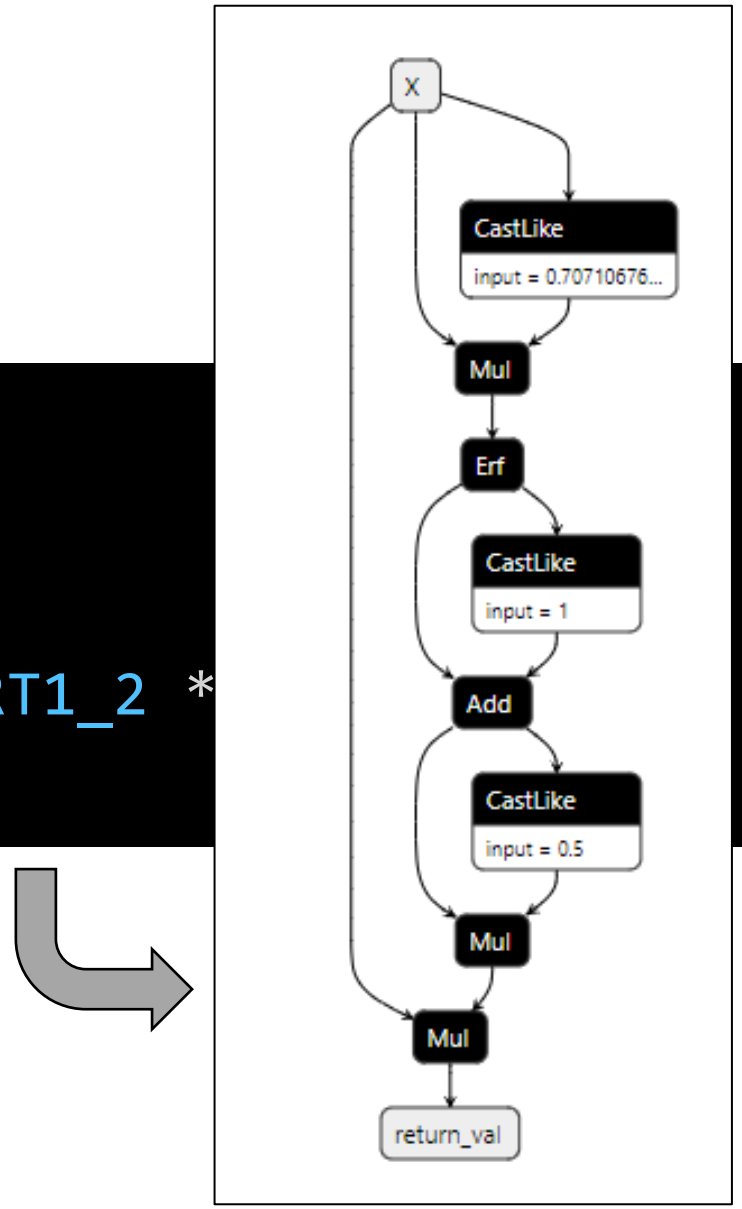
ONNX Script: Authoring ONNX In Python

G. Ramalingam

Microsoft

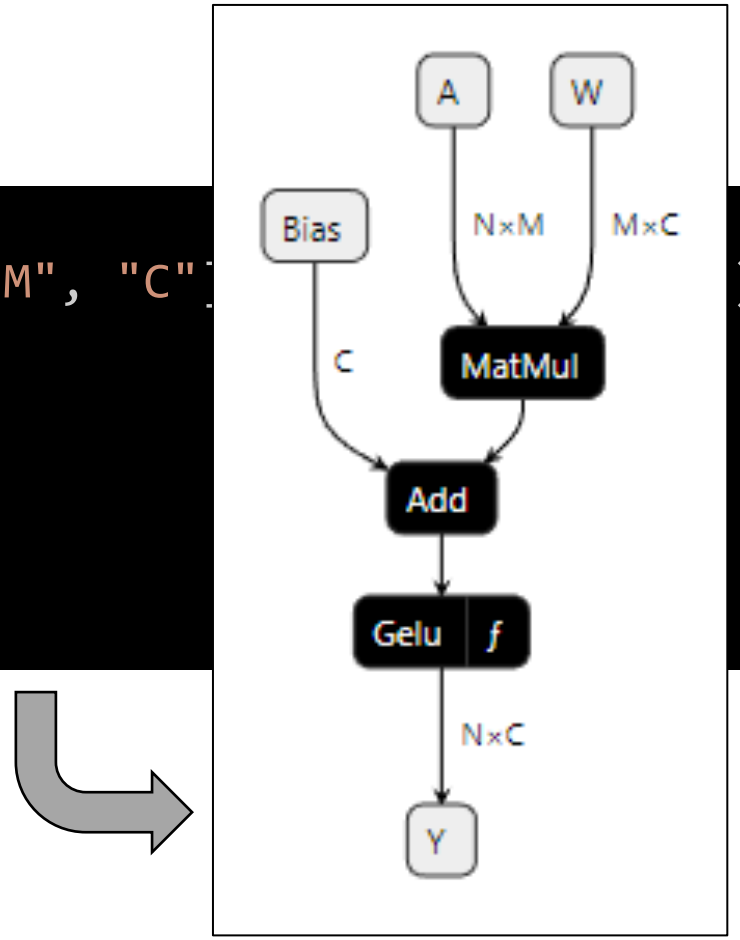
Example: Gelu definition

```
M_SQRT1_2 = math.sqrt(0.5)
@script()
def Gelu(X):
    phiX = 0.5 * (op.Erf(M_SQRT1_2 *
    return X * phiX
```



Example: A simple model

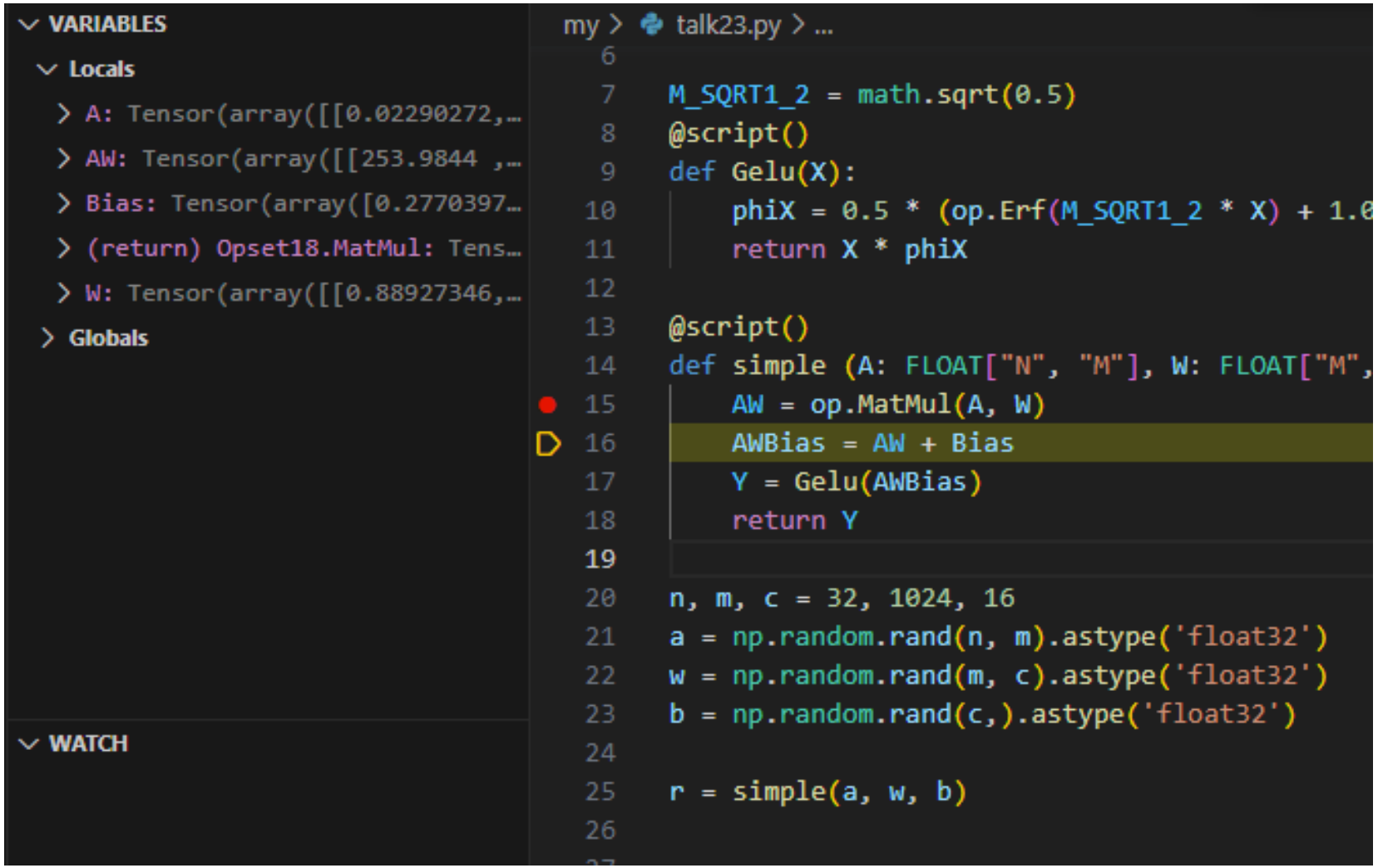
```
@script()
def simple (A: FLOAT["N", "M"], W: FLOAT["M", "C"]
      -> FLOAT["N", "C"]:
    AW = op.MatMul(A, W)
    AWBias = AW + Bias
    Y = Gelu(AWBias)
    return Y
```



Another example (with control-flow)

```
def Dropout(data, ratio, training_mode, seed):
  if (training_mode):
    rand = op.RandomUniformLike(data, seed=seed, dtype=FLOAT)
    mask = (rand >= ratio)
    output = op.Where(mask, data, 0) / (1.0 - ratio)
  else:
    mask = op.Expand(True, op.Shape(data))
    output = data
  return (output, mask)
```

Support for execution and debugging



Higher Order Ops (Scan, SequenceMap, …)

```
from onnxscript import script, graph

@script()
def CumulativeSum(X):
    @graph()
    def Sum(sum_in, next):
        sum_out = sum_in + next
        scan_out = op.Identity(sum_out)
        return sum_out, scan_out

    all_sum, cumulative_sum = op.Scan(0, X, body=Sum, num_scan_inputs=1)
    return cumulative_sum
```

Motivation and Uses

- Functions are a key extensibility mechanism in ONNX
  - Exposes richer (op) API to model developers
  - Retains smaller (core) op surface area to be supported by a backend
  - Enables development and use of optimized kernels on-demand
- Enable easy development of function-ops
  - Function ops in the ONNX standard
  - Custom ops to be added to a model
  - Libraries of custom ops
  - Simplify development of ONNX exporter from a framework
- Ongoing effort to define torchlib as a library of ONNX functions
  - For use in Pytorch’s ONNX Exporter

Other Uses

- Simplify ONNX Exporters

- Experimentation with ONNX
- Create ONNX test-cases
- Debug ONNX backends and/or ONNX models

Summary: Features

- Constant literals (0, 1, [-1], …)
- Automatic cast of constants (enables polymorphic code like X+1)
- Operator symbols as syntactic sugar (+, *, …)
- Indexing/Slicing (e1[0], e1[1:5], …)
- Nested expressions
- Control-flow
- Nested functions (for graph attributes)
- Type and shape annotations

- Operator sets, ops, their types and documentation

https://github.com/microsoft/onnxscript/