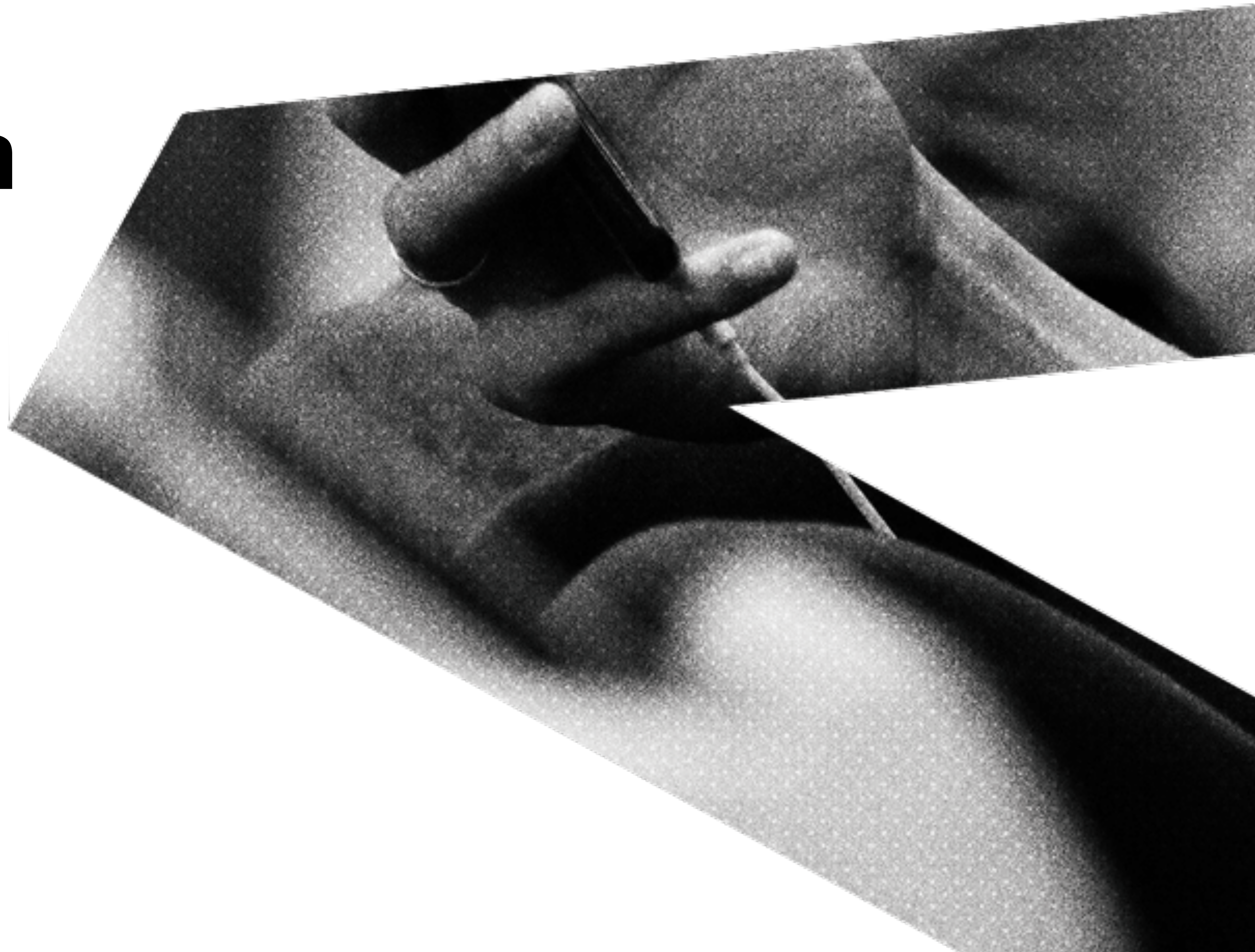
ZAMA

Recent advances in homomorphic compilation

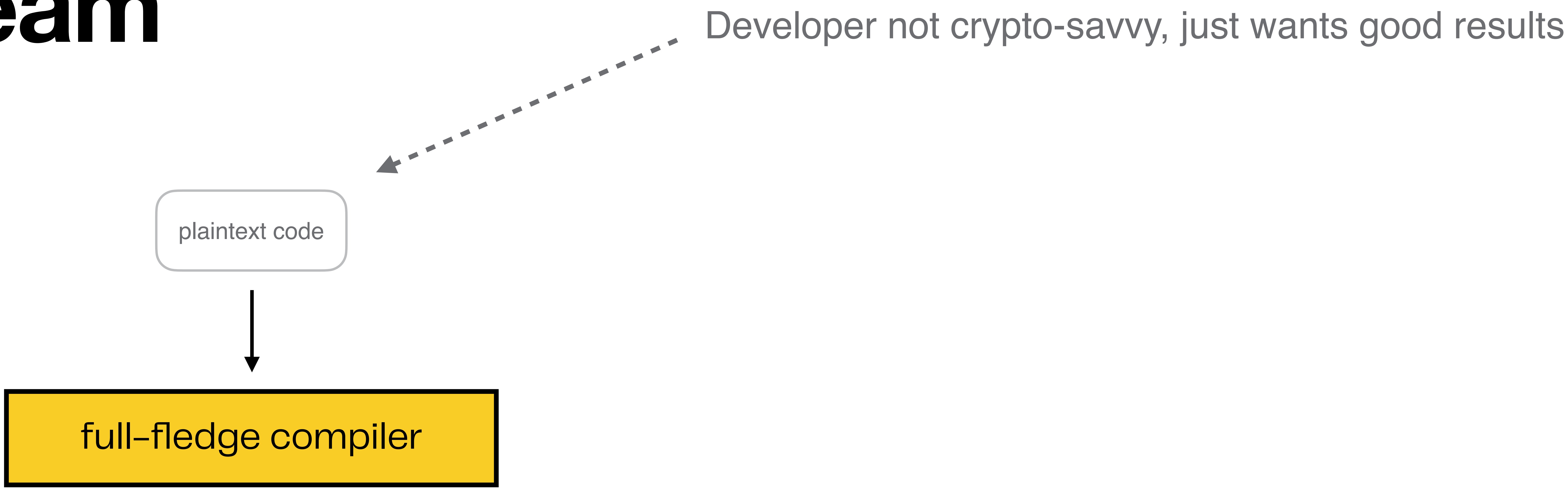


Pascal Paillier, Zama

FHE.org conference 2023

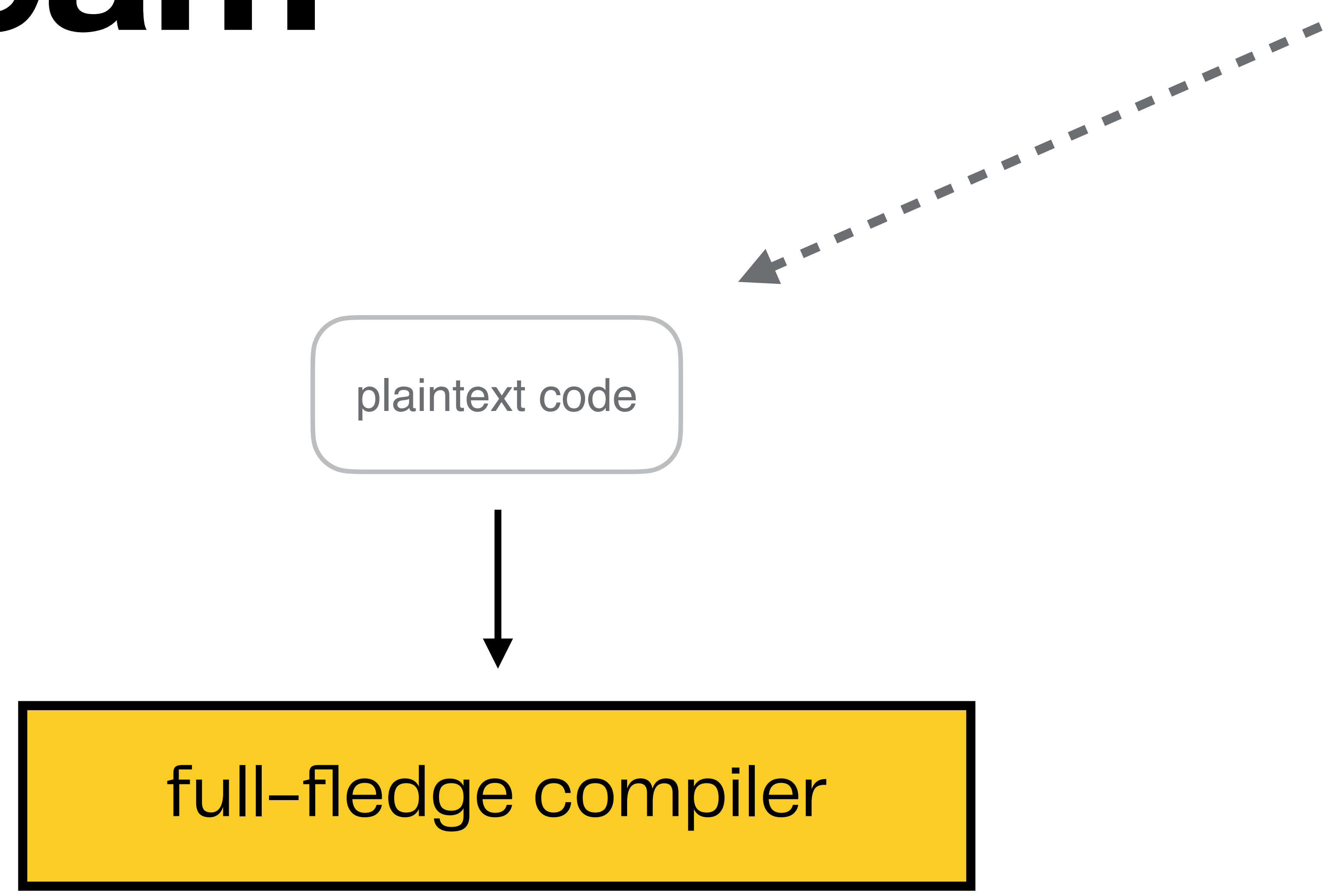
What is homomorphic compilation?

The big dream



The big dream

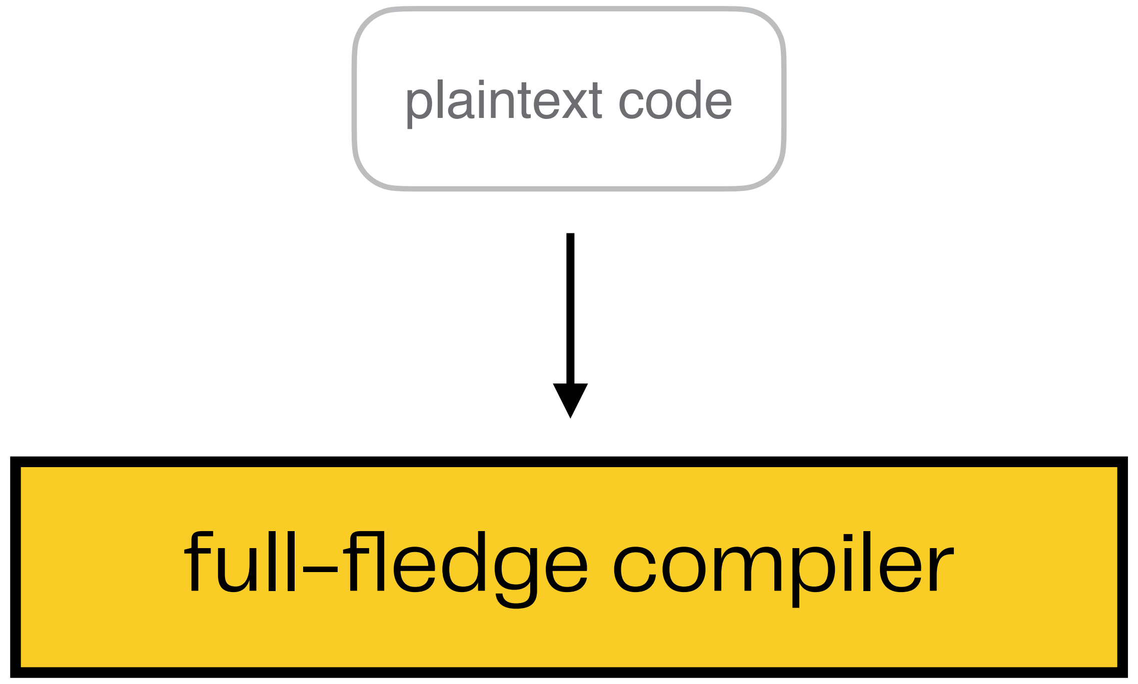
Developer not crypto-savvy, just wants good results



The big dream

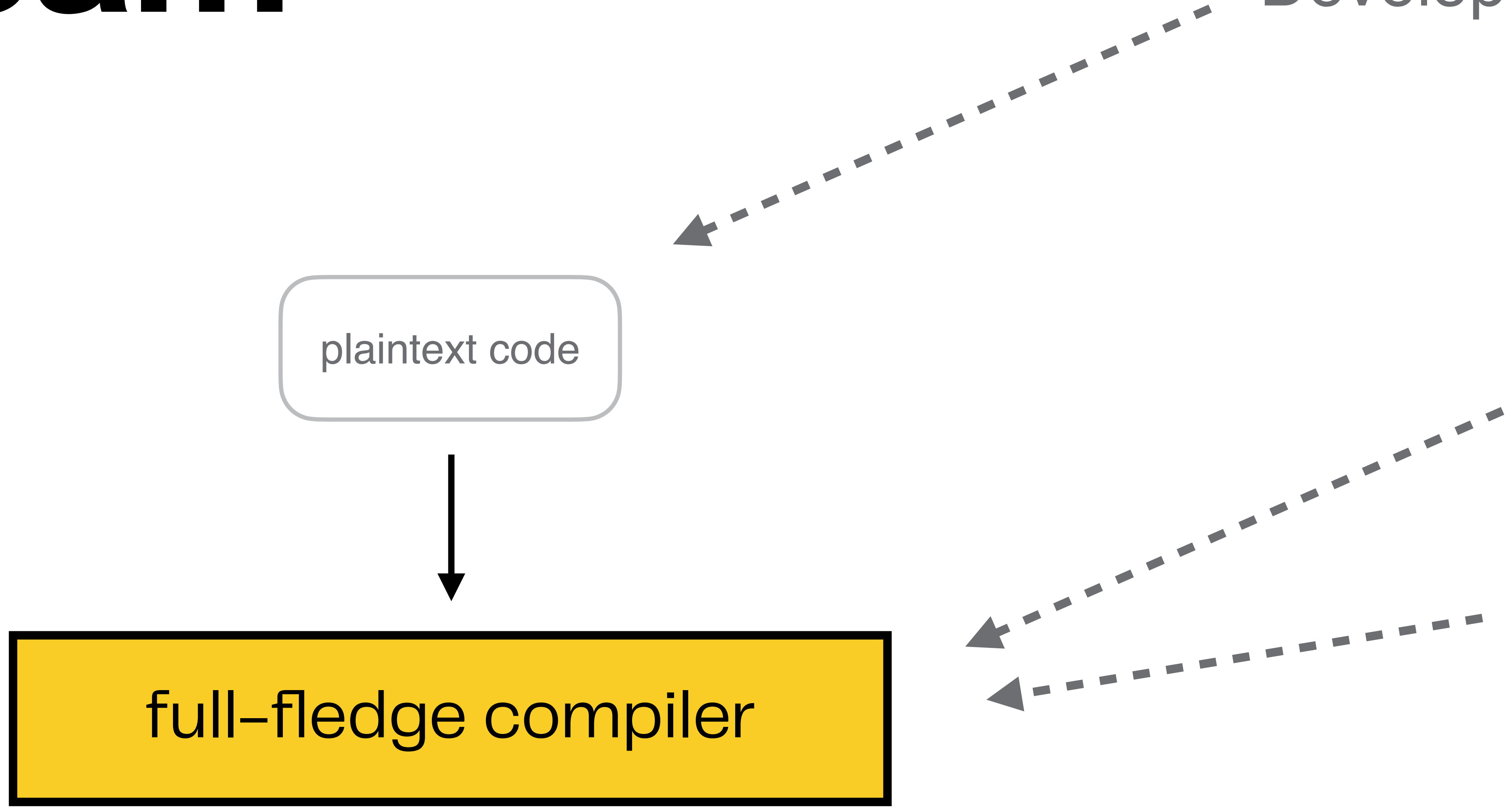
Developer not crypto-savvy, just wants good results

1 Converts input code into an FHE circuit



The big dream

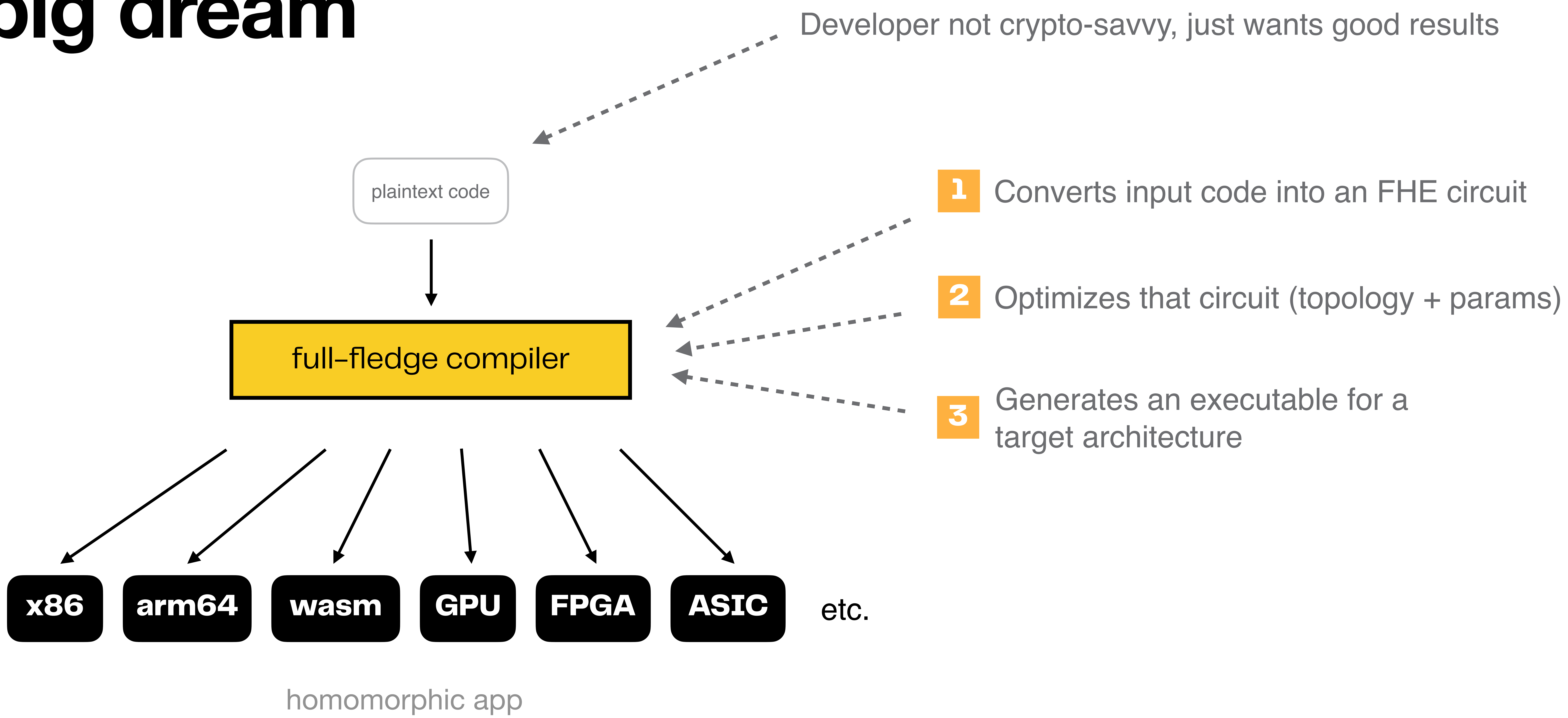
Developer not crypto-savvy, just wants good results



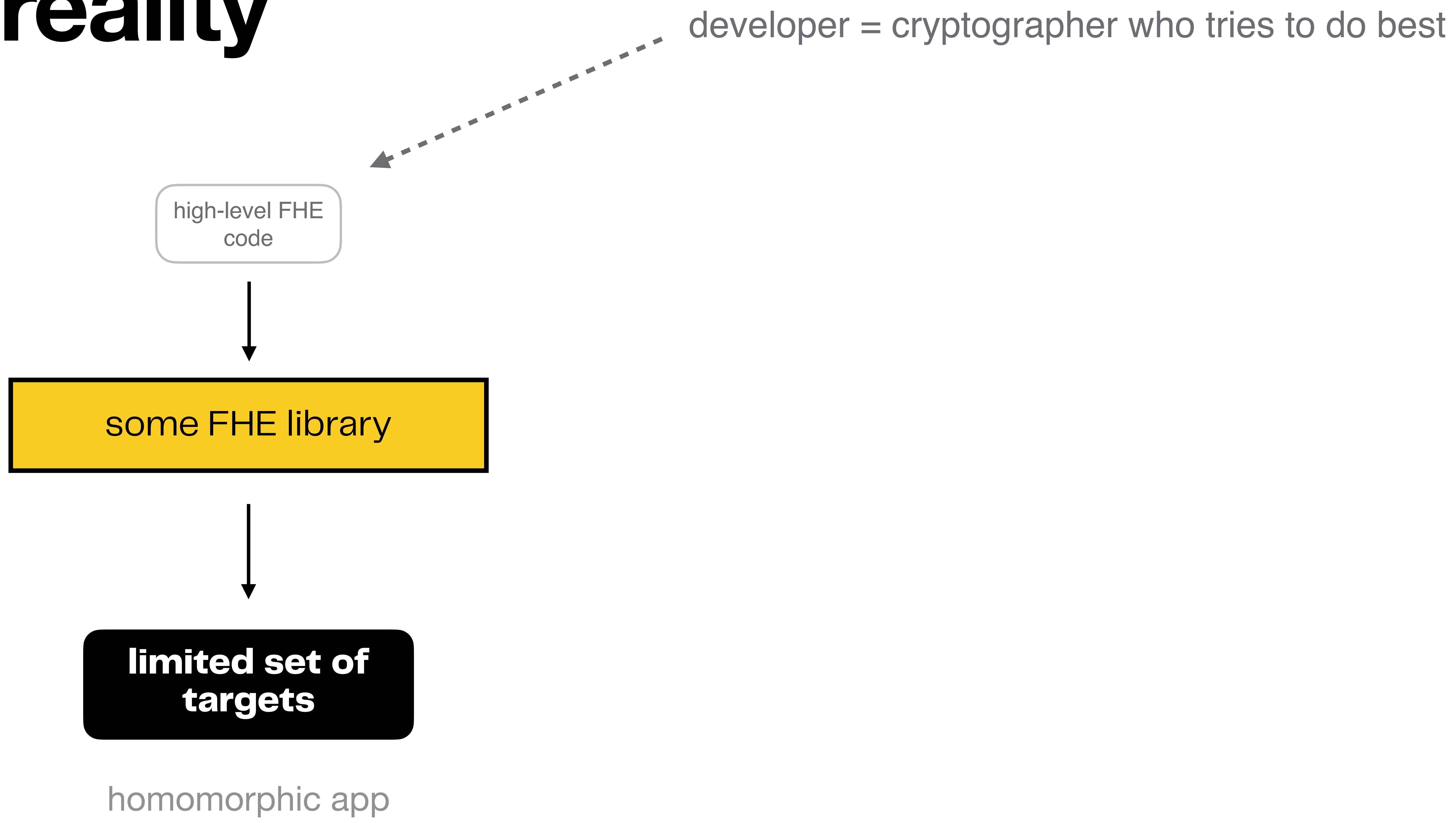
1 Converts input code into an FHE circuit

2 Optimizes that circuit (topology + params)

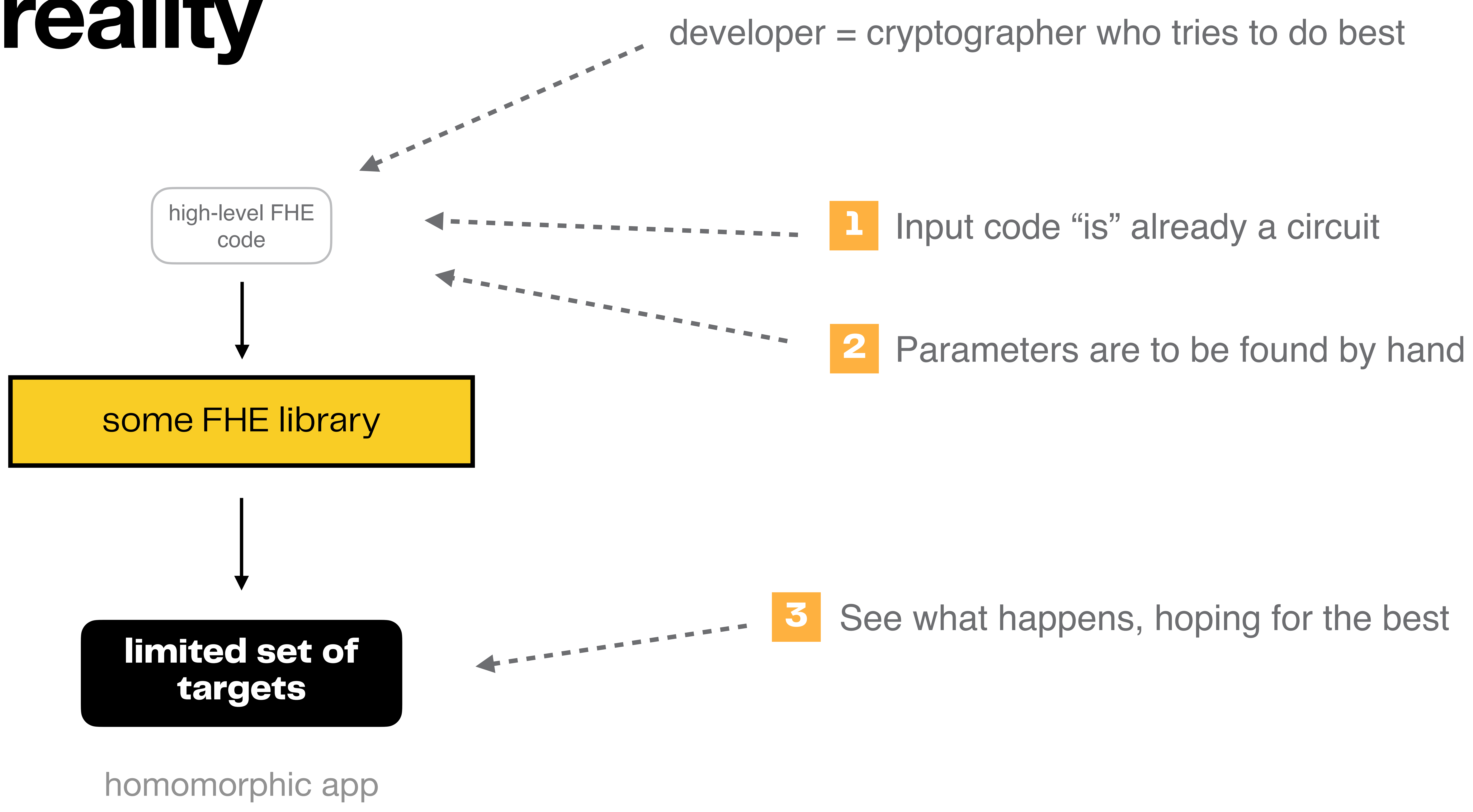
The big dream



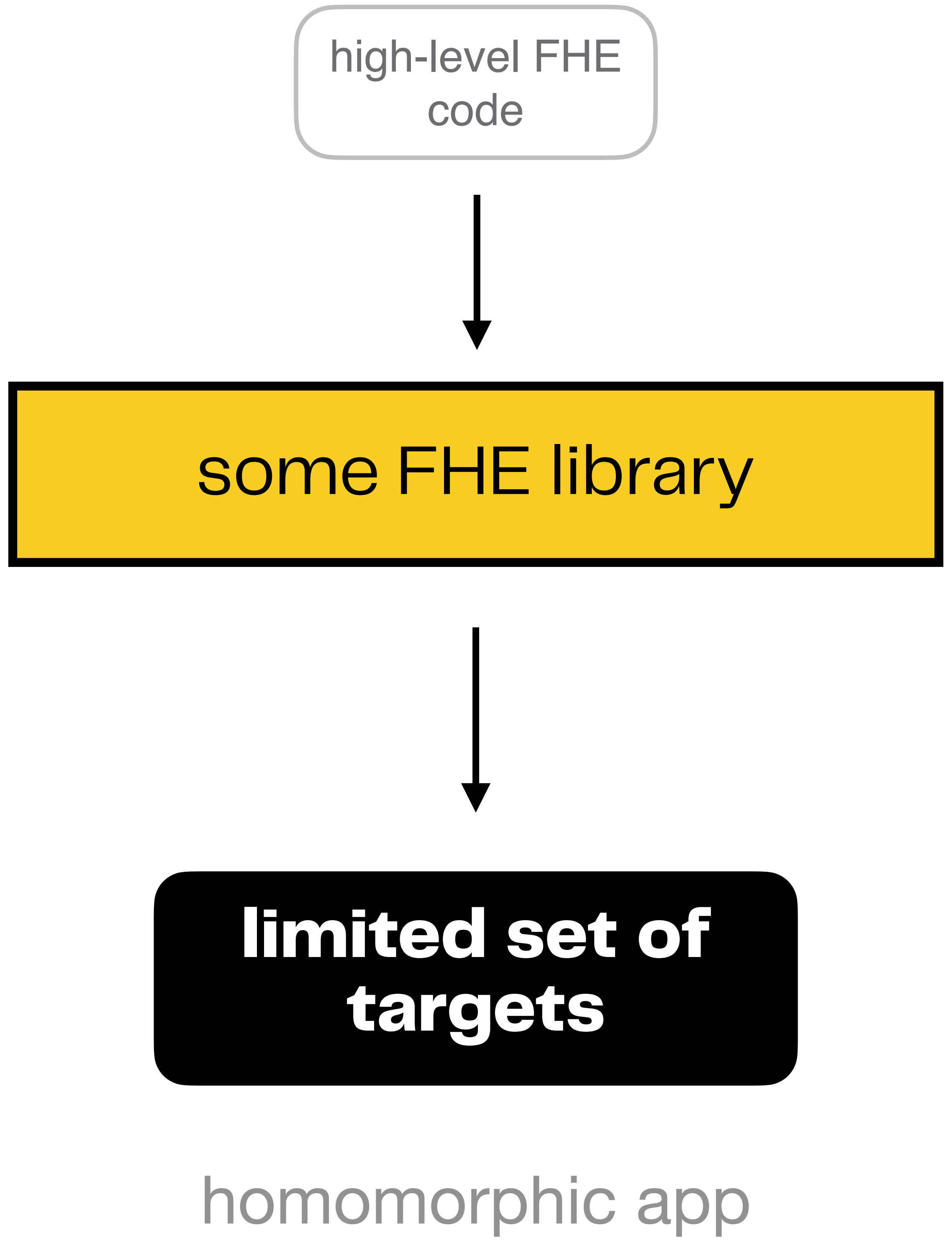
The (poor) reality



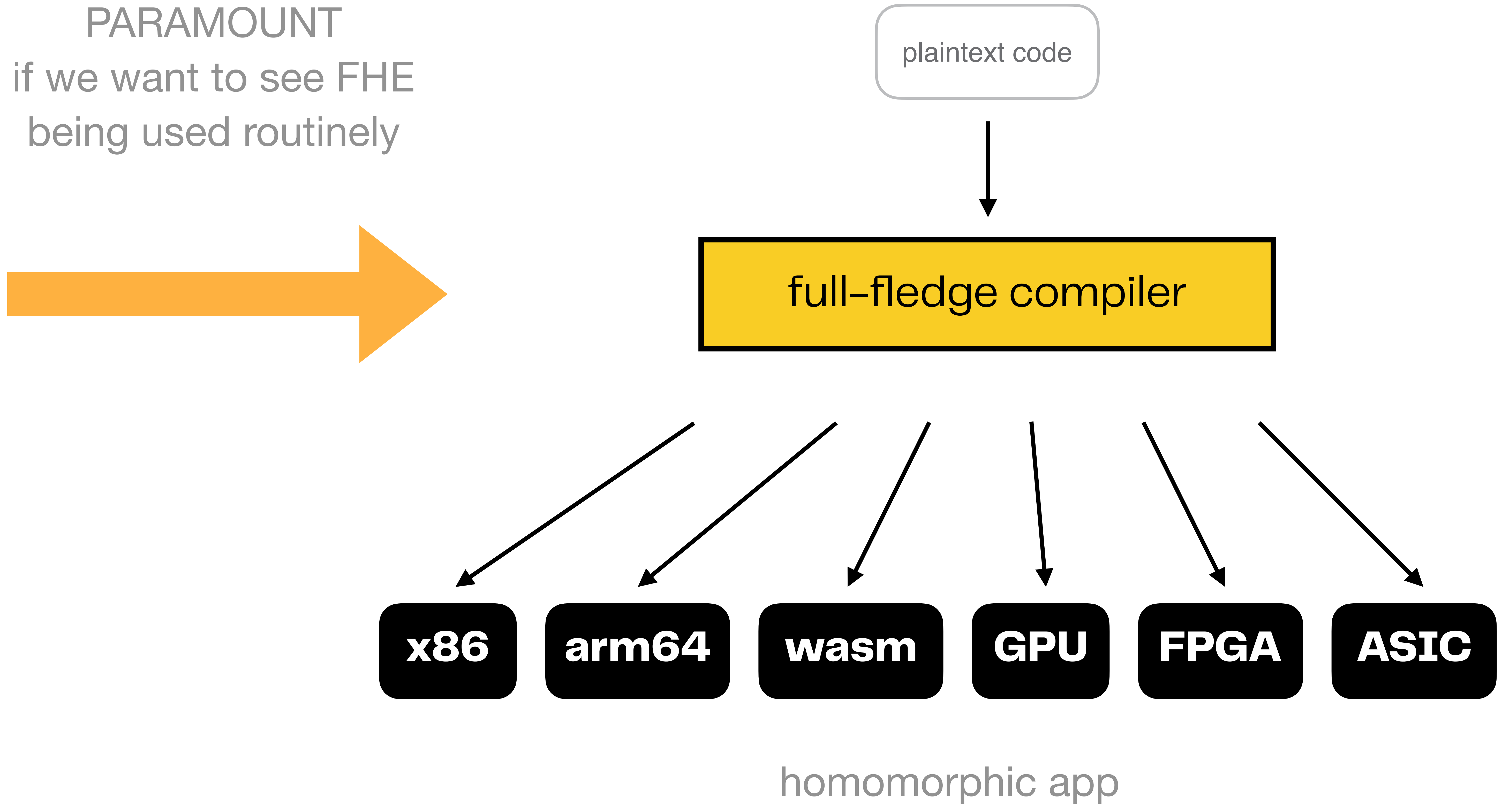
The (poor) reality



What is blocking this transition?



PARAMOUNT
if we want to see FHE
being used routinely

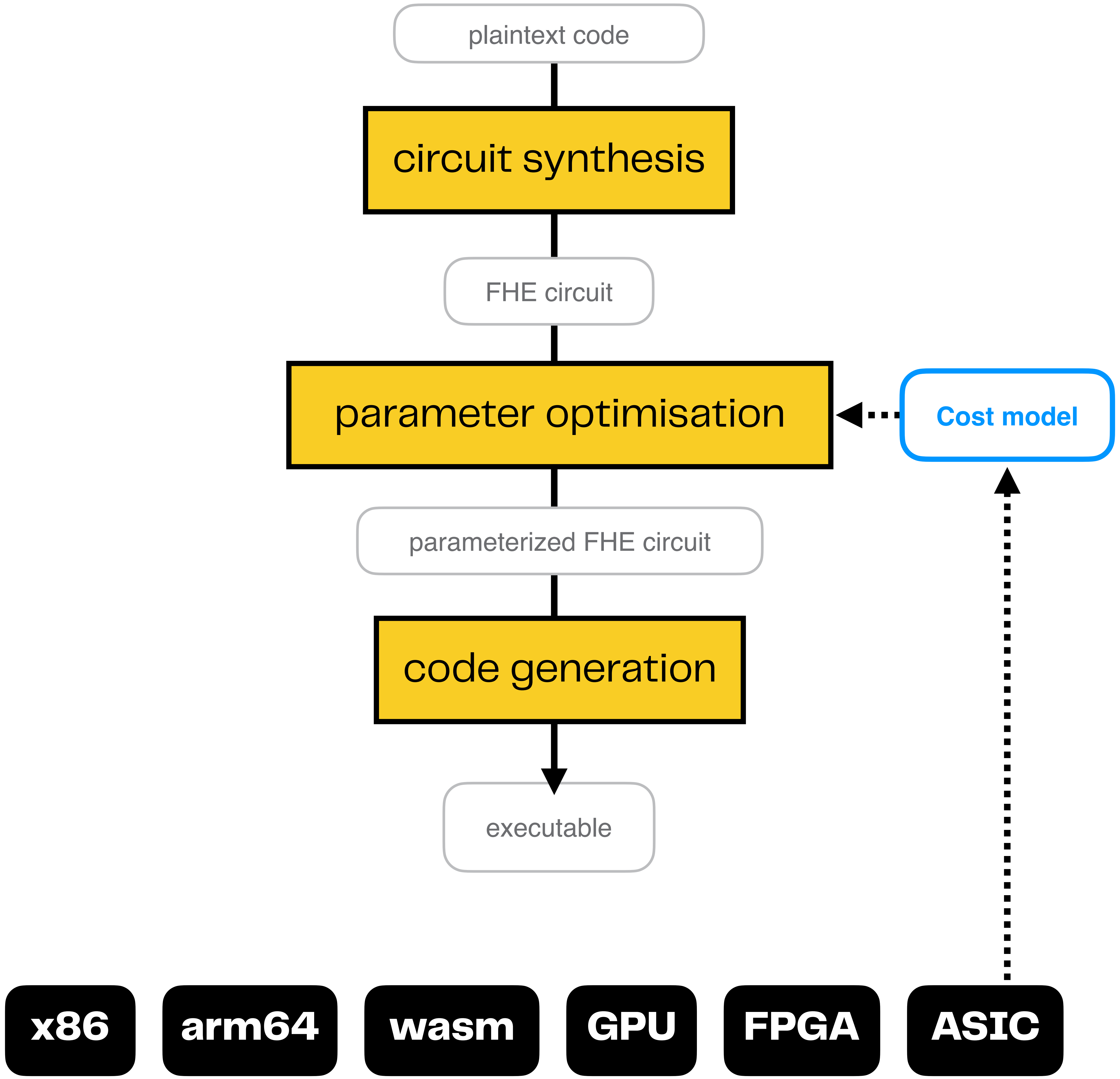


The inner ingredients of hom. compilers

1 Converts input code into an FHE circuit

2 Optimizes that circuit (topology + params)

3 Generates an executable for a target architecture

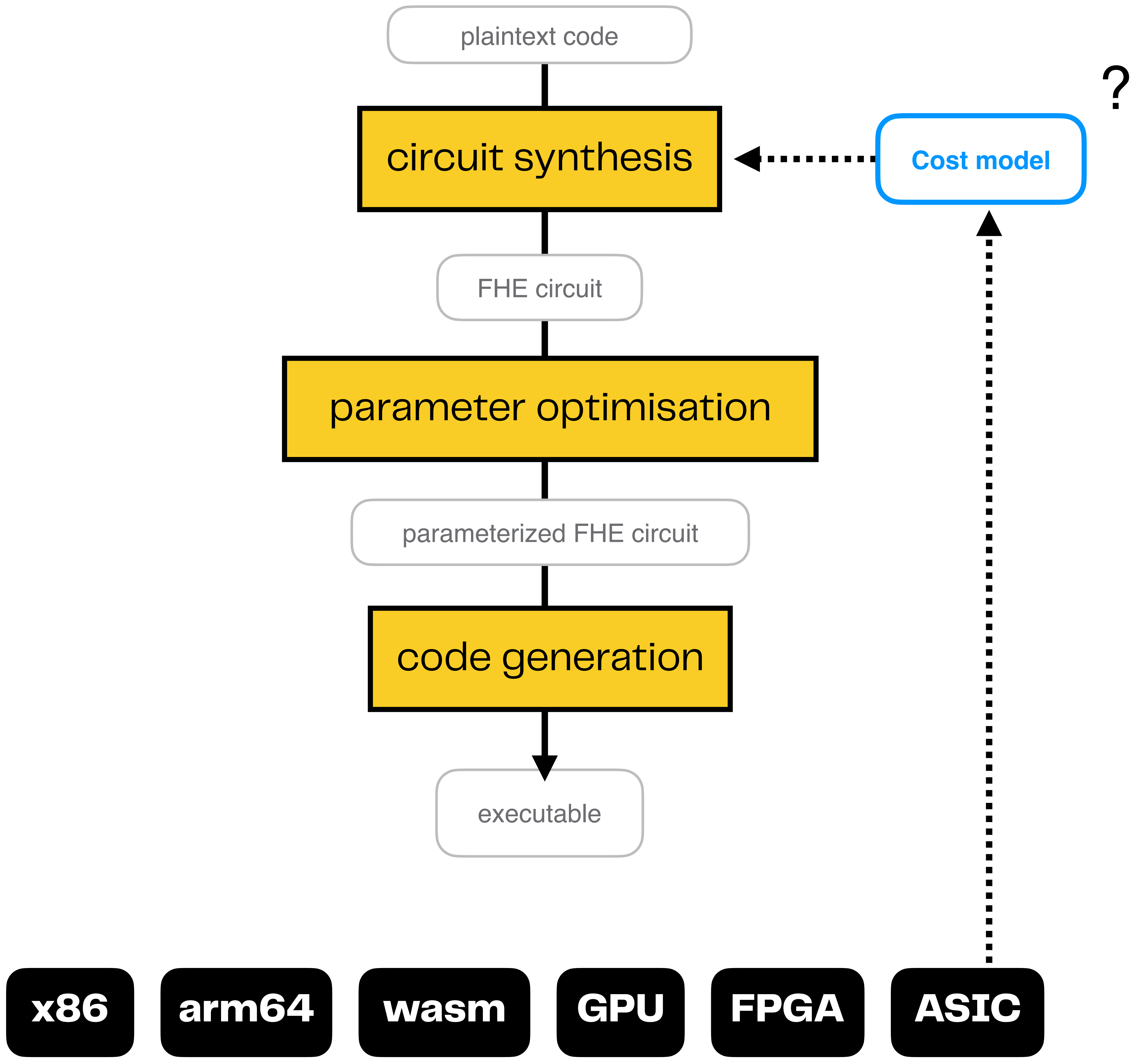


The inner ingredients of hom. compilers

1 Converts input code into an FHE circuit

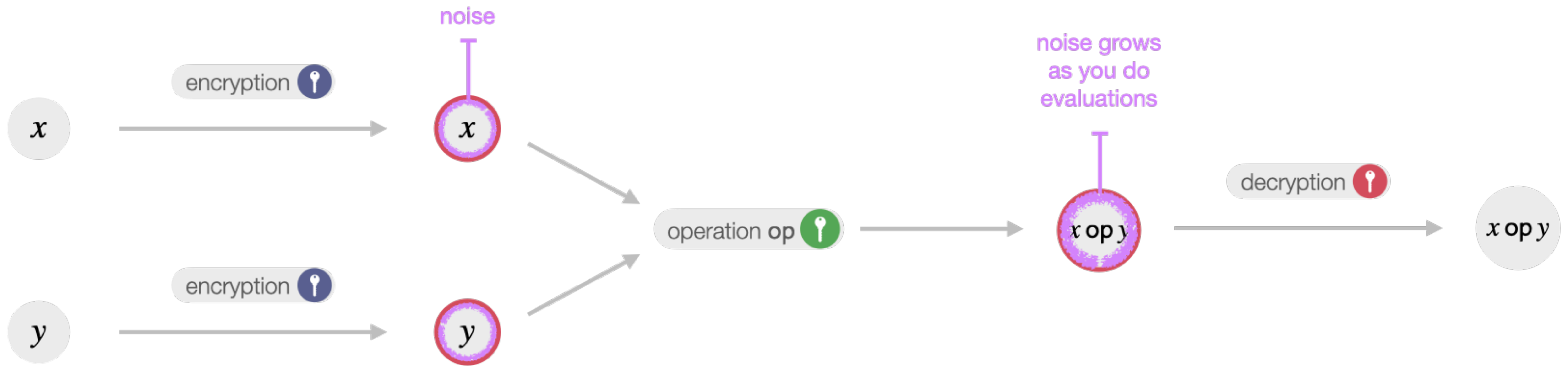
2 Optimizes that circuit (topology + params)

3 Generates an executable for a target architecture

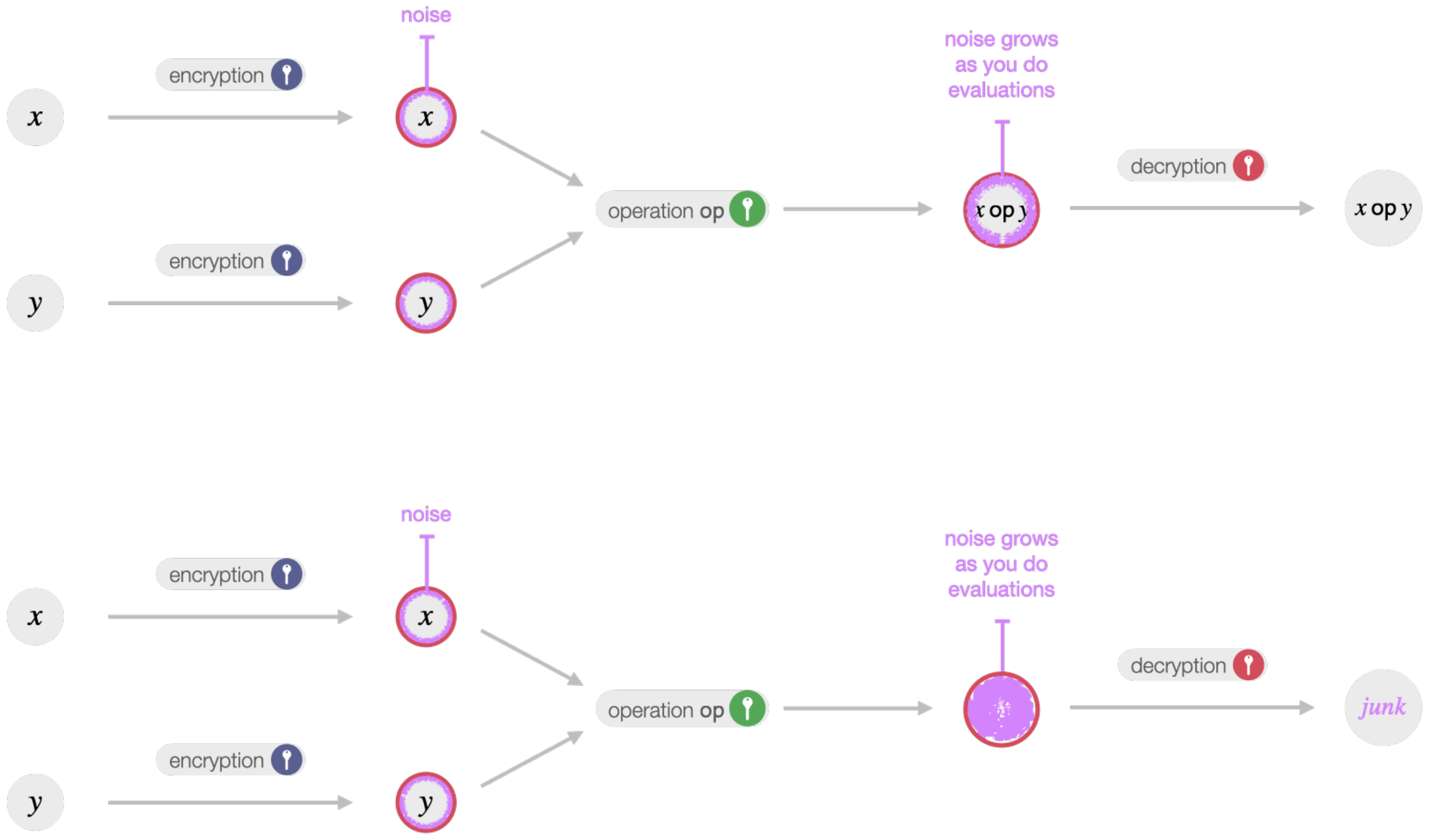


What we are building at Zama

Homomorphic Encryption: Basic Concepts

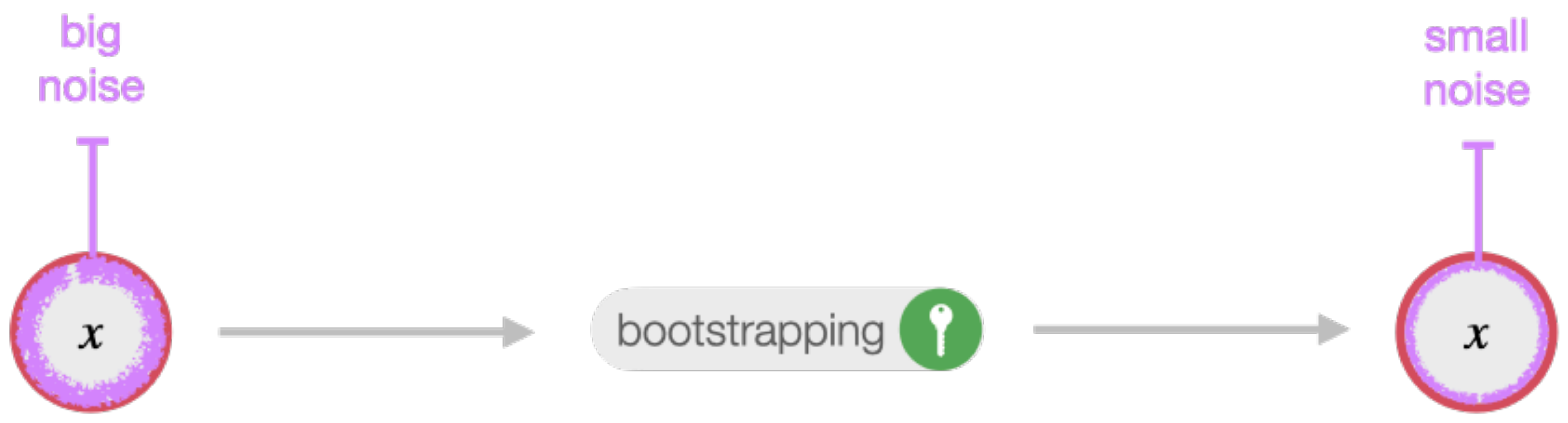


Homomorphic Encryption: Basic Concepts



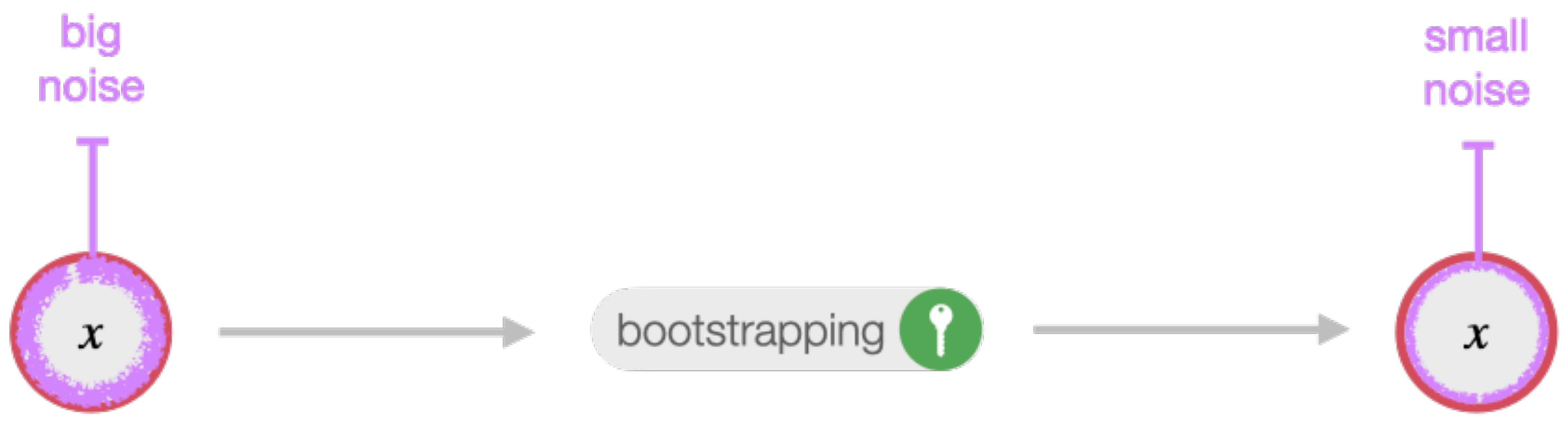
Homomorphic Encryption: Basic Concepts

Bootstrapping



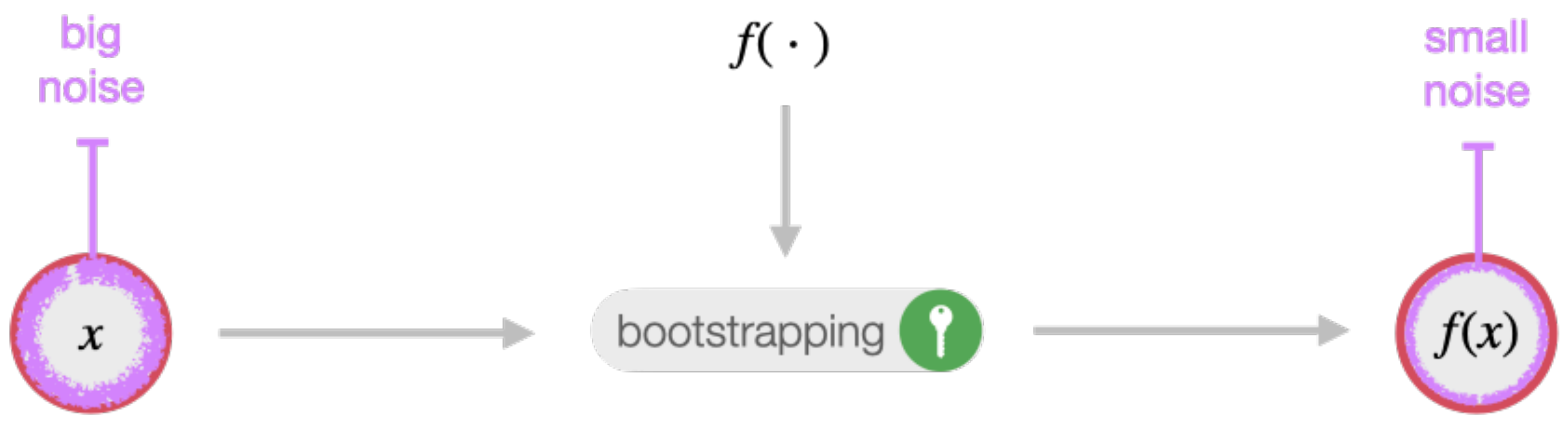
Homomorphic Encryption: Basic Concepts

Bootstrapping

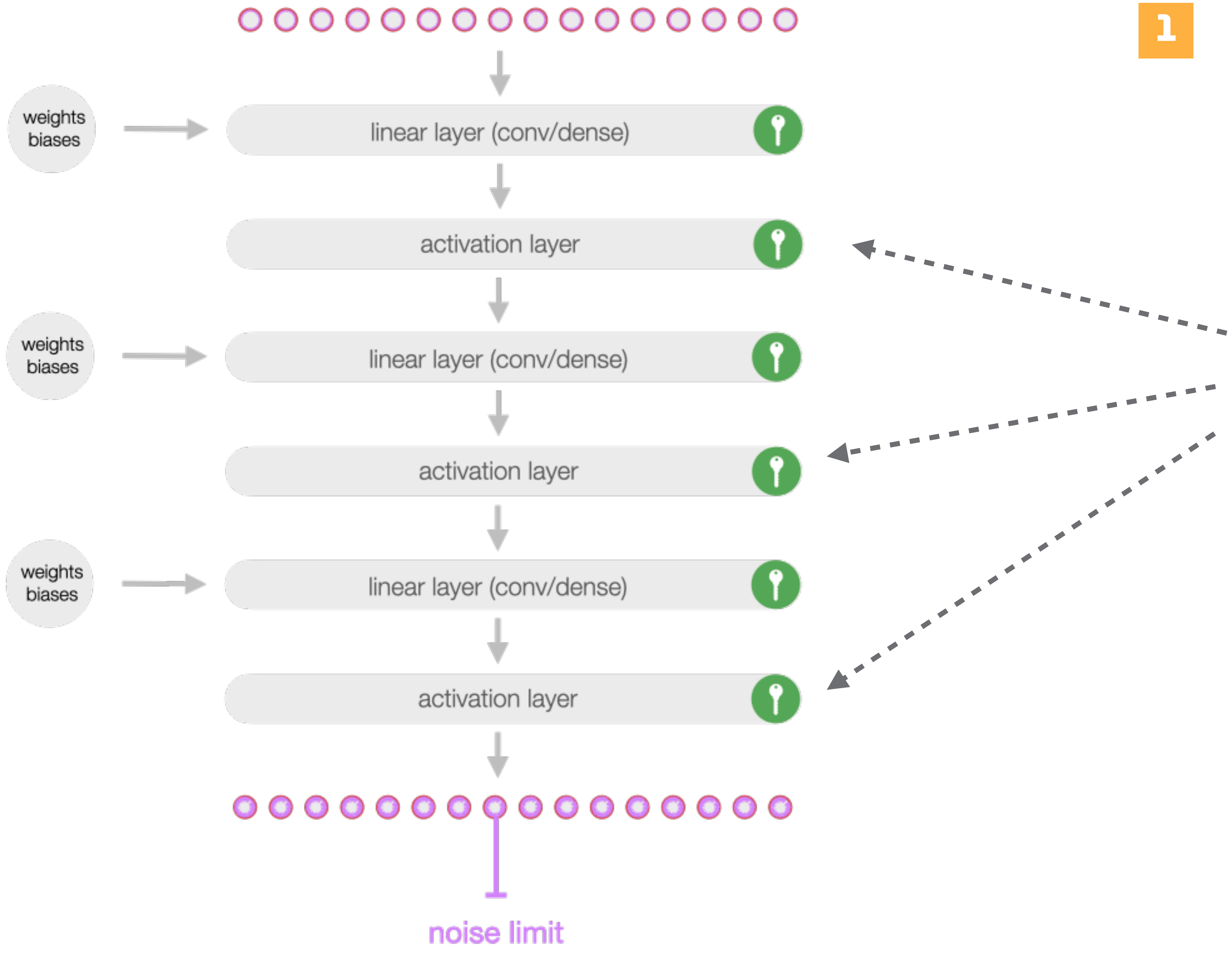


Programmable Bootstrapping

(PBS)



Homomorphic Inference: Leveled HE

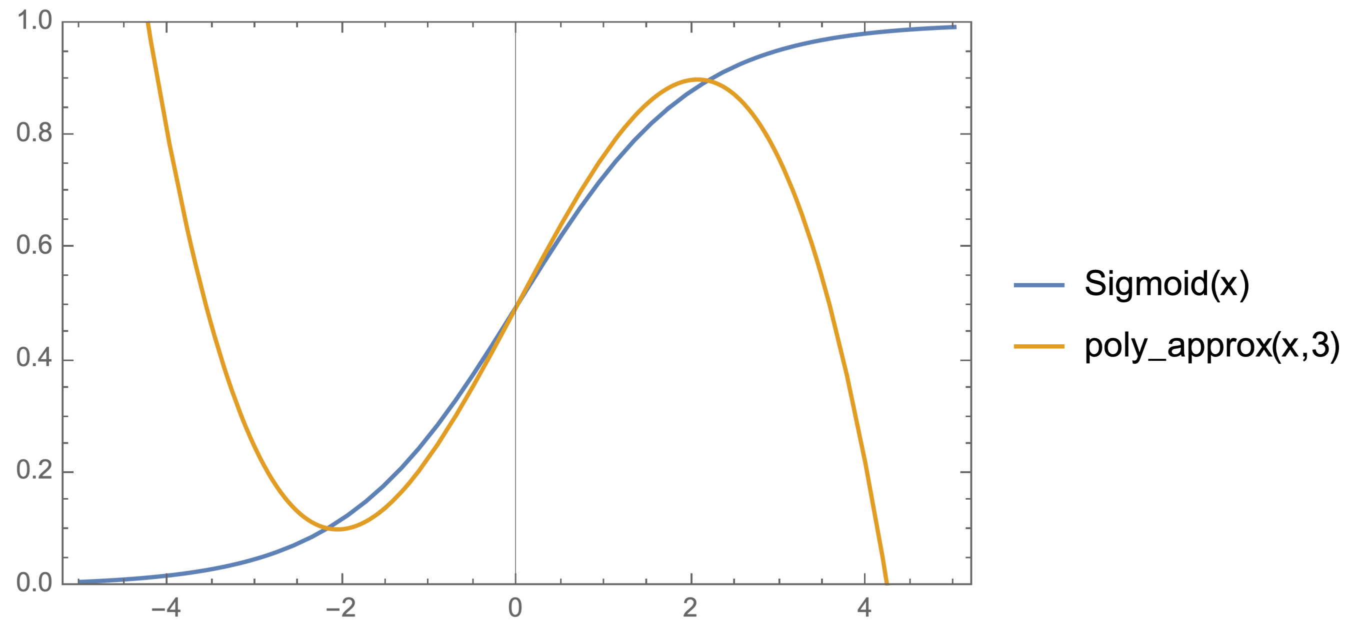


1

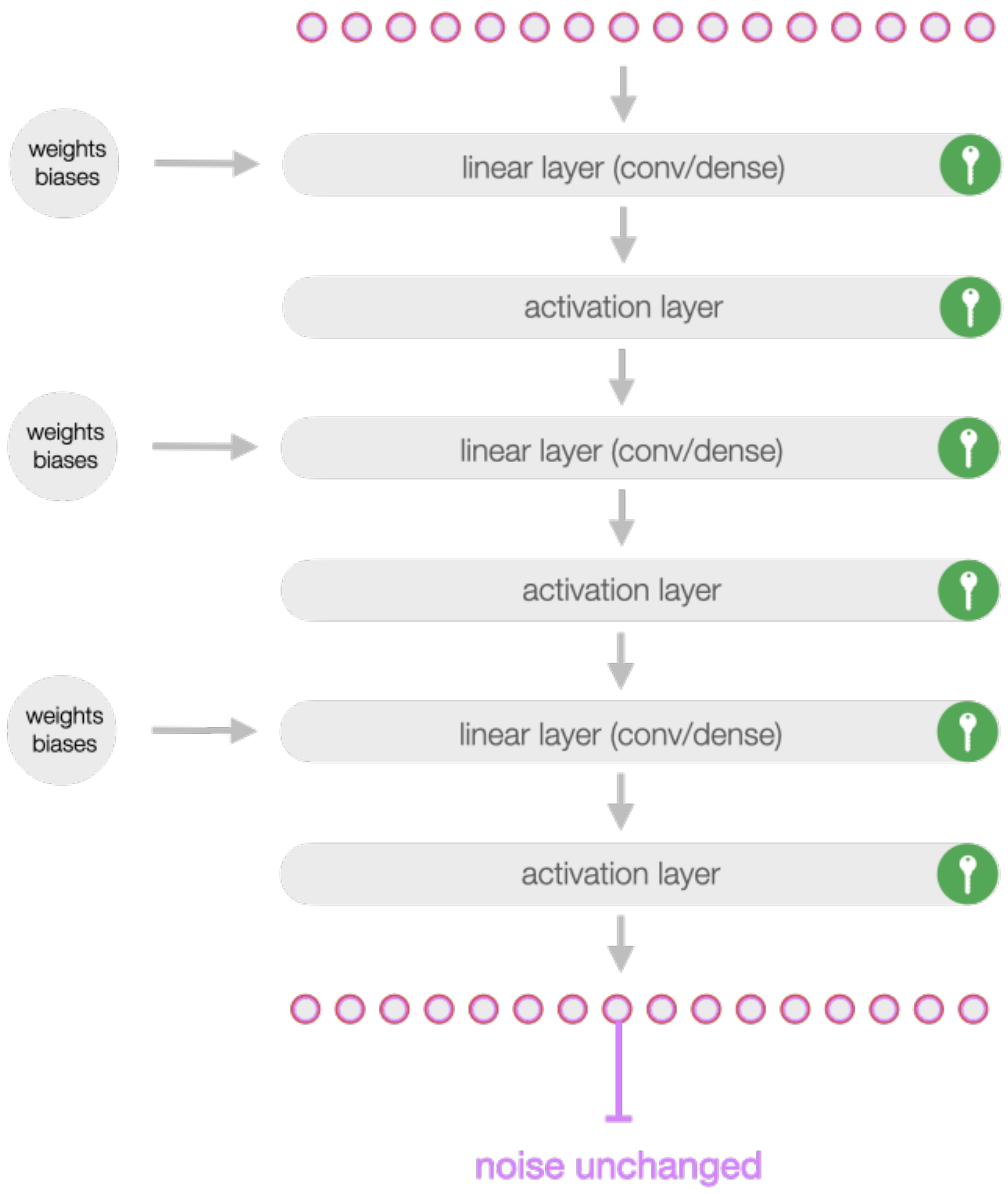
No bootstrapping: way too slow!

2

Activation layers: polynomial approximation



Homomorphic Inference: Fully HE

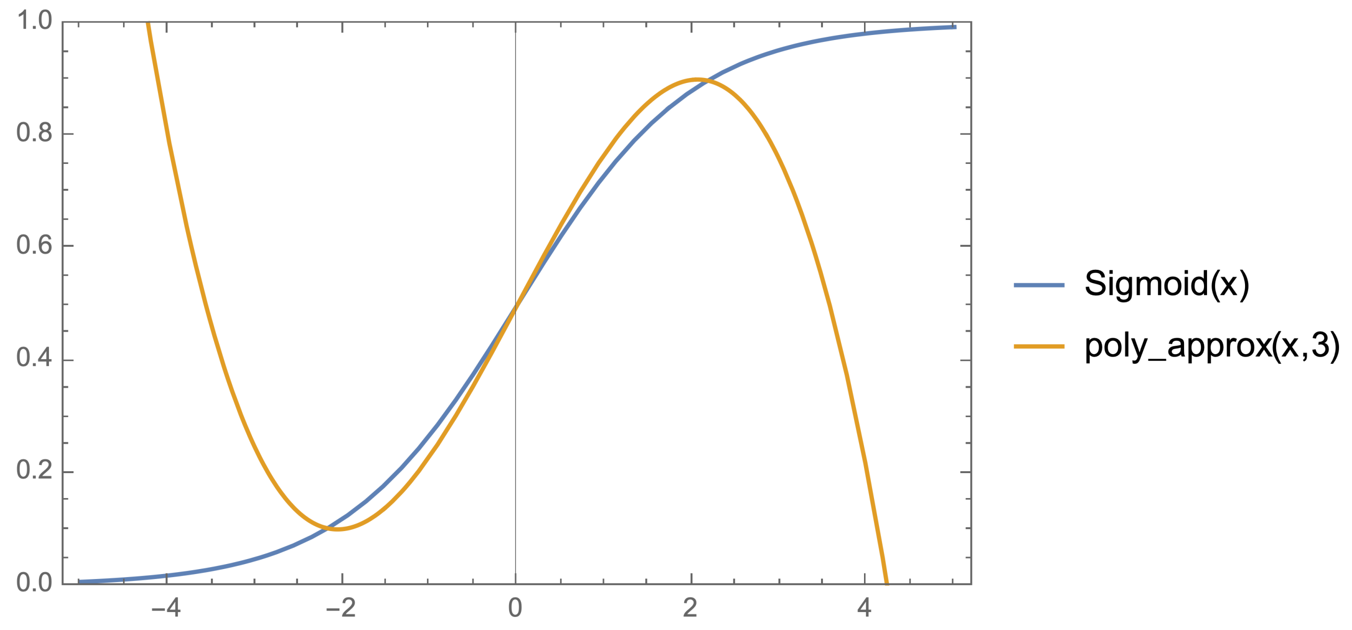


1

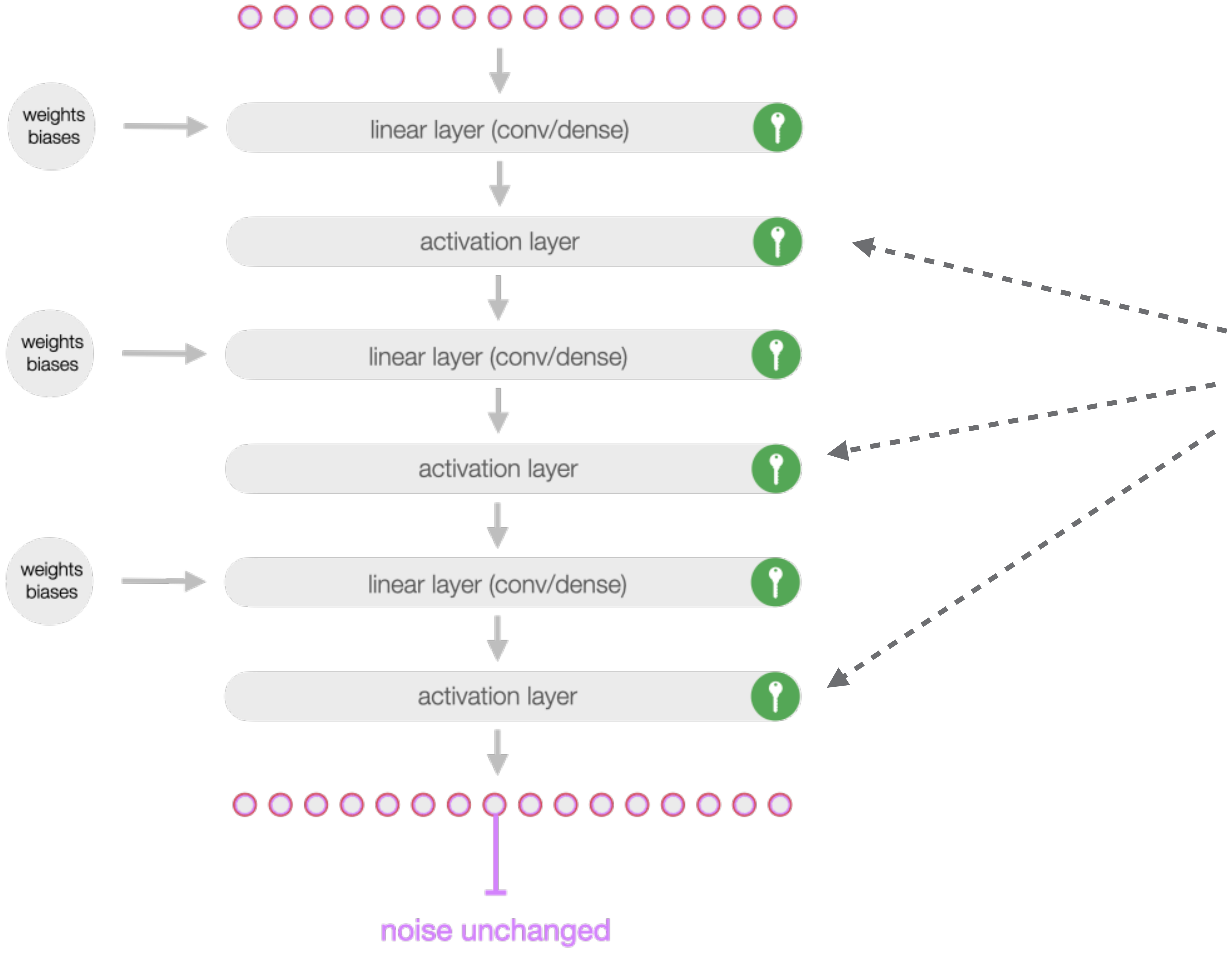
Bootstrapping (but super slow)

2

Activation layers: polynomial approximation

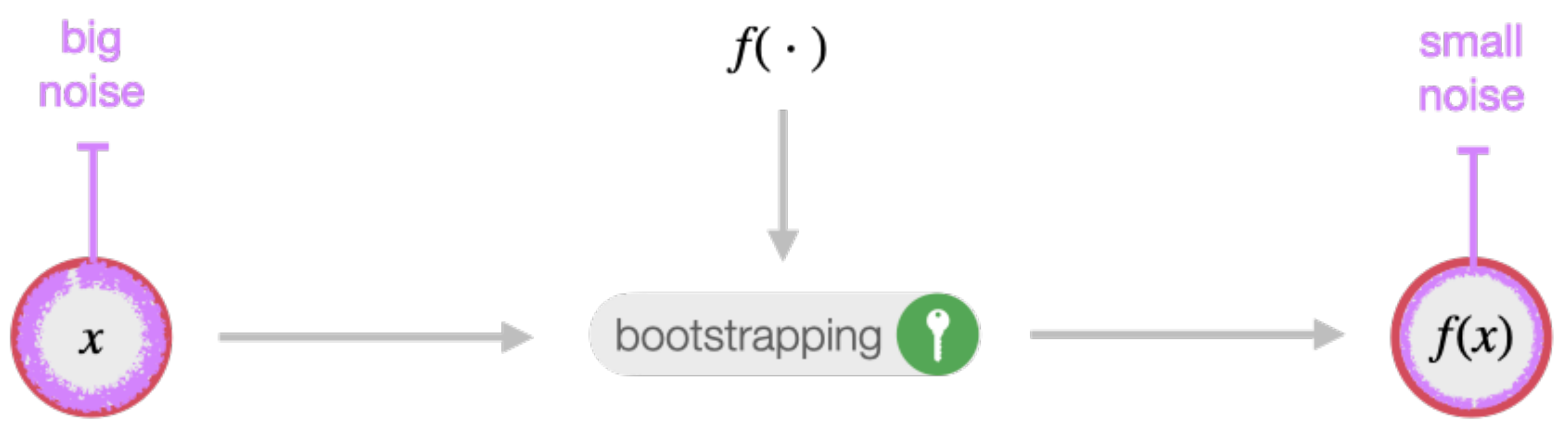


Homomorphic Inference: The "Zama way"



1+2

Activation layers: programmable bootstrapping



PBS is super fast: ~20 ms

A new computational paradigm

ZAMA

Kolmogorov Superposition Theorem (KST)

1957

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{2n+1} g_i \left(\sum_{j=1}^{n} f_{ij}(x_j) \right)$$

univariate

A new computational paradigm

ZAMA

Kolmogorov Superposition Theorem (KST) 1957

$$f\left(x_1, x_2, \ldots, x_n\right) = \sum_{i=1}^{2n+1} g_i\left(\sum_{j=1}^{n} f_{ij}\left(x_j\right)\right)$$

univariate

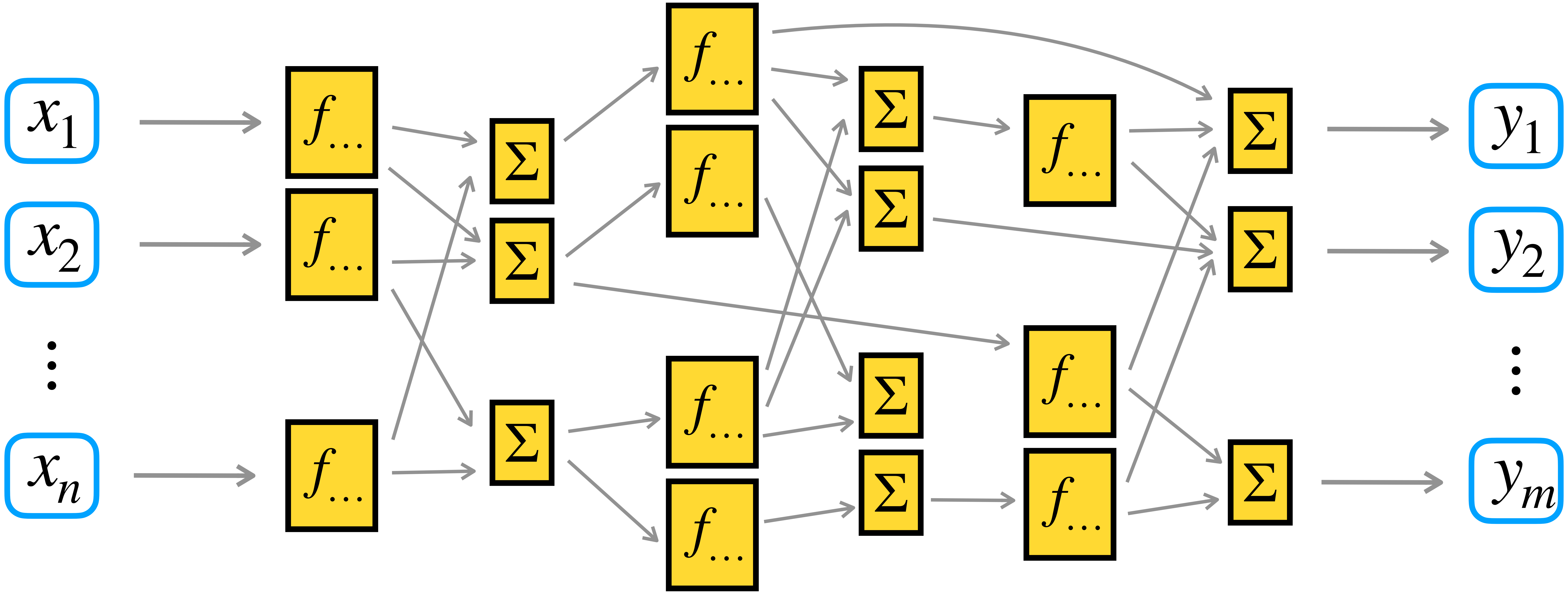
Ridge decomposition or approximation

$$f(x_1, \ldots, x_n) \approx \sum_{i=1}^{r} g_i\left(\sum_{j=1}^{n} a_{ij} \cdot x_j\right)$$

univariate $a_{ij} \in \mathbb{Z}$

A new computational paradigm

circuit of univariate functions



= graph mixing univariate functions and linear combinations

The Concrete Compiler

(and a glance at the whole Concrete stack)

Exact vs. approximate computing with TFHE

	plaintexts	continuous torus encoding	discretized torus encoding	noisy discretized torus	encrypted domain
exact paradigm	$m \in \mathbb{Z}_p$ • p odd • p even	$\mu \in \mathbb{R}/\mathbb{Z}$ $\mu = \frac{m}{p} \mod 1$	$\mu \in \mathbb{Z}_{2^{32}}$ $\mu = \left\lfloor \frac{2^{32}}{p} \right\rceil m \mod 2^{32}$	$\mu + \varepsilon \in \mathbb{Z}_{2^{32}}$ pdf $\left(\frac{\varepsilon}{2^{32}}\right) \approx \mathcal{N}(0,\sigma)$	$\mathrm{LWE}_{sk}\left(\mu + \varepsilon\right)$ $\in \mathbb{Z}_{2^{32}}^{n+1}$
approximate paradigm	$x \in [x^-, x^+] \subset \mathbb{R}$ pdf$(x) \approx \mathcal{N}(x, \sigma_x)$ $\Pr\left[x \notin [x^-, x^+]\right] = 0$	$\mu \in \mathbb{R}/\mathbb{Z}$ $\mu = \frac{x - x^-}{x^+ - x^-} \mod 1$	$\mu \in \mathbb{Z}_{2^{32}}$ $\mu = \left\lfloor \frac{x - x^-}{x^+ - x^-} \cdot 2^{32} \right\rceil \mod 2^{32}$		

Modular TFHE circuits

exact paradigm

$m \in \mathbb{Z}_p$

M = multisum

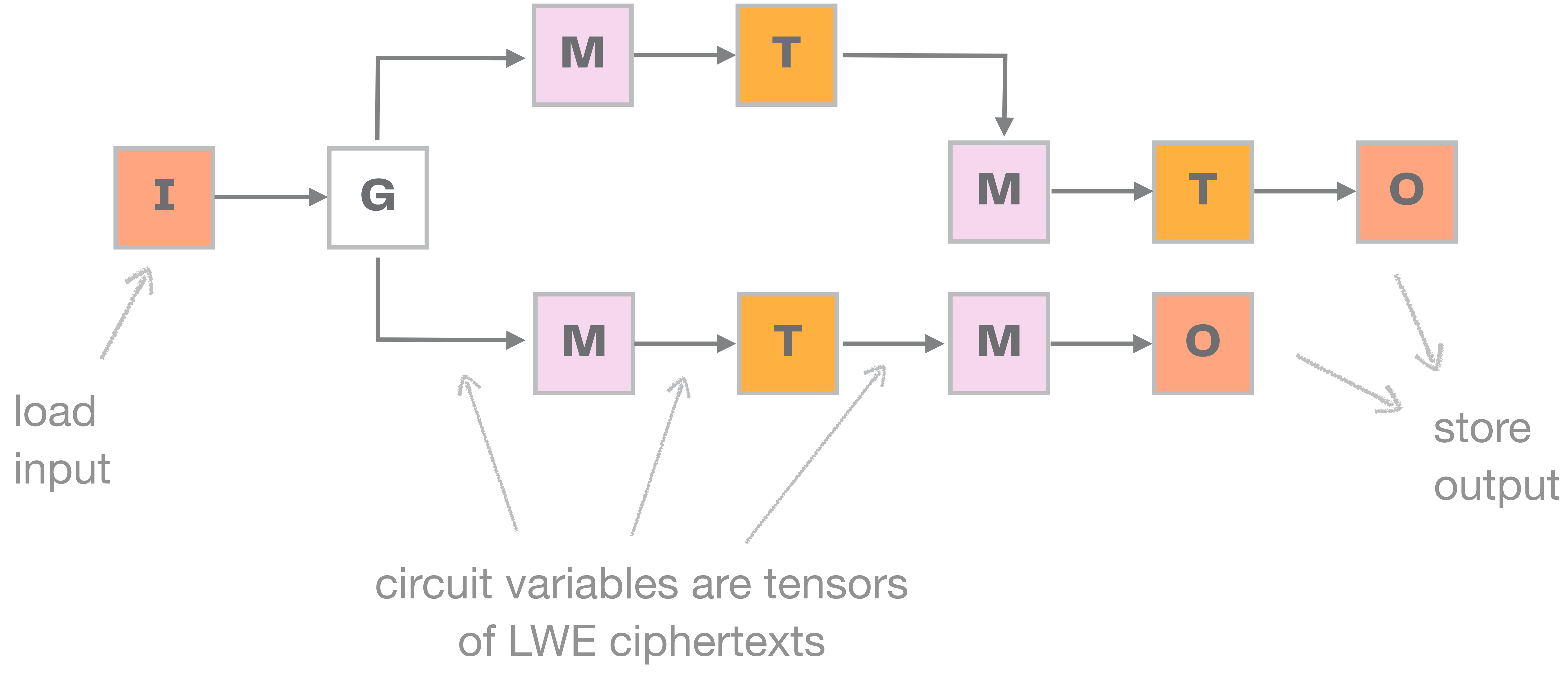
T = lookup table

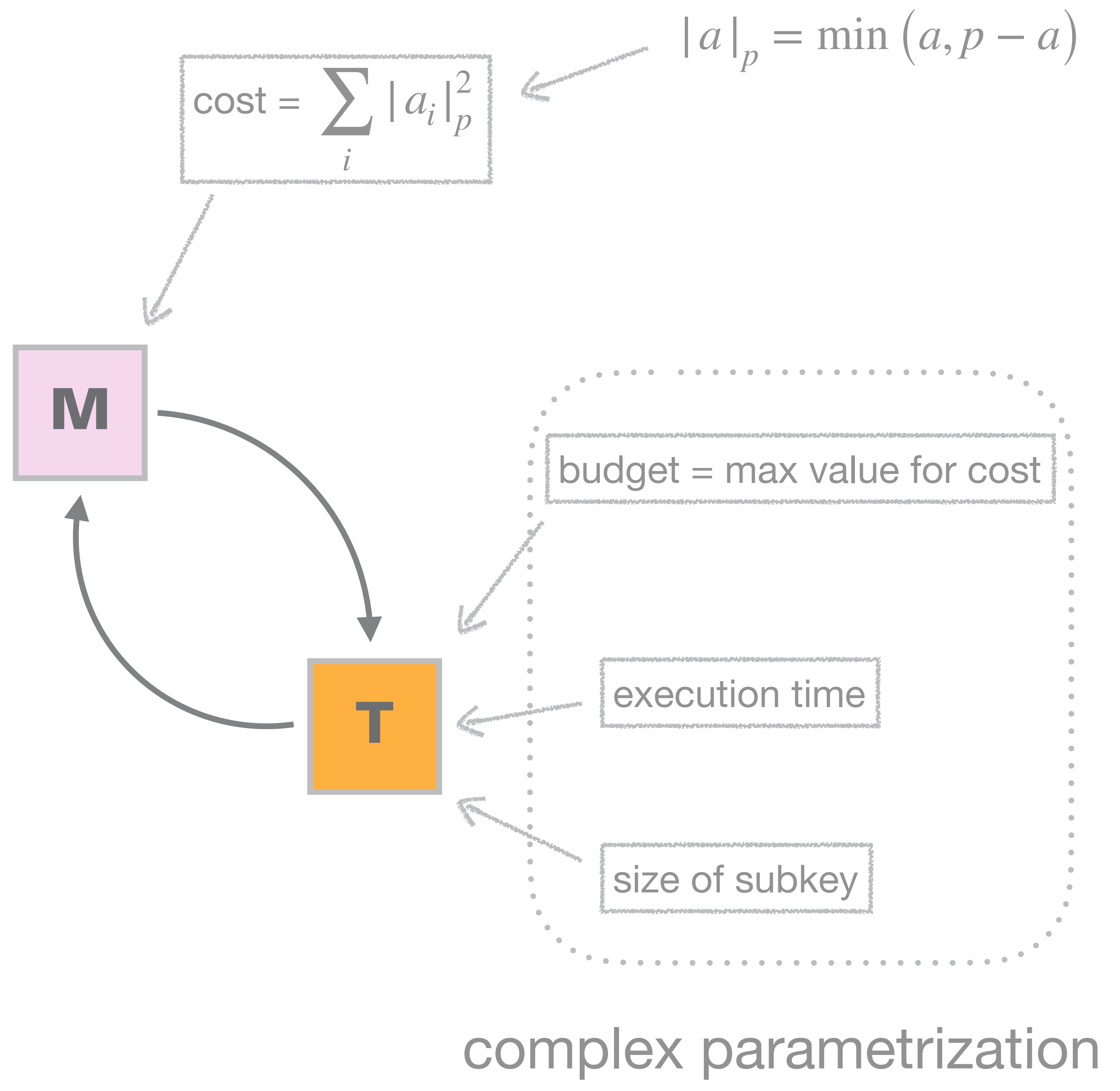
G = grab

- concatenation
- splits
- reshape

I = input

O = output



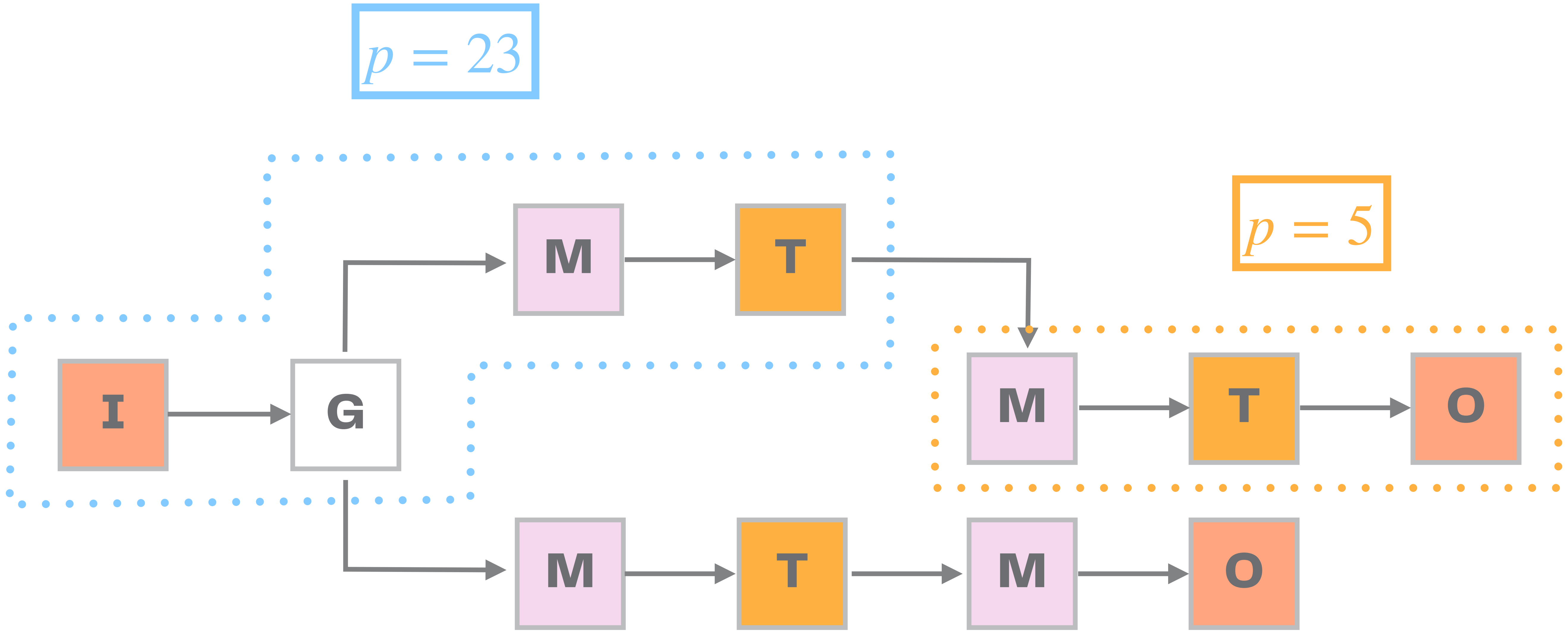


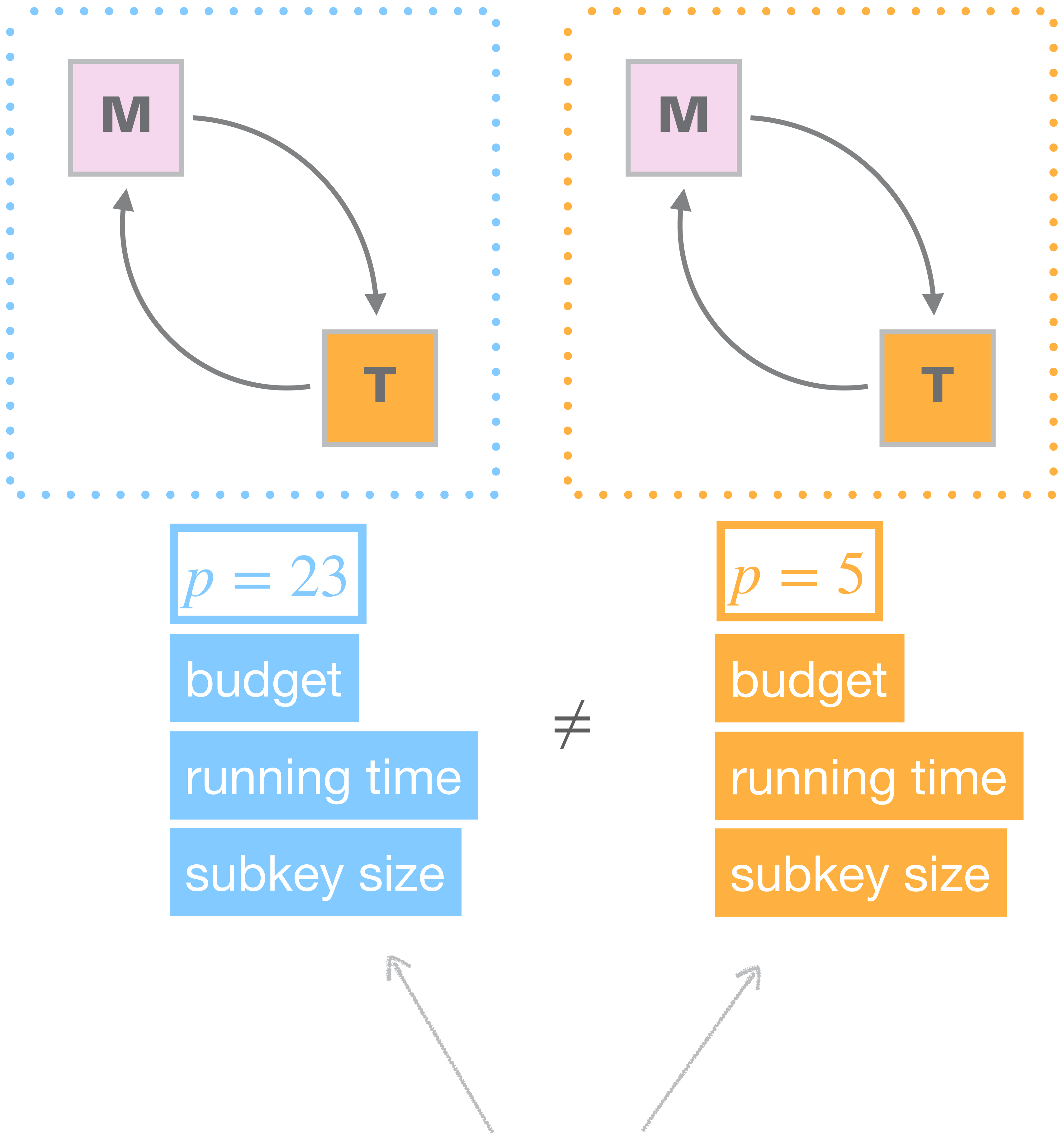
Multi-modular TFHE circuits

exact paradigm

$m \in \mathbb{Z}_p$

Circuits typically switch back and forth between several moduli $p_1, \ldots, p_k$



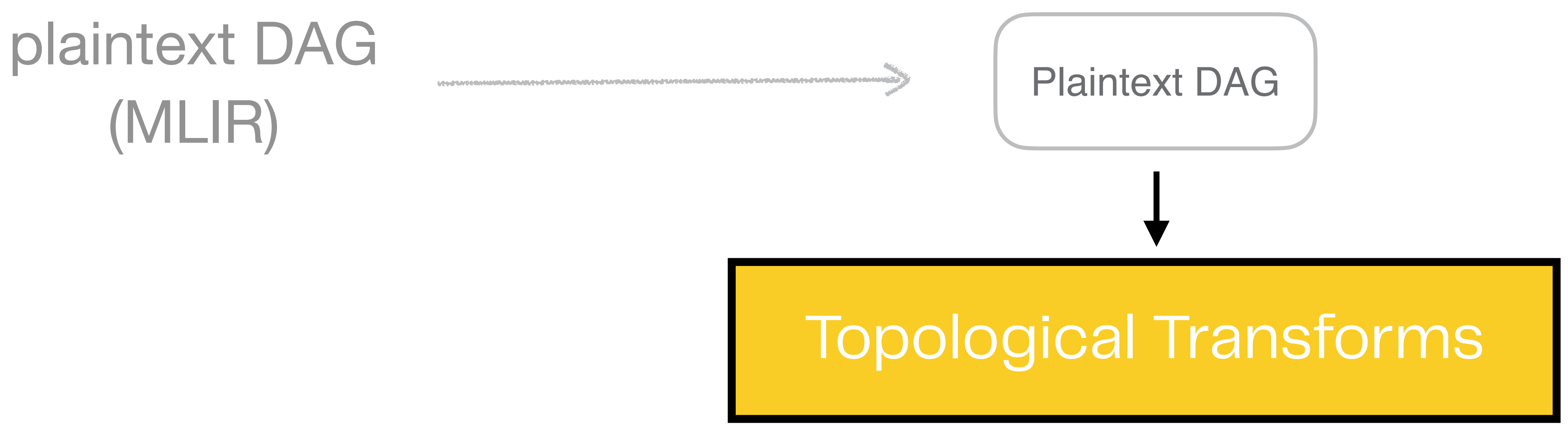


automated generation of optimal parameters from circuit topology

The Concrete Compiler (CC)

The Concrete Compiler (CC)

The Concrete Compiler (CC)

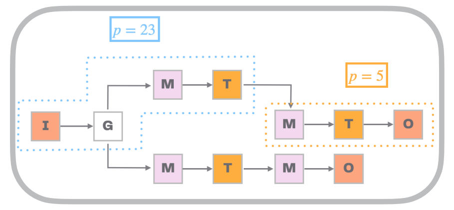


The Concrete Compiler (CC)

plaintext DAG
(MLIR)

Plaintext DAG

Topological Transforms



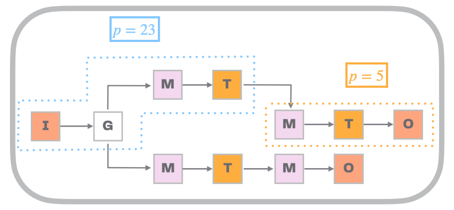
The Concrete Compiler (CC)

plaintext DAG
(MLIR)

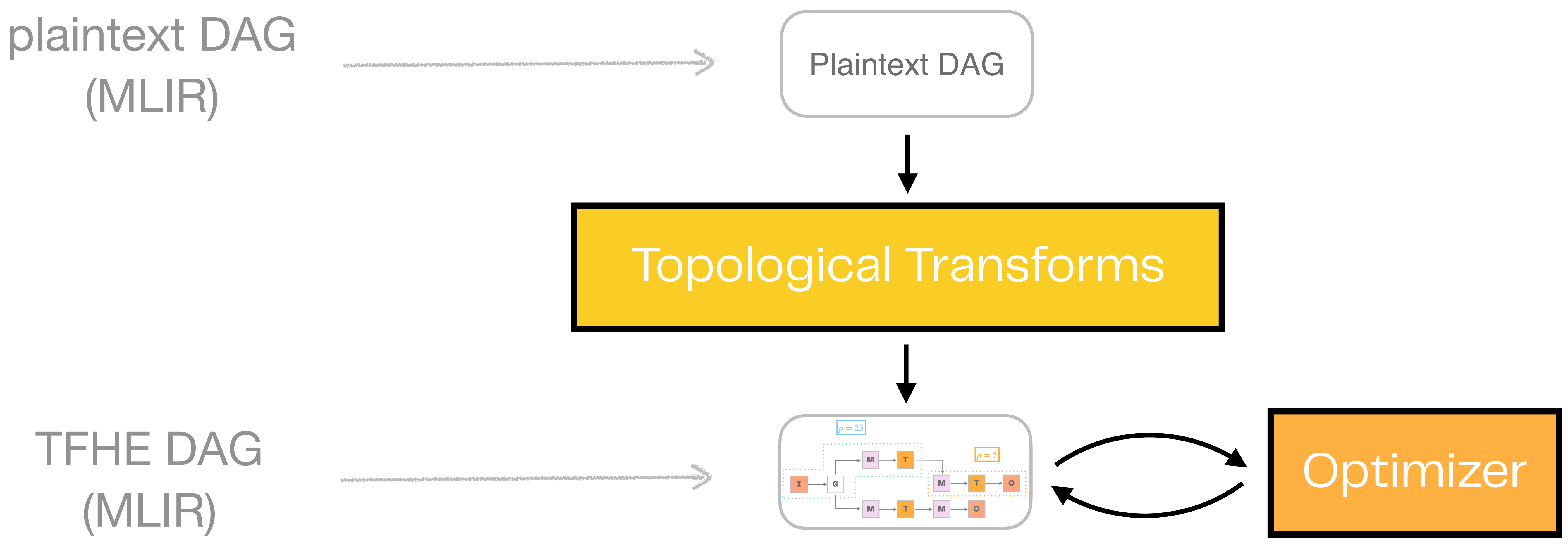
Plaintext DAG

Topological Transforms

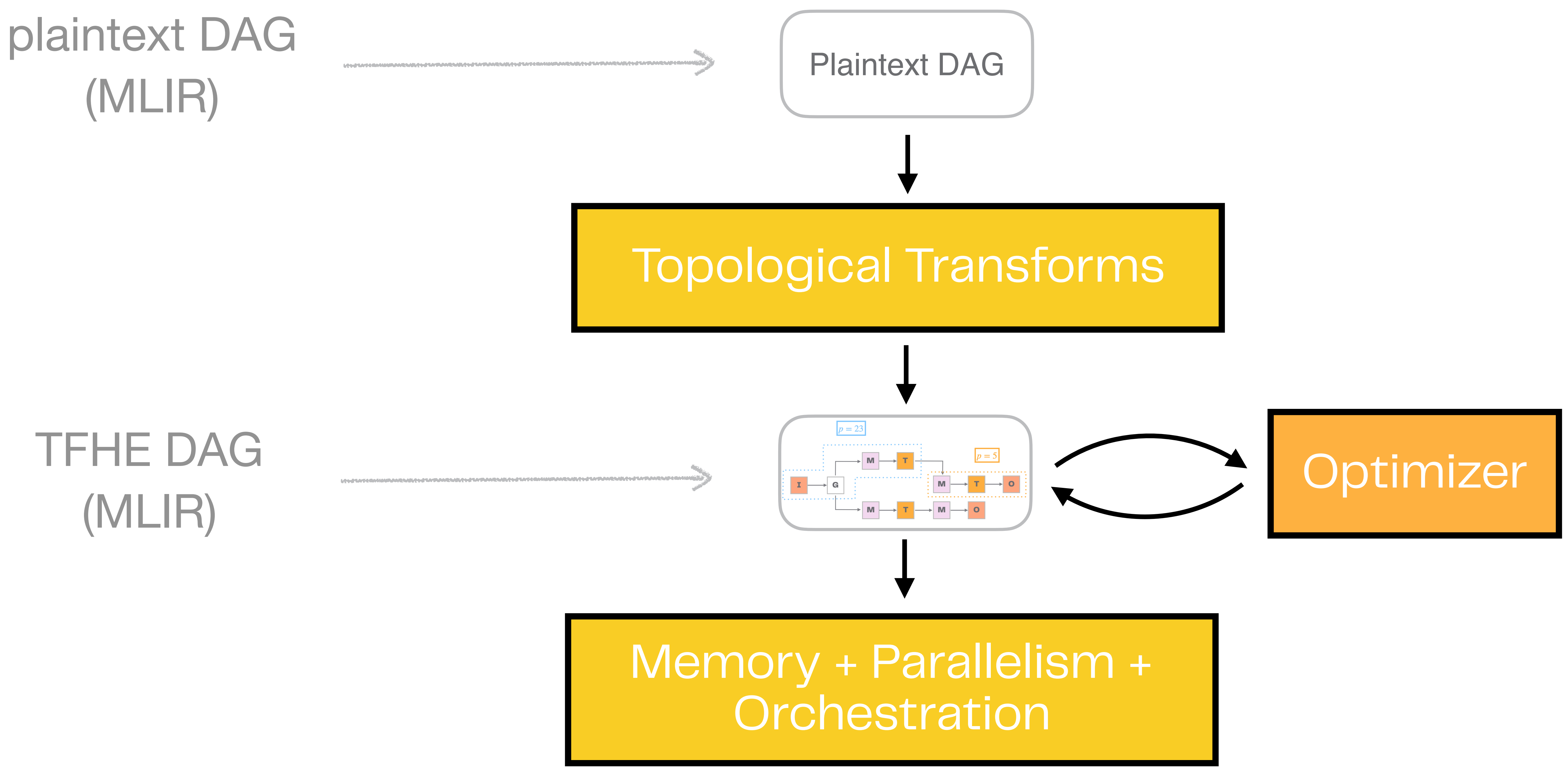
TFHE DAG
(MLIR)



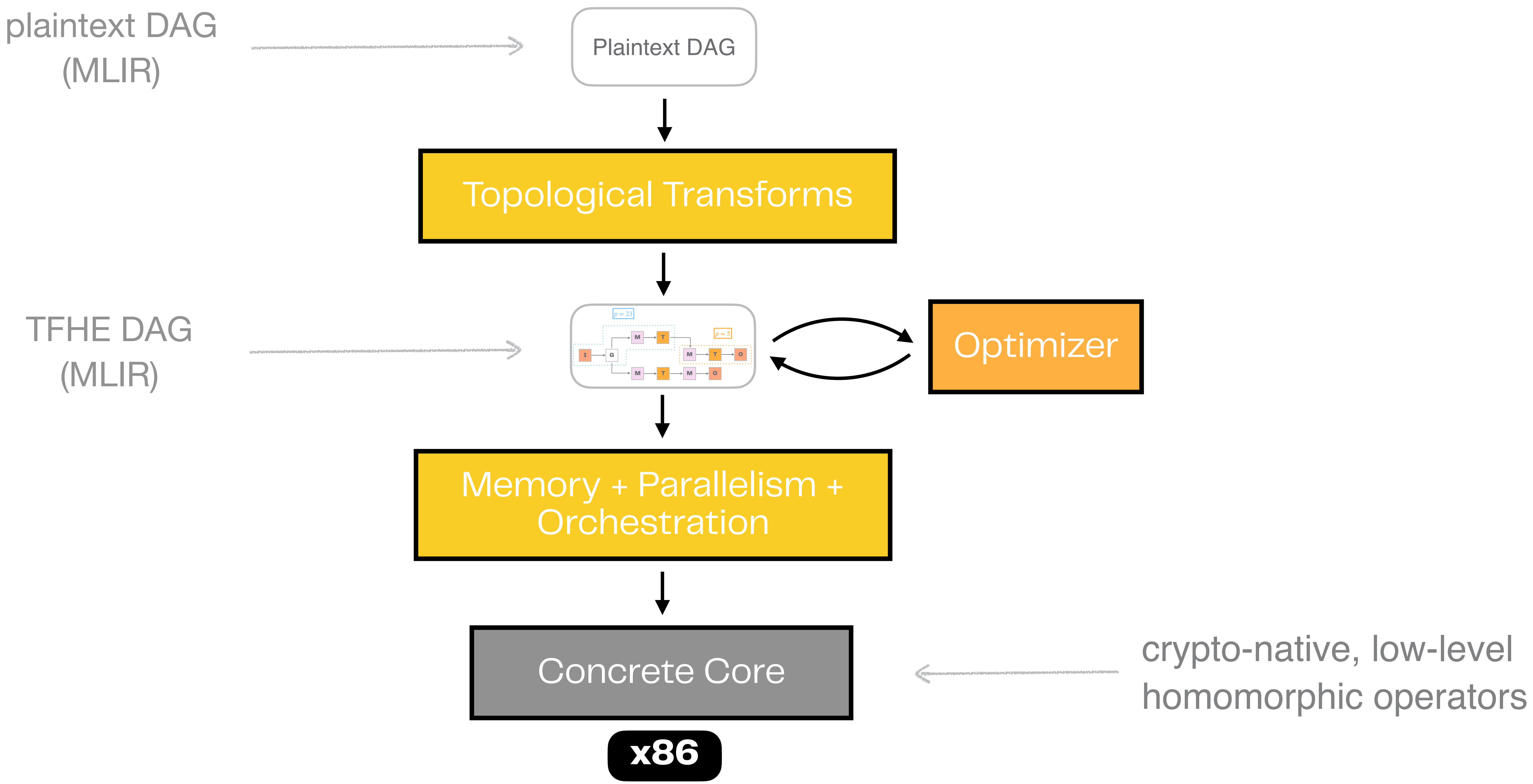
The Concrete Compiler (CC)



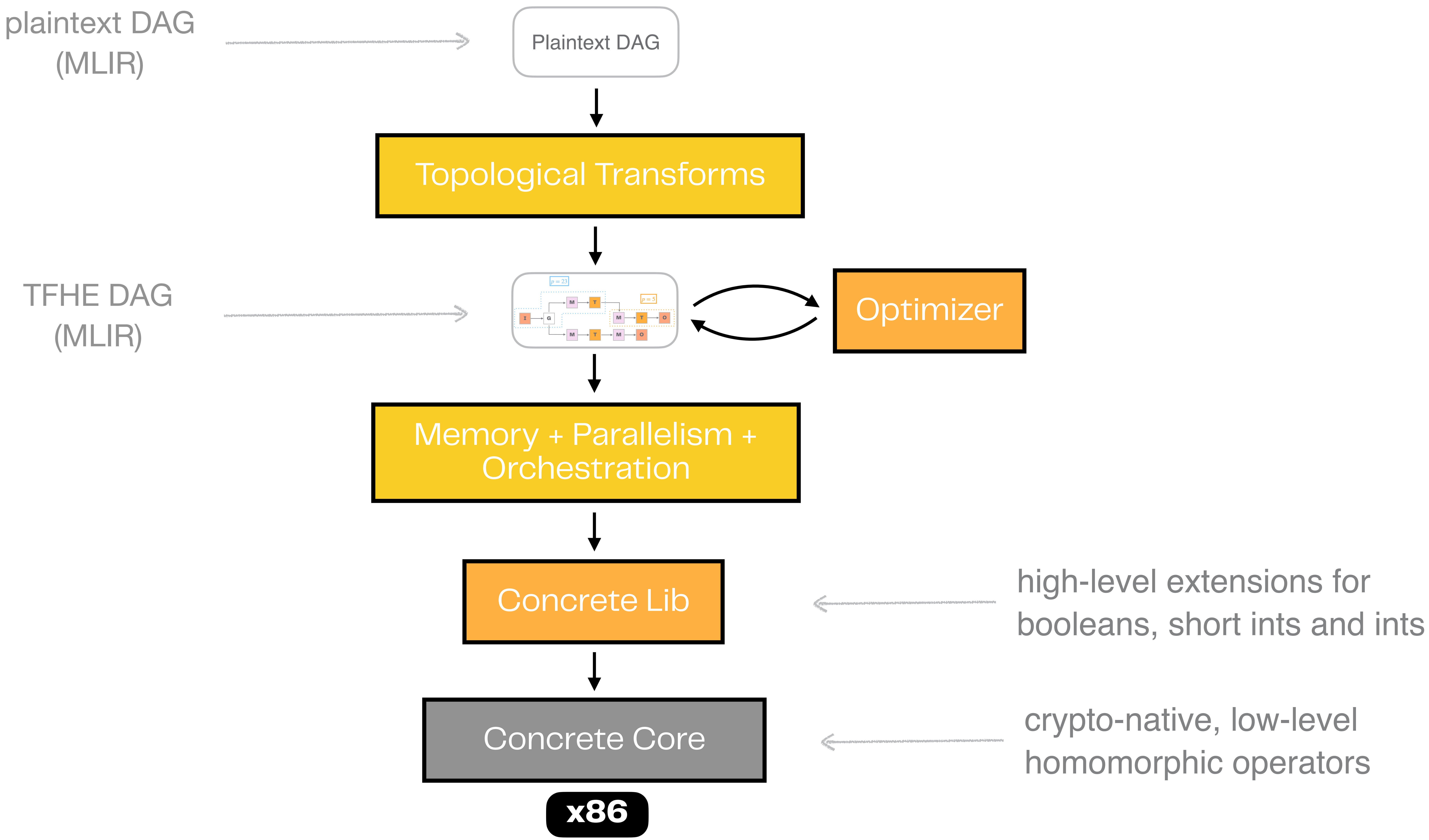
The Concrete Compiler (CC)



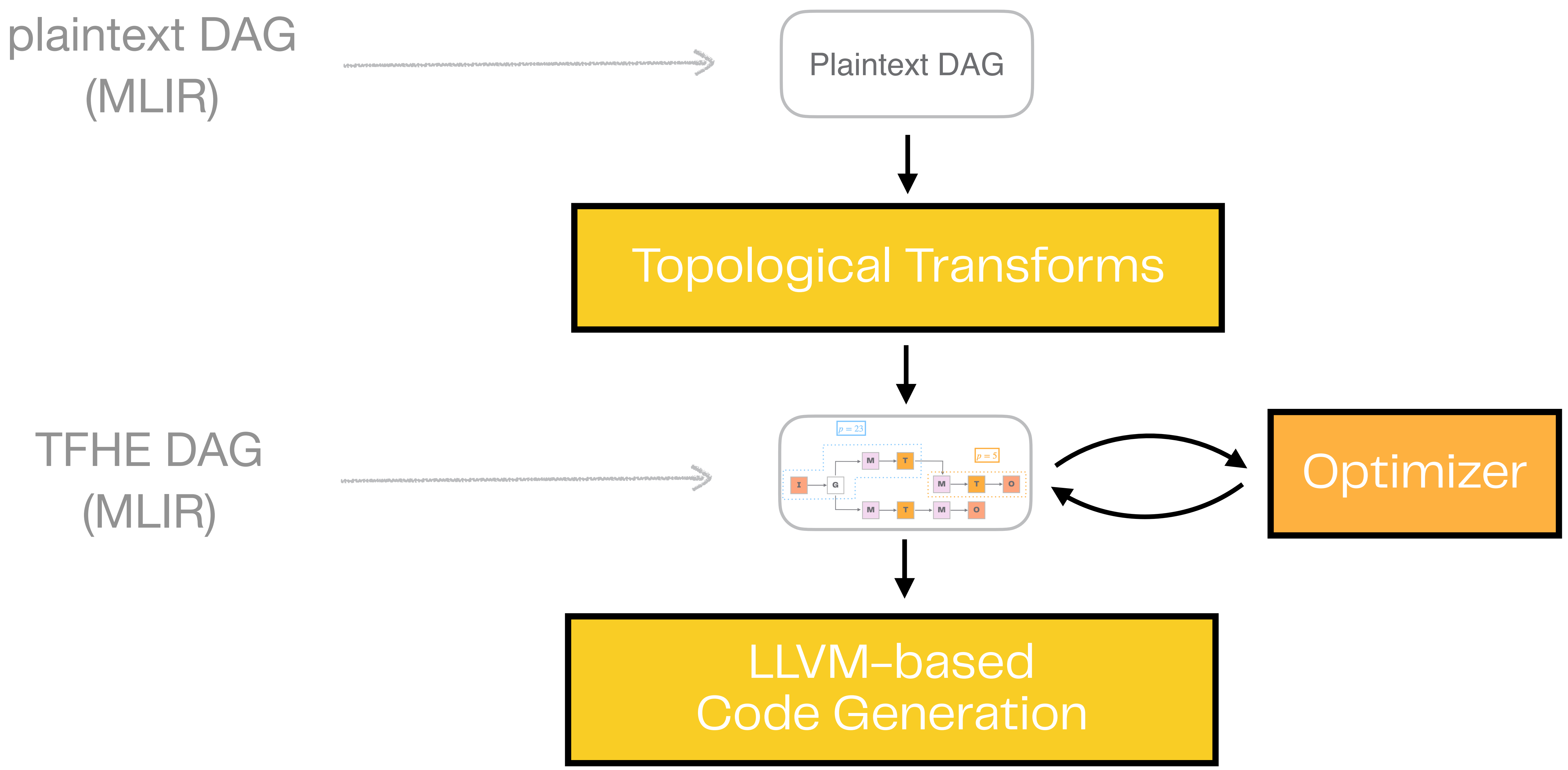
The Concrete Compiler (CC)



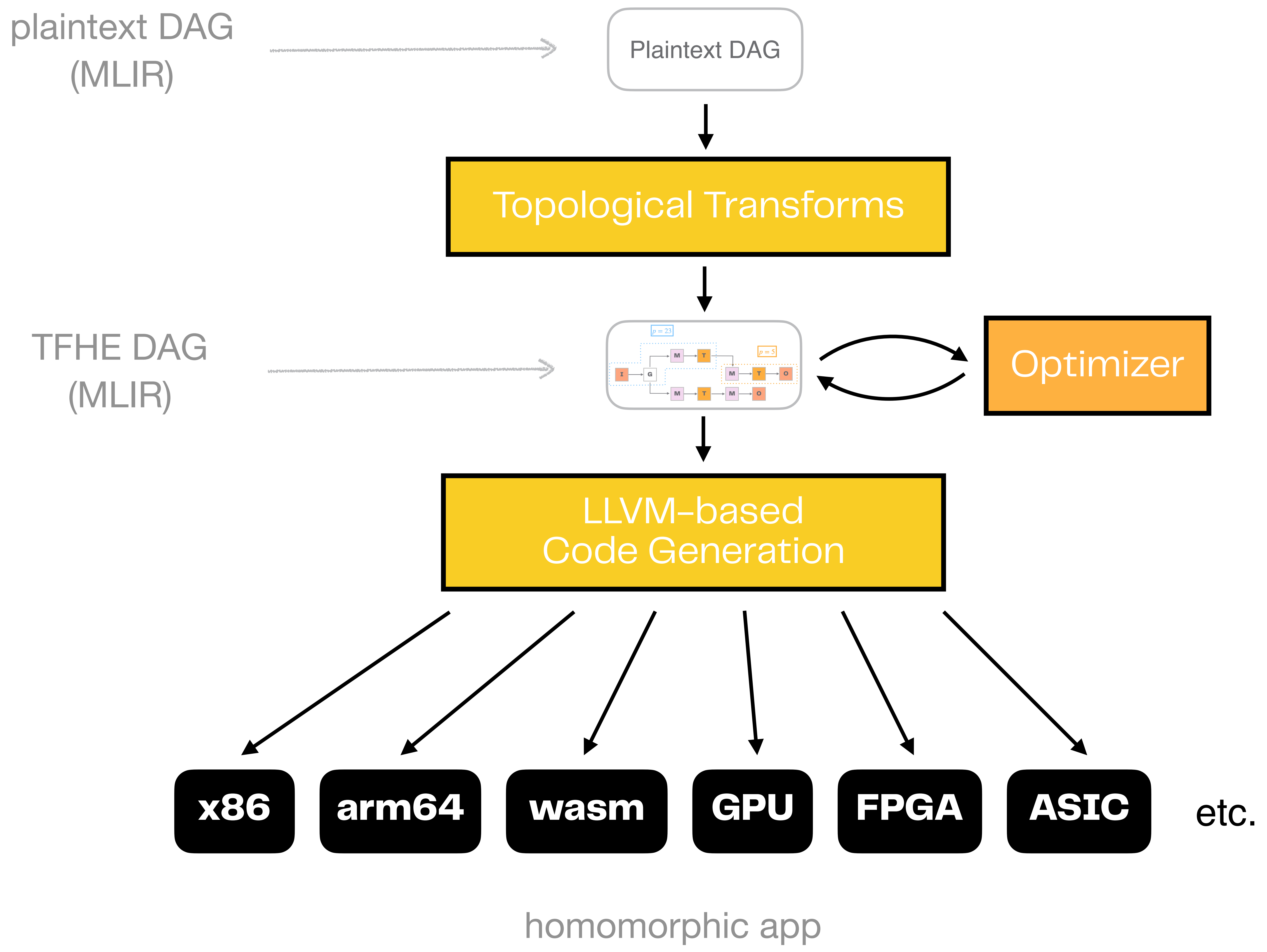
The Concrete Compiler (CC)



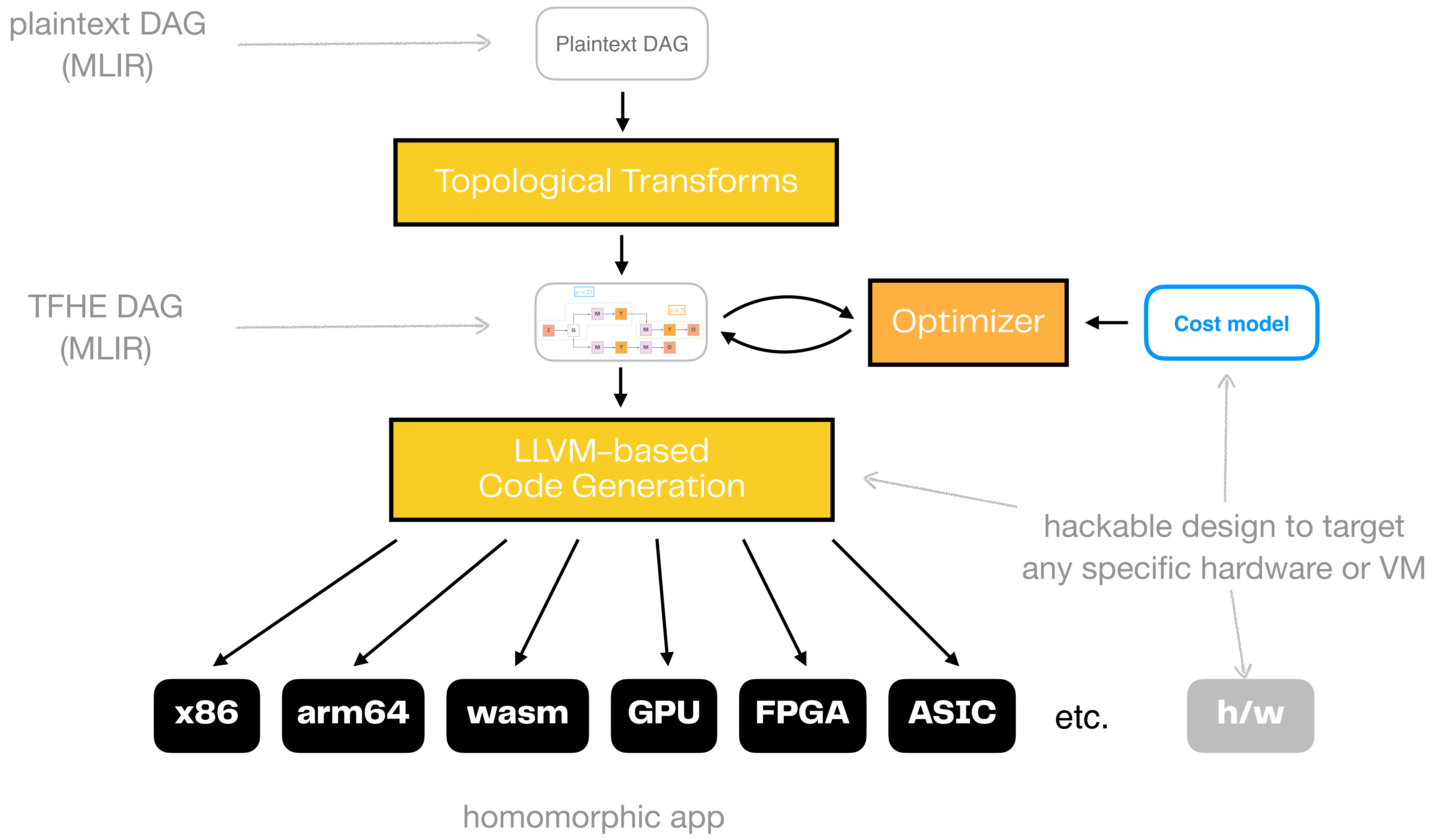
The Concrete Compiler (CC)



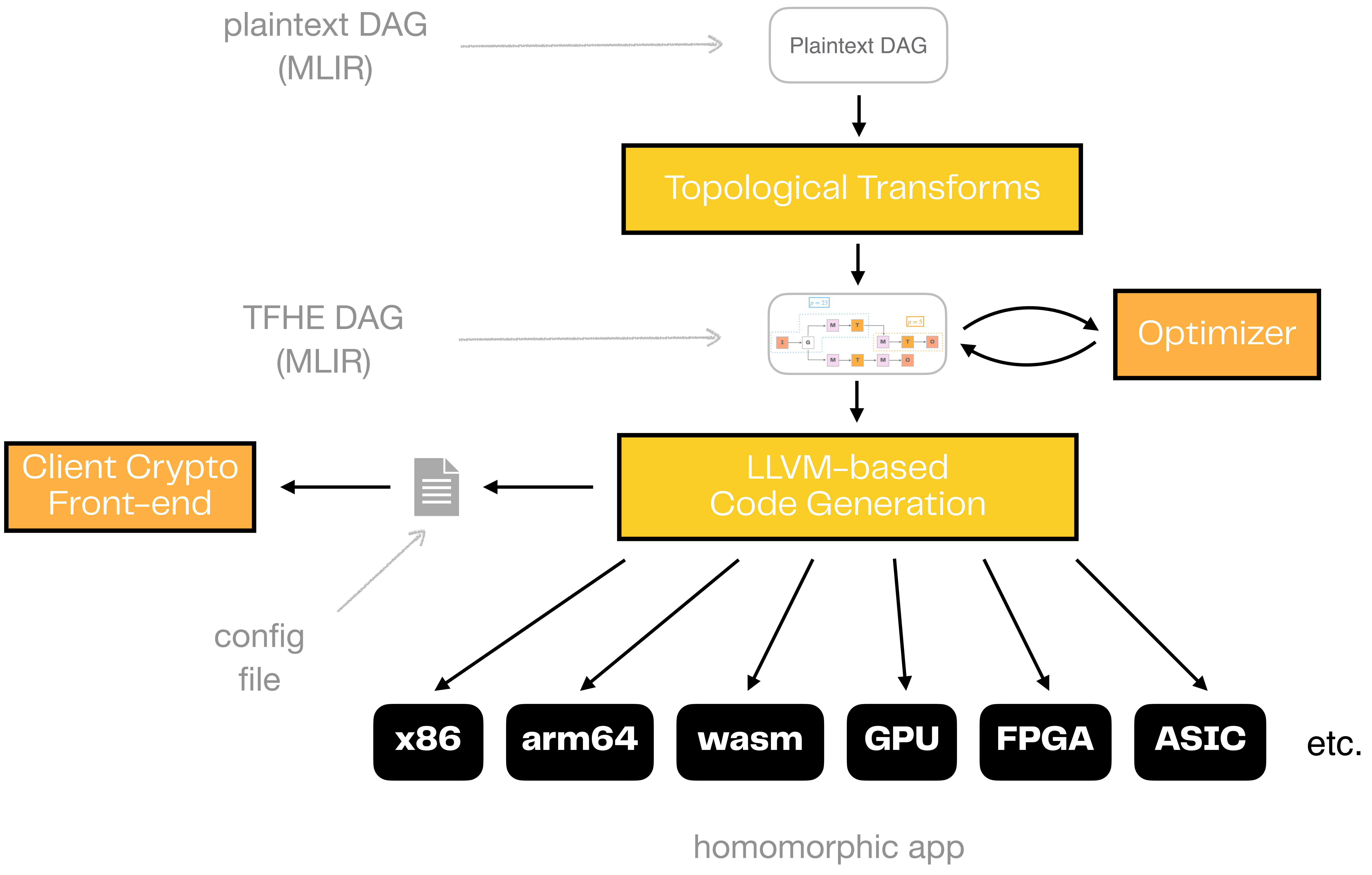
The Concrete Compiler (CC)



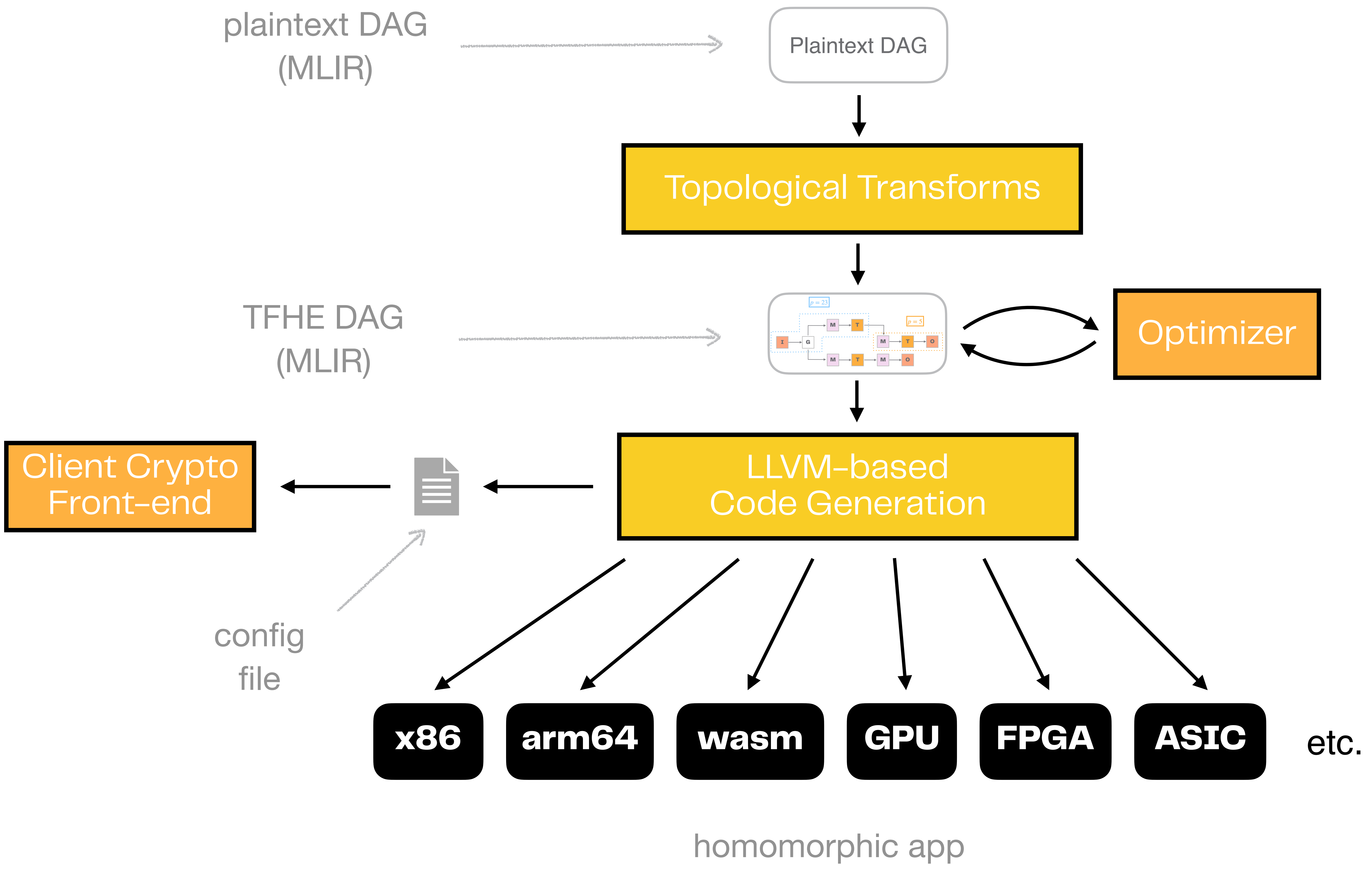
The Concrete Compiler (CC)



The Concrete Compiler (CC)



The Concrete Compiler (CC)



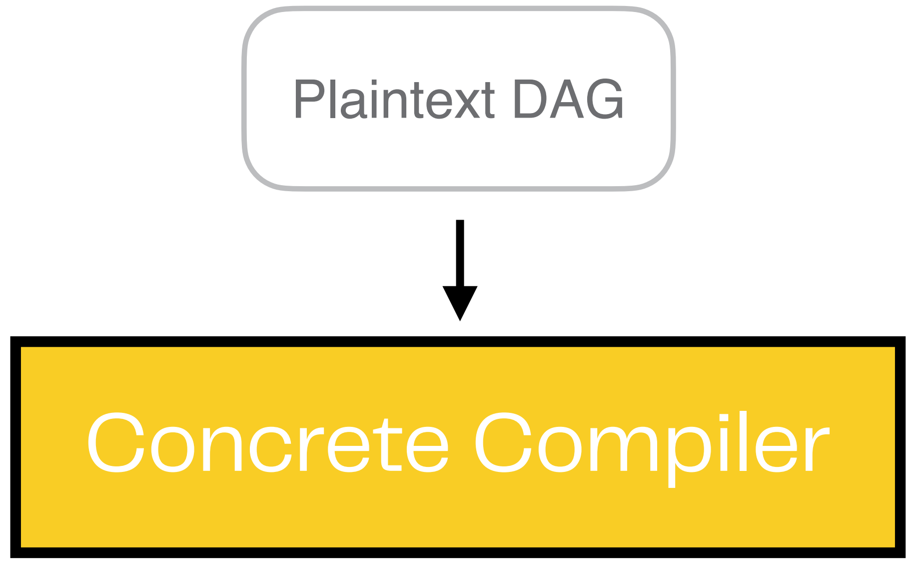
+ TFHE.rs

(for handmade circuits)

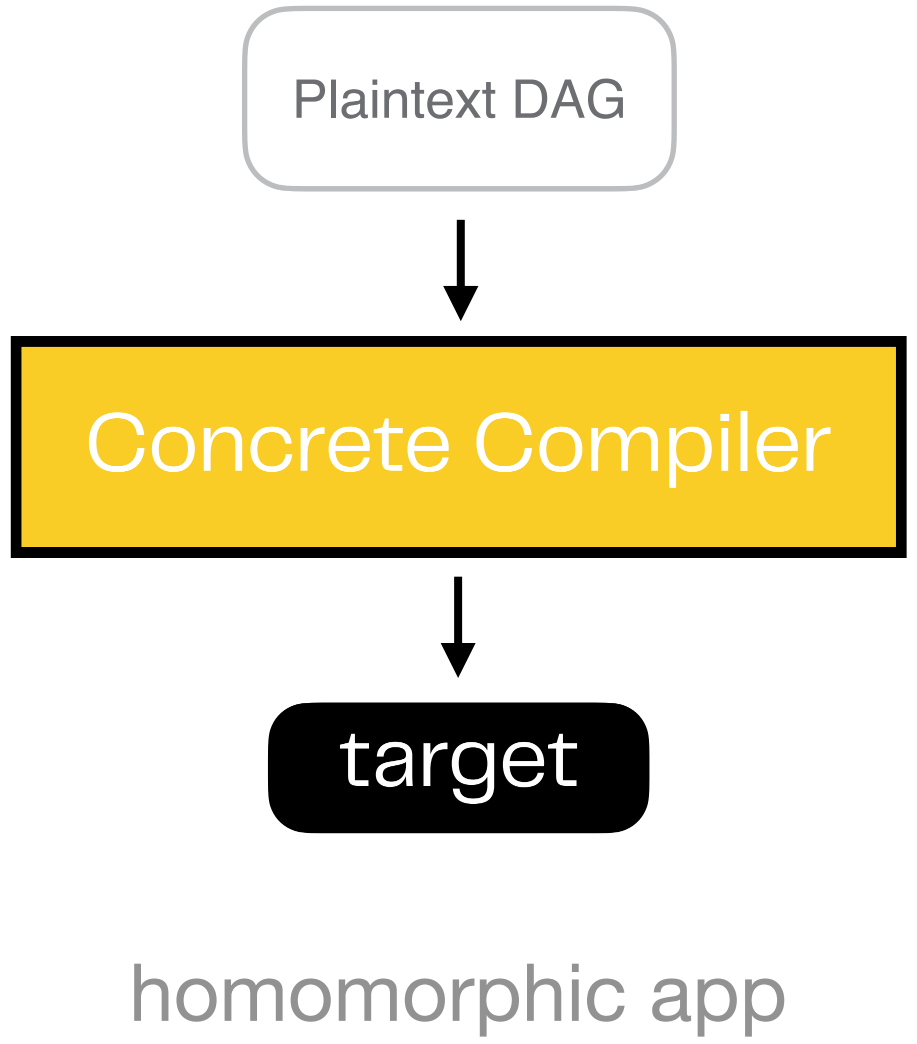
The Concrete stack is a versatile framework

Concrete Compiler

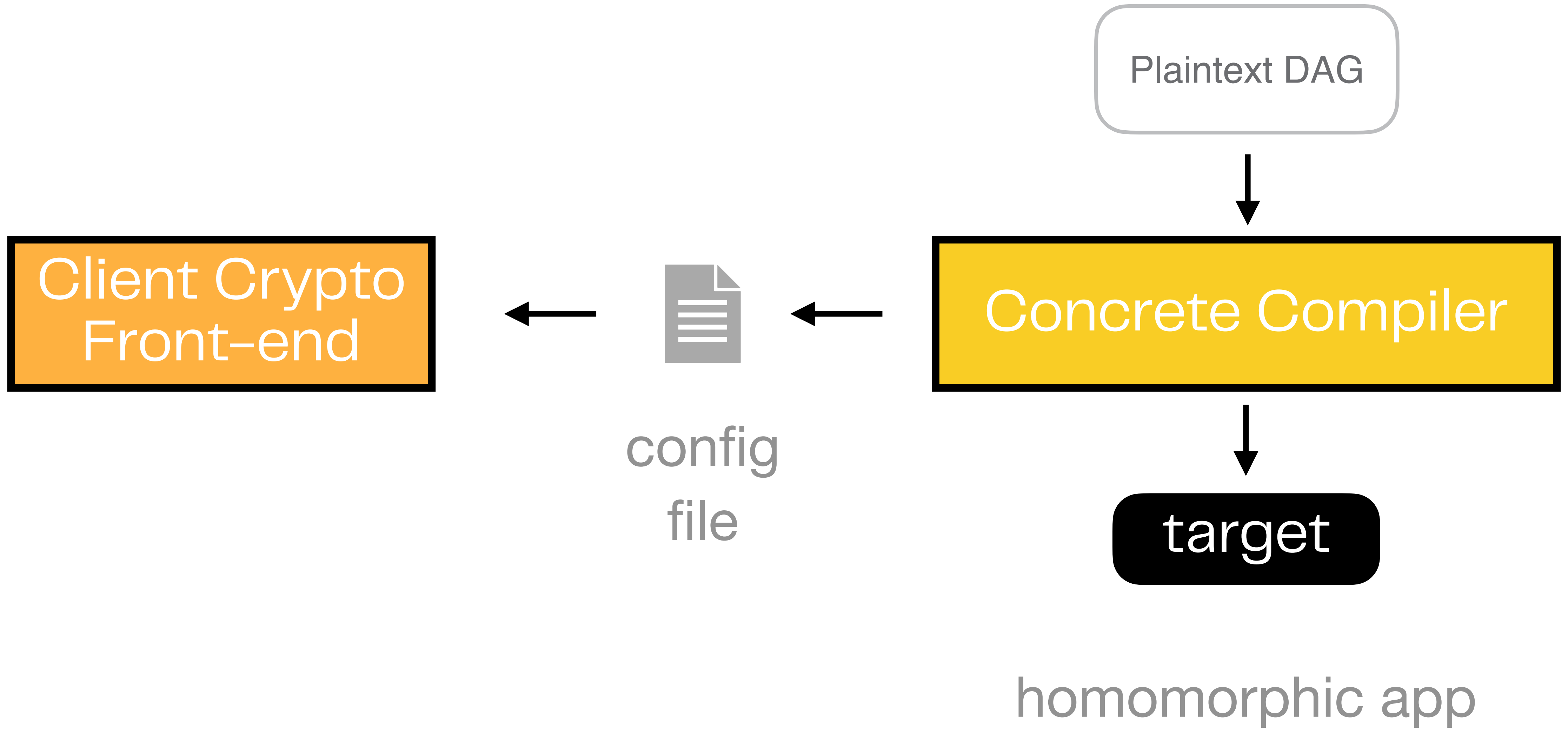
The Concrete stack is a versatile framework



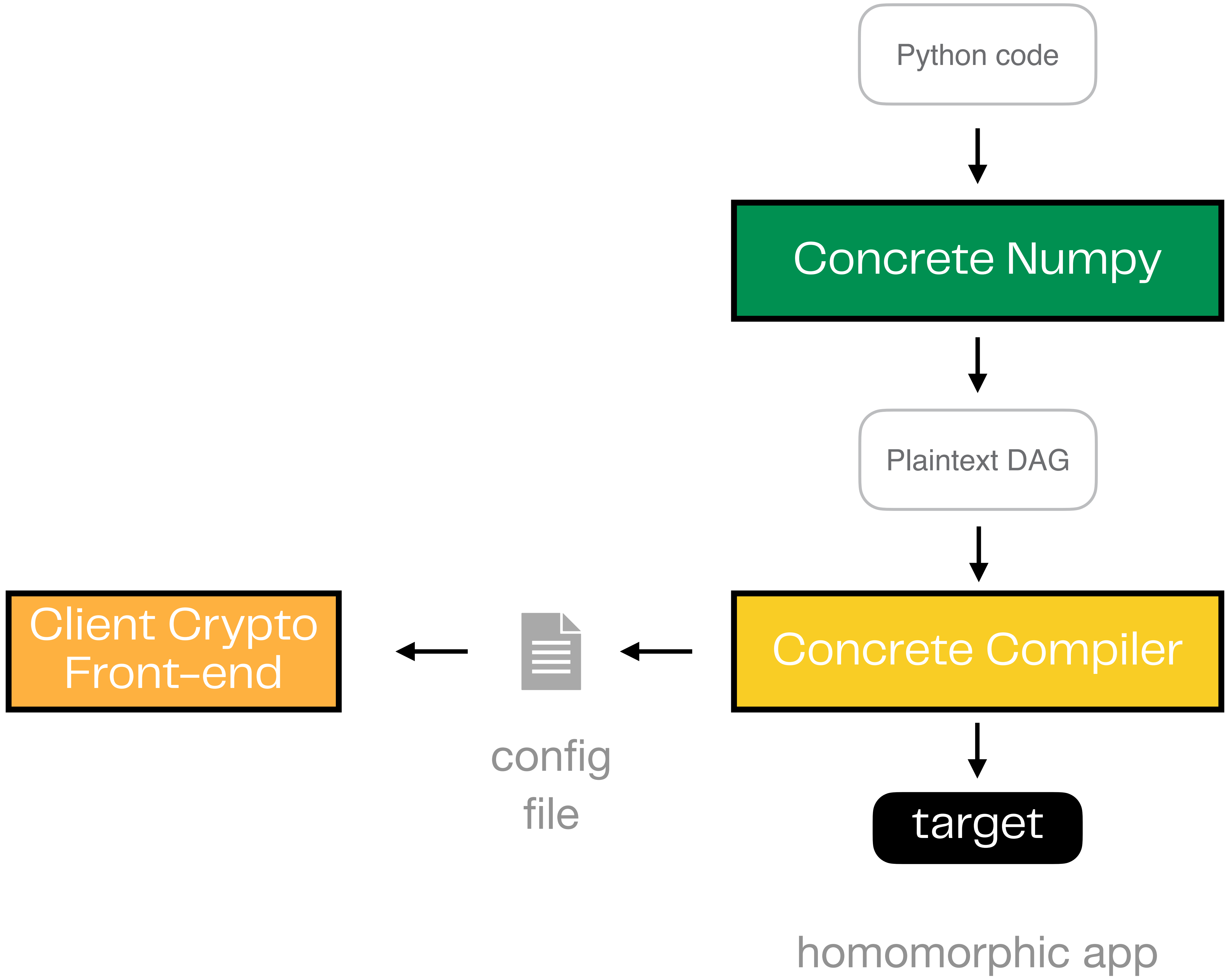
The Concrete stack is a versatile framework



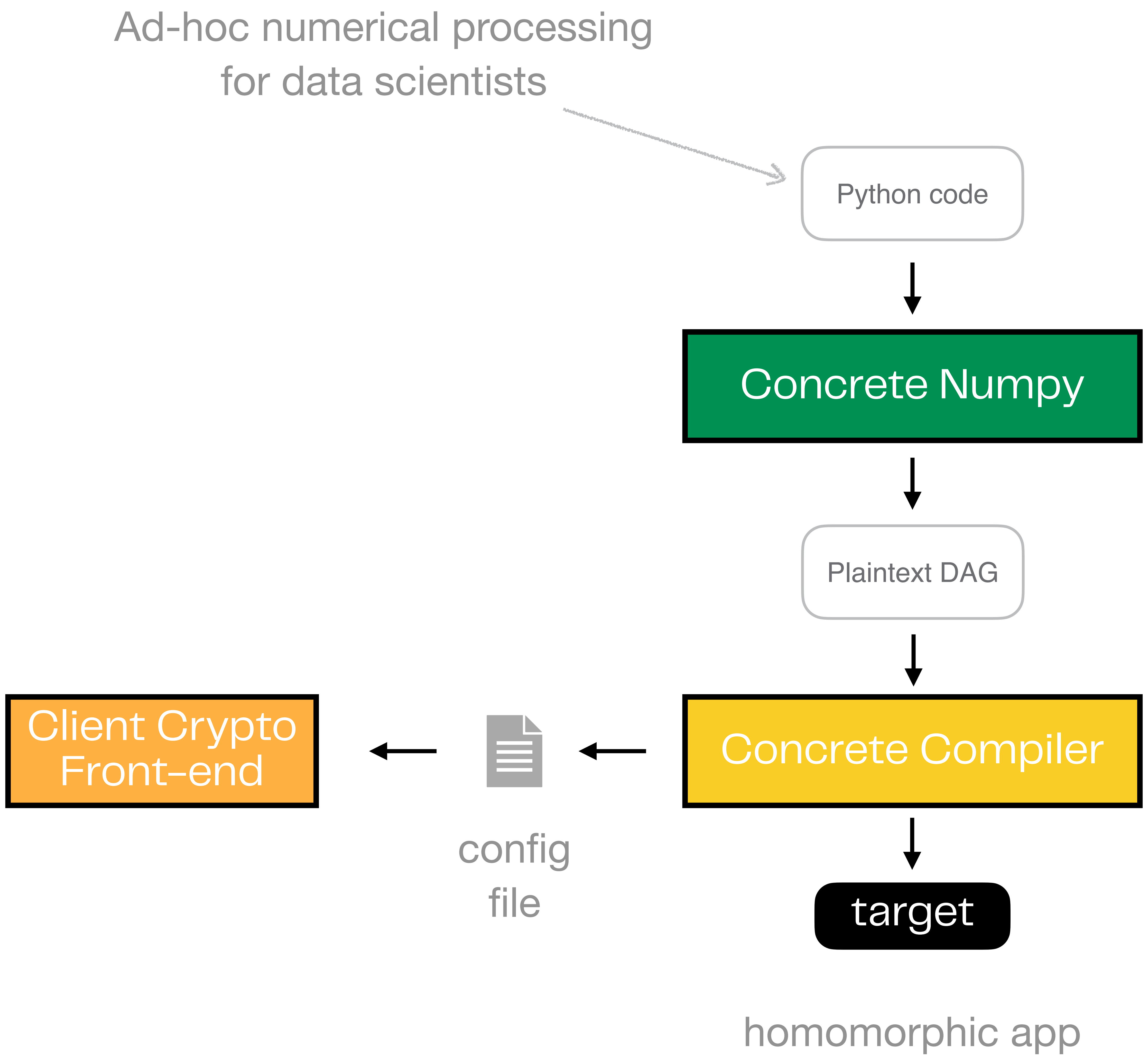
The Concrete stack is a versatile framework



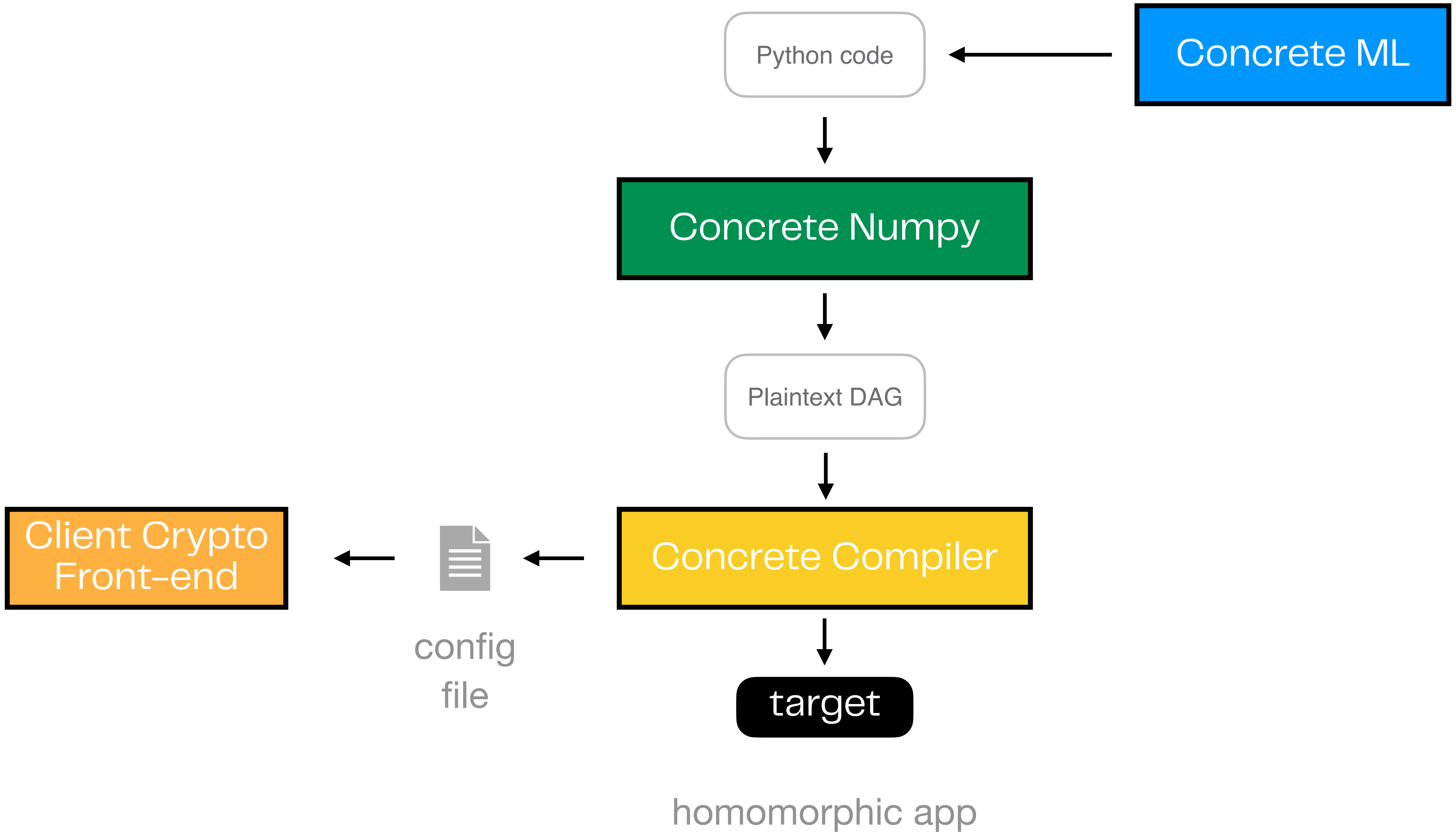
The Concrete stack is a versatile framework



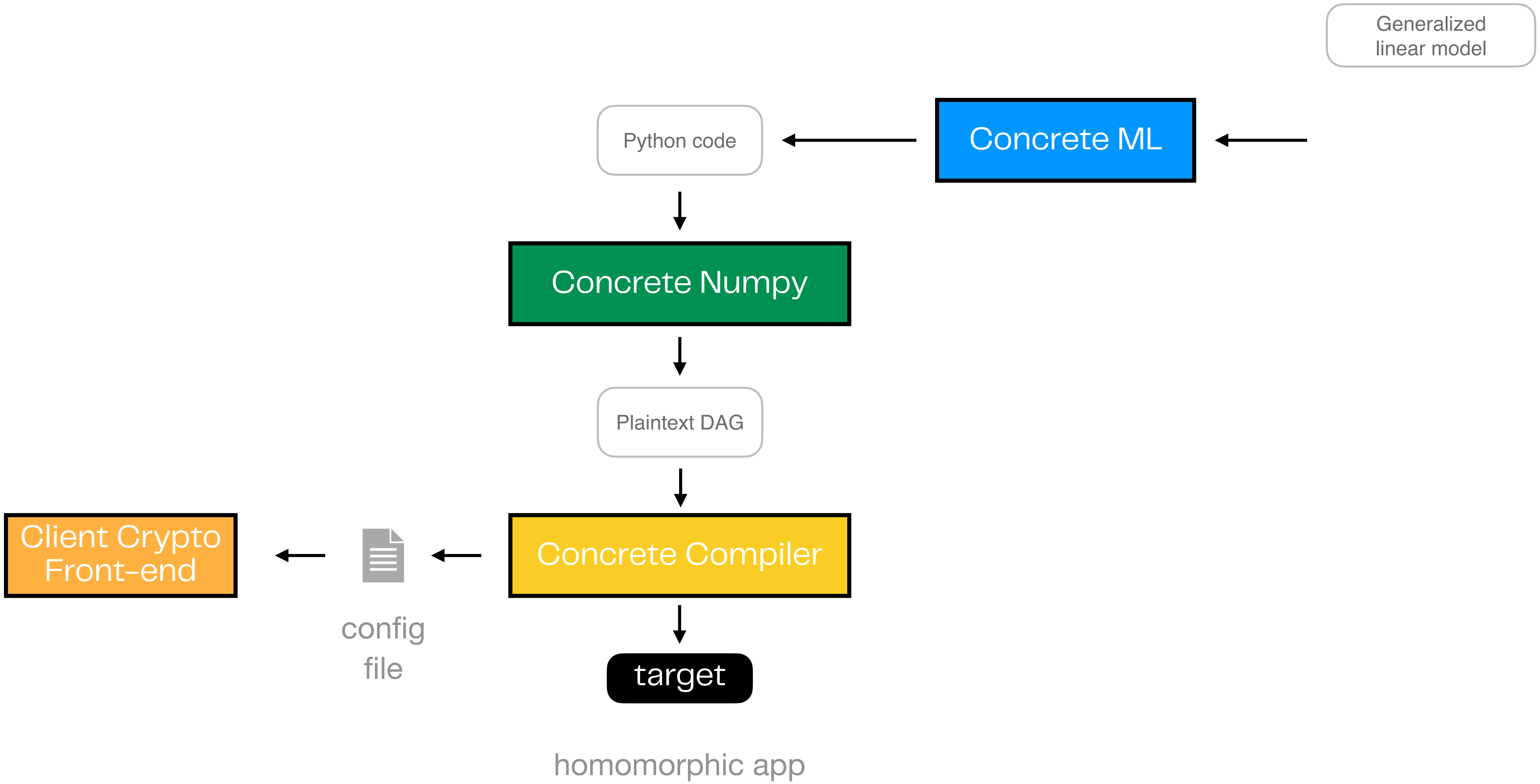
The Concrete stack is a versatile framework



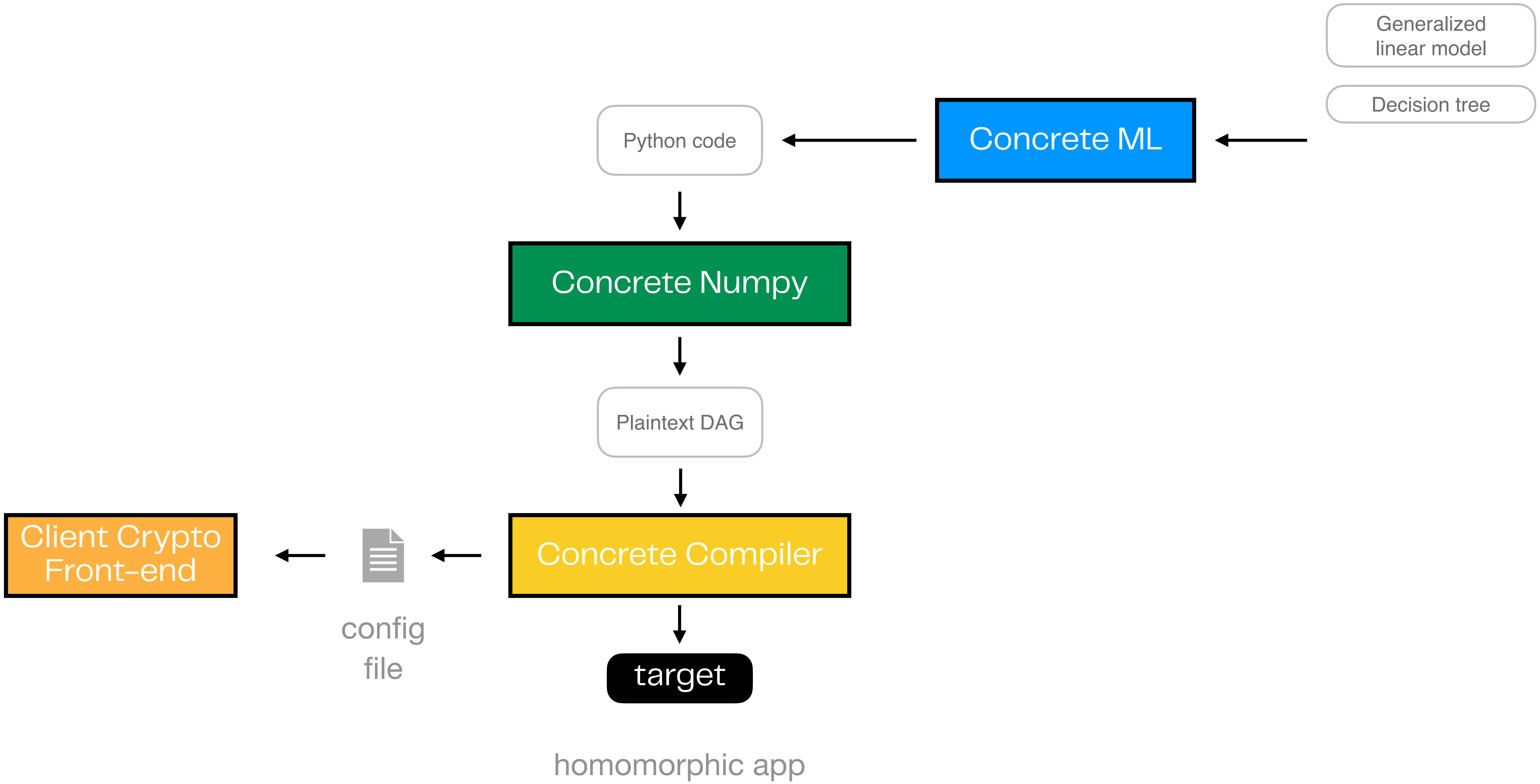
The Concrete stack is a versatile framework



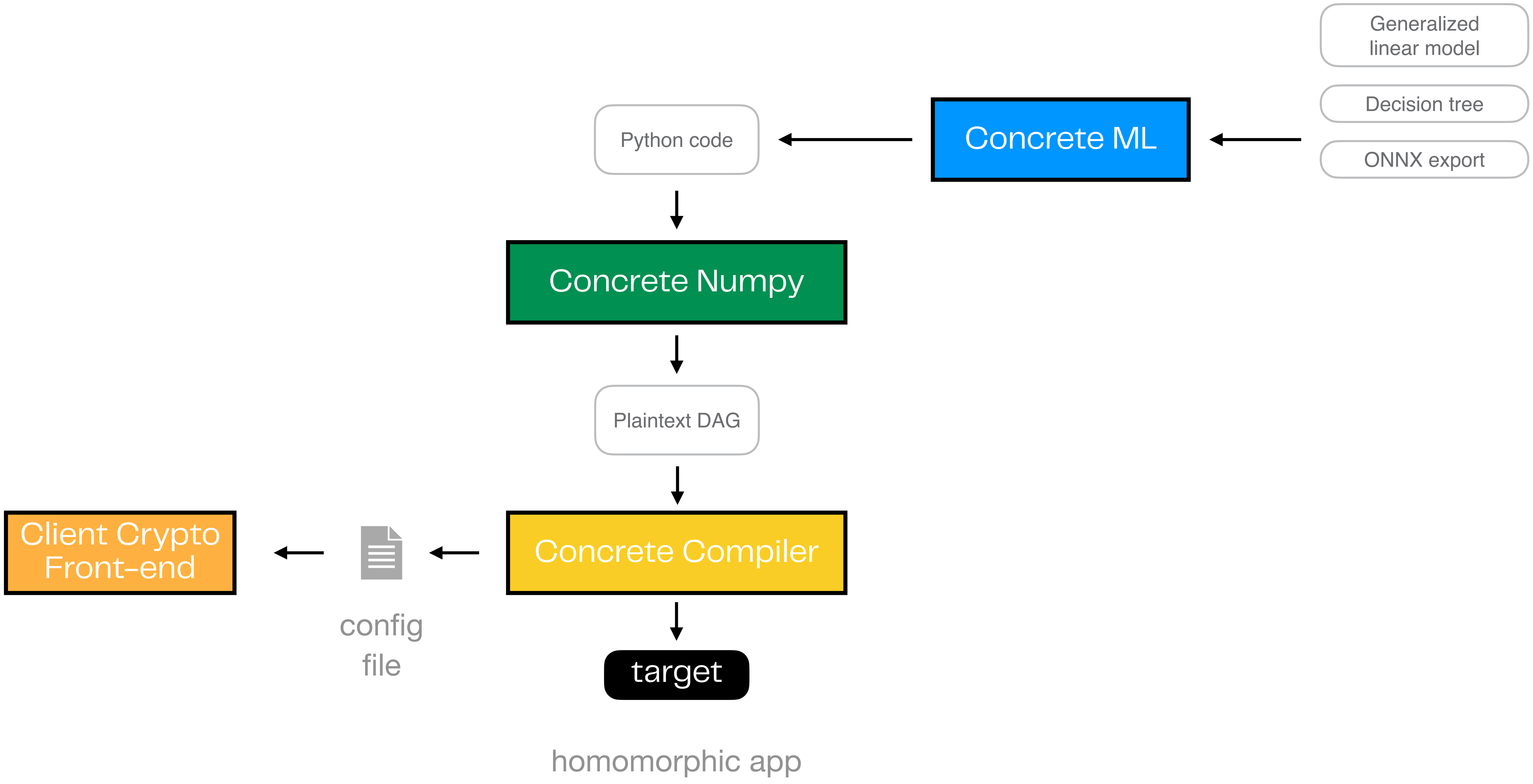
The Concrete stack is a versatile framework



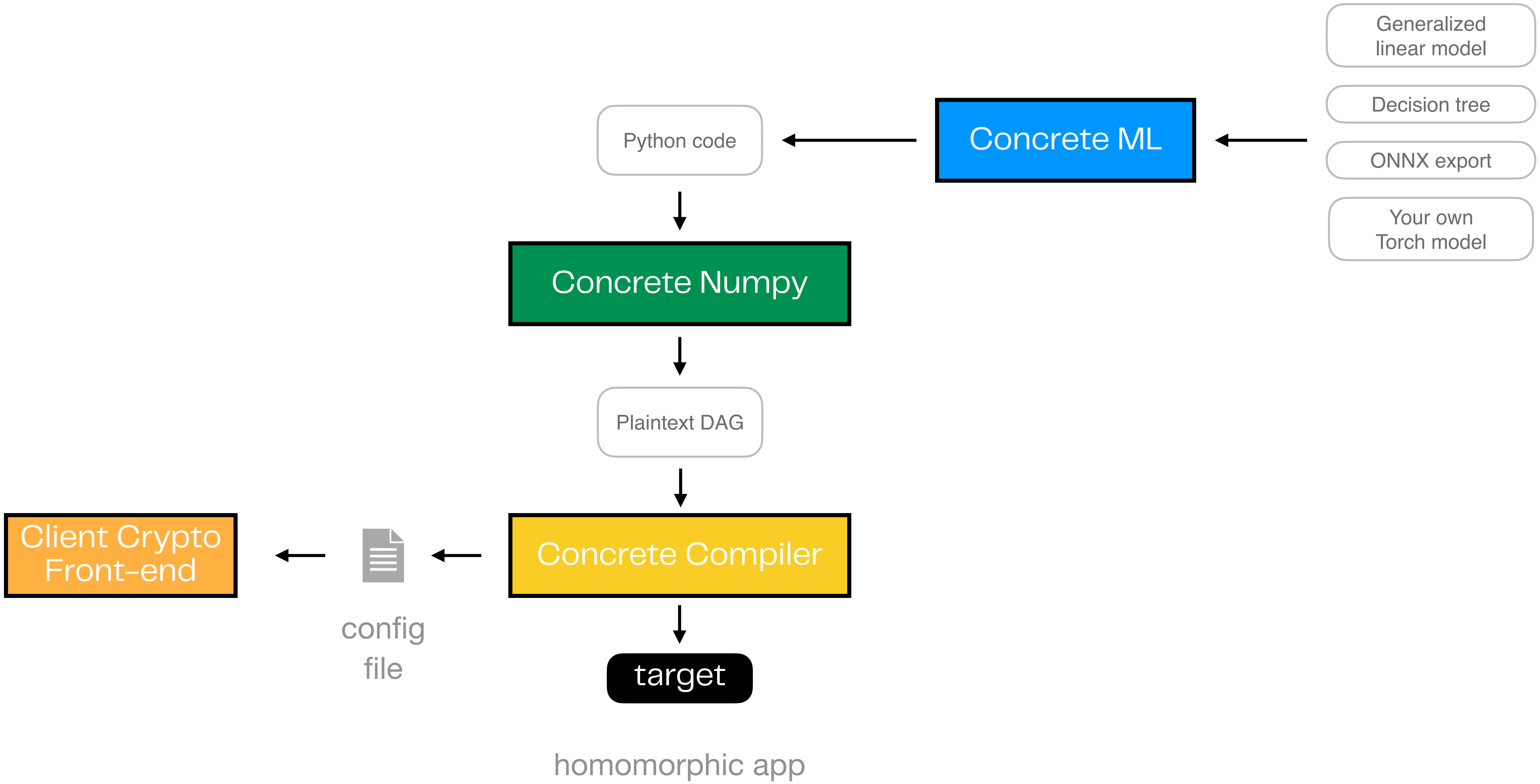
The Concrete stack is a versatile framework



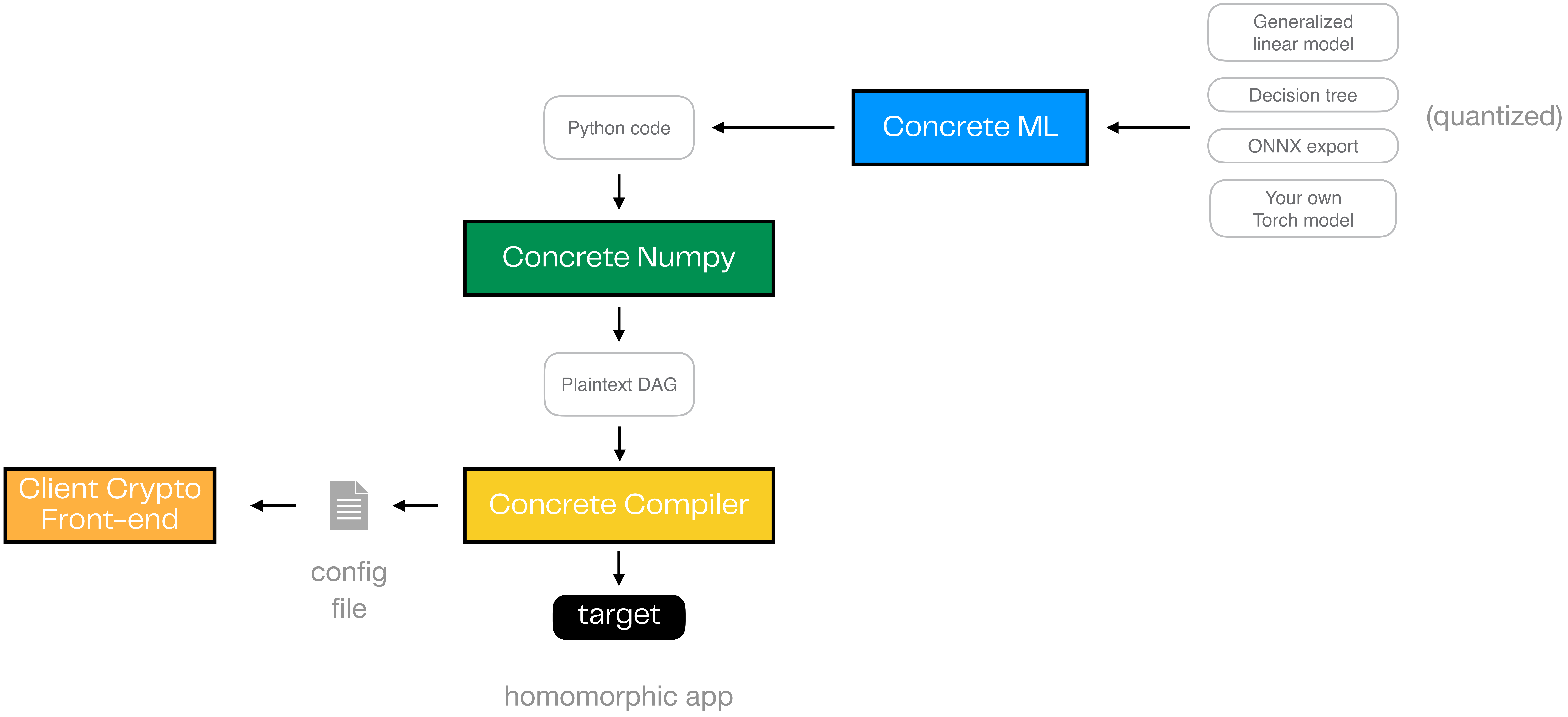
The Concrete stack is a versatile framework



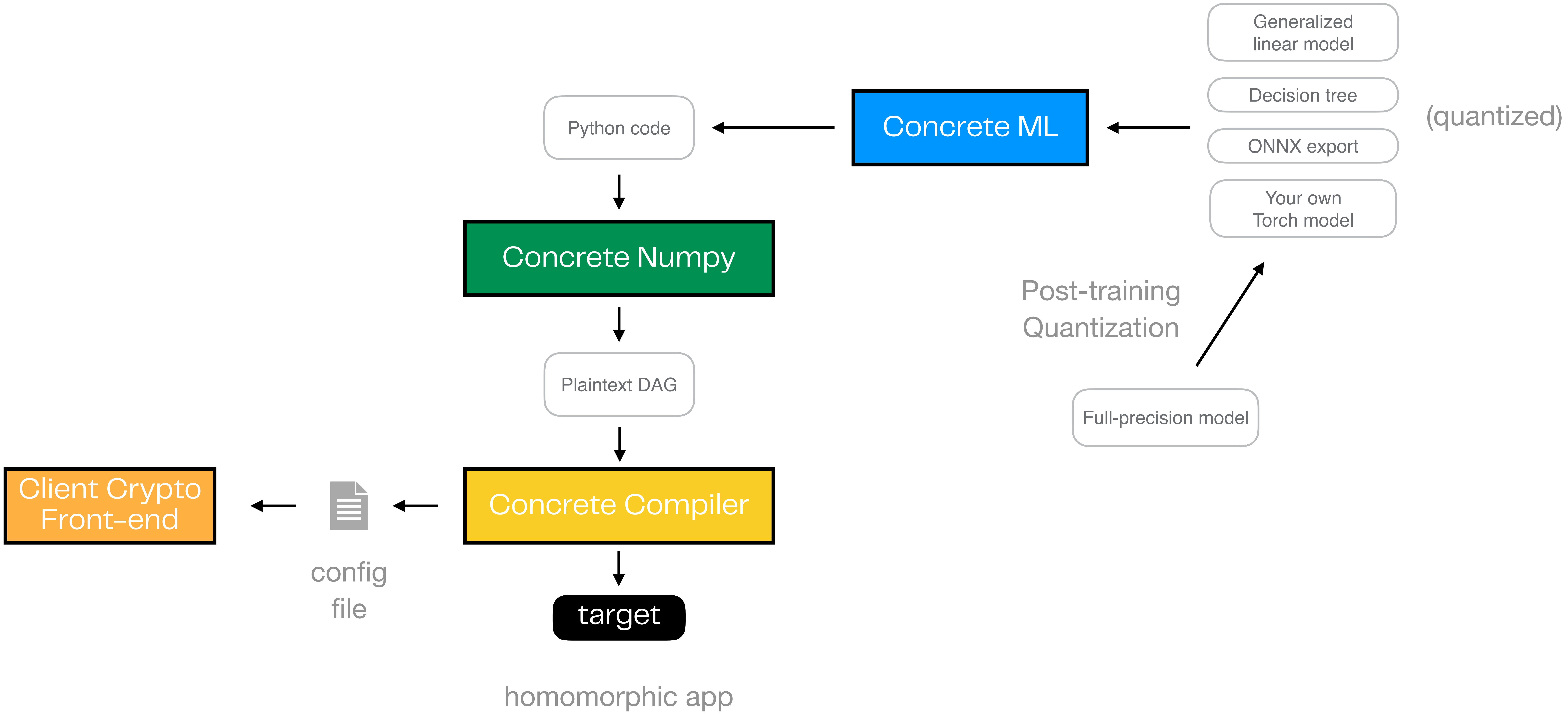
The Concrete stack is a versatile framework



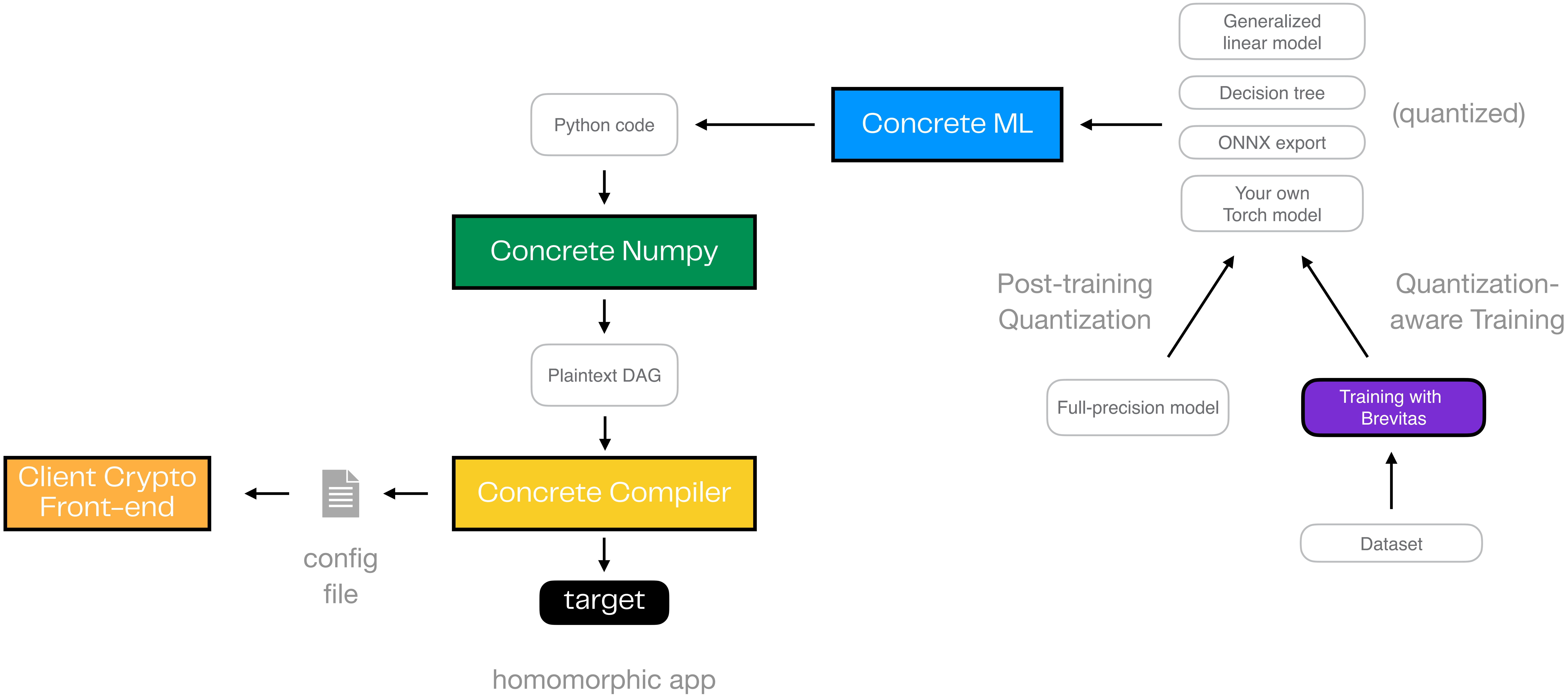
The Concrete stack is a versatile framework



The Concrete stack is a versatile framework



The Concrete stack is a versatile framework



Check it out!

Homomorphic Everything

Clone from **https://github.com/zama-ai** and get support on https://discord.fhe.org/ (#concrete channel)