# Program Analysis in Relay

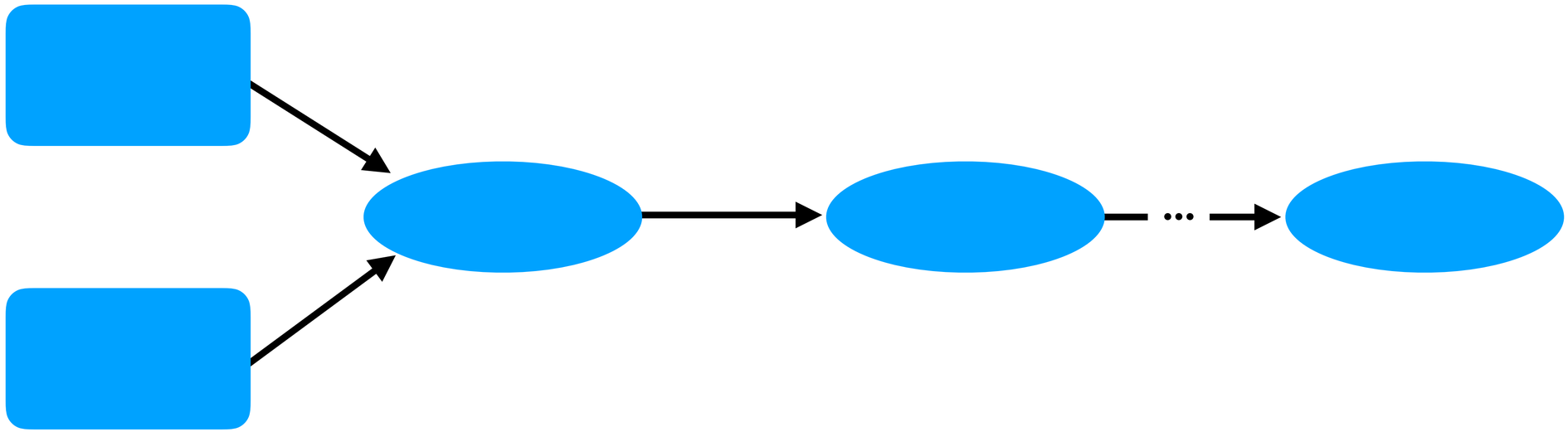
Gus Smith

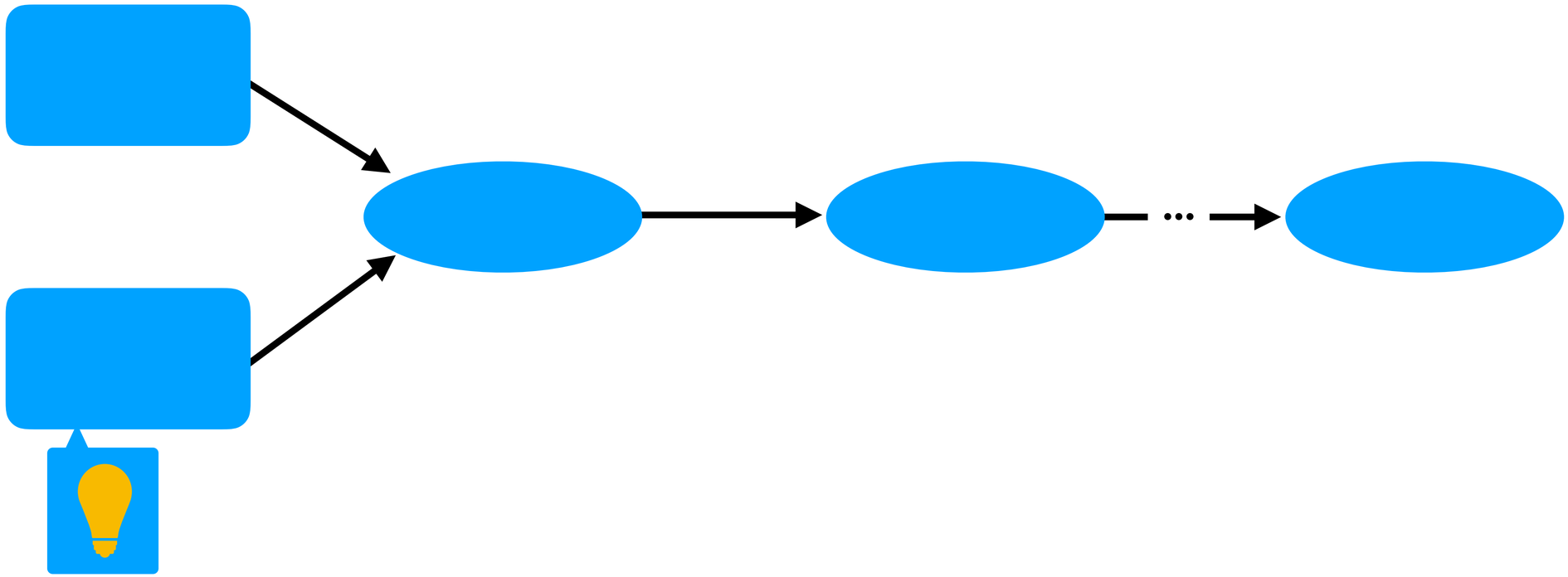
December 5th, 2019

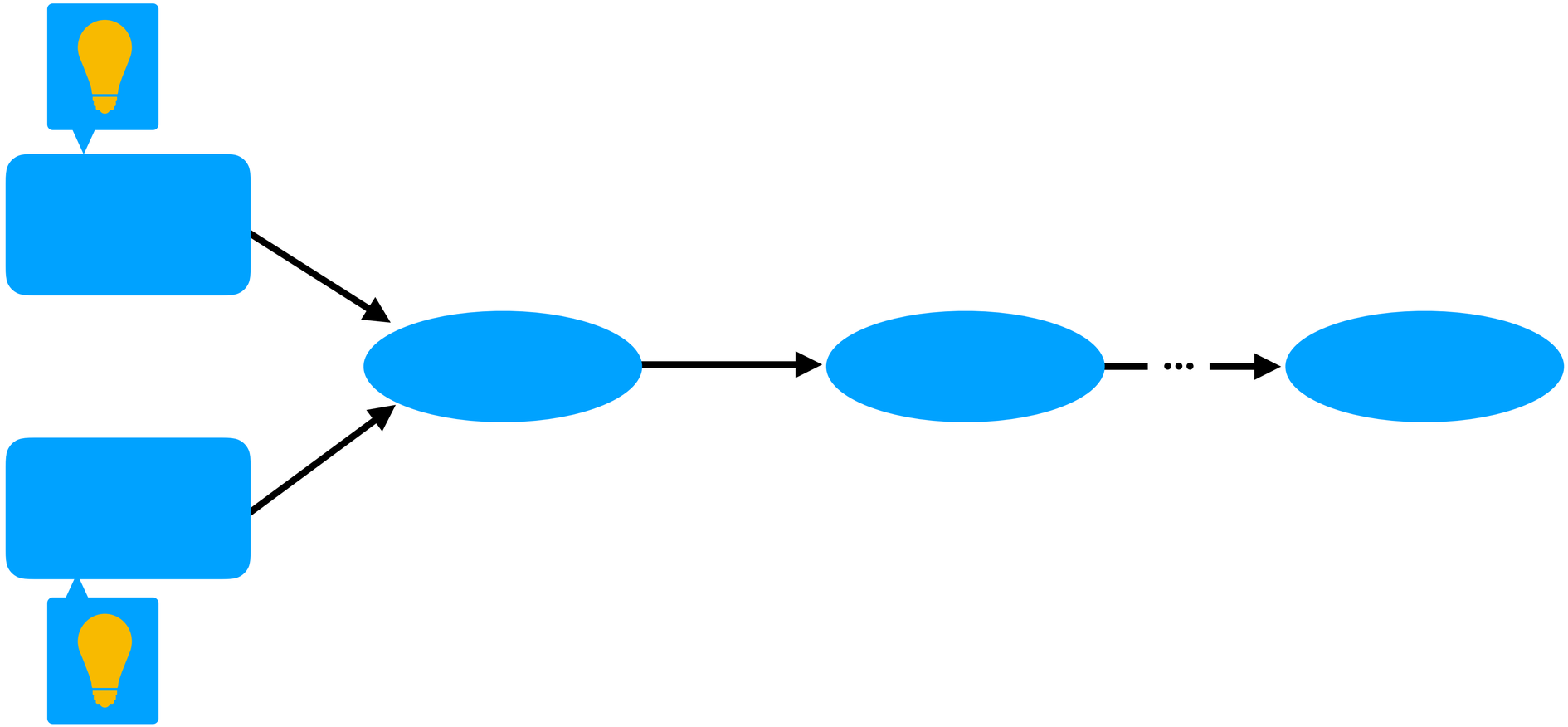
sampl

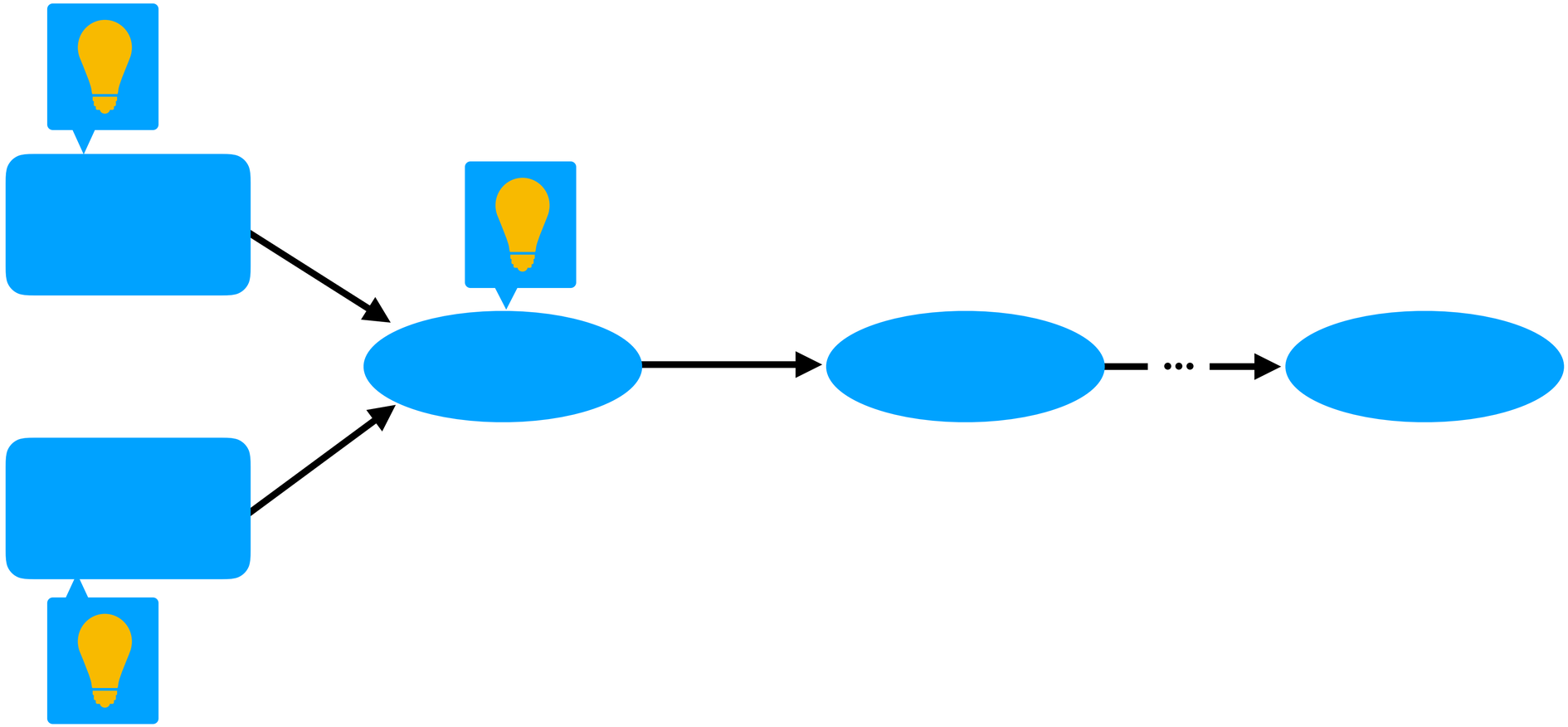
PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

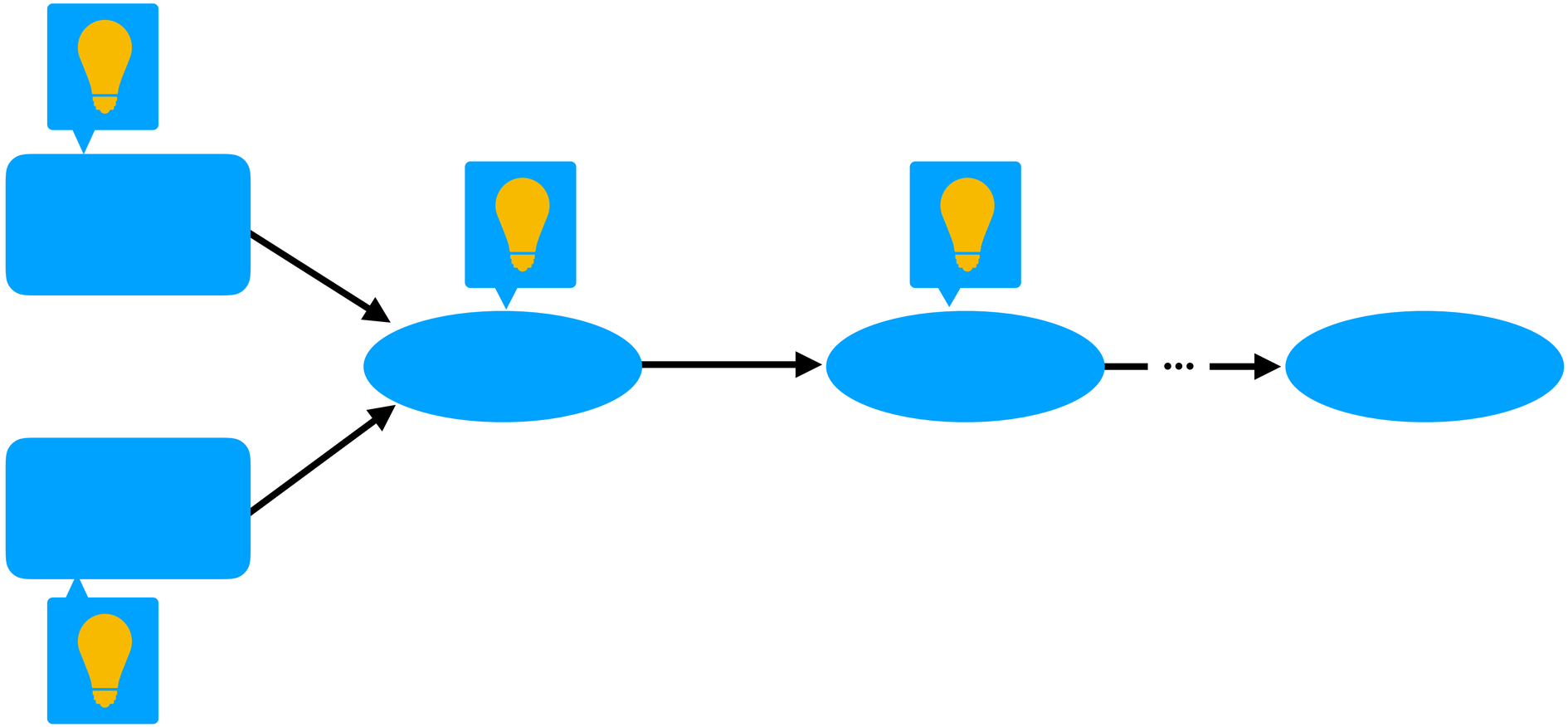
tvm.ai

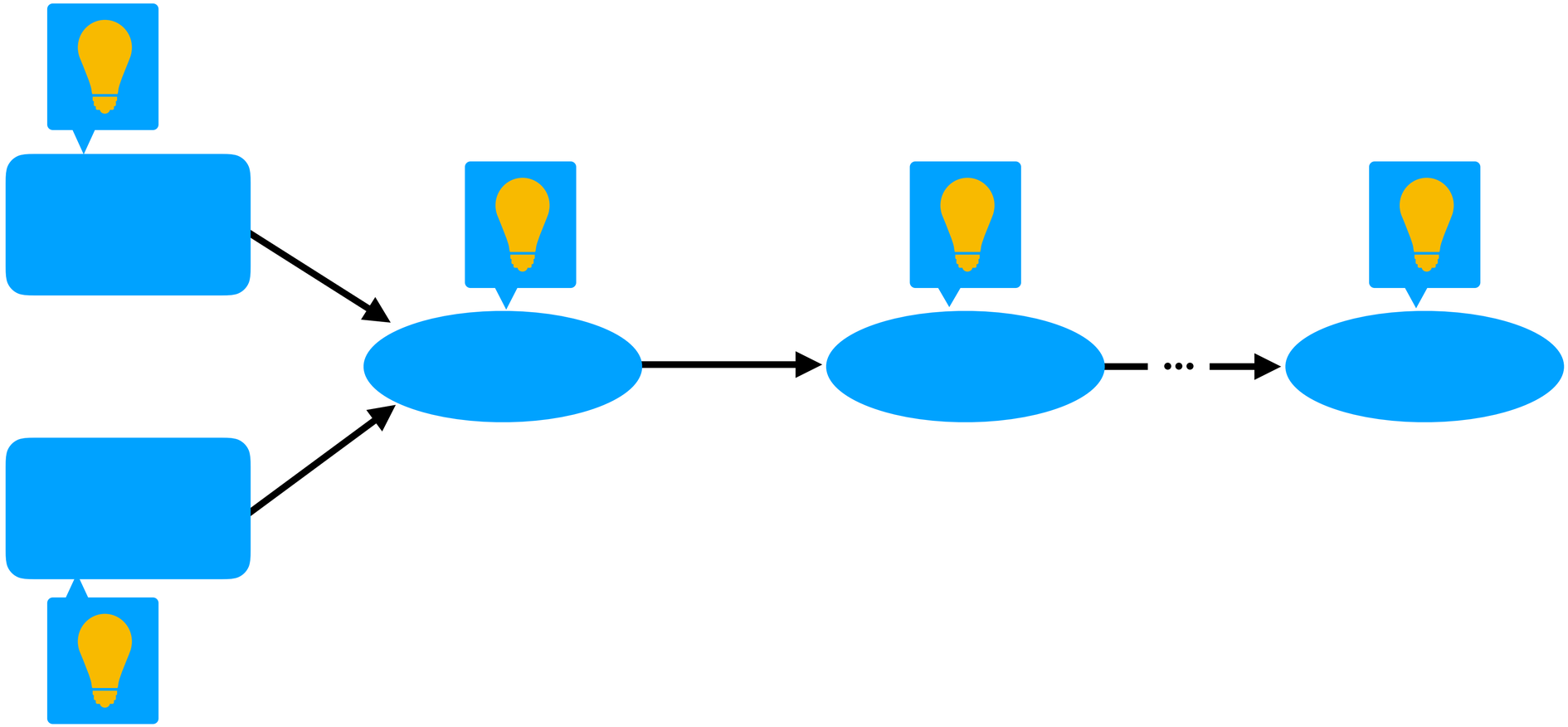


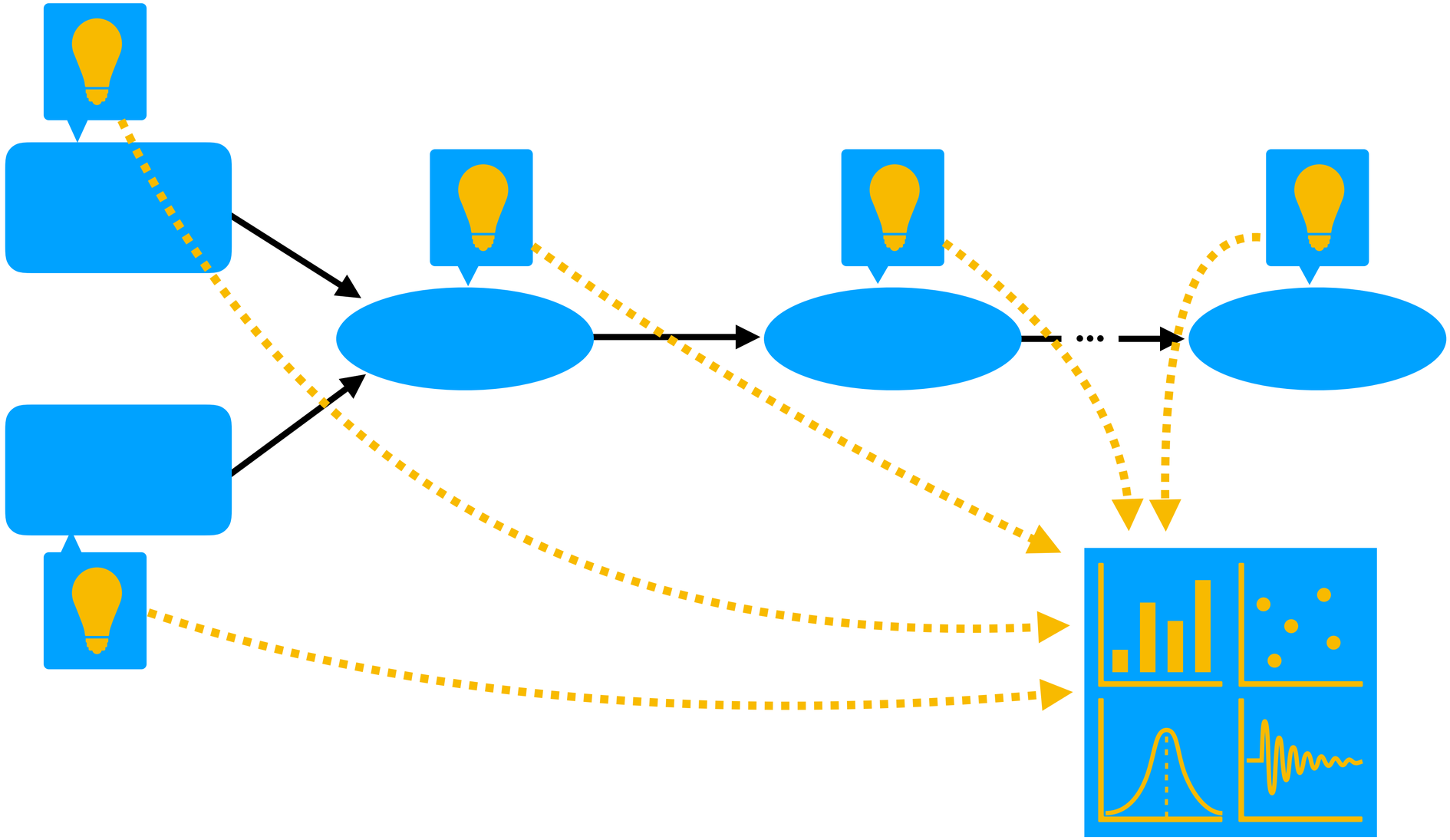












```
class MacCounter : private ExprVisitor {
 public:
  MacCounter() {
    count_ = 0;
  }
  static int64_t GetTotalMacNumber(const Expr& expr) {
    LOG(INFO) << "This pass only counts MACs in direct CONV 2D, "
              << "CONV 2D Transpose and Dense ops";
    MacCounter counter;
    counter(expr);
    return counter.count_;
  }

 private:
  void VisitExpr_(const CallNode* call_node) final {
    static const auto& fprep =
        Op::GetAttr<FMacCount>("FMacCount");
    auto f = fprep.get(call_node->op, nullptr);
    if (f != nullptr) count_ += f(GetRef<Call>(call_node));
    ExprVisitor::VisitExpr_(call_node);
  }

  int64_t count_;
};
```

```
class MacCounter : private ExprVisitor {
 public:
  MacCounter() {
    count_ = 0;
  }
  static int64_t
    LOG(INFO) <<
              <<
    MacCounter c
    counter(expr
    return count
  }

 private:
  void VisitExpr
    static const
        Op::GetA
    auto f = fpr
    if (f != nul
    ExprVisitor::
  }

  int64_t count_;
};
```

```
class FindDef : private ExprVisitor {
 private:
  VarMap<Expr> expr_map_;

  void VisitExpr_(const LetNode* l) final {
    CHECK_EQ(expr_map_.count(l->var), 0);
    expr_map_[l->var] = l->value;
    VisitExpr(l->value);
    VisitExpr(l->body);
  }
};
```

There’s no Relay-sanctioned way to build program analyses!

This leads to problems:

This leads to problems:

- Duplication of effort

This leads to problems:

- Duplication of effort
- High barrier to entry for new developers

This leads to problems:

- Duplication of effort
- High barrier to entry for new developers
- Less readability and maintainability

# [Relay][RFC] Analysis Infrastructure #3895

Open **MarisaKirisame** opened this issue on Sep 4 · 0 comments

**MarisaKirisame** commented on Sep 4 Contributor

RN there is already a few analysis in relay.
For example, quantize analyze for the best range, an WIP bitpack analyze for the correct layout, Partial Eval do a trivial analysis for functions id, ANF do analysis for scope...
One can even say that type inference is an analysis.
And annotations like stop_fusion is analysis as well.

RN there is two way to deal with analysis:
returning a data structure
special annotate node.

# [Relay][RFC] Analysis Infrastructure #3895

Open **MarisaKirisame** opened this issue on Sep 4 · 0 comments

# [RFC] Data-flow Analysis Functionality on TVM IR #4468

New issue

Open **DKXXXL** opened this issue 4 hours ago · 4 comments

**DKXXXL** commented 4 hours ago • edited

## Problem

When developing program passes on TVM IR (the one once was **Halide IR**), it is normal to ask for all sorts of information requiring program analysis, for example, live variable analysis for dead code elimination. This requirement becomes urgent when TVM has to directly issue intrinsic and the subsequent processing stage (for example LLVM) **cannot analyze the program because of these intrinsic**.

**Assignees**

No one assigned

**Labels**

None yet

**Projects**

None yet

```
class FindDef : private ExprVisitor {
 private:
  VarMap<Expr> expr_map_;

  void VisitExpr_(const LetNode* l) final {
    CHECK_EQ(expr_map_.count(l->var), 0);
    expr_map_[l->var] = l->value;
    VisitExpr(l->value);
    VisitExpr(l->body);
  }
};
```

**Needs documentation!**

```
1  class FindDef : private ExprVisitor {
2   private:
3    VarMap<Expr> expr_map_;
4
5    void VisitExpr_(const LetNode* l) final {
6      CHECK_EQ(expr_map_.count(l->var), 0);
7      expr_map_[l->var] = l->value;
8      VisitExpr(l->value);
9      VisitExpr(l->body);
10   }
11 };
```

**Needs documentation!**

```
1  class FindDef : private ExprVisitor {
2   private:
3    VarMap<Expr> expr_map_;
4
5    void VisitExpr_(const LetNode* l) final {
6      CHECK_EQ(expr_map_.count(l->var), 0);
7      expr_map_[l->var] = l->value;
8      VisitExpr(l->value);
9      VisitExpr(l->body);
10   }
11 };
```

**Needs a standard data interchange format!**

**Needs documentation!**

```
1  class FindDef : private ExprVisitor {
2   private:
3    VarMap<Expr> expr_map_;
4
5    void VisitExpr_(const LetNode* l) final {
6      CHECK_EQ(expr_map_.count(l->var), 0);
7      expr_map_[l->var] = l->value;
8      VisitExpr(l->value);
9      VisitExpr(l->body);
10   }
11 };
```

**Needs a standard data interchange format!**

**...and needs to be discoverable/accessible!**

*What do we want in an analysis framework?*

*What do we want in an analysis framework?*

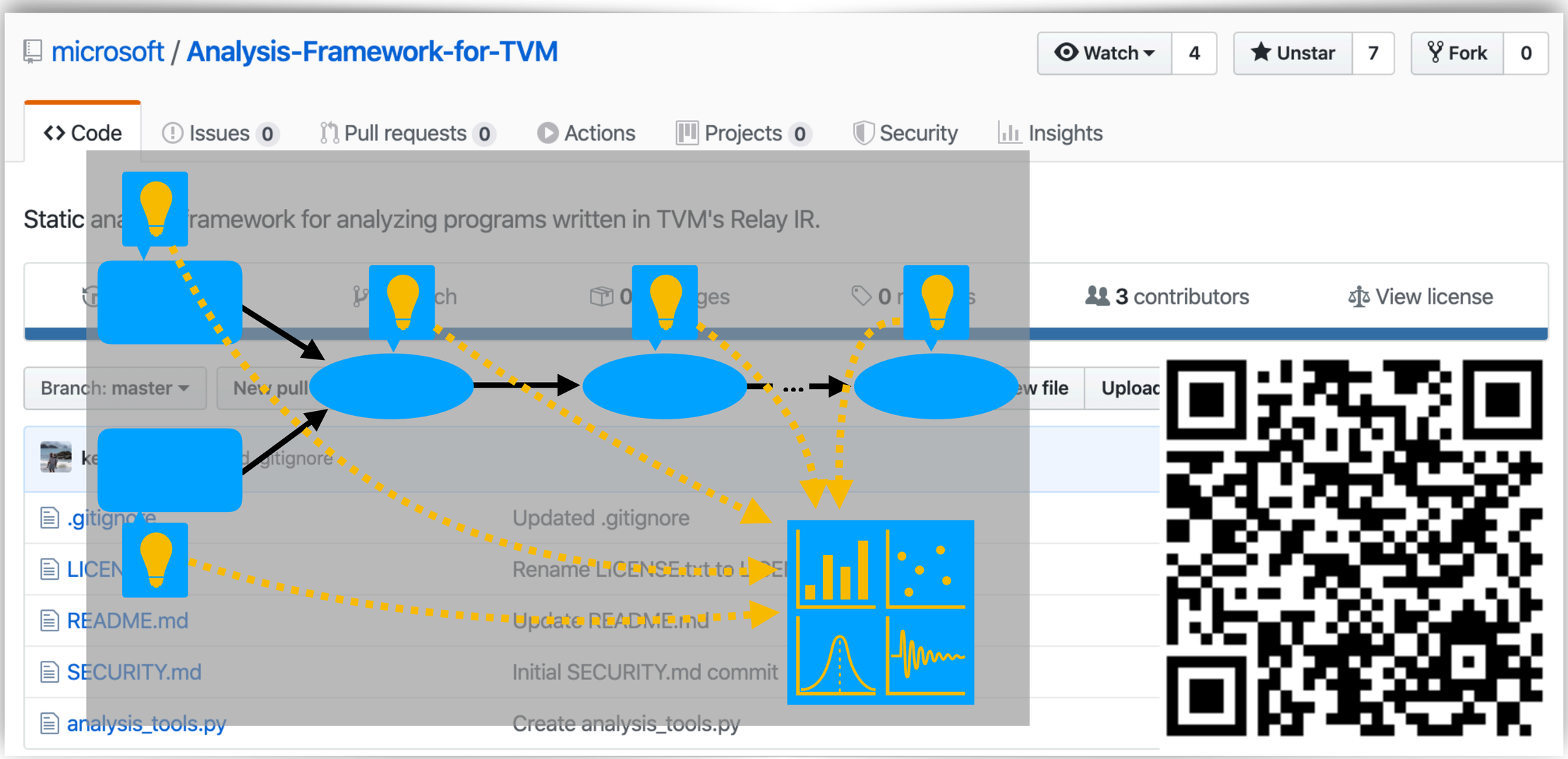
- Supports many types of program analyses

*What do we want in an analysis framework?*

- Supports many types of program analyses
- Quick to write new analyses

*What do we want in an analysis framework?*

- Supports many types of program analyses
- Quick to write new analyses
- Promotes composing analyses together




https://github.com/microsoft/Analysis-Framework-for-TVM

508 lines (508 sloc) | 16.4 KB

Raw Blame

In [ ]:
```
import tvm
from tvm import relay
import tvm.relay.analysis_tools
```

We'll start by examining a simple Relay program:

In [ ]:
```
program = relay.const(1) - (relay.var('x') * relay.var('y'))
```

This simple analysis pass visits all `Calls`. It uses the `AnalysisPass` helper method `_add_detail` to attach analysis results to an expression. In this case, it attaches an analysis result named `'readable_name'` to the `Call` being visited. `_add_detail` is one of the main conveniences added by this simple analysis framework.

In [ ]:
```
class GetReadableName(relay.analysis_tools.AnalysisPass):
    def visit_call(self, call):
        super().visit_call(call)
        self._add_detail(call, readable_name=call.op.name)
```

https://github.com/gussmith23/tvm/blob/analysis-framework-demo/demo.ipynb

# Moving forward

## [RFC] Program Analysis Framework in Relay #4449

Open **gussmith23** opened this issue 4 days ago · 0 comments

**gussmith23** commented 4 days ago — Contributor

Please also see #3895, which is **@MarisaKirisame**'s RFC around a specific change to support analyses in Relay.

This RFC pertains to building a centralized, comprehensive program ana
primary uses of program analyses in Relay: for generating analysis data
things such as quantization, and for generating human-readable data **us**
exploring Relay programs. Whereas Marisa's request pertains more to th
inspired by the second use-case, and motivated by a desire to build a fr
cases. Such frameworks exist -- see LLVM's framework, which is design
both the compiler and the developer.

I built a small analysis framework for Relay this past summer at Microsof
readable analyses of Relay programs. A demo of this framework can be

https://github.com/apache/incubator-tvm/issues/4449