Ariadne: Analysis for Machine Learning Programs

Fabian Schiebel

Seminar: Advanced Techniques in Software Analysis

University of Paderborn

06.07.2020

Example*: Image recognition

```
import tensorflow as tf
...
def conv_net(x_dict, n_classes, dropout, reuse,
                                        is_training):
    with tf.variable_scope('ConvNet', reuse=reuse):
        x = x_dict['images']
        # MNIST data input is a 1-D vector of
        #   784 features (28*28 pixels)
        # Reshape to match picture format
        #   [Height x Width x Channel]
        # Tensor input become 4-D:
        #   [Batch Size, Height, Width, Channel]
        x = tf.reshape(x, shape=[-1, 28, 28, 1])
        ...
```

Example*: Image recognition

```
import tensorflow as tf
...
def conv_net(x_dict, n_classes, dropout, reuse,
                                        is_training):
    with tf.variable_scope('ConvNet', reuse=reuse):
        x = x_dict['images']
        # MNIST data input is a 1-D vector of
        #   784 features (28*28 pixels)
        # Reshape to match picture format
        #   [Height x Width x Channel]
        # Tensor input become 4-D:
        #   [Batch Size, Height, Width, Channel]
        x = tf.reshape(x, shape=[-1, 28, 28, 1])
        ...
```

Approach

We need

- A type system for tensors
- A dataflow analysis
  - Use the WALA framework

Approach

We need

- A type system for tensors
- A dataflow analysis
  - Use the WALA framework

Approach

We need

- A type system for tensors
- A dataflow analysis
  - Use the WALA framework

The Type System

- *Python types* π:
  - Basic types: `Int`, `channel` ...
  - Record types: Dictionaries
  - Function types
- Tensor types: $[d_1, \ldots, d_n \ \mathit{of}\ \pi]$

The Type System

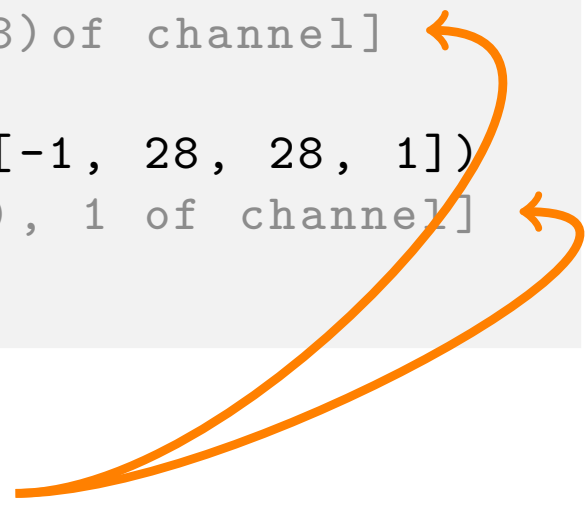
Example*

```
import tensorflow as tf
...
def conv_net(x_dict, n_classes, dropout, reuse,
                                              is_training):
    with tf.variable_scope('ConvNet', reuse=reuse):
        # x: [batch, y(28) * x(28)of channel]
        x = x_dict['images']
        x = tf.reshape(x, shape=[-1, 28, 28, 1])
        # x: [batch, y(28), x(28), 1 of channel]
        ...
```

The Type System

Example*

```
import tensorflow as tf
...
def conv_net(x_dict, n_classes, dropout, reuse,
                                              is_training):
    with tf.variable_scope('ConvNet', reuse=reuse):
        # x: [batch, y(28) * x(28)of channel]
        x = x_dict['images']
        x = tf.reshape(x, shape=[-1, 28, 28, 1])
        # x: [batch, y(28), x(28), 1 of channel]
        ...
```

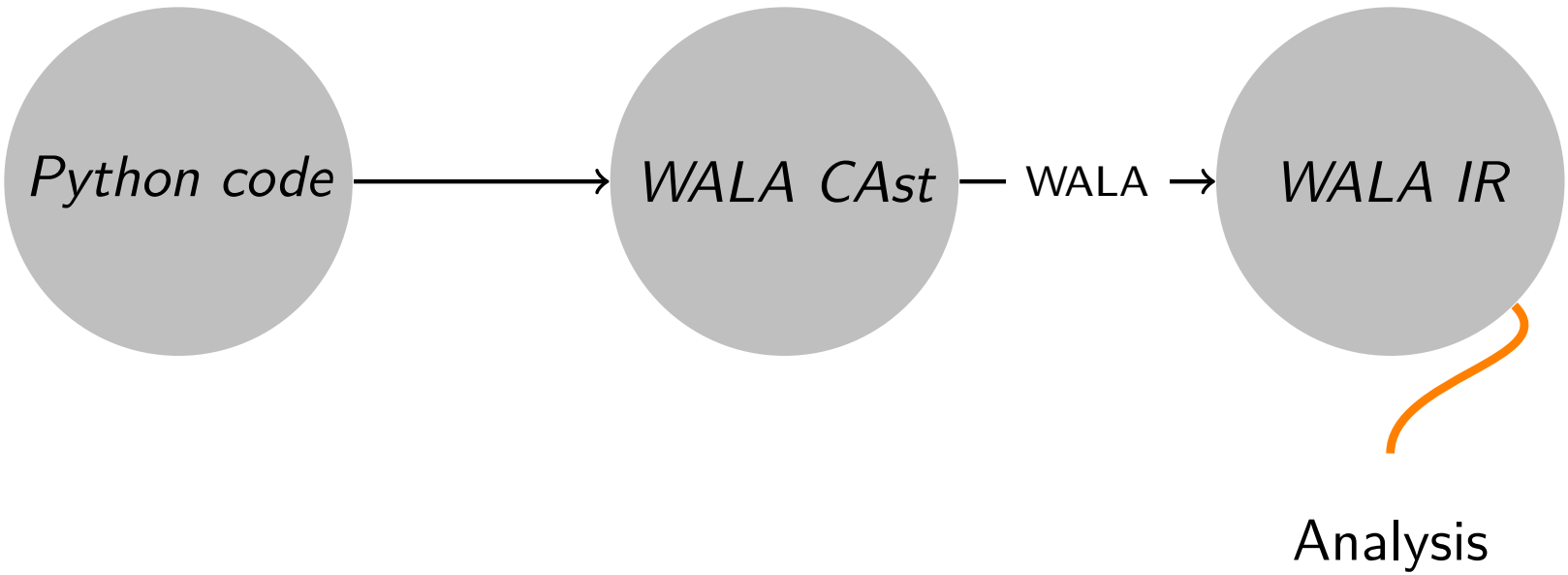


Modeling Python in WALA

Workflow

Modeling Python in WALA

Workflow

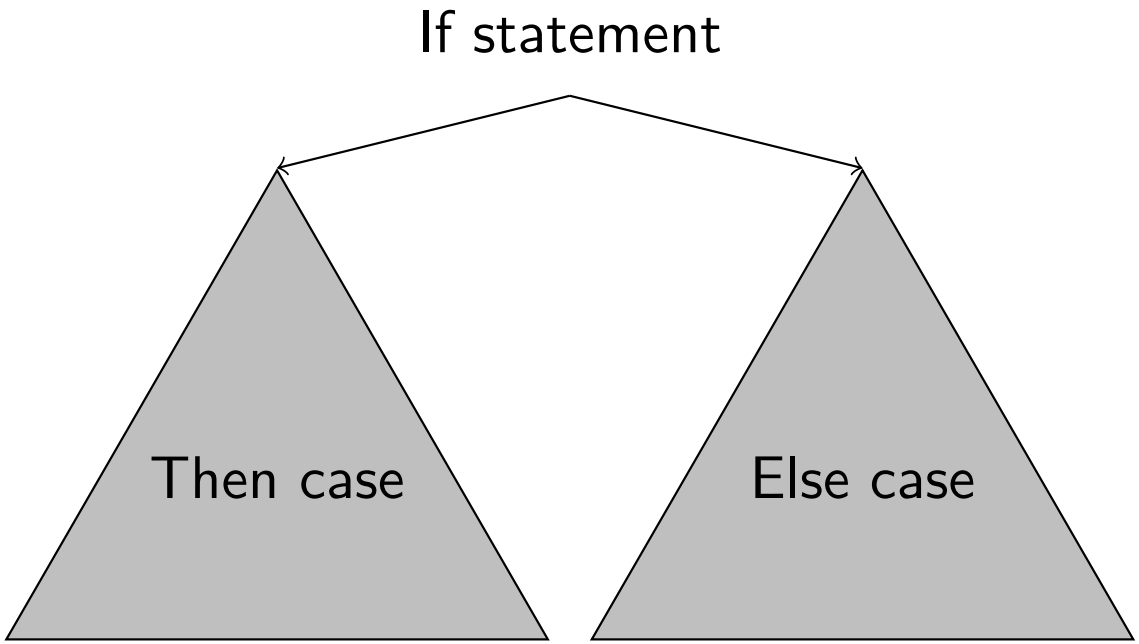


Modeling Python in WALA

Workflow

Modeling Python in WALA

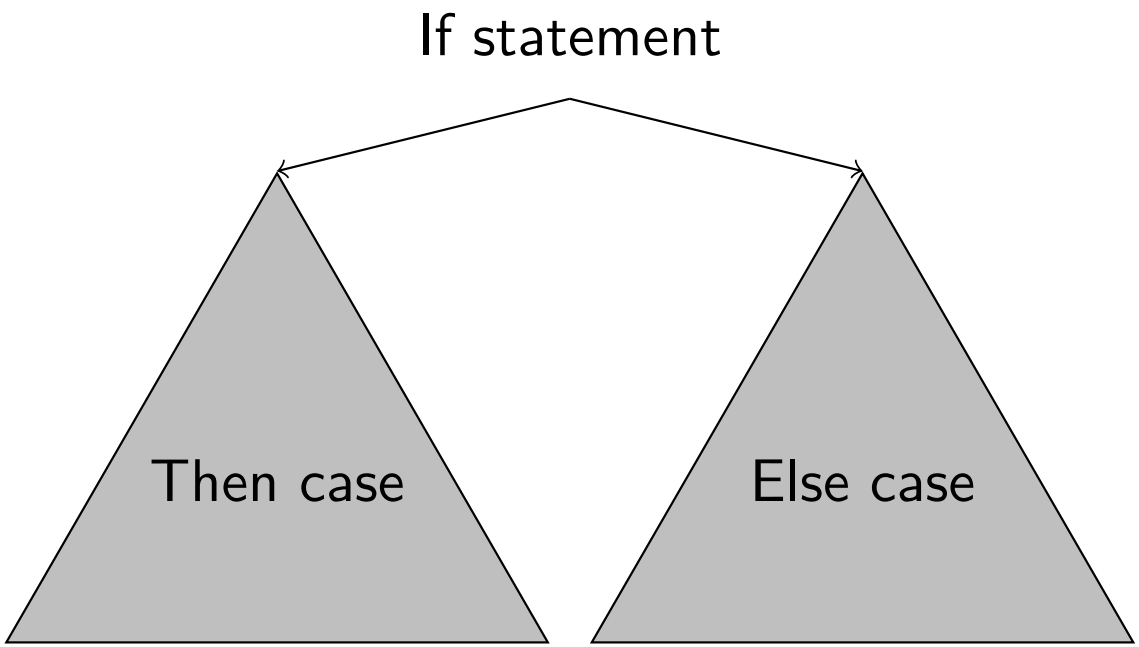
Creating the CAst



- A few language constructs cannot be represented
- Create new nodes

Modeling Python in WALA

Creating the CAst



- A few language constructs cannot be represented
- Create new nodes

Modeling Python in WALA

Creating the IR

Why do we need to create the IR?

- Custom CAst nodes
- Custom semantics of existing nodes

Modeling Python in WALA

Creating the IR

Why do we need to create the IR?

- Custom CAst nodes
- Custom semantics of existing nodes

Modeling Python in WALA

Creating the IR: Call sites*

```
class Foo:
    def foo(self, a):
        ...

x = Foo()
```

Modeling Python in WALA

Creating the IR: Call sites*

```
class Foo:
    def foo(self, a):
        ...

x = Foo()
```

```
x.foo(42)
```

Modeling Python in WALA

Creating the IR: Call sites*

```
class Foo:
    def foo(self, a):
        ...

x = Foo()
```

```
x.foo(42)
```

```
y = x.foo
y(42)
```

Modeling Python in WALA

Creating the IR: Call sites*

```
class Foo:
    def foo(self, a):
        ...

x = Foo()
```

```
x.foo(42)
```

```
y = x.foo
y(42)
```

```
Foo.foo(x, 42)
```

Modeling Python in WALA

Creating the IR: Call sites*

```
class Foo:
    def foo(self, a):
        ...

x = Foo()
```

```
x.foo(42)
```

```
y = x.foo
y(42)
```

```
Foo.foo(x, 42)
```

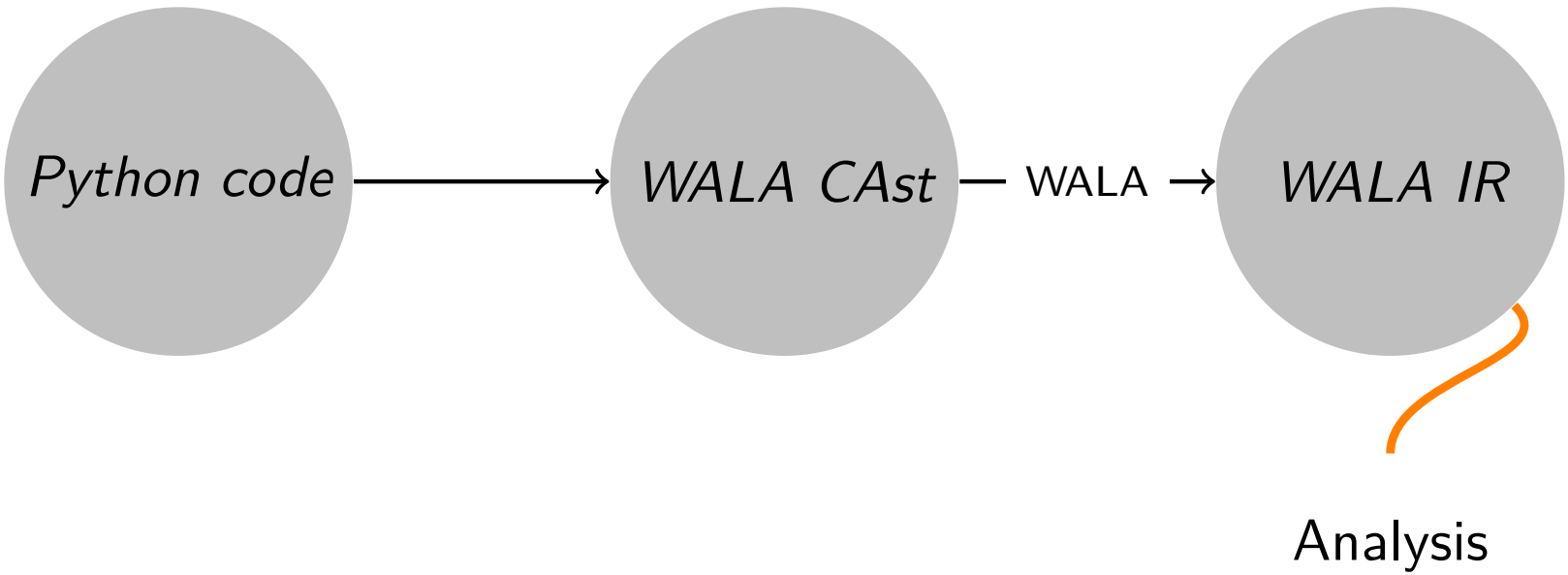
```
x.foo = Foo.foo
x.foo(x, 42)
```

Defining the Dataflow Analysis

Workflow

Defining the Dataflow Analysis

Workflow



Defining the Dataflow Analysis

Goal: Precise dataflow analysis

- Modeling of *Python* semantics
- Modeling of *TensorFlow* API
- Helper Analyses

Defining the Dataflow Analysis

Goal: Precise dataflow analysis

- Modeling of *Python* semantics
- Modeling of *TensorFlow* API
- Helper Analyses

Defining the Dataflow Analysis

Pointer Analysis*

Which variables alias?

```
import tensorflow as tf
...
mnist = input_data.read_data_sets("/tmp/data/")
    ...
def conv_net(x_dict, n_classes, dropout, reuse,
                                        is_training):
    with tf.variable_scope('ConvNet', reuse=reuse):
        x = x_dict['images']
        # MNIST data input is a 1-D vector of
        #   784 features (28*28 pixels)
        x = tf.reshape(x, shape=[-1, 28, 28, 1])
        ...
```

Defining the Dataflow Analysis

Pointer Analysis*

Which variables alias?

```
import tensorflow as tf
...
mnist = input_data.read_data_sets("/tmp/data/")
    ...
def conv_net(x_dict, n_classes, dropout, reuse,
                                        is_training):
    with tf.variable_scope('ConvNet', reuse=reuse):
        x = x_dict['images']
        # MNIST data input is a 1-D vector of
        #   784 features (28*28 pixels)
        x = tf.reshape(x, shape=[-1, 28, 28, 1])
        ...
```

Defining the Dataflow Analysis

Callgraph Analysis

Which functions may be called?

```
class Foo:
    def baz():
        ...
class Bar:
    def baz():
        ...

x = Foo()
x.baz()

x = Bar()
x.baz()
```

Defining the Dataflow Analysis

Callgraph Analysis

Which functions may be called?

```
class Foo:
    def baz():
        ...
class Bar:
    def baz():
        ...

x = Foo()
x.baz()

x = Bar()
x.baz()
```

Defining the Dataflow Analysis

Dataflow Graph

Dataflow from y to x: $x \prec y$

```
# Assignment

x = y
```

```
# Return

def foo():
    ...
    return y

x = foo()
```

```
# Function call

def foo(x):
    ...

foo(y)
```

Defining the Dataflow Analysis

Tensor Estimate

Tensor estimate $T(x)$ by induction:

Base case: Type annotations

Induction step:

- $T(x) \subseteq T(y) \quad \textit{if} \quad x \prec y$
- $T(x) \subseteq z \quad \textit{if} \quad x \prec \texttt{reshape}(y, z) \wedge \exists z_i \in S(z) : T(y) \doteq z_i$
- other *TensorFlow* APIs

Defining the Dataflow Analysis

Tensor Estimate

Tensor estimate $T(x)$ by induction:

Base case: Type annotations

Induction step:

- $T(x) \subseteq T(y) \quad \textit{if} \quad x \prec y$
- $T(x) \subseteq z \quad \textit{if} \quad x \prec \texttt{reshape}(y, z) \wedge \exists z_i \in S(z) : T(y) \doteq z_i$
- other *TensorFlow* APIs

Defining the Dataflow Analysis

Tensor Estimate

Tensor estimate $T(x)$ by induction:

Base case: Type annotations

Induction step:

- $T(x) \subseteq T(y) \quad \textit{if} \quad x \prec y$
- $T(x) \subseteq z \quad \textit{if} \quad x \prec \texttt{reshape}(y, z) \wedge \exists z_i \in S(z) : T(y) \doteq z_i$
- other *TensorFlow* APIs

Evaluation

- Modeled four *TensorFlow* API functions
- Analyzed six ML programs

- No false positives
- Fast execution

Evaluation

- Modeled four *TensorFlow* API functions
- Analyzed six ML programs

- No false positives
- Fast execution

Conclusion

- Machine Learning programs in dynamic languages (*Python*)
- Type system for tracking tensors
- Dataflow Analysis
  $\Rightarrow$ Modeling *Python* in WALA
- Fast execution and precise results

References

*) Julian Dolby, Avraham Shinnar, Allison Allain, and Jenna Reinen. **Ariadne: Analysis for Machine Learning Programs**. *In Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 1-10, 2018.

Questions?