Semantics of ONNX graphs

Dumitru Potop Butucaru – Inria

SONNX presentation – March 12, 2025

ONNX graph

- ONNX specification
  - Acyclic graph of nodes
    - Cf. https://onnx.ai/about.html, https://onnx.ai/onnx/intro/concepts.html
    - Certain nodes (loops, test) can have sub-graphs
    - Functions provide another form of hierarchy
      - Both for specification and execution
  - Nodes are expected to be side effect-free
    - Cf. https://onnx.ai/onnx/repo-docs/IR.html
    - Attention: no side effects is not the same as deterministic
      - Randomness: RandomNormal, RandomUniform, Dropout…

ONNX graph

- ONNX graph semantics
  - Cf. https://onnx.ai/onnx/repo-docs/IR.html
  - Inference semantics – the graph is a stateless function
  - Training semantics – stateful function with parameters as states
    - Parameter = ONNX graph initializer
    - Init method assigning initial (usually random) values to parameters
    - Train method performing one step of back-propagation and gradient descent
    - Inference method

DAG – directed acyclic graphs

- Nodes with inputs and outputs connected by directed arcs
  - Every input of a node must and every graph output be alimented by either:
    - one output of another node
    - one input
    - This property is a particular case of Static Single Assignment
- The graph must be acyclic
  - If node A (transitively) depends on B and B depends on A, then the graph is incorrect

DAG dataflow execution

- DAG execution starts in a state where
  - all DAG inputs have a value
  - all node outputs are undefined

- Dataflow node execution
  - When all inputs of the node have a value, the node is activated
  - On an activated node, the function associated with the node can be applied to compute the value of all outputs based on that of the inputs

- DAG execution semantics
  - Interleaving: execute one activated node at a time
  - True concurrency: multiple activated nodes can be executed concurrently

DAG dataflow execution guarantees

- Completeness
  - DAG execution always completes, i.e. all nodes are executed and all node outputs are assigned a value, regardless of execution semantics

- Determinism
  - If the graph contains no random operators, then for given initial input values the computed node and graph output values are the same, regardless of the execution semantics and the node interleaving (execution order).
  - When pseudo-random nodes are involved, determinism is guaranteed if each random node receives the same seed (jax-style deterministic “randomness”).

Limits to DAG expressiveness

- Can only represent stateless computations
  - Feedforward networks
  - Stateful behaviors unrolled over fixed-size vectors
    - E.g. working on a sliding window of inputs

- Cannot:
  - Efficiently represent stateful computations (unrolling cost is linear in the size of the unrolling window)
  - Represent computations that depend on the indefinite past (e.g. a roadsign detected some time ago)