先进编译实验室
Advanced Compiler

编译论坛

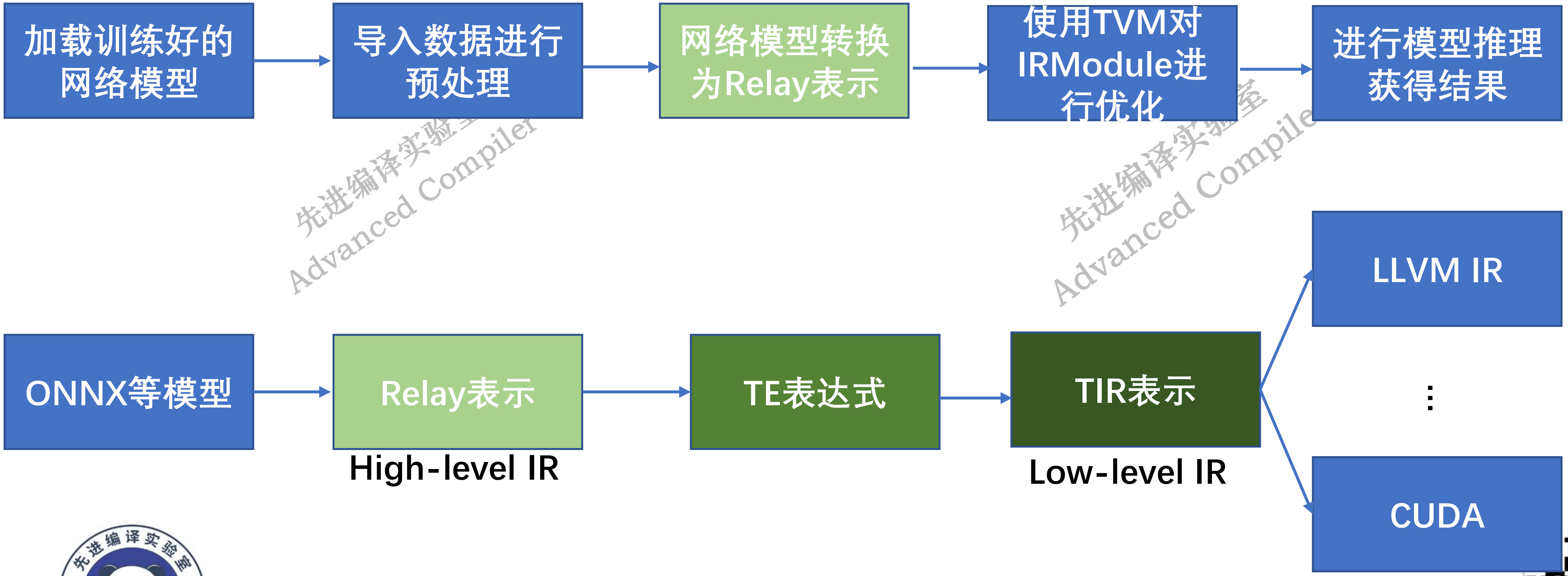
TVM编译流程与中间表示分析 I

Relay中间表示转换与分析

嘉宾：桂中华

先进编译实验室
Advanced Compiler

TVM编译流程

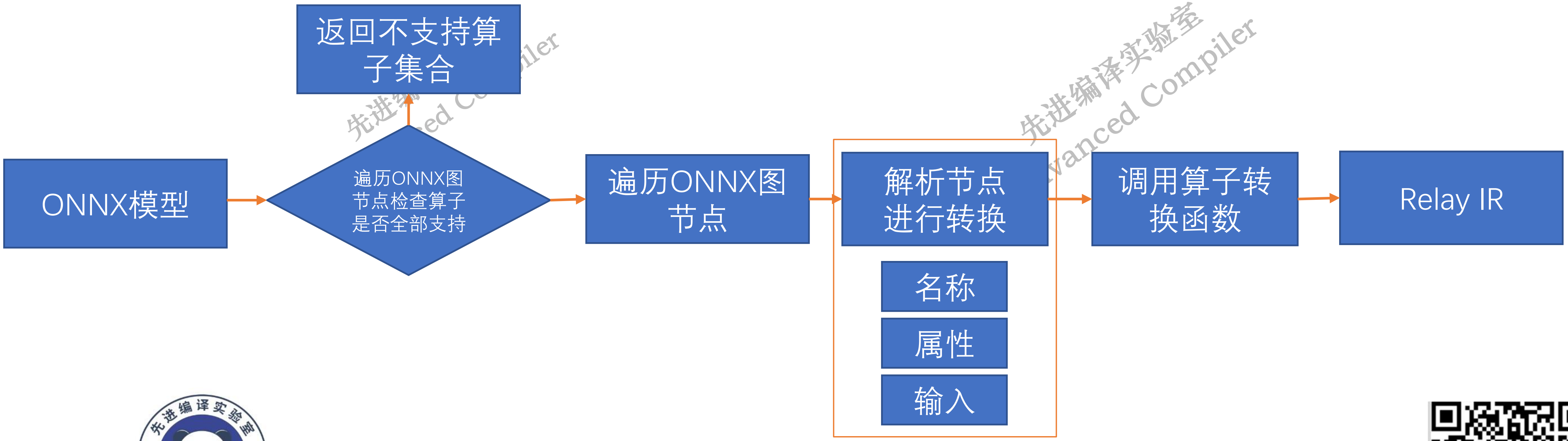

加载训练好的网络模型
导入数据进行预处理
网络模型转换为Relay表示
使用TVM对IRModule进行优化
进行模型推理获得结果
ONNX等模型
Relay表示
High-level IR
TE表达式
TIR表示
Low-level IR
LLVM IR
⋮
CUDA

TVM模型接入

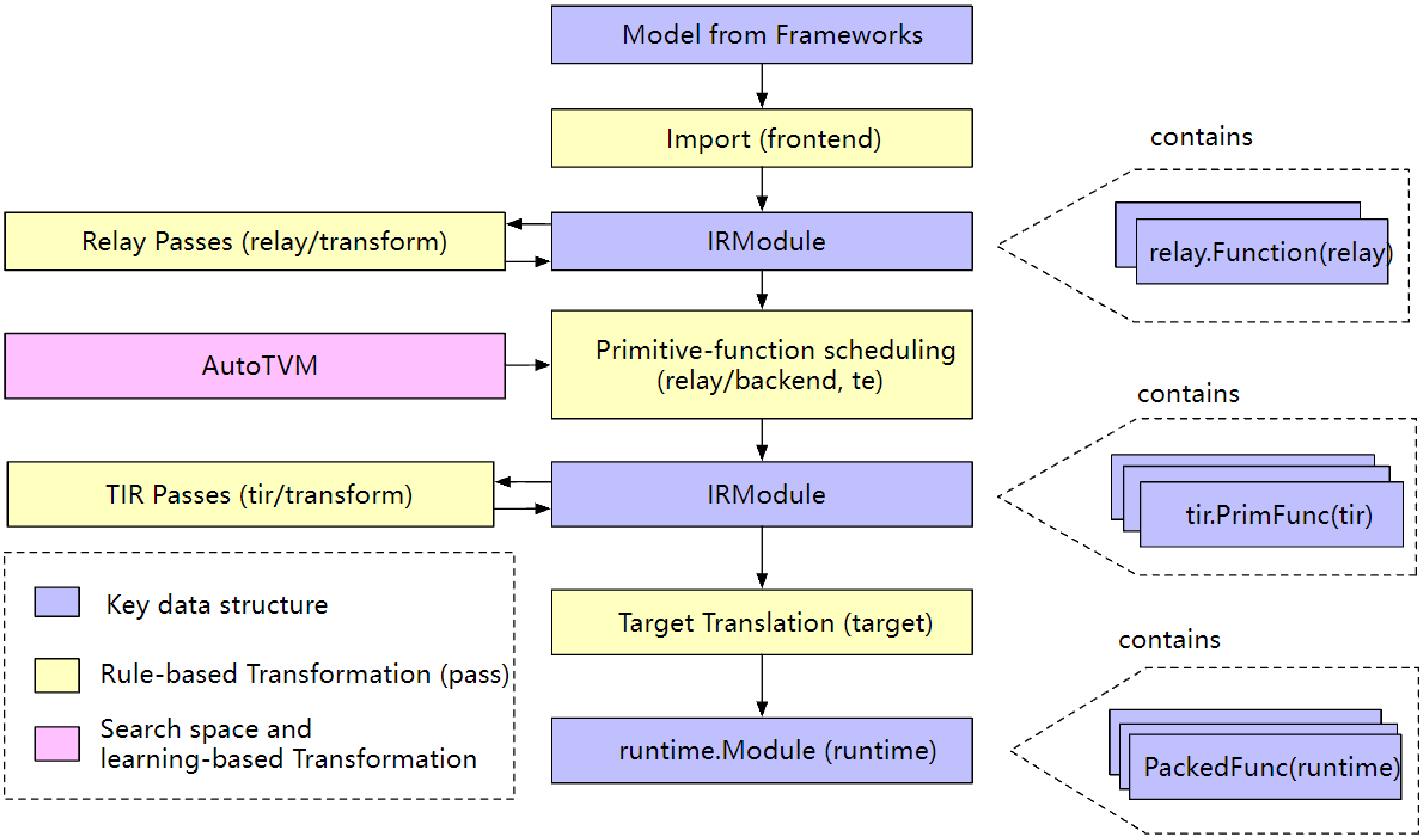
先进编译实验室

```
mod, params = relay.frontend.from_onnx(onnx_model, shape_dict)
```

不支持算子需要开发转换函数



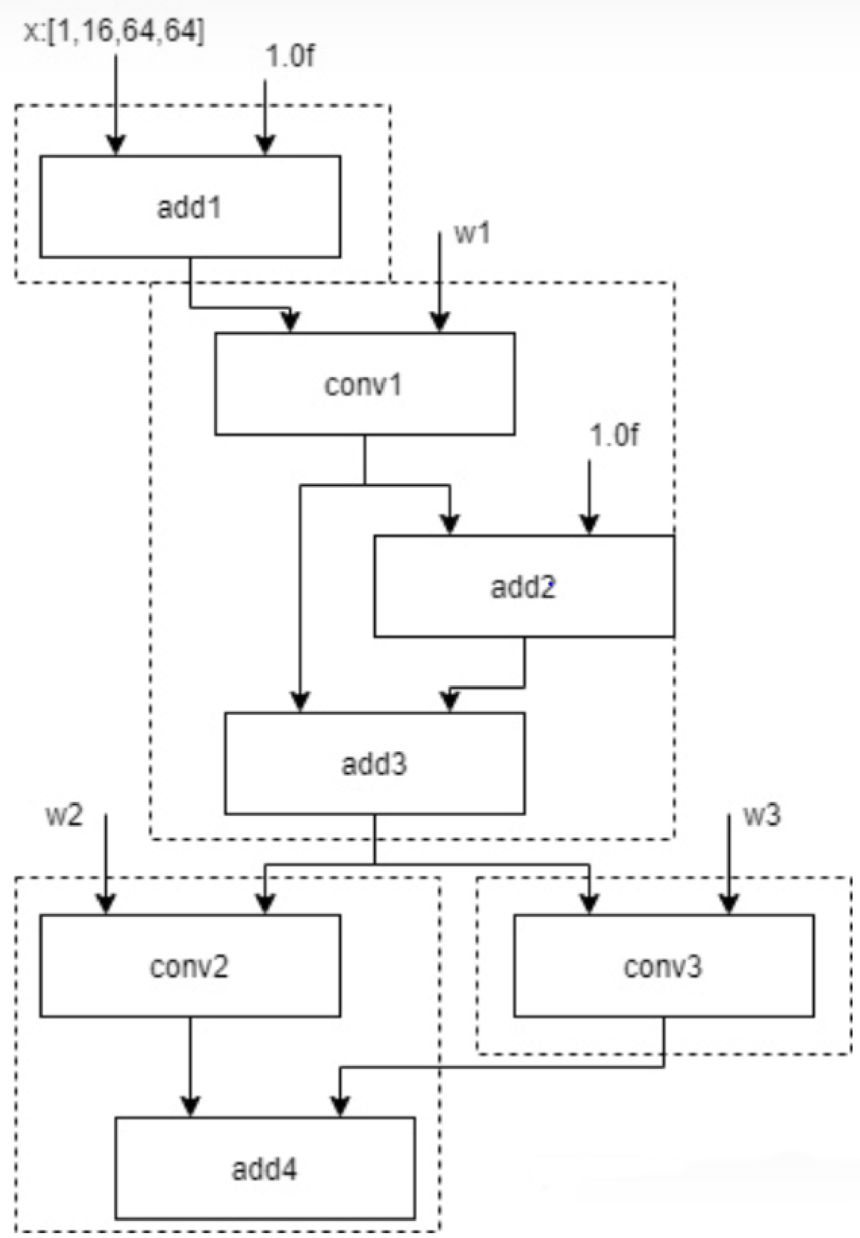
编译流程


Model from Frameworks
Import (frontend)
Relay Passes (relay/transform)
IRModule
contains
relay.Function(relay)
AutoTVM
Primitive-function scheduling (relay/backend, te)
TIR Passes (tir/transform)
IRModule
contains
tir.PrimFunc(tir)
Target Translation (target)
runtime.Module (runtime)
contains
PackedFunc(runtime)
Key data structure
Rule-based Transformation (pass)
Search space and learning-based Transformation

打印出得到的IRModule（Relay IR）

先进编译实验室

```
fn (%x: Tensor[(1, 16, 64, 64), float32], %w1, %w2, %w3)
{
  %0 = add(%x, 1f);
  %1 = nn.conv2d(%0, %w1, padding=[0, 0, 0, 0],
  channels=16, kernel_size=[1, 1]);
  %2 = add(1f, %1);
  %3 = add(%1, %2);
  %4 = nn.conv2d(%3, %w2, padding=[0, 0, 0, 0],
  channels=16, kernel_size=[1, 1]);
  %5 = nn.conv2d(%3, %w3, padding=[0, 0, 0, 0],
  channels=16, kernel_size=[1, 1]);
  add(%4, %5)
}
```



Relay IR

- 语言特点：

函数式IR:可微分、函数作为一等公民、支持高阶函数、闭包、代数数据类型ADT、支持递归等控制流、支持lambda表达式、支持let-binding（类似Ocaml和Haskell语言）

- 类型系统（静态、可推断）：

int、float等DType、FuncType 、TensorType、ToupleType…支持类型推导

- 表达范围（Expr）：

1.数据流

变量Variable 、常量、函数、算子、ADT构造、调用Call、元组、let-binding…

2.控制流

if-else-then、ADT表达式匹配、函数递归调用…

- 下一代Relay：Relax

同时表示和优化高层级和低层级的 IR，支持动态 shape 模型的表达和优化（partial lowering思想）

tvm.ir基础设施

Relay与TIR共用一套TVM IR基础设施，最为核心的是Type和Expr（Expression）两个概念

Type： 包含基础数据类型int、float等DType，复杂数据类型Tensor类型、Touple元组类型等

1.PrimType 继承自Type，描述为运行时基础数据类型uint8,int,float等

2.FuncType 继承自Type，类似C++模板函数，记录函数参数类型，返回值类型等

3.TensorType继承自BaseTensorType，具有shape成员，是TVM不支持动态shape的因素之一

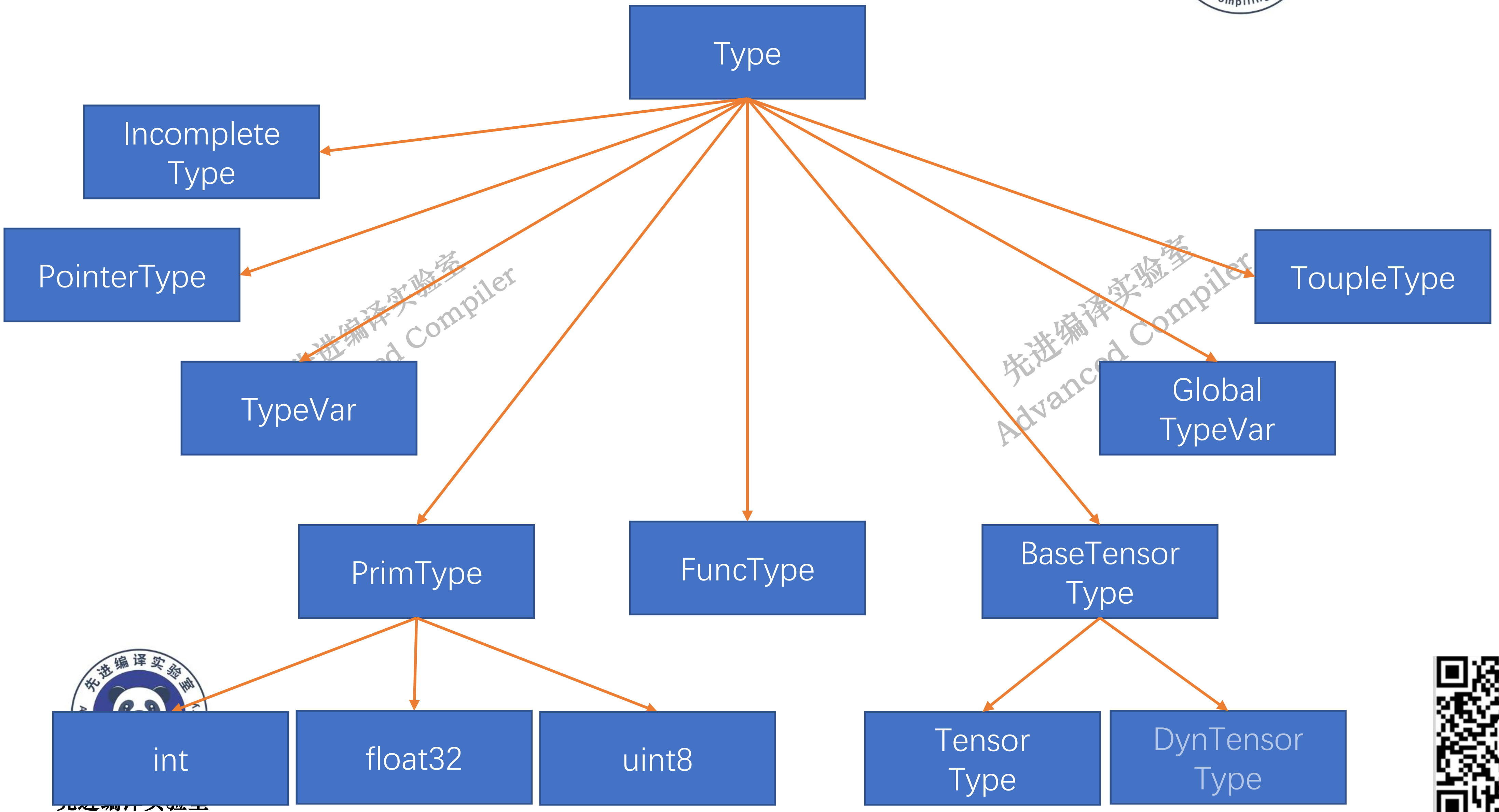
Expr： 包含IR 中的各种表达式元素，分为PrimExpr和RelayExpr

1.PrimExpr 处理TIR中基础类型数据，用于low-level的代码优化和分析

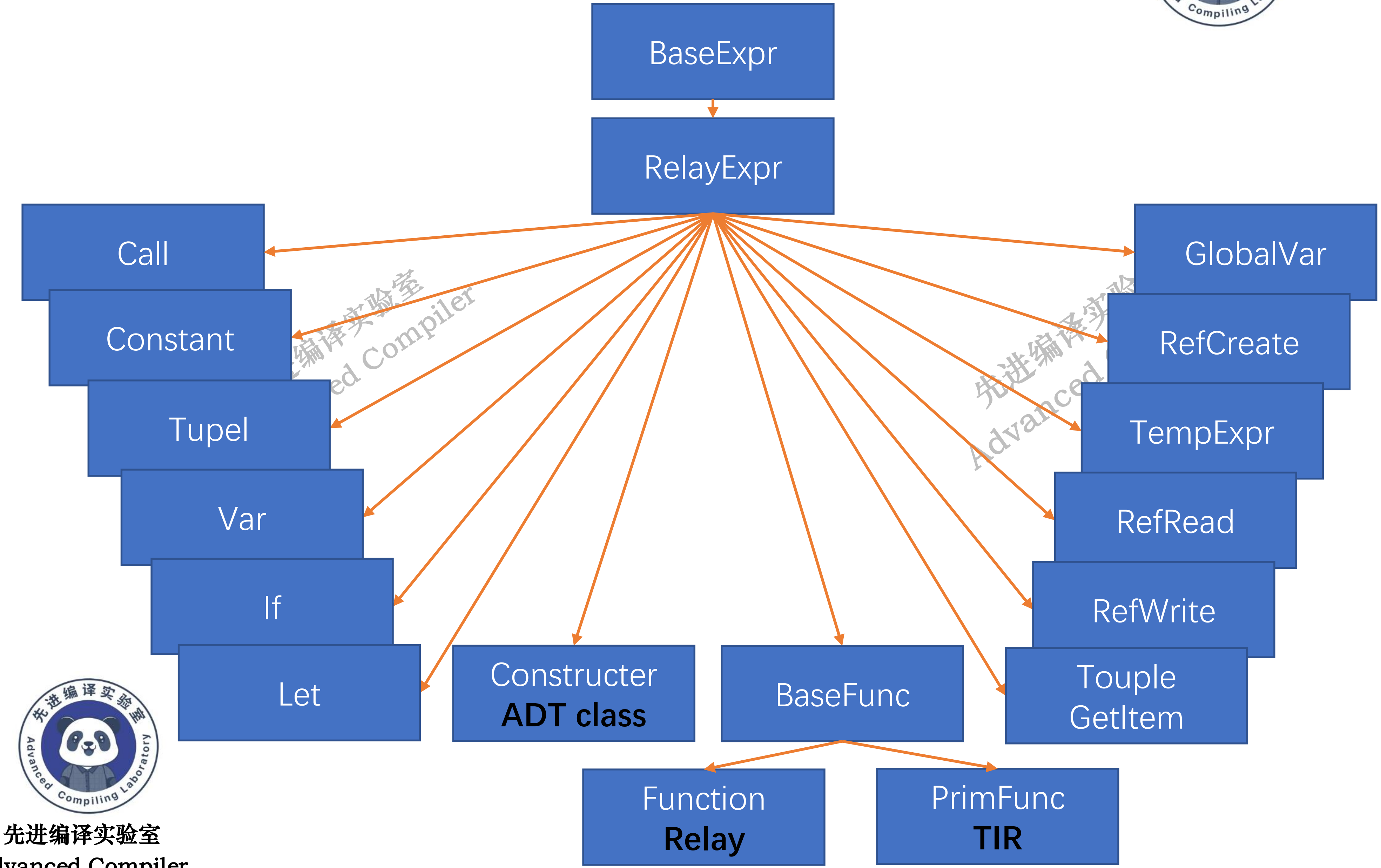
实现为PrimExpr继承自BaseExpr（Expr基类）

2.RelayExpr 所有非PrimExpr类的基类（比如Op类），也继承自BaseExpr（Expr基类）

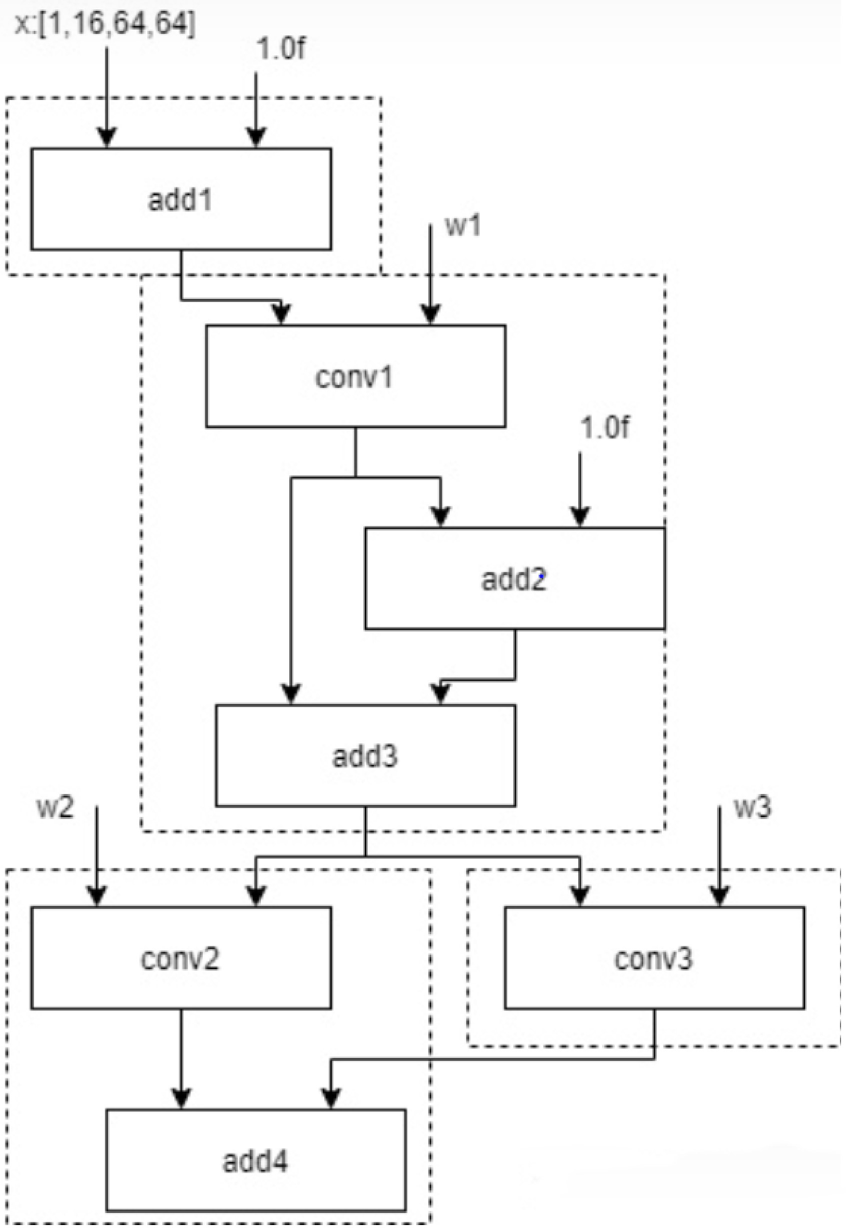
tvm.ir基础设施Type（部分）

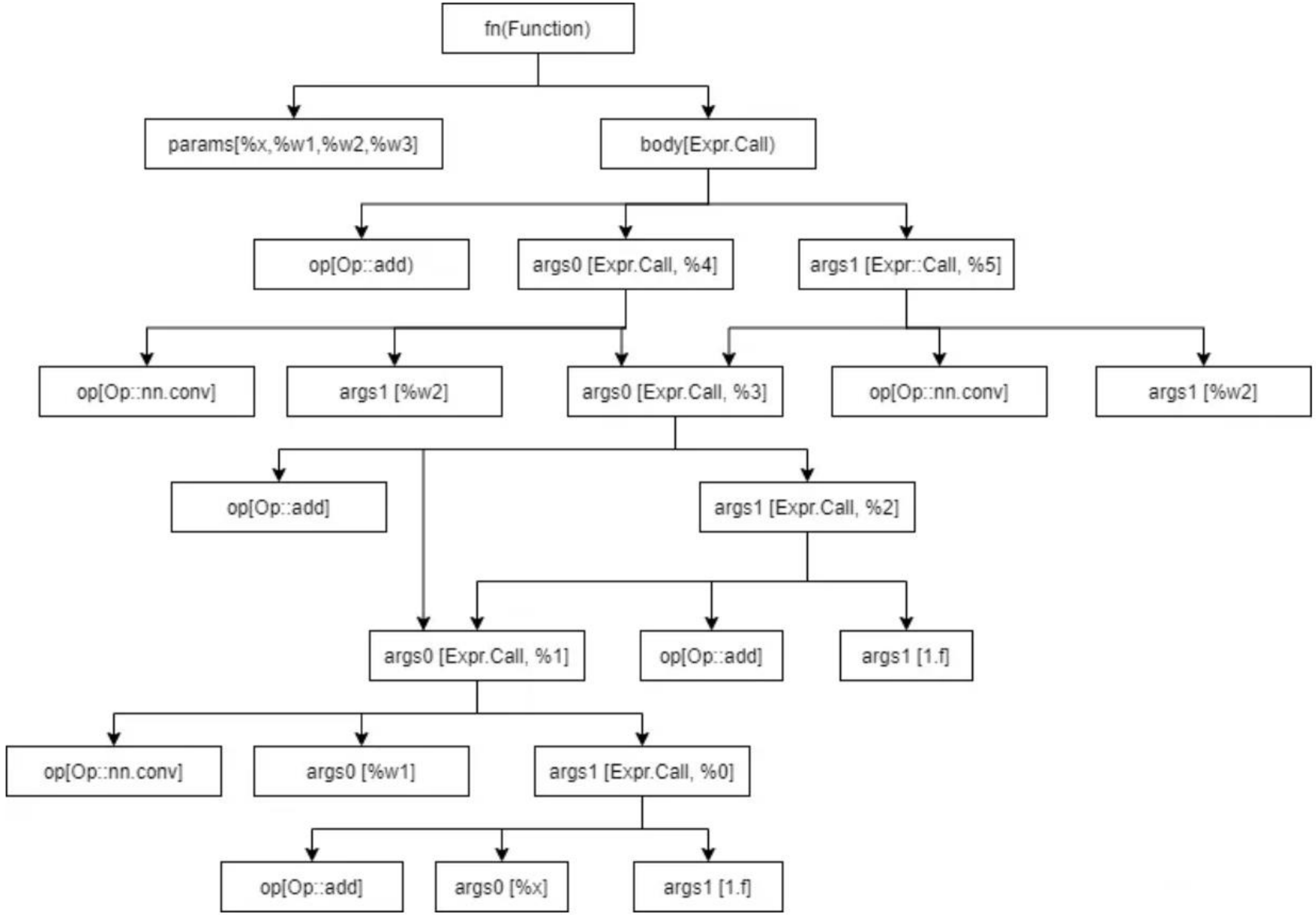


tvm.ir基础设施RelayExpr（部分）



Relay IR对应的数据结构图





参考文献

[1] Jared Roesch, Steven Lyubomirsky, Logan Weber, Josh Pollock, Marisa Kirisame, Tianqi Chen, Zachary Tatlock. Relay: a new IR for machine learning frameworks. MAPL@PLDI 2018: 58-68

[2] https://www.zhihu.com/people/archer-88-72

[3] https://www.zhihu.com/people/sunny-yuan-77

[4] https://www.zhihu.com/people/zhang-xiao-yu-45-67-74

先进编译实验室
Advanced Compiler

编译论坛

TVM编译流程与中间表示分析 II

TE表达式、Tensor IR中间表示转换与分析

嘉宾：桂中华

先进编译实验室
Advanced Compiler

开启tvm优化之旅

前面的整个过程只是获取了TVM优化的对象，还没正式进入优化流程。

通过relay.build函数传入获取的Relay IRModule，目标硬件设备target进行编译（tvm.build）

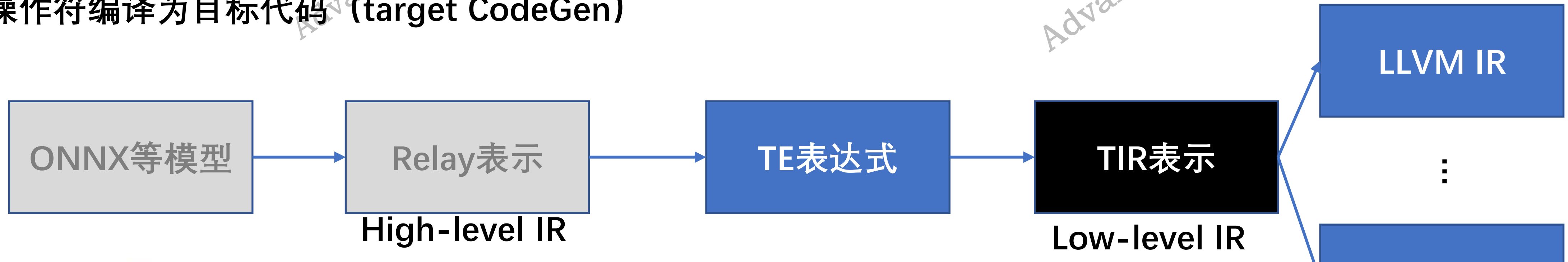
主要分为以下流程：

1.Relay层计算图优化（常量折叠、公共子表达式消除、算子layout变换、算子规范化、算子融合…）

2.通过查询运算符注册表来查找算子实现（TOPI），为算子生成计算表达式和调度(Compute+Schedule)

3. lower为TIR，硬件相关优化

4.将操作符编译为目标代码（target CodeGen）




```
with tvm.transform.PassContext(opt_level=1):
    lib = relay.build(mod, target=target, params=params)
```

Tensor Expression张量表达式

语言特点：

支持张量，纯函数式语言，表达式不具有副作用，支持lambda表达式，主要用于TVM中算子的定义和优化

计算描述compute：

```
k = te.reduce_axis((0, K), "k")
A = te.placeholder((M, K), name="A")
B = te.placeholder((K, N), name="B")
C = te.compute((M, N), lambda x, y: te.sum(A[x, k] * B[k, y], axis=k), name="C")
```

调度模板schedule：

```
s = te.create_schedule(C.op)
xo, yo, xi, yi = s[C].tile(C.op.axis[0], C.op.axis[1], 32, 32)
(k,) = s[C].op.reduce_axis
ko, ki = s[C].split(k, factor=4)
s[C].reorder(xo, yo, ko, ki, xi, yi)
```

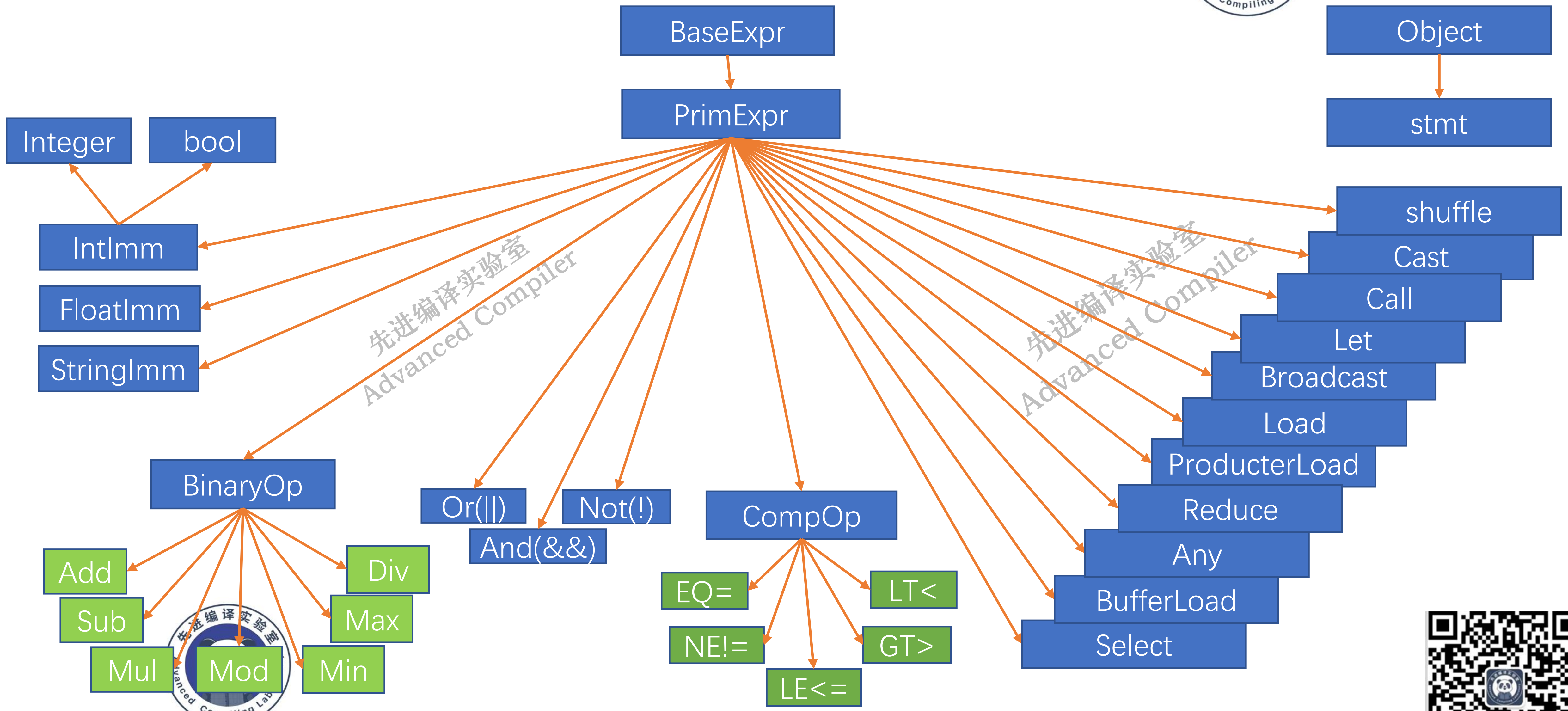
打印TIR：

```
print(tvm.lower(s, [A, B, C], simple_mode=True))
```

Tensor IR描述

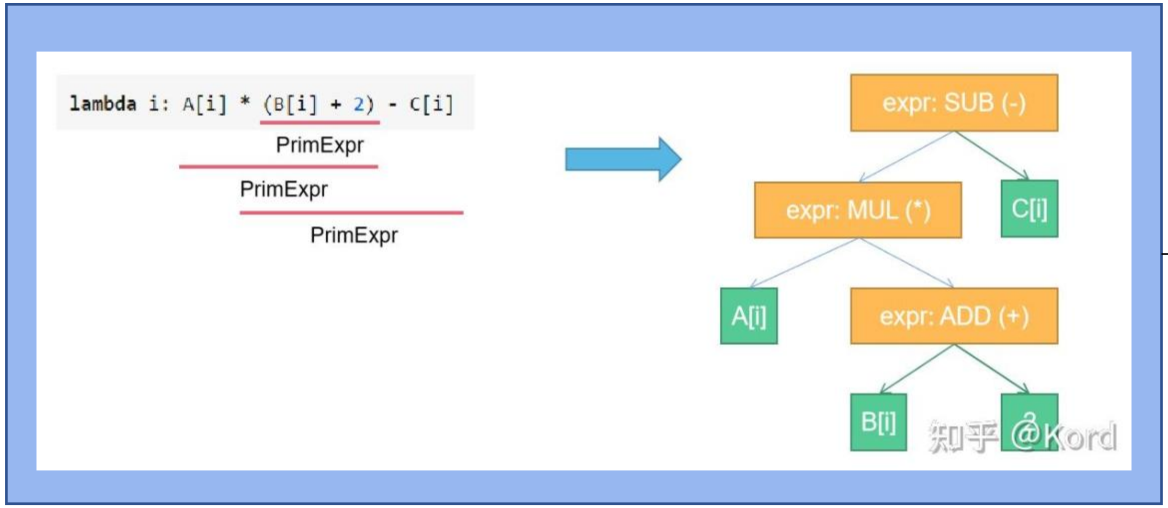

```
@main = primfn(A_1: handle, B_1: handle, C_1: handle) -> ()
 attr = {"from_legacy_te_schedule": True, "global_symbol": "main", "tir.noalias": True}
 buffers = {
         A: Buffer(A_2: Pointer(float32), float32, [1024, 1024], []),
         B: Buffer(B_2: Pointer(float32), float32, [1024, 1024], []),
         C: Buffer(C_2: Pointer(float32), float32, [1024, 1024], [])}
 buffer_map = {A_1: A, B_1: B, C_1: C} {
 for (x.outer: int32, 0, 32) {
  for (y.outer: int32, 0, 32) {
   for (x.inner.init: int32, 0, 32) {
    for (y.inner.init: int32, 0, 32) {
     C_3: Buffer(C_2, float32, [1048576], [])[((((x.outer*32768) + (x.inner.init*1024)) + (y.outer*32)) + y.inner.init)] = 0f32
    }
   }
   for (k.outer: int32, 0, 256) {
    for (k.inner: int32, 0, 4) {
     for (x.inner: int32, 0, 32) {
      for (y.inner: int32, 0, 32) {
       let cse_var_3: int32 = (y.outer*32)
       let cse_var_2: int32 = ((x.outer*32768) + (x.inner*1024))
       let cse_var_1: int32 = ((cse_var_2 + cse_var_3) + y.inner)
       C_3[cse_var_1] = (C_3[cse_var_1] + (A_3: Buffer(A_2, float32, [1048576], [])[((cse_var_2 + (k.outer*4)) + k.inner)]*B_3:
Buffer(B_2, float32, [1048576], [])[((((k.outer*4096) + (k.inner*1024)) + cse_var_3) + y.inner)]))
     }}}}}}
```

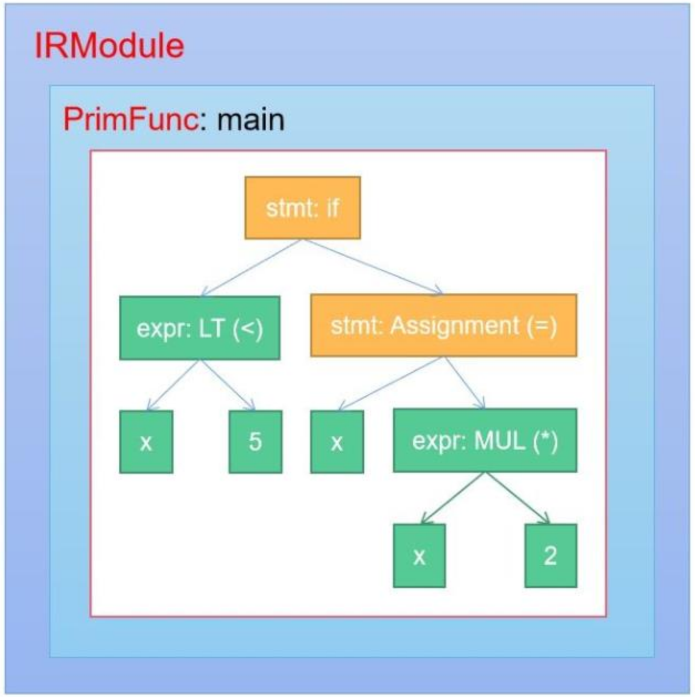
tvm.ir基础设施PrimExpr(部分)

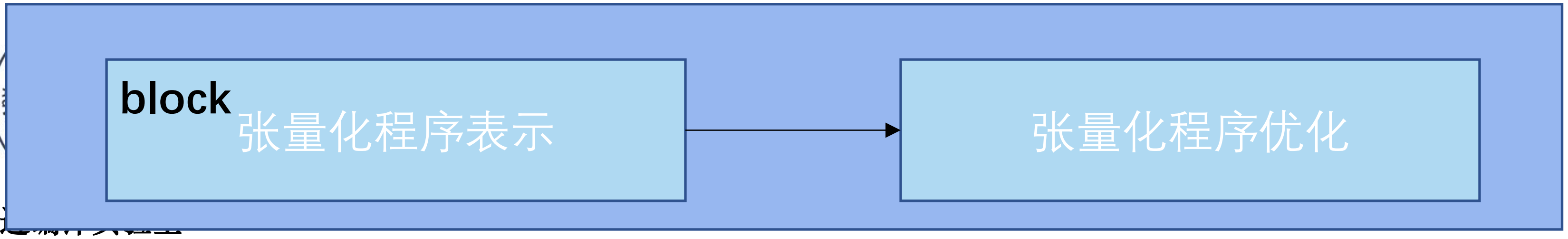


Tensor IR

表示常量、变量声明和赋值、内存分配、基本运算、函数的调用、控制分支、循环等概念







张量化程序表示和优化

```
for yo, xo, ko in grid(16, 16, 16):            outer loop
  with block():
    vy, vx, vk = ...                            signatures
    block_siguatures
    with init():
      for y, x in grid(4, 4):                   init stmt
        C[vy*4+y, vx*4+x] = 0.0
    for y, x, k in grid(4, 4, 4):
      C[vy*4+y, vx*4+x] +=                      body
        A[vy*4+y, vk*4+k] * B[vk*4+k, vx*4+x]
```

Block Signature

```
Iterator domain and binding values
vy: spatial_axis(length=16, binding_value=yo)
vx: spatial_axis(length=16, binding_value=xo)
vk: reduce_axis (length=16, binding_value=ko)

Producer-consumer dependency relations
read  A[vy*4:vy*4 + 4, vk*4:vk*4 + 4]
read  B[vk*4:vk*4 + 4, vx*4:vx*4 + 4]
write C[vy*4:vy*4 + 4, vx*4:vx*4 + 4]
```

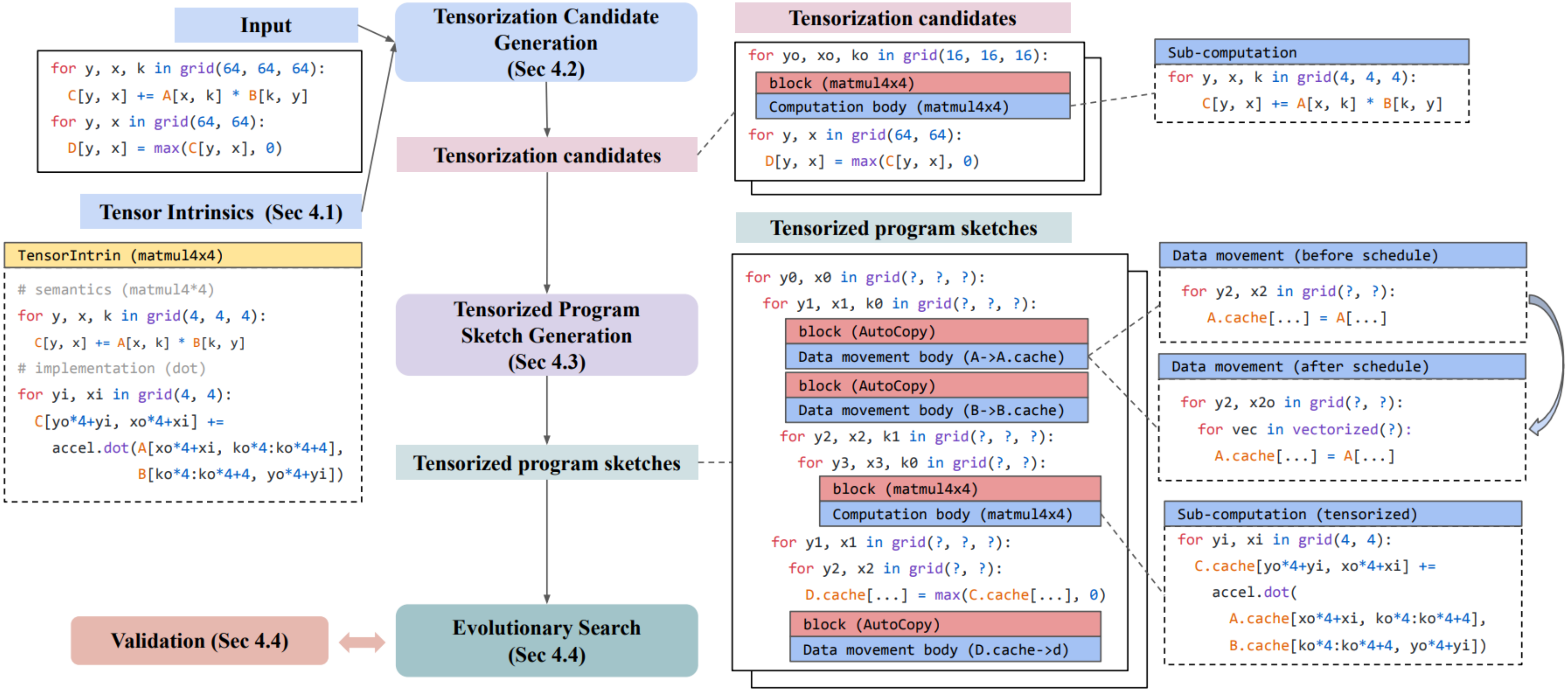
```
for i, j in grid(64, 64):   <- Loop Tiling
  block_C (vi, vj = i, j)
  C[vi, vj] = dot(A[vi, :], B[:, vj])
for i, j in grid(64, 64):
  block_D (vi, vj = i, j)
  D[vi, vj] = max(C[vi, vj], 0)
```

```
for i0, j0 in grid(8, 8):          Reverse Compute_at
  for i1, j1 in grid(8, 8):
    block_C(vi, vj = i0*8 + i1, j0*8 + j1)
    C[vi, vj] = dot(A[vi, :], B[:, vj])
for i, j in grid(64, 64):
  block_D (vi, vj = i, j)
  D[vi, vj] = max(C[vi, vj], 0)
```

```
for i0, j0 in grid(8, 8):
  for i1, j1 in grid(8, 8):
    block_C(vi, vj = i0*8 + i1, j0*8 + j1)
    C[vi, vj] = dot(A[vi, :], B[:, vj])
  for i1, j1 in grid(8, 8):
    block_D(vi, vj = i0*8 + i1, j0*8 + j1)
    D[vi, vj] = max(C[vi, vj], 0)
```

张量化程序表示和优化

先进编译实验室



先进编译实验室

Advanced Compiler

参考文献

[1] Jared Roesch, Steven Lyubomirsky, Logan Weber, Josh Pollock, Marisa Kirisame, Tianqi Chen, Zachary Tatlock. Relay: a new IR for machine learning frameworks. MAPL@PLDI 2018: 58-68

[2] Siyuan Feng, Bohan Hou, Hongyi Jin, Wuwei Lin, Junru Shao, Ruihang Lai, Zihao Ye, Lianmin Zheng, Cody Hao Yu, Yong Yu, Tianqi Chen:TensorIR: An Abstraction for Automatic Tensorized Program Optimization. CoRR abs/2207.04296 (2022)

[3] https://tvm.apache.org/docs/tutorial/tensor_expr_get_started.html#example-2

[4] https://www.zhihu.com/people/archer-88-72

[5] https://www.zhihu.com/people/sunny-yuan-77

[6] https://www.zhihu.com/people/zhang-xiao-yu-45-67-74