LambdaGen

A GPU code generator powered by recursion schemes

András Leitereg

Computer Science MSc

Faculty of Informatics, Eötvös Loránd University

Supervisor: Dániel Berényi

GPU Lab, Wigner RCP

Emberi Erőforrások Minisztériuma

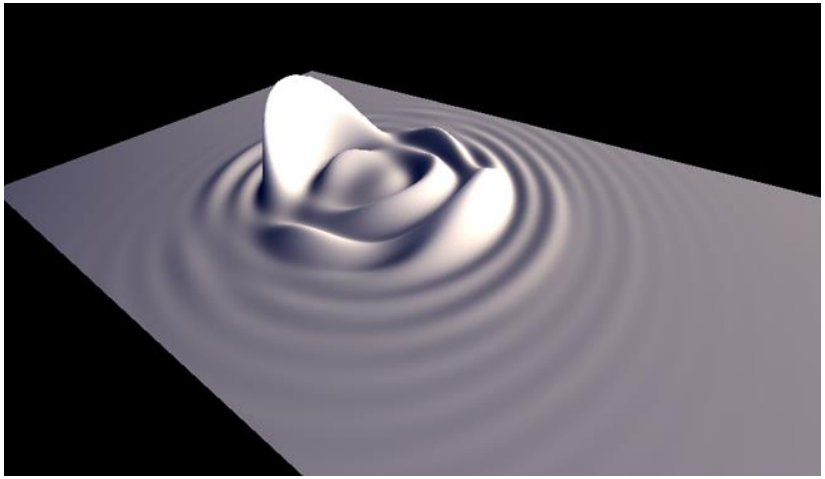
Overview

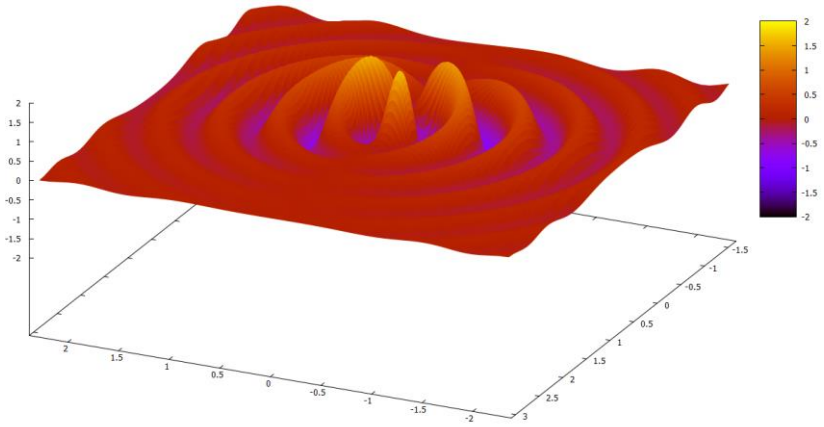
- Motivation
- Vector operations
- LambdaGen
  - Transformations
  - Annotation
  - Memory model
- Summary

Is there a better representation of linear algebraic computations than matrix-vector operations?

Can we deduce the optimal memory usage and parallelization from the vector expression itself?

Motivation - Example





Motivation – Existing libraries

- Existing libraries:
  - Fixed dimensional structure
  - Non-customizable operations

$$C_{ik} = \sum_j A_{ij} v_j B_{jk}$$

Eigen

BLAS

Armadillo

Blitz++

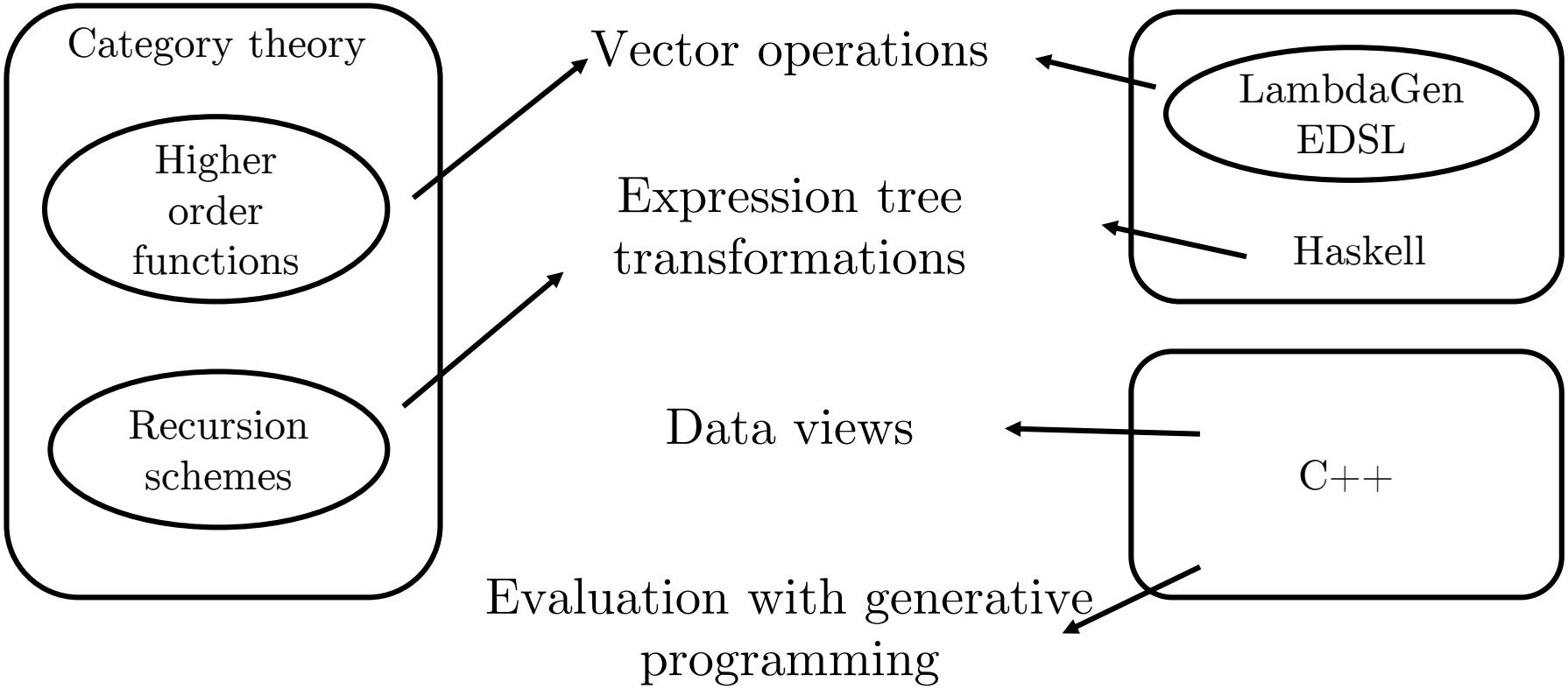
boost C++ LIBRARIES

Boost

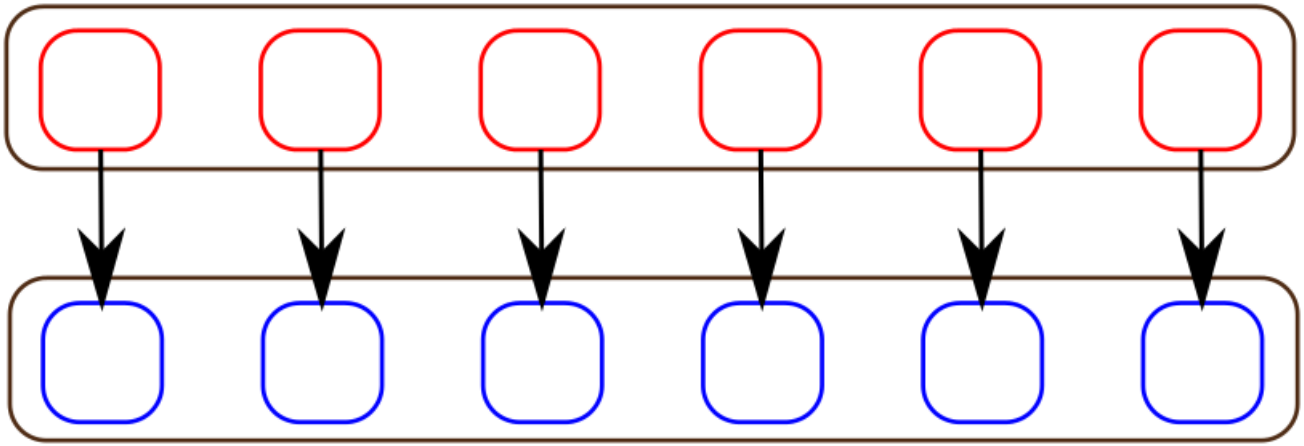
Alternative approach

- Higher order functions
- Data views
- General transformations on the expression trees
- Evaluation with generative programming

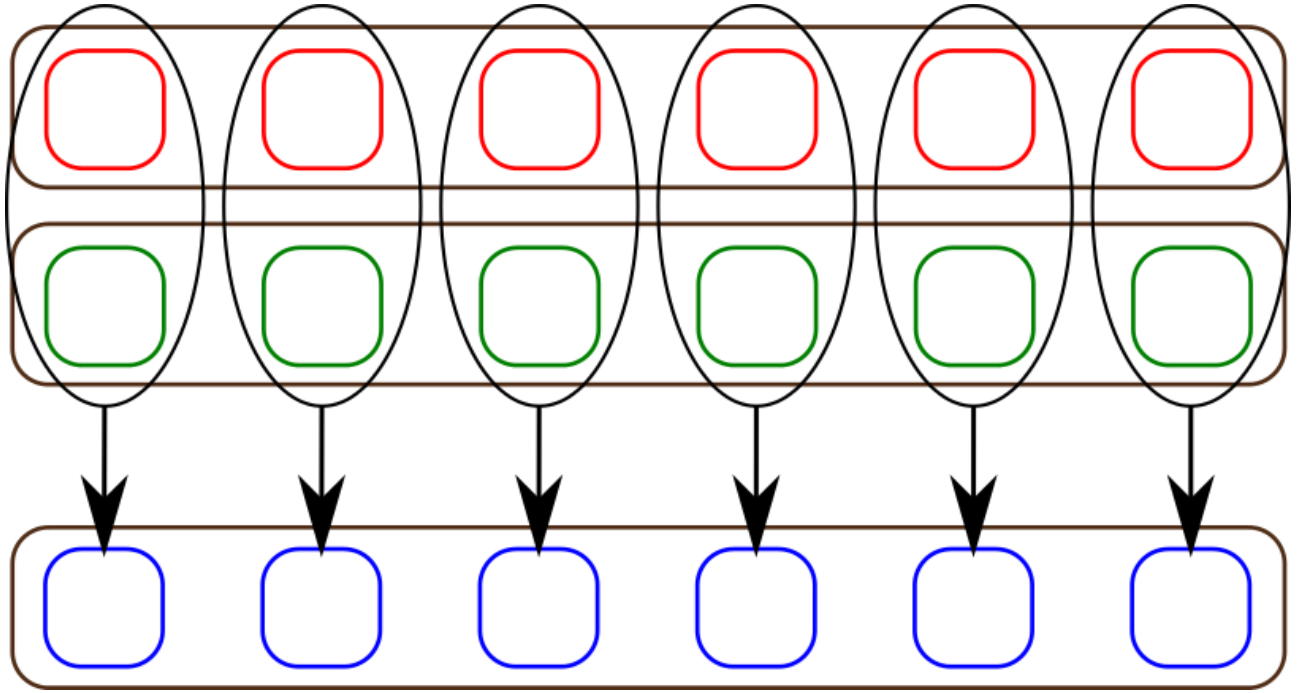
Tools


Category theory
Higher order functions
Recursion schemes
Vector operations
Expression tree transformations
Data views
Evaluation with generative programming
LambdaGen EDSL
Haskell
C++

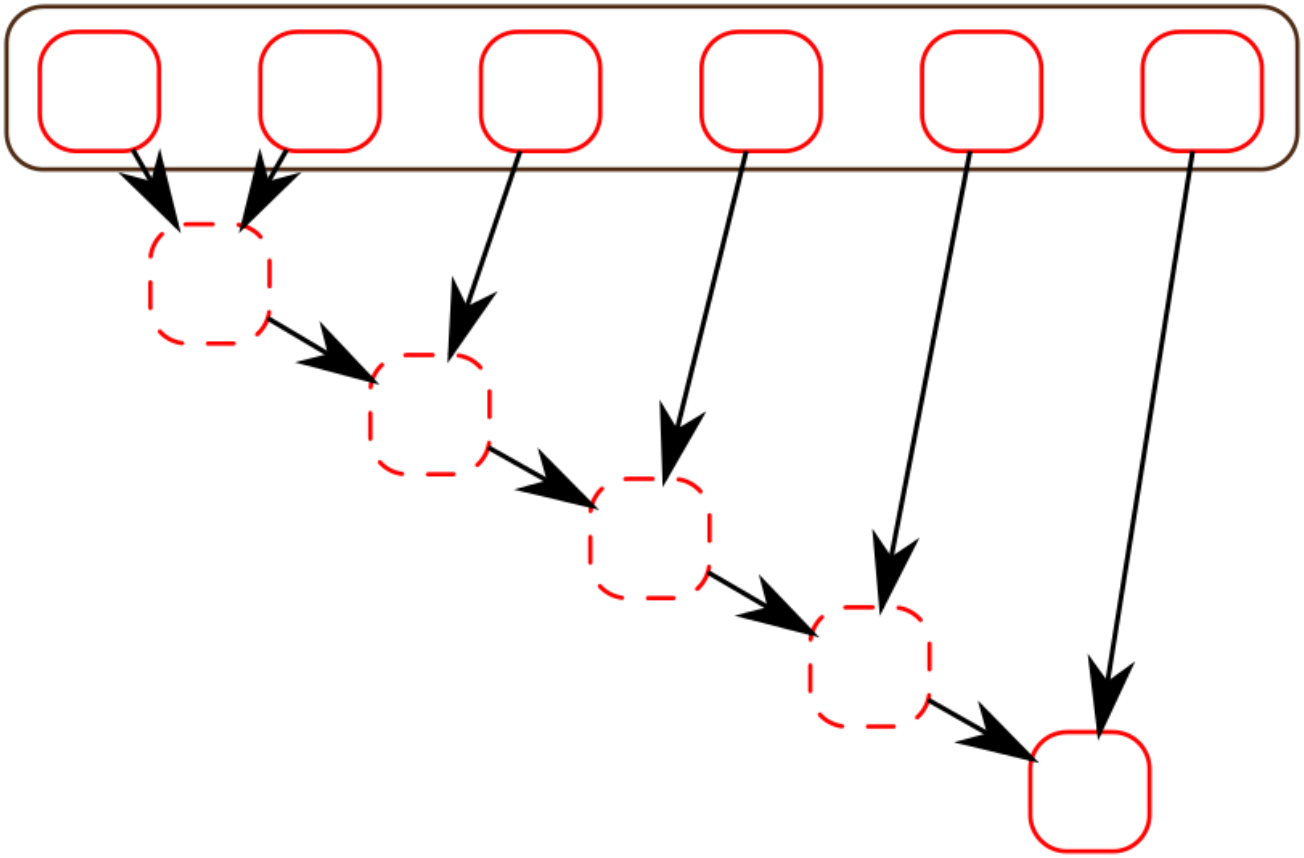
Vector operations - Map



Vector operations - Zip

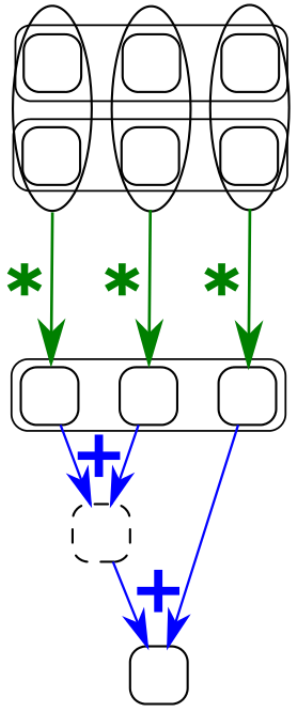


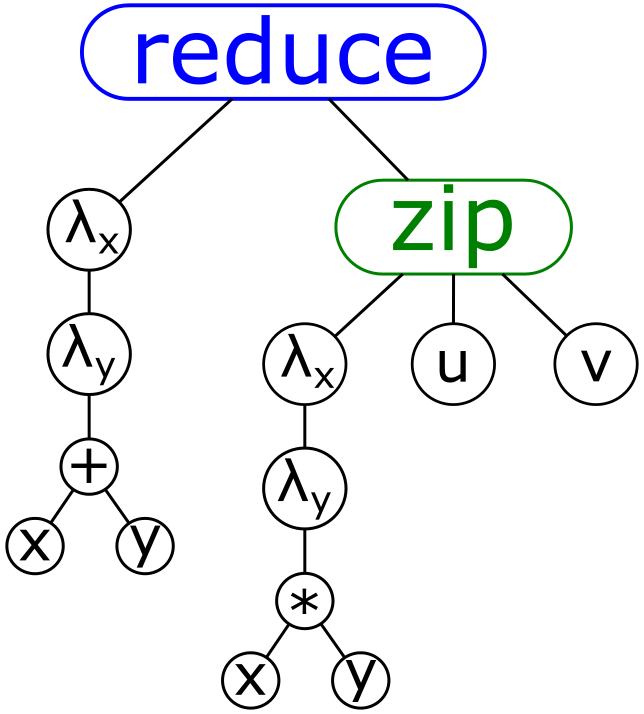
Vector operations - Reduce



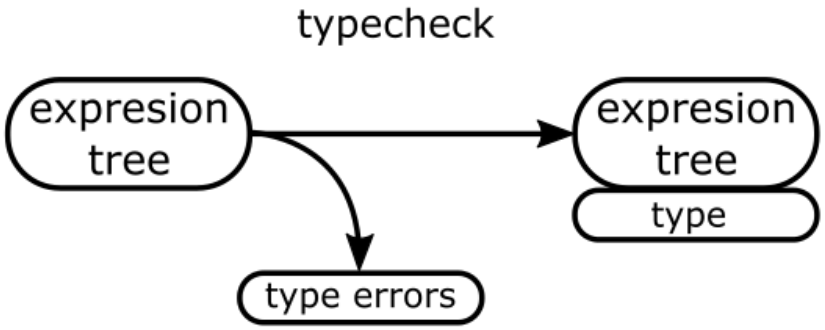
Vector operations - Example

$$\sum_i u_i \, v_i$$

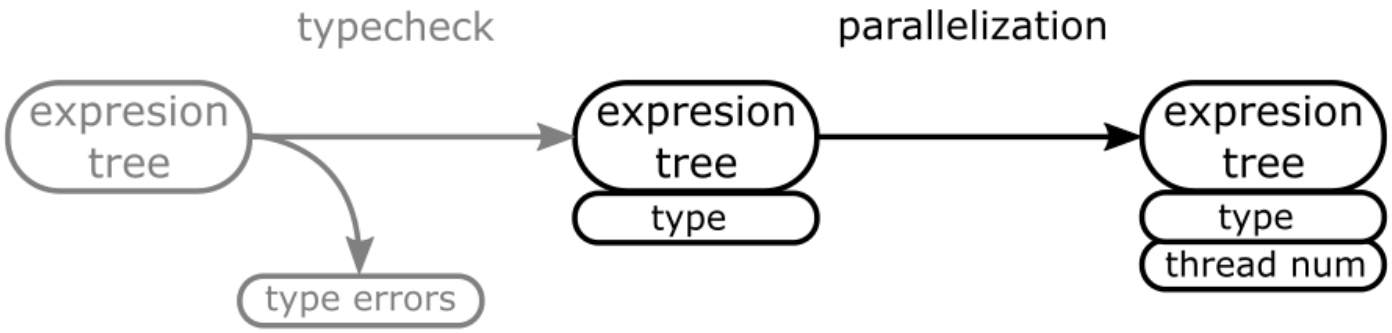




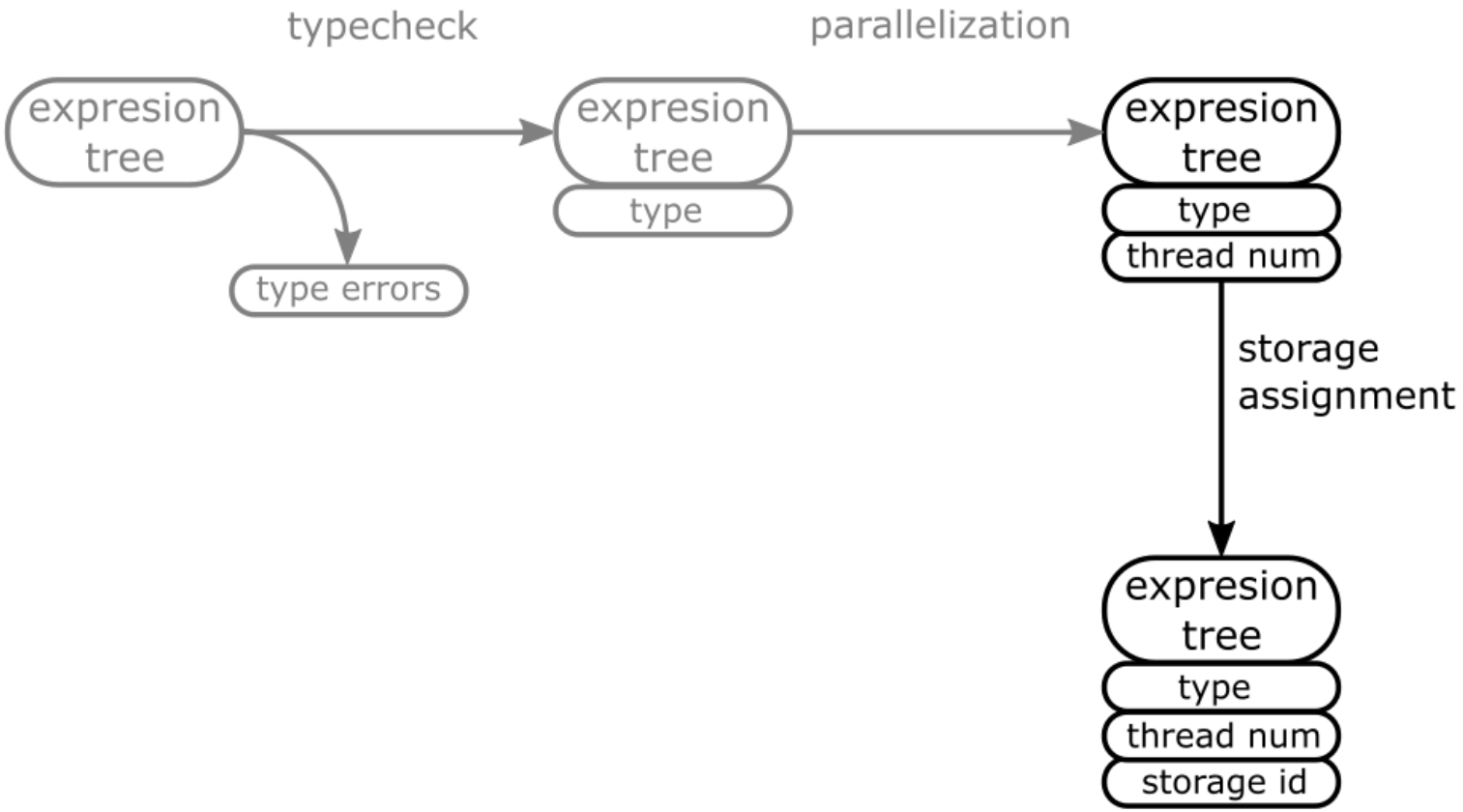
LambdaGen - Transformations



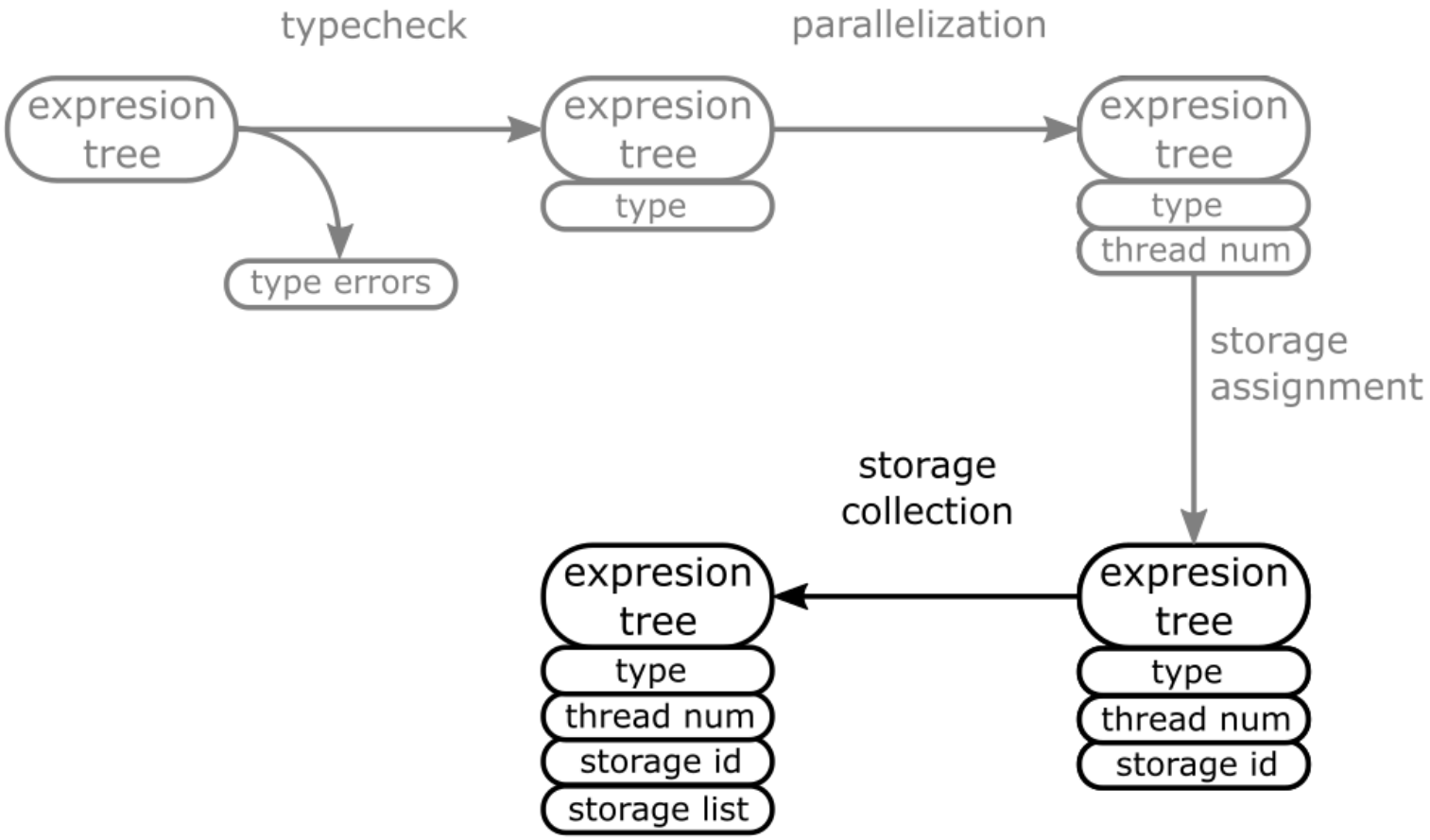
LambdaGen - Transformations



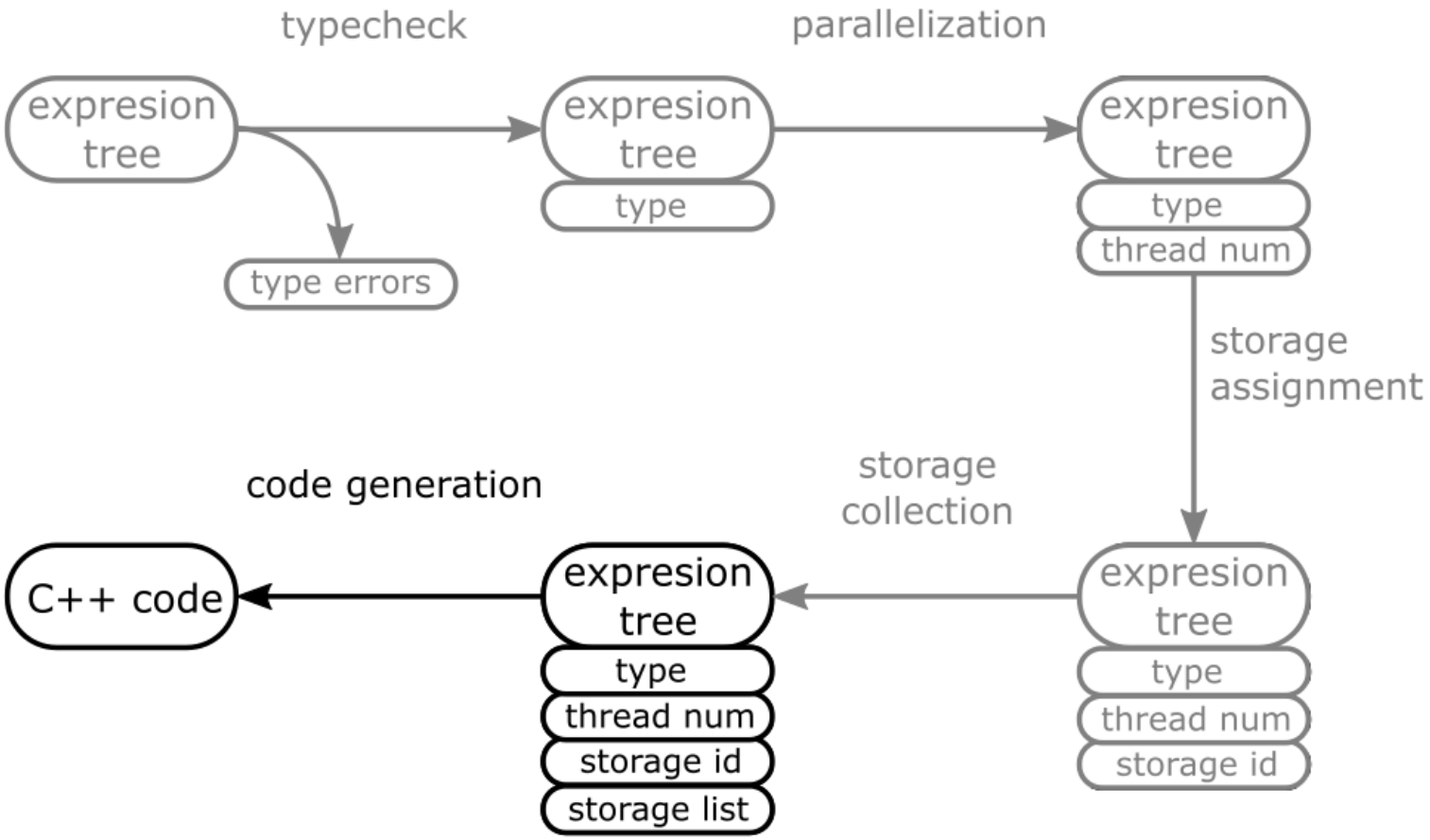
LambdaGen - Transformations



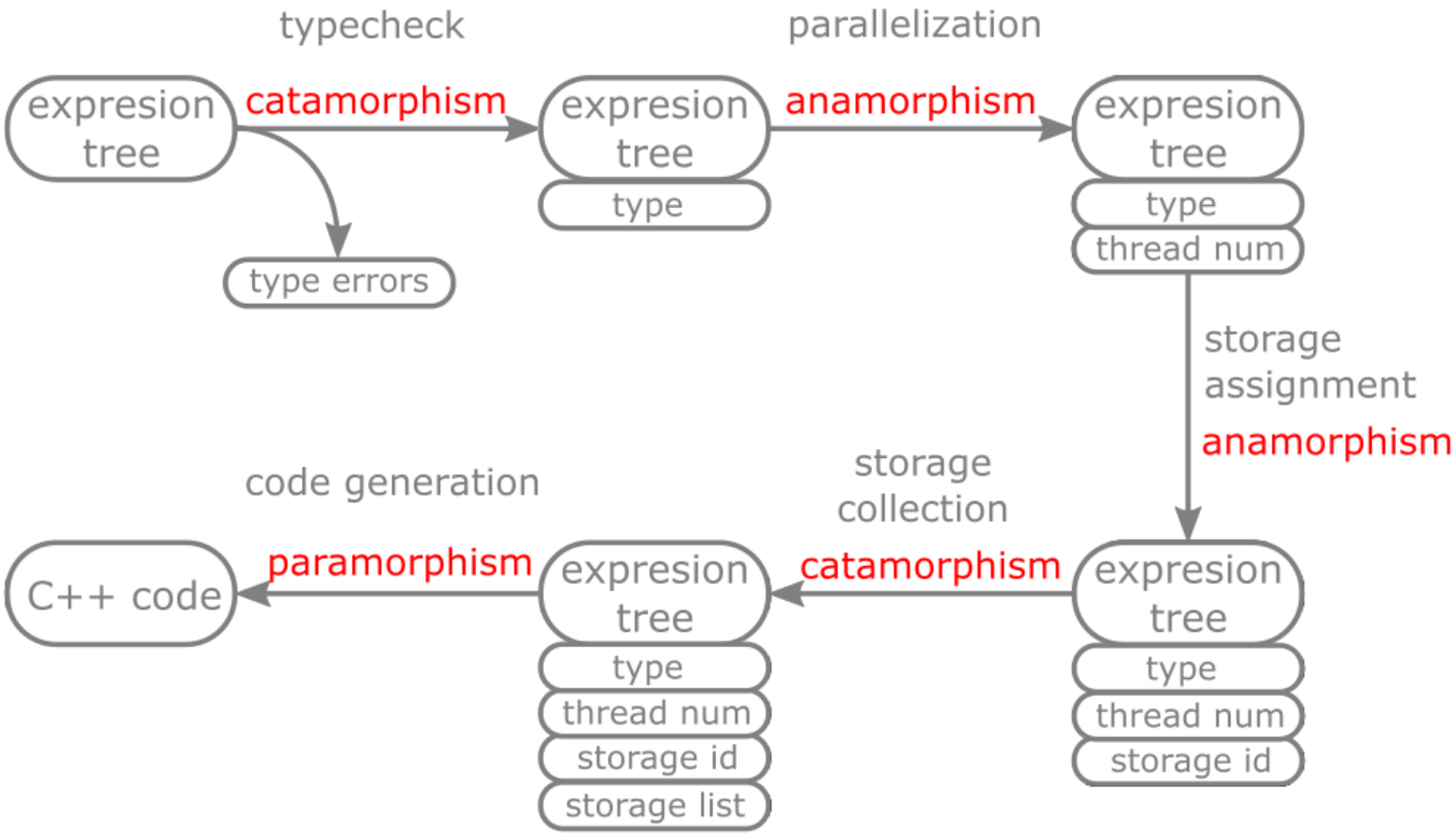
LambdaGen - Transformations



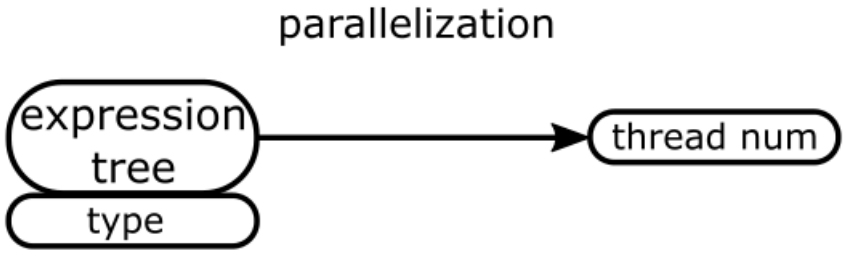
LambdaGen - Transformations



LambdaGen - Transformations

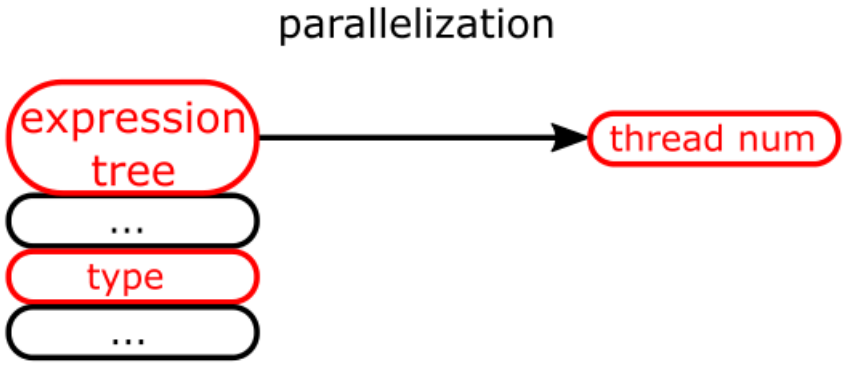


LambdaGen - Annotation



This is the algorithm we'd like to implement.

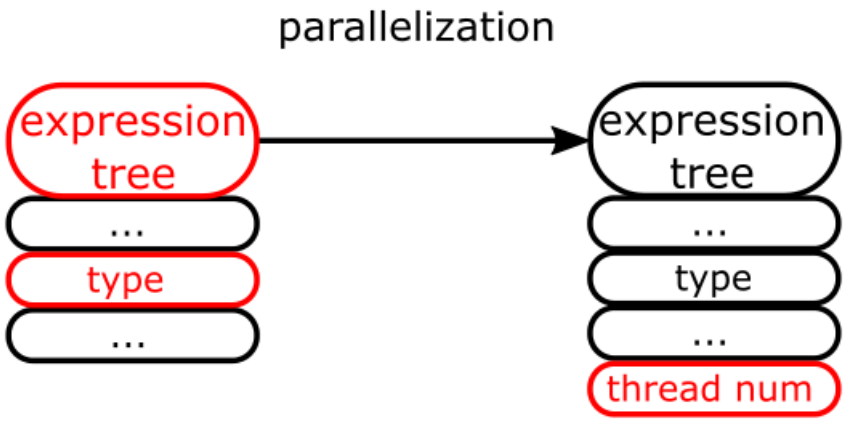
LambdaGen - Annotation



Extensible records (Vinyl):

- Abstraction over the specific annotations (and thus the type of the tree nodes)
- Explicitly require some annotations

LambdaGen - Annotation



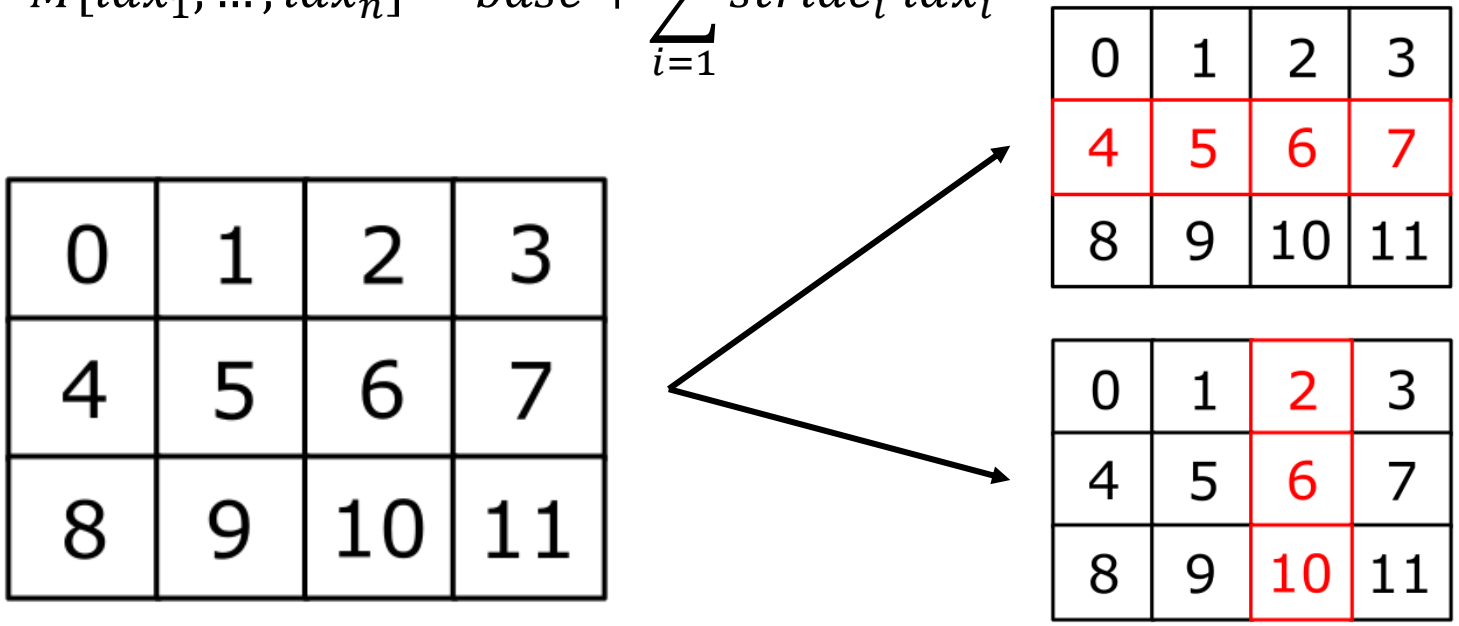
We'd like to make a chain of transformations

Transformation adapter algorithm:

Extends the annotations of the tree with the new value

LambdaGen - Memory model

$$M[idx_1, \ldots, idx_n] = base + \sum_{i=1}^{n} stride_i \; idx_i$$



LambdaGen - SYCL

- C++11 lambdas supported
- Debug mode: triSYCL
- Standard C++
- pointer = accessor
  pointer arithmetic = ?

Benefits of LambdaGen

- The assembled set of operations is
  - Customizable, optimizable, dimension independent
- Constructions are explicit on type level
  - Fewer mistakes, less debugging
- Extendable, abstract transformation system
  - The system scales well to describe complex program transformations

Summary

- ✓Linear algebra with Map/Reduce/Zip
- ✓High level tree transformations
- ✓Efficient memory handling
- ✓Evaluation with generated code

Further plans:

- Hierarchical evaluation
- Optimizations

Thanks!

github.com/leanil/LambdaGen

The LambdaGen EDSL

```
reduce
     (lam x (lam y (add x y)))
     (zip
          (lam x (lam y (mul x y)))
          u
          v)
```

Catamorphism

```
cata :: (F a -> a) -> fix F -> a
cata alg = alg . fmap (cata alg) . out
```