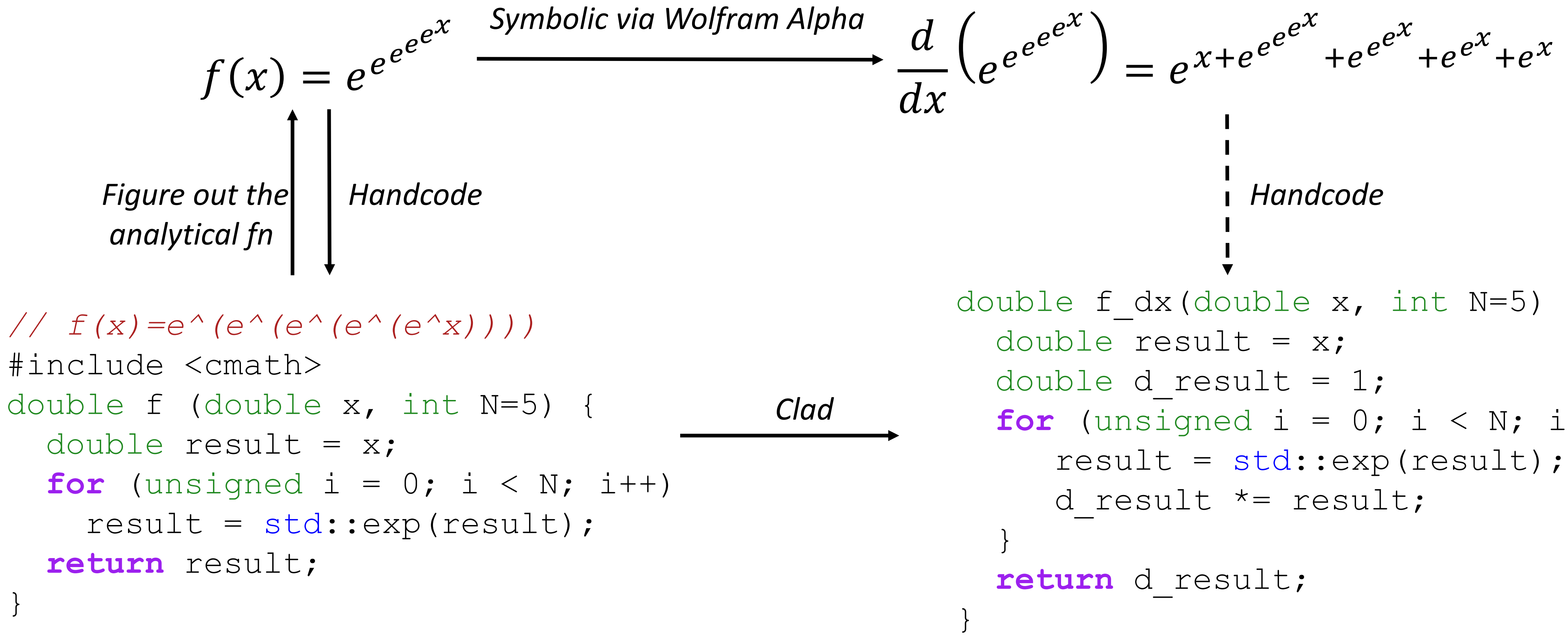
Clang Reflection

Reworked cppyy is coming back as a general capability layered directly on top of the compiler. Easier to install, deploy and develop. See Aaron & Vipul’s talk on Thursday.

Differentiable Programming



Automatic Differentiation & Clad



$f(x) = e^{e^{e^{e^{e^x}}}}$

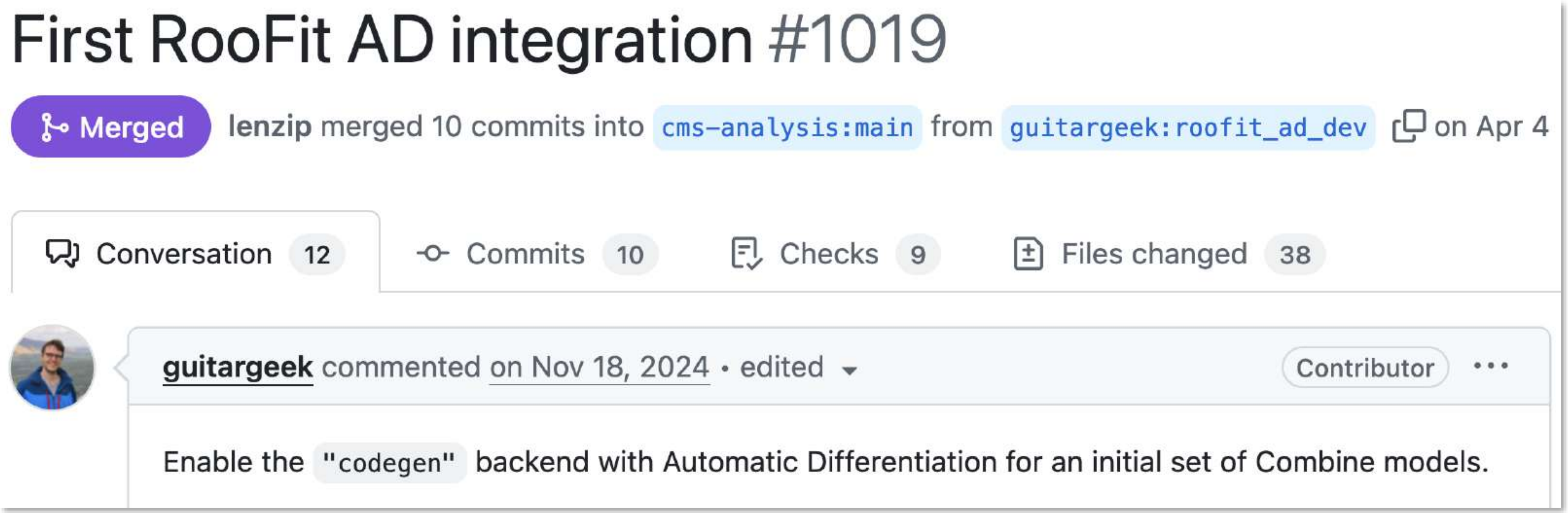
$\frac{d}{dx}\left(e^{e^{e^{e^{e^x}}}}\right) = e^{x+e^{e^{e^{e^x}}}+e^{e^{e^x}}+e^{e^x}+e^x}$

```
// f(x)=e^(e^(e^(e^(e^x))))
#include <cmath>
double f (double x, int N=5) {
  double result = x;
  for (unsigned i = 0; i < N; i++)
    result = std::exp(result);
  return result;
}
```

```
double f_dx(double x, int N=5) {
  double result = x;
  double d_result = 1;
  for (unsigned i = 0; i < N; i++) {
     result = std::exp(result);
     d_result *= result;
  }
  return d_result;
}
```

Integration in CMS Combine

Work steered mostly via CAT hackathons. Thank you Aliya Nigamova and Piergiulio Lenzi!



First RooFit AD integration #1019

Merged lenzip merged 10 commits into `cms-analysis:main` from `guitargeek:roofit_ad_dev` on Apr 4

guitargeek commented on Nov 18, 2024 • edited

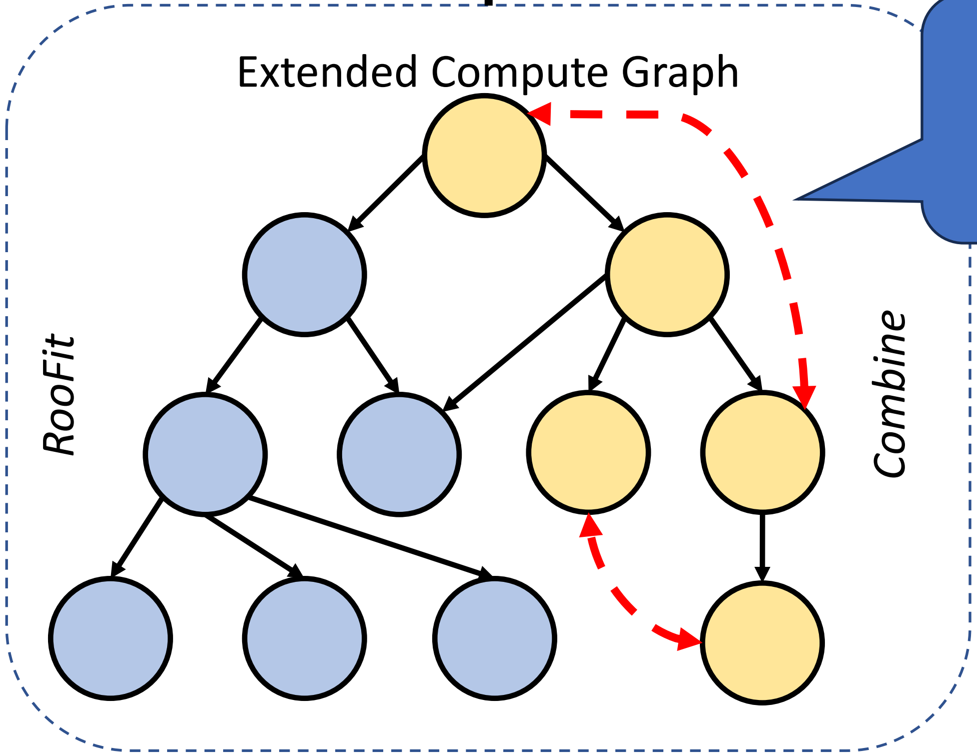
Enable the `"codegen"` backend with Automatic Differentiation for an initial set of Combine models.

CMS Combine Compute Graph

No need to recompile RooFit

Visit each node

```
void codegenImpl(RooAddPdf&, CodegenContext&);
void codegenImpl(RooChebychev&, CodegenContext&);
...
void codegenImpl(ProcessNormalization&, CodegenContext&);
void codegenImpl(FastVerticalInterpHistPdf2&, CodegenContext&);
```



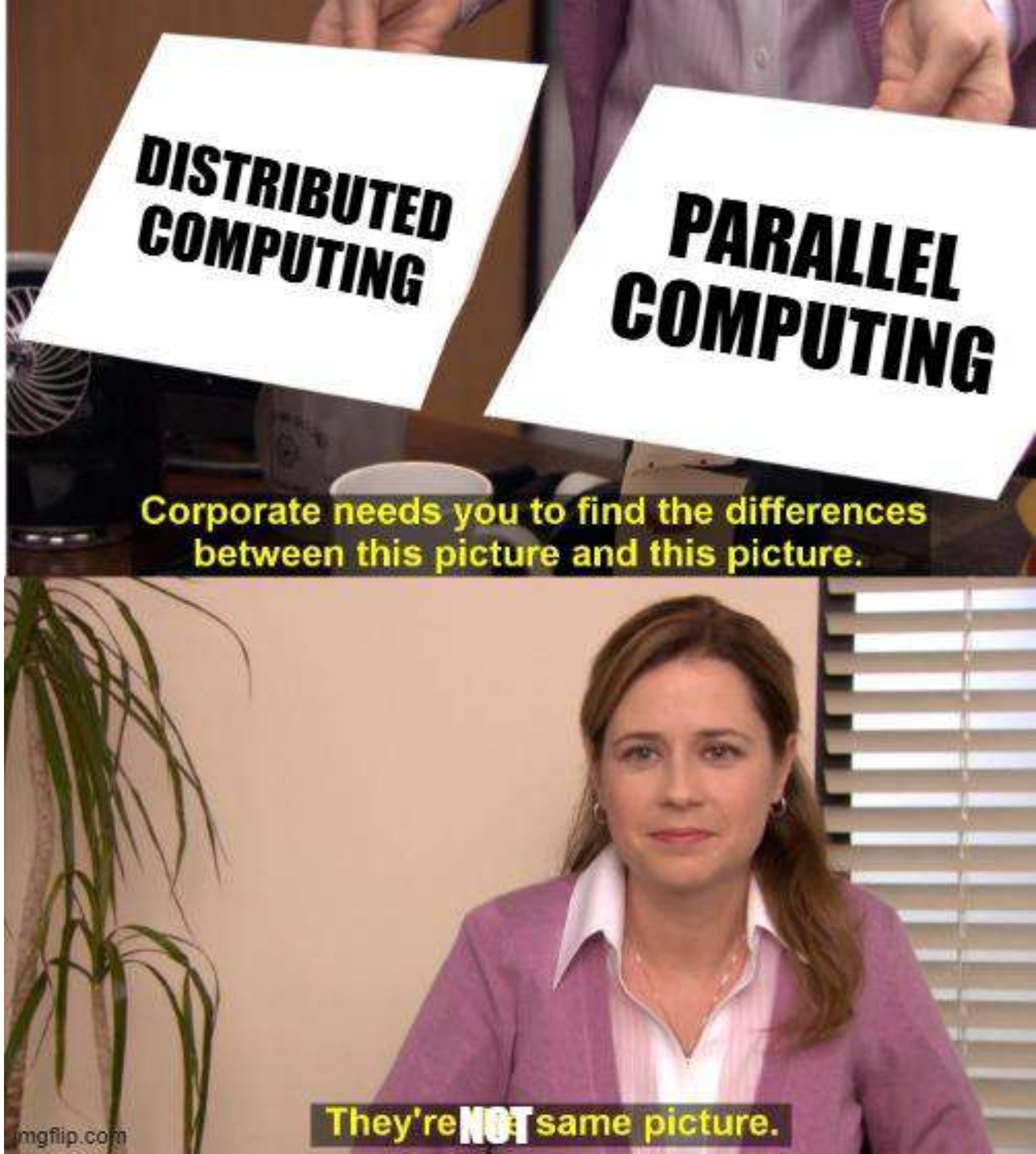
More cleanup is needed to avoid layering violations

Clad → Δ → *Optimize* → FCN

```
// ProcessNormalization::n_exp_bindijet_proc_qqH[ thetaList
otherFactorList=(r_qqH) ] = 0.95
const double t20 = RooFit::Detail::MathFuncs::processNormal
      0.950000, 1, 0, 1, t19, xlArr + 6, nullptr, xlArr + 6

// RooNLLVar[ pdf=model_s weightVar=_weight _weight_sumW2=_
for (int loopIdx1 = 0; loopIdx1 < 1; loopIdx1++) {
  nll_result += RooFit::Detail::MathFuncs::nll(t25, obs[3],
}
```

Parallel Programming


DISTRIBUTED COMPUTING
PARALLEL COMPUTING
Corporate needs you to find the differences between this picture and this picture.
They're NOT same picture.
imgflip.com

Expanding to Biology & Cancer Research

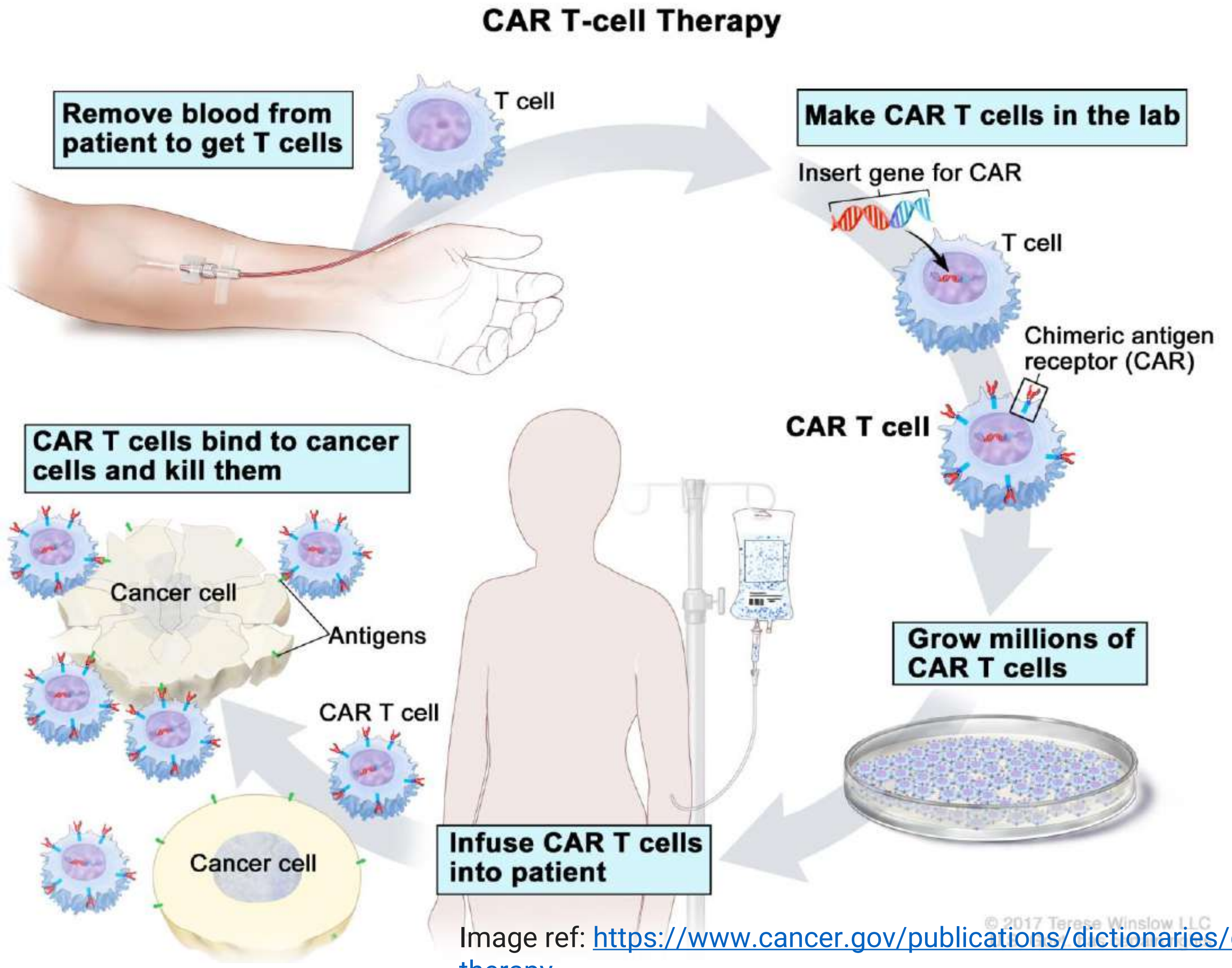

CAR T-cell Therapy
Remove blood from patient to get T cells
T cell
Make CAR T cells in the lab
Insert gene for CAR
T cell
Chimeric antigen receptor (CAR)
CAR T cell
Grow millions of CAR T cells
Infuse CAR T cells into patient
CAR T cell
CAR T cells bind to cancer cells and kill them
Cancer cell
Antigens
Cancer cell
© 2017 Terese Winslow LLC

Image ref: https://www.cancer.gov/publications/dictionaries/cancer-terms/def/car-t-cell-therapy

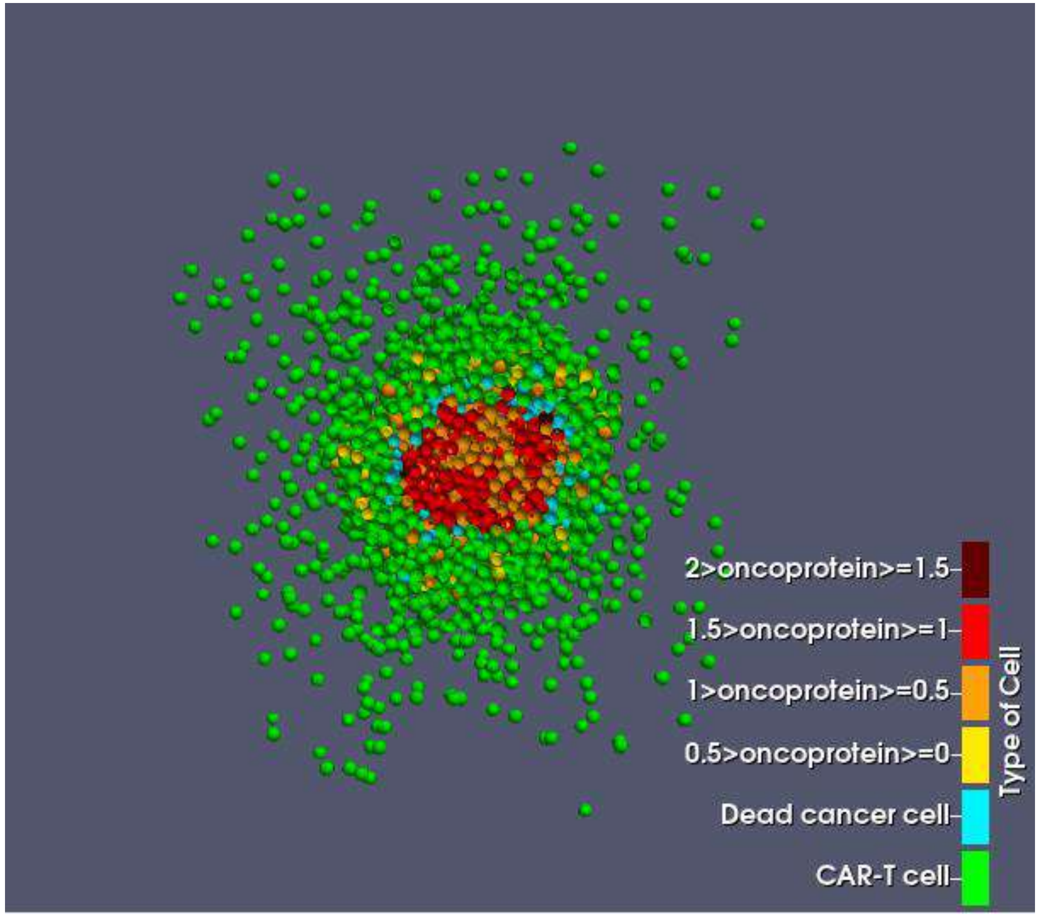
CAR T-cell therapy: A type of immunotherapy that engineers T-cells to recognize and kill cancer cells.

CARTopiaX: Open-Source Simulation of Tumor Organoids and CAR T-Cell Therapy

CARTopiaX

Biological Digital Twin Applications:

- The prototype is based on cutting-edge research published by Luciana Melina Luque et al.,2024 in [Nature]
- Implemented as an agent-based model with BioDynaMo and ROOT
- More in Salva’s talk on Friday



A 3D visualization of the sliced tumor and CAR T-cell treatment used in this example, rendered in ParaView

Open Mentorship Model

- Remote, long-term, project-driven training rooted in real research and production code
- Participants contribute to live compiler and systems projects, not isolated classroom exercises
- Growth pathway: contributor, maintainer, reviewer, researcher
- Supported through public funding, GSOC programs, and institutional collaborations
- Since 2020: 50+ contributors across 5 continents, now working at institutions including CERN, MIT, ETH, OpenAI, NVIDIA, Apple, Bloomberg, and Qualcomm

The model scales mentorship not by lowering barriers, but by raising shared infrastructure.

How You Can Engage

- Mentor or host project contributors
- Present, collaborate, or co-develop research directions
- Establish a partnership program

Open, connected compiler infrastructure accelerates scientific discovery

Thank you!