EC7.18

ONNX Embedded profile - Specification (On Going)

contact@confiance-ai.fr | www.confiance.ai

Contributors

	Name	Organisation	Role
Responsible for the deliverable	Marie-Charlotte TEULIERES	Airbus Protect	Author
Co-authors	Christophe GABREAU	Airbus	Co-author
	ONNX Community Members	IRT-Saint Exupery | IRT System-X | Airbus | ONERA | Safran Electronics | CEA | Embraer	Contributors

Document Control

Revision	Date	Commentary	Author
v1.0			

Contents

A. Introduction

Embedded ML systems are subject to Authority regulation constraints and, in the future, to a demonstration of conformity to EUROCAE WG114 (oing) guidance currently discussed in the forum of the joint standardization group EUROCAE WG-114 / SAE G-34. In particular, it includes objectives about the preservation of the semantics and the properties of the model during the transformations of the designed ML Model into an implemented ML Model. The transition between the design process and the implementation process is defined by the Machine Learning Model Description (MLMD). Based on previous work done to evaluate format supporting MLMD, it has been decided to focus on Open Neural Networks eXchange (ONNX) format. The format does not complete all requirements defined in EUROCAE WG114 (oing). The MLMD | ONNX Community has been initiated to propose a specification for an embedded profile of ONNX.
The objective of the document is to propose a summary of the "Topics" which were discussed during the community workshops. At the stage of discussion, it is not a specification, the work is still in progress.

B. Topics

B.1. Develop the operator semantic and specification

B.1.1 Rationale

Propose a complete specification of operators, unambiguous, no subject to interpretation or approximation. The specification should use a set of well-defined operators/primitives.

B.1.2 Needs

The objective is to specify the behaviour (which can be provided through an executable code). Compliance levels should be defined and attributed according to certification level of the system. Needs are dependent of the compliance level choosen.
The list of needs is given by, but not limited to :

- **Operators**
  Operators semantic should be Unambiguous, complete, and no subject to interpretation
  The specification should rely on a set of well defined basic operators
  Include the behaviour of the operator, and parameters, attributes description
- **Representation**
  It should define Data Types
  Numbers representation has to be expressed
  The uncertainty related to floating point operations should be captured
- **Compliance level** definition (for example : bit-level, functional-level)

B.1.3 Solutions

Two ways to specify an operator have been identified :

- Propose a set of property that must be hold,
- Propose an algorithm, as an operational procedure

B.1.3.1 Formal Specification Language

The formal specification language is strictly a specification. It is expressed using pre/post conditions. It does not give indication on how to implement the function. Compliance with the specification is done by verifying that, assuming pre-conditions hold, that the post conditions expressed by the specification hold.

Pros	Cons
• Formal • It is a strictly a specification, i.e., there is no indication whatsoever of the way it shall be implemented.	• Verifying compliance with the specification requires either to use some formal method/tool (e.g., FramaC) or, if verification is done using testing, a means to execute the specification. • Not always very easy to stick to a pure specification. In many cases, specifying the function boils down to describing how it can be implemented

Examples (non-exhaustive list) :

- ACSL / FramaC
- Why3 / WhyML
- B/event-B
- CoQ
- Specification of Mathematical library such as BLAS
- Other existing specification such as TOSA

B.1.3.2 Reference implementation

The implementation is done using a language with a clear semantics and correct implementation. Providing an algorithm does not mean that the implementation has to copy it. Compliance with the specification is done by comparing the results generated by the implementation under test and those generated by the reference implementation. The algorithm shall not refer to mathematical objects, and computational objects.

Pros	Cons
• Ease the implementation • Is executable and can be used as a test oracle	• The specification is implicit. Verifying compliance requires executing the specification. • The specification may be considered as an example of implementation • The specification relies on the semantic of the programming language

B.1.3.3 Other solutions

Other solutions may be explored such as :

- Check at what has been done by other lib and framework. Ex : See e.g., Intel's OneAPI specification for convolutions here
- Propose an implementation in infinite precision. For example : use of the python rationales.

B.2. Execution order and control flow

B.2.1 Rationale

The ML model semantics is specified through the MLMD (Machine Learning Model Description, concept developed in ARP6983). the ML model can be hosted on one or several SW/HW items. To this purpose, the ML Constituent architecture decomposes the MLMD into MLMIDs (one per item). The objective is to specify how the MLMIDs interact between each others (data/-control flow) to replicate the MLMD execution. We should be able to express the operation ordering.

B.2.2 Needs

The list of needs is given by, but not limited to :

- **Express Scheduling Order** : Express all possible implementation, specify which implementations are valid. Define a static scheduling.
- **Decompose a Neural Network** : a Neural Network can possibly be deployed onto one or several HW/SW items, i.e. the MLMD is decomposed into several MLMIDs (one per item)
- **Demonstrate the decomposition is correct** : The objective is to demonstrate that the union of the MLMIDs is consistent with the MLMD description/semantics.

B.2.3 Solutions

B.2.3.1 Petri Nets

Petri nets can express all possible implementation for a model. The solution has been proposed in Pagetti (2023), presented to the ONNX community [1]Presentation available on the Confiance.AI sharepoint. The solution has been presented using NNEF.
Petri Nets are compliant with the expression of scheduling order. Paths of the petri nets specify all the possible implementation|scheduling. And fulfill the need of DataFlow semantic expression which is neither yet proposed by NNEF nor ONNX.
In order to include the split of neural networks, it has been proposed to include subgraphs and extension of the syntax to express interaction between subgraphs. It introduces a colored petri nets to be compared to initial petri nets demonstrating all possible implementations in order to prove the preservation of the semanctic.

Pros	Cons
•	• •

B.2.3.2 Dataflow

Dataflow is a solution allowing to cover :

- Parallelization
- Grouping components
- Pipelining
- Splitting
- Allocation

- Static scheduling (in order)

Pros	Cons
• Cover a large part of needs • Easy to understand	• Complex low-level aspects (such as shared memory, tiling, copying, blocking, vector execution) may need a dedicated language[a]. • Static scheduling (in time) • Dynamic allocation / scheduling not covered

[a] See work of Inria - CORSE

For more information, please refer to the presentation made by Dumitru Potop on the 13th of December.

B.2.3.3 Other solutions

Here is a list, non-exhaustive, of solutions to be explored :

- Automata
- SysML V2.0 or UML UMV V2.x activity diagrams specification. UML V2.x activity diagrams allows representing the flow of data and synchronization between activities.
- CircuitFlow.
- SystemC
- DeepDSL.
- Synchronous dataflow.

B.3. Syntax Specification

B.3.1 Rationale

The syntax should be textual and human readable. In order to be open to manual code, and auditability.

B.3.2 Needs

The syntax should :

- be human readable without the use of tools
- be textual
- be open to manual code

B.3.3 Solutions

B.3.3.1 DSL

The solution can be developed by defining a specific DSL that will capture all the concepts and constructs of ONNX. The implementation can either be :

- Standalone
- Embedded in a more general language (ex : Scala)

B.3.3.2 API

Use an existing API to create a computational graph. Examples : Python ONNX API, Keras etc.

Pros	Cons
• Using Python would leverage all Python constructs • Part of the semantics would be already defined by Python's • Representation would be familiar to Data Scientists • Generating an ONNX representation in the form of a Python code would allow the direct verification of the correctness of the representation: the "execution" of the representation would create an ONNX file that could be compared to the original one.	• Verbosity (e.g., all relations between elements must be constructed using some API call). • For the approach to be really useful, the generated representation must be correct / executable code.

B.4. Summary

Topic	Rationale	Needs	Solutions
Develop the operators semantic specification	Propose a complete specification of operators, unambiguous, no subject to interpretation or approximation. The specification should use a set of well-defined operators/primitives.	• Unambiguous, complete, no subject to interpretation semantic description of all operators. • Include the behaviour of the operator, and parameters, attributes description • Data Types • Numbers representation • Objective is to specify the behaviour (which can be provided through an executable code) • Define the compliance level (for example : bit-level, functional-level)	• Formal specification languages : ACSL Why3 • Existing specification of mathematical libraries BLAS • Existing specification TOSA
Execution Order | Control Flow	The ML model semantics is specified through the MLMD (Machine Learning Model Description, concept developed in ARP6983). the ML model can be hosted on one or several SW or HW items. To this purpose, the MLC architecture decomposes the MLMD into MLMIDs (one per item). The objective is to specify how the MLMIDs interact between each others (data/control flow) to replicate the MLMD execution. We should be able to express the operation ordering.	• Express the scheduling order • Splitting a neural network in order to deploy on several items • Demonstrate split is coherent to description and semantic	• Using Petri Nets to express possible implementation for a model (see example on NNEF – ref C.Pagetti and A.Gauffriau) • Integrate Dataflow on ONNX • Automata

Topic	Rationale	Needs	Solutions
Syntax Specification	Should be textual, and be human readable.		Define a specific DSL (“Standalone” or embedded in a more general language) Use an existing API (e.g. ONNX Python API)
Real time requirements (TBC)	Link to execution order topic.	Express the worst case execution time (1 inference step)	

C. Annex

C.1. Annex 1 : ONNX materials

Here is a list of materials to familiarize yourself with ONNX.

C.1.1 Introduction

ONNX, is an open-source project proposing a Machine Learning Model Description. It is composed of all the tools for execution and interoperability between frameworks. ONNX is a representation format for ML Model, and can be completed by ONNX Runtime as an inference engine. ONNX has been initiated by Facebook and Microsoft and is supported by a community of partners, such as Microsoft, AWS, NXP etc. The format is entitled to evolve a lot thanks to Special Interest Groups or Working Groups. There is a large community using the format, and there is work in development to improve the format on several subjects and for sure on Quantization. ONNX is based on protocol buffers, a technology for data serialization and retrieval of structured data. Protocol buffers is a technology developed by Google. Protocol buffers are characterized by:

- Format
  Based on Protocol buffers, Base 128 Varints, it is binary file format. The protobuf specification for an ONNX model is available at `onnx-ml.proto`, `onnx-data.proto`
  Converters from binary format, to text file format,
- Operations
  Serialization is not deterministic, the same data message can have different representations.
  The format is given with a set of operations.
  It has not a formal standard, so it may be evolutive,
- Interoperability
  Converters are included in the use of protocol buffers for most frameworks such as Caffe, PyTorch, Tensorflow, Keras etc.
  The diversity of frameworks allows to represent several types of ML models, among other the connection with sklearn (regression, decision tree etc.).
- The specification is available at
  ONNX operator description
  ONNX Intermediate Representation
  ONNX Runtime

Bibliography

EUROCAE WG114, S. G. (on going). Arp 6983 : Process Standard for Development and Certification/Approval of aeronautical Safety-Related Products implementing Artificial intelligence.

Pagetti, A. G. . C. (2023). Formal description of ml models for unambiguous implementation. arXiv:2307.12713.

Title: ONNX Embedded profile - Specification (On Going)

Keywords: Certification, Machine Learning, Implementation, Description

The document details discussions of the ONNX | MLMD Community. Rationales, needs and potential solutions have been proposed regarding 3 main ideas to be included in an ONNX to be in accordance to EUROCAE WG114 (oing).

Our partners: