SONNX WG

Towards an ONNX profile for critical systems

Eric JENN[1], Jean SOUYRIS[2], Mohammed BELCAID[3], Henri BELFY[4], Sebastian BOBLEST[5], Jean-Loup FARGES[6], Cong LIU[7], Eduardo MANINO[8], Salomé MARTY-LAURENT[2], Dumitru POTOP-BUTUCARU[9], Jean-Baptiste ROUFFET[10], Mariem TURKI[1], Nicolas VALOT[11], Franck VEDRINE[12]

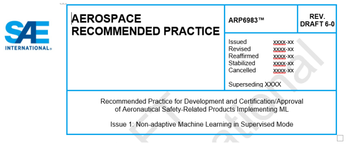
(1) IRT Saint Exupery, (2) Airbus, (3) CS Sopra-Steria, (4) Thales AVS, (5) BOSCH, (6) ONERA, (7) Collins Aerospace , (8) U of Manchester, (9) INRIA, (10) Airbus Protect, (11) Airbus Helicopter, (12) CEA LIST

Agenda

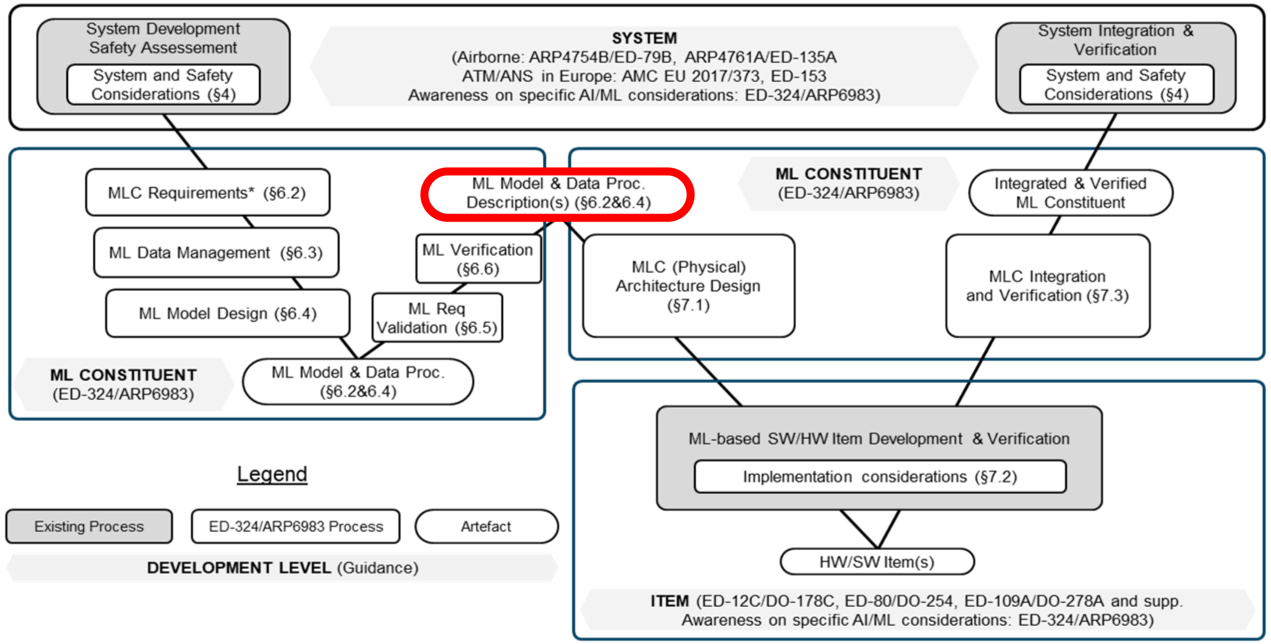
- Objectives of the SONNX working group
- The working group
- Some results
- Next…

Objectives of the SONNX WG

Towards a safe profile…



- General objective
  - Provide a language to describe ML models
- SONNX objectives
  - **Complete** ONNX standard
    - Clarify semantics of operators and graph…
    - Remove ambiguities…
  - **Restrict** the ONNX standard
    - To simplify compliance demonstration with respect to standards (esp. aero standards)
  - Provide a simple **reference implementation** compliant with the standard



Expectations

The ARP 6983 MLMD

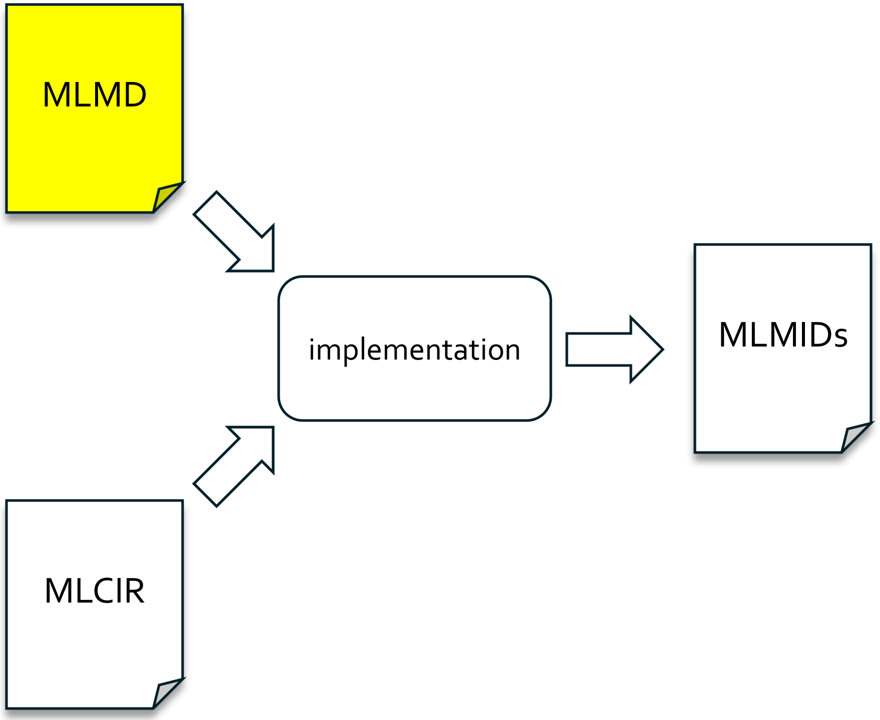
Table D3 - ML Model Design process

	Objective		Activity	Applicability by Assurance Level					Output		Control Category by Assurance Level				
	Description	Ref	Ref	A AL1 SWAL1	B AL2 N/A	C AL3 SWAL2	N/A AL4 SWAL3	D AL5 SWAL4	Data Item	Ref	A AL1 SWAL1	B AL2 N/A	C AL3 SWAL2	N/A AL4 SWAL3	D AL5 SWAL4
5	The ML Model description is developed.	5.4.1.g	5.4.3.6			○	○	○	MLMD	7.4.7			①	①	①


AEROSPACE RECOMMENDED PRACTICE
ARP6983™
REV. DRAFT 6-0
Issued xxxx-xx
Revised xxxx-xx
Reaffirmed xxxx-xx
Stabilized xxxx-xx
Cancelled xxxx-xx
Superseding XXXX
Recommended Practice for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing ML
Issue 1: Non-adaptive Machine Learning in Supervised Mode

In §5.4.3.6 "ML Model Description"

a. The ML model logical architecture is described
b. The ML model hyperparameters are described
c. The ML model parameters are described
d. The analytical/algorithmic syntax and semantics of the ML Model […] are described in an unambiguous manner in the ML Model description to facilitate [allow?] their implementation.
e. The replication criterion (either exact or approximated) is defined from the MLC requirements and if applicable from the ML Model requirements:
f. The execution environment of the ML Model is described.
g. Any necessary dependence on the learning environment (e.g., library, format) is explicitly mentioned.
h. Any information that should not be part of the implemented ML Model is removed or explicitly identified as "not part of the ML Model description".



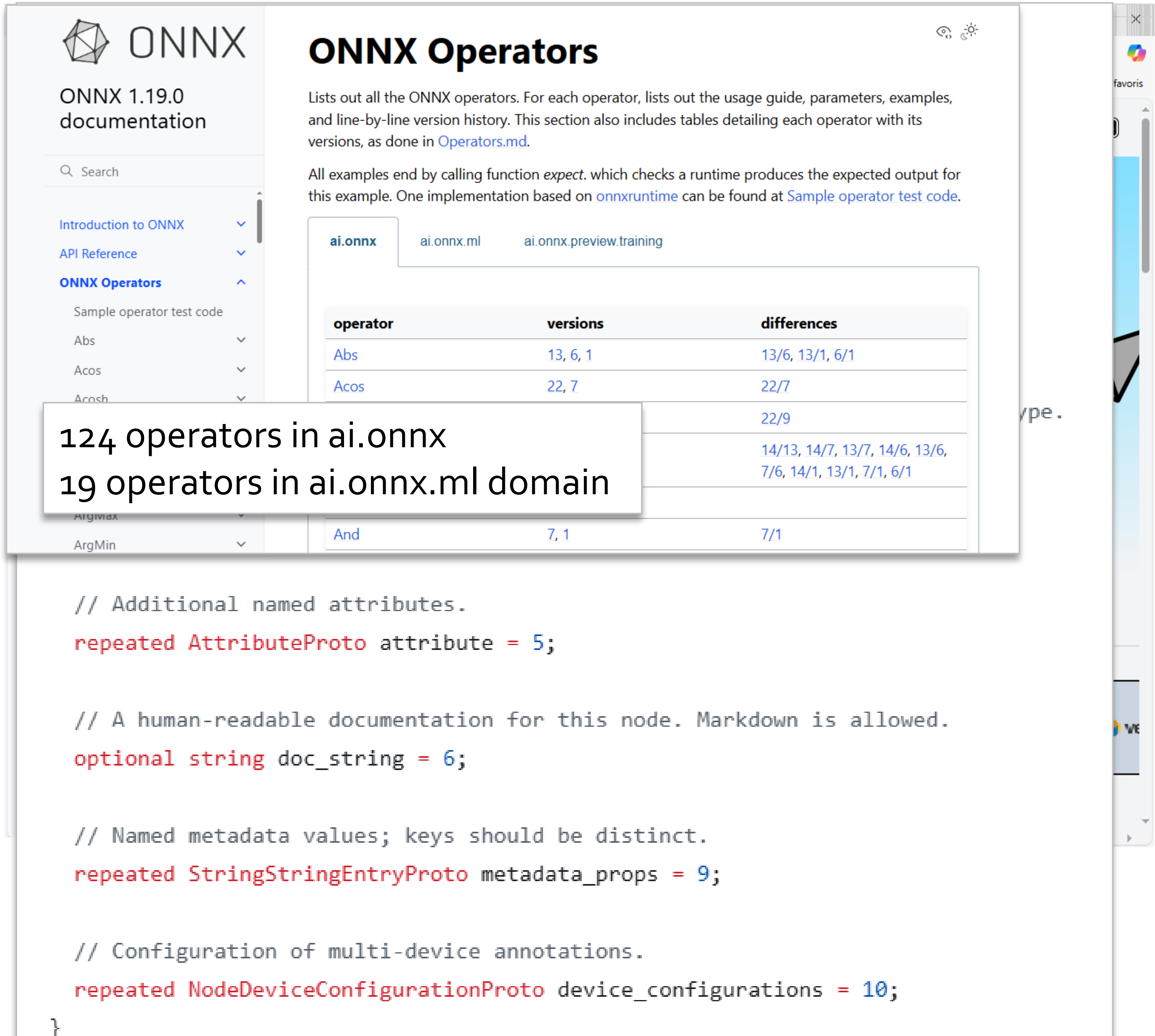
Why ONNX?

- Are there other candidate “standards“?
  - Vendor-neutral standards
    - Neural Network Exchange Format (NNEF) from Khronos Group (https://www.khronos.org/nnef)
      - Pretty good, but not really supported by tool vendors…
    - PMML
      - Not for deep neural networks
  - Non vendor-neutral standard
    - TensorFlow saved model
    - Torchscript
    - Core ML
    - Etc.

    By definition: not cross-platform…
- ONNX is supported by a large set of tools (see https://onnx.ai/supported-tools.html)
  - First meeting with ONNX in 2023/07/13

What is ONNX?

- A set of operators
- An API
- An Intermediate Representation (IR) described using Protobuf
- A runtime (ONNXruntime) [managed as a separate project in [ONNX Runtime | Home]



Who is ONNX?

- Supported by major companies in ML, SW, and HW

SONNX

Meetings and attendance

- 15 bi-weekly meetings (see minutes at https://github.com/ericjenn/working-groups/blob/ericjenn-srpwg-wg1/safety-related-profile/meetings/minutes.md)
- 1 workshop on formal methods
- Actual participation
  - Between 6-15 people per meeting

CEA, INRIA, IRT Saint-Exupery, ISAE SupAero, ONERA, TUM

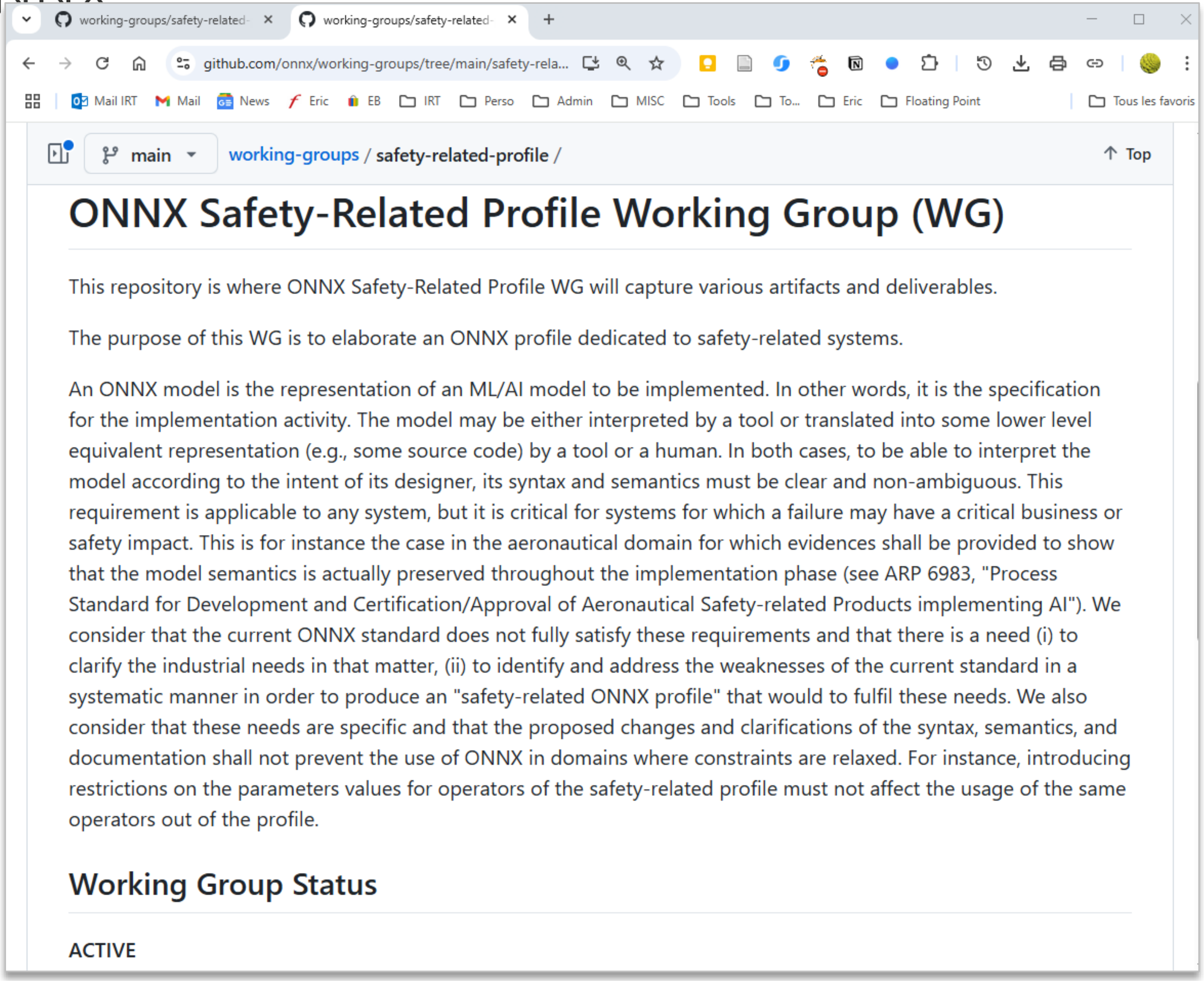
- **Aeronautics** : Airbus Helicopter, Airbus Operations, Airbus Protect, Embraer, Safran Electronics and Defense, THALES AVS, THALES Research and Technologies, DGA-TA
- **Space** : Airbus Defence and Space
- **Automotive** : Bosch, Ampere
- **Naval**: Naval Group
- **Industry** : Trumpf, Crosscontrol
- **Energy**: ARCYS
- **Other**: SopraSteria, Mathworks, Infineon, ANSYS (discussion)

ONNX

SONNX
Work area

https://github.com/onnx/working-groups/tree/main/safety-related-profile

(latest) https://github.com/ericjenn/working-groups/tree/ericjenn-srpwg-wg1/safety-related-profile


main working-groups / safety-related-profile / Top

ONNX Safety-Related Profile Working Group (WG)

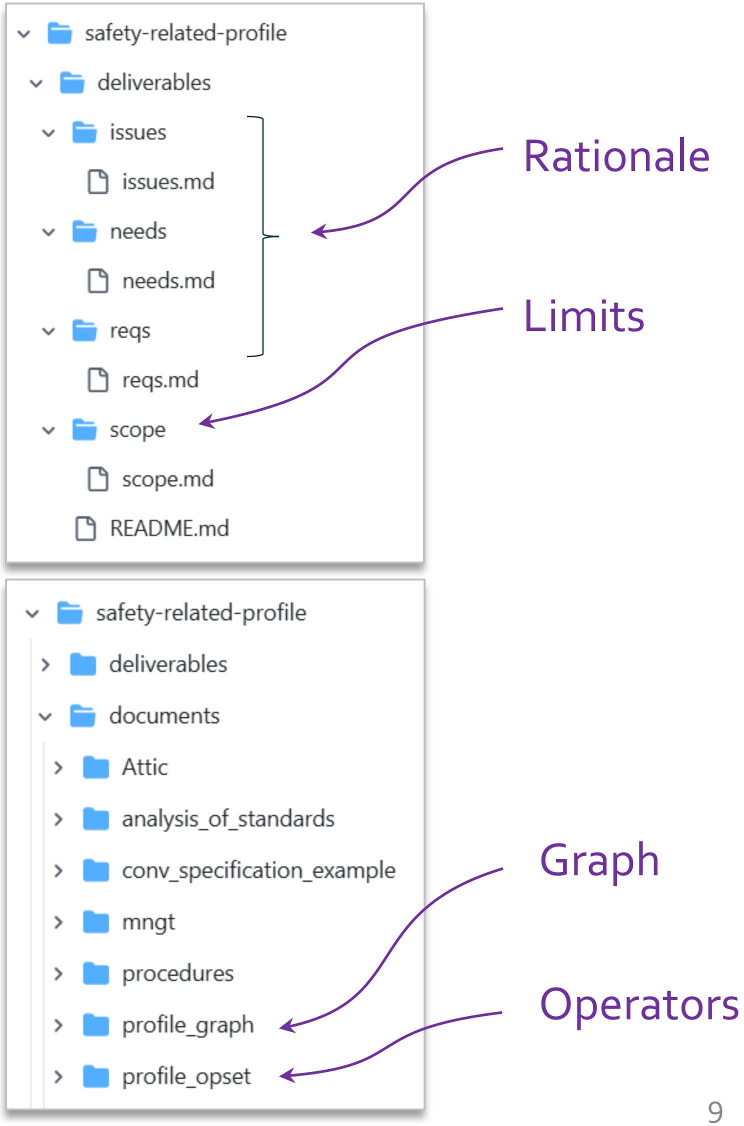
This repository is where ONNX Safety-Related Profile WG will capture various artifacts and deliverables.

The purpose of this WG is to elaborate an ONNX profile dedicated to safety-related systems.

An ONNX model is the representation of an ML/AI model to be implemented. In other words, it is the specification for the implementation activity. The model may be either interpreted by a tool or translated into some lower level equivalent representation (e.g., some source code) by a tool or a human. In both cases, to be able to interpret the model according to the intent of its designer, its syntax and semantics must be clear and non-ambiguous. This requirement is applicable to any system, but it is critical for systems for which a failure may have a critical business or safety impact. This is for instance the case in the aeronautical domain for which evidences shall be provided to show that the model semantics is actually preserved throughout the implementation phase (see ARP 6983, "Process Standard for Development and Certification/Approval of Aeronautical Safety-related Products implementing AI"). We consider that the current ONNX standard does not fully satisfy these requirements and that there is a need (i) to clarify the industrial needs in that matter, (ii) to identify and address the weaknesses of the current standard in a systematic manner in order to produce an "safety-related ONNX profile" that would to fulfil these needs. We also consider that these needs are specific and that the proposed changes and clarifications of the syntax, semantics, and documentation shall not prevent the use of ONNX in domains where constraints are relaxed. For instance, introducing restrictions on the parameters values for operators of the safety-related profile must not affect the usage of the same operators out of the profile.

Working Group Status

ACTIVE



So

our objective is to extend/improve ONNX…

but

is there anything to improve?

ONNX “issues”

ONNX failed conversion survey

- Are there empirical evidences of incompleteness, inconsistencies, etc.?
- Converters fail…
  - See Wenxin Jiang, Arav Tewari, et al, Interoperability in Deep Learning: A User Survey and Failure Analysis of ONNX Model Converters, Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 1466–1478,Vien 2024
- … often with the bad mode…
- … but no root cause leading to the spec…

Finding 4. Location: Most failures are in *Node Conversion* (74%).
Finding 5. Symptom: The most common symptoms in DL model converters are *Crash* (56%) and *Wrong Model* (33%).
Finding 6. Causes: *Crashes* are largely due to *Incompatibilities* and *Type Problems*. *Wrong models* are largely due to *Type Problems* and *Algorithmic Errors*.

ONNX “issues”

ONNX github issues

- See discussion https://github.com/onnx/onnx/issues/3651
- See ssues labelled **topic: spec clarification**

Describe the bug

When noop_with_empty_axes == 1 & axes is empty, in ONNX spec, it will return input tensor directly.
But in reference in onnx, it is mismatch. it returned np.square of the input tensor

```
Xavier Dupré, 13个月前 | 2 authors (Xavier Dupré and others)
class ReduceSumSquare_18(OpRunReduceNumpy):
    def _run(self, data, axes=None, keepdims=1, noop_with_empty_axes=0):  # type: ignore
        if self.is_axes_empty(axes) and noop_with_empty_axes != 0:  # type: ignore      liqun Fu, 17
            return (np.square(data),)

        axes = self.handle_axes(axes)
        keepdims = keepdims != 0  # type: ignore
```

This is complicated. Agree that there is a mismatch, but is the bug in the specification or implementation?

My personal interpretation is that this is a bug in the specification, not implementation, for the following reason: the attributes serve to define the set of axes being reduced: specifically, it is a flag to allow the empty list to indicate that all axes must be reduced (or that no axes must be reduced). Now, even if zero axes are reduced, it makes sense to compute the square. ReduceSumSquare is not actually a reduction-op: it is a reduction-op Sum applied to the square of the input.

Note: as of 2025, this issue has been corrected.

ONNX “issues”

ONNX github issues

- Rounding and numerical precision
  - `Cast` operator rounding (#3876, #5004)
    - No mention to truncation or rounding…
  - `DequantizeLinear` (#6132)
    - y = (x - x_zero_point) * x_scale , with x and x_zero_point with the same dtype. What happens if x - x_zero_point is outside the rage of dtype?

  Differences between the (Python) reference implementation and onnxrutime

- Shape and broadcasting semantics
  - `LayerNormalization` broadcasting (#5666 [closed])
    - LayerNormalization operator spec did not define how the optional bias `B` and scale `Scale` inputs should be broadcast when their shapes differ from the input.
  - `Shape` operator axis clamping (#6862)
    - *The spec says "axes will be clamped to [0, r-1]" where r is rank, implying an exclusive upper bound for slicing. But because `Shape` returns [start:end) with exclusive end, the correct clamping range for `end` should be `[0, r]` (inclusive of r). The issue also observes the spec omits what happens if `start > end`*

ONNX “issues”
ONNX github issues

- Shape and broadcasting semantics (continued)
  - `Slice` (#2433) [closed]
    - The spec’s recommendation to use `INT_MAX` for an open-ended slice did not cover reverse slicing cases.
  - *r]` (inclusive of r). The issue also observes the spec omits what happens if `start > end`*

- Operator semantics
  - `RandomNormal`, `RandomUniform` (# 6408)
    - The operator mentions a see attribute, but doesn’t said anything about its behavior. If the operator is stateless, the same value will be generated each time it is called. If it is statefull, it’ll generate different values, but according to the same sequence. The onnxruntime and reference implementation behave differently.

- Input and output typing
  - Integer attribute range (#5281)
    - Constraint about values (negative or not) shall be specified…

ONNX "issues"

Laconic and lacunar documentation

Problem: *what is a convolution?*

Conv - 22 ↑ Back to top

Summary

The convolution operator consumes an input tensor and a filter, and computes the output.

(Excerpt of ONNX doc.)

In general, the up-scaled space has dimensions $(B, C, X_1, X_2, \ldots)$, the down-scaled space has shape $(B, c, x_1, x_2, \ldots)$, and the filter has dimensions $(c, C, f_1, f_2, \ldots)$. The following equations will suppose two *spatial* dimensions, but generalization to more dimensions is straightforward.

In case of the `conv` operation, for each batch index $b \in [0..B)$ and for each $k_2 \in [0..c)$, the output is calculated as:

$$\text{output}[b][k_2][i_1][i_2] = \sum_{k_1=0}^{C-1} \sum_{j_1=0}^{f_1-1} \sum_{j_2=0}^{f_2-1} \widetilde{\text{input}}[b][k_1][i_1 \cdot s_1 + j_1 \cdot d_1 - p_1][i_2 \cdot s_2 + j_2 \cdot d_2 - p_2] \cdot \text{filter}[k_2][k_1][j_1][j_2]$$

(Excerpt of NNEF doc.)

Problem

- What is the value used for of padding in a convolution?
  - 0?

ONNX “issues”

Opset resolution, naming ambiguity

Problem

- An ONNX Function is a design artefact used to:
    1. define a composition of operators (ex: Relu Function is defined through Max Operator)
    2. define a composition of Nodes in the Graph as a reusable sub-graph (local function)
- Opsets are referenced in the Model element, and in each Function definition.
- Ex : Model import Opset v14, Model local function Relu import Opset v15.

- The Opset resolution is not specified:

// The (domain, name, overload) tuple must be unique across the function protos in this list.
// In case of any conflicts the behavior (whether the model local functions are given higher priority,
// or standard operator sets are given higher priotity or this is treated as error) is defined by
// the runtimes.

ONNX "issues"

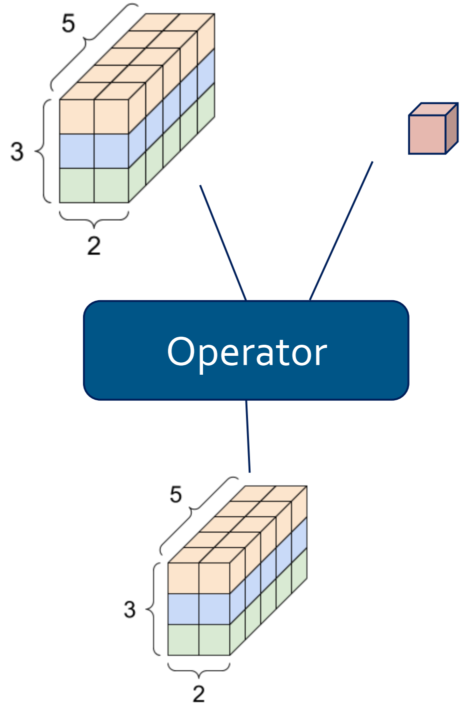
Polymorphism

Problem 1: data types in function parameters

- Function tensor input and output data type and shape are not specified
- To avoid type inference in the implementation, do we need to particularize the semantics for any concrete data type ?

Problem 2: shape broadcasting

- Operator Tensor input shall be of the same element type and shape.
- unless tensor shape broadcasting is enabled.
- Broadcasting logic shall be non ambiguous if supported by safety profile.



17

ONNX “issues”
Overloading

Problem:

- IR version 10 introduces the overloading capability, i.e. to have several definitions for the same function, and select them using a new ‘overloading’ field.

➔ Proposal: The overloading logic shall be reviewed (non ambiguous ?).

ONNX "issues"

Dynamic (Node input) vs static (Node attribute)

Problem:

The semantics is not clear that Node input is dynamic and Node attribute is static.

As attributes can take Tensor values, these values might come from other Nodes (constant or not)

The ONNX trend follows pytorch : more dynamic capabilities.

E.g.: https://onnx.ai/onnx/operators/onnx__Dropout.html, the ratio attribute in opset 10 was moved to input in recent opsets.

➔ Proposal: Do we follow the trend or do we restrict ? Consequence : compatibility.

Challenges for ONNX

Provide an accurate and precise description of the ML model leaving no room to interpretation and approximations…

- Complete the definition and documentation of
  - The operator semantics
  - The graph semantics

  for all datatypes
  - The ONNX abstract (metamodel) and concrete (format) syntax

In what order are the operators of a graph executed?

Compliance with dataflow constraints. Sufficient?

ONNX runtime

- Default execution order uses Graph::ReverseDFS() to generated topological sort
- Priority-based execution order uses Graph::KahnsTopologicalSort with per-node priority

Challenges for ONNX

Provide an accurate and precise description of the ML model leaving no room to interpretation and approximations…

- Complete the definition and documentation of
  - The operator semantics
  - The graph semantics
  - The ONNX abstract (metamodel) and concrete (format) syntax
- *Also consider other features to…*
  - Facilitate traceability
  - Improve understandability
  - Etc.

For instance…

Use doc string to
- enforce the documentation of the meaning of each dimensions of tensors…
- add traceability data

ONNX "issues"

Default values

Problem:

The semantics is not clear that Node input is dynamic and Node attribute is static.

As attributes can take Tensor values, these values might come from other Nodes (constant or not)

The ONNX trend follows pytorch : more dynamic capabilities.

E.g.: https://onnx.ai/onnx/operators/onnx__Dropout.html, the ratio attribute in opset 10 was moved to input in recent opsets.

➔ Proposal: Do we follow the trend or do we restrict ? Consequence : compatibility.

Deliverables

Status

(D1.a) Safety-related Profile **Scope** Definition (2024/11/01)

(D1.b.x) End users **needs** for domain x (2024/12/01)

(D1.c) **Consolidated needs** for all industrial domains (2025/01/01)

(D2.a) ONNX safety-related Profile **requirements** (2025/02/01)

(D3.a) ONNX Safety-related profile - proof of concept (2024/12/01)

(D3.b) ONNX Safety-related profile – graph (2025/05/01)

(D3.c) ONNX Safety-related profile – operators (2025/12/31)

(D3.d) ONNX Safety-related profile – format (2025/12/31)

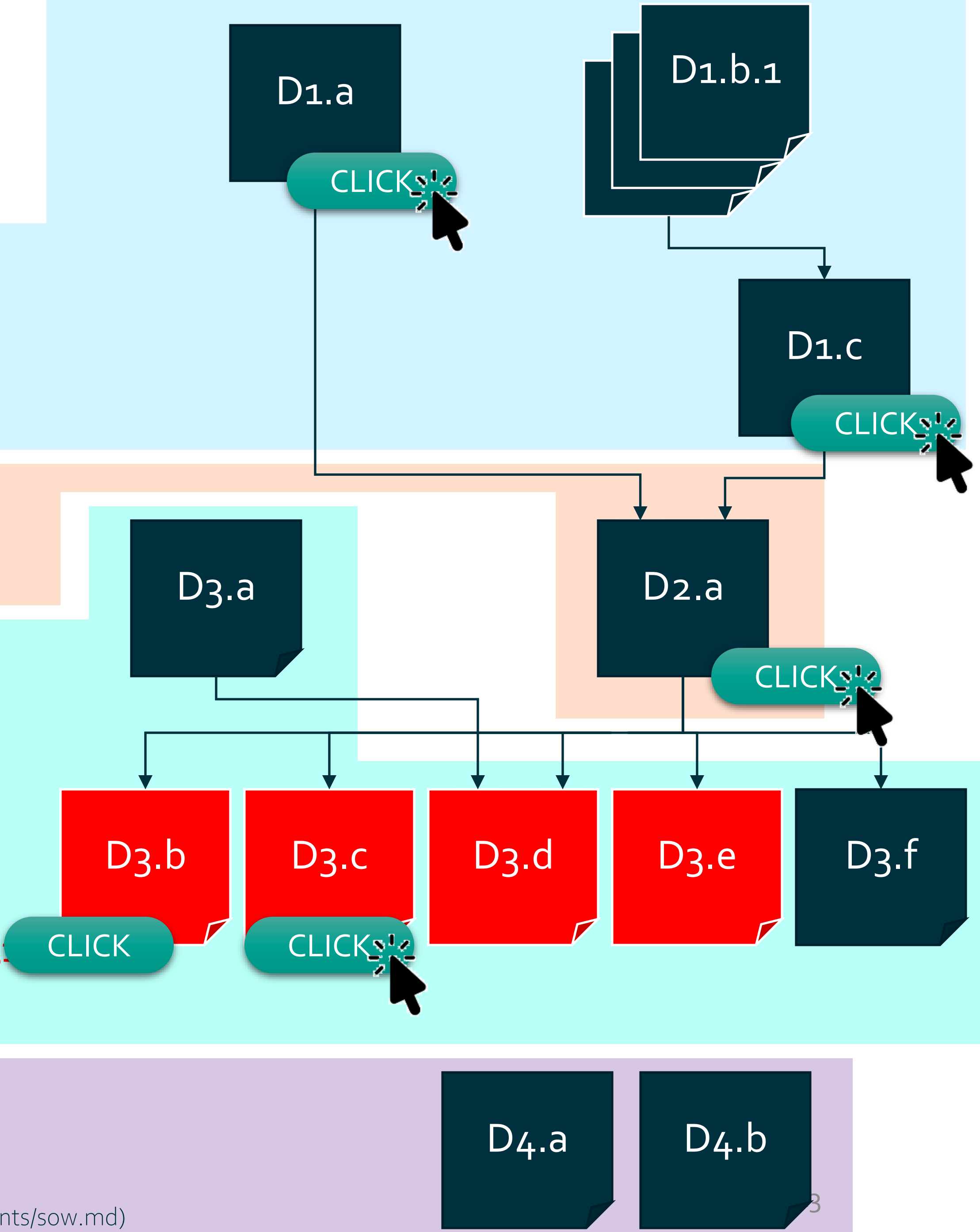
(D3.e) ONNX Safety-related profile reference implementation (2025/12/3

(D3.f) ONNX Safety-related profile rules (2025/01/31)

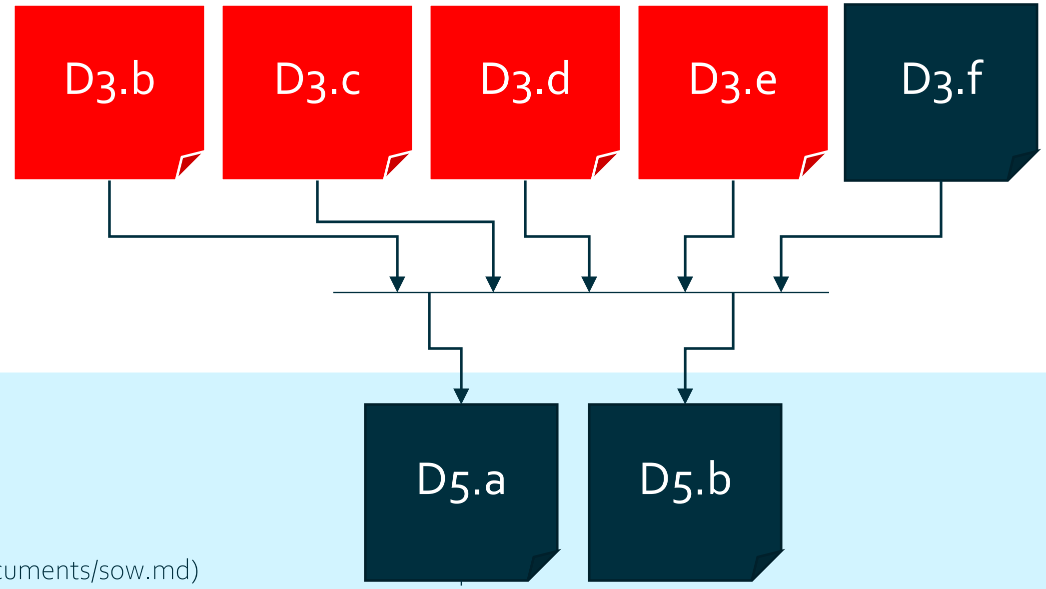
(D4.a) ONNX Safety-related profile **verification** report

(D4.b) ONNX Safety-related profile **validation** report

(detailed WP is available at https://github.com/ericjenn/working-groups/blob/main/safety-related-profile/documents/sow.md)



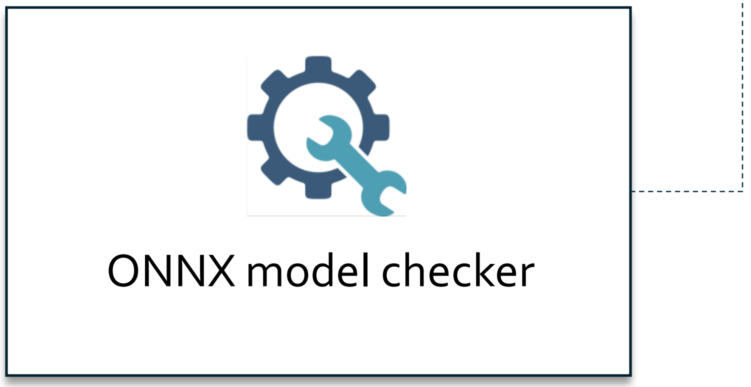
Deliverables
Status



(D5.a) Expression of the **needs** / tool list (2025/01/31)

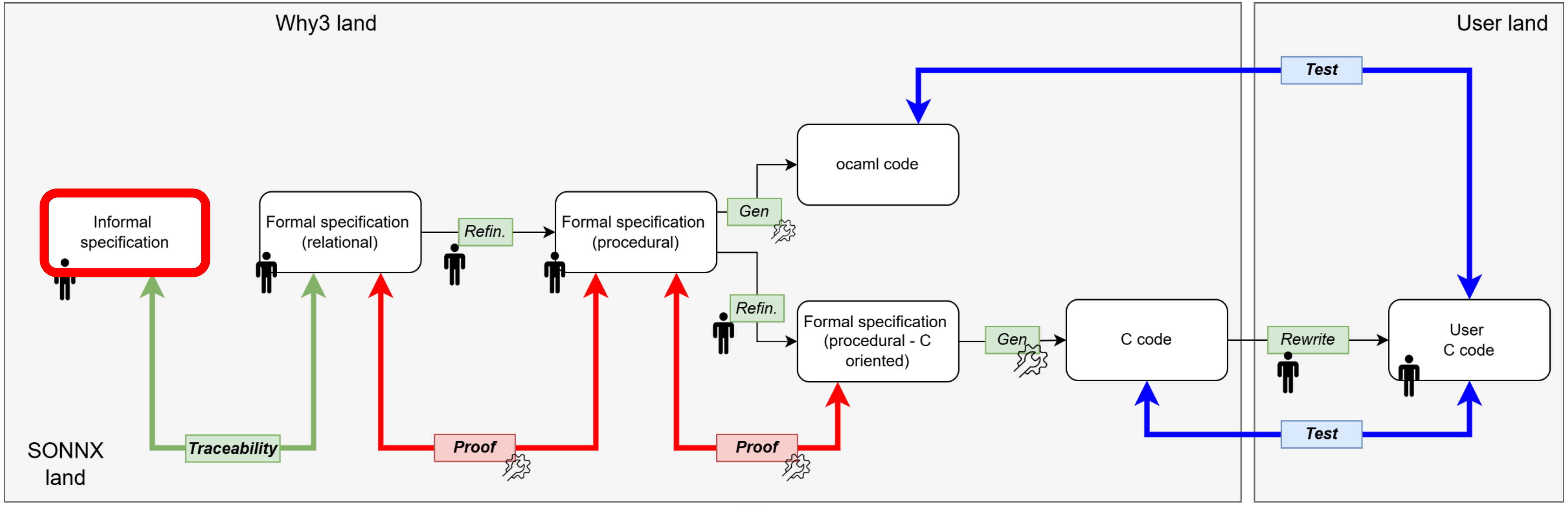
(D5.b) **Requirements** of tool <tool>(2025/12/31)

(detailed WP is available at https://github.com/ericjenn/working-groups/blob/main/safety-related-profile/documents/sow.md)



Deliverables

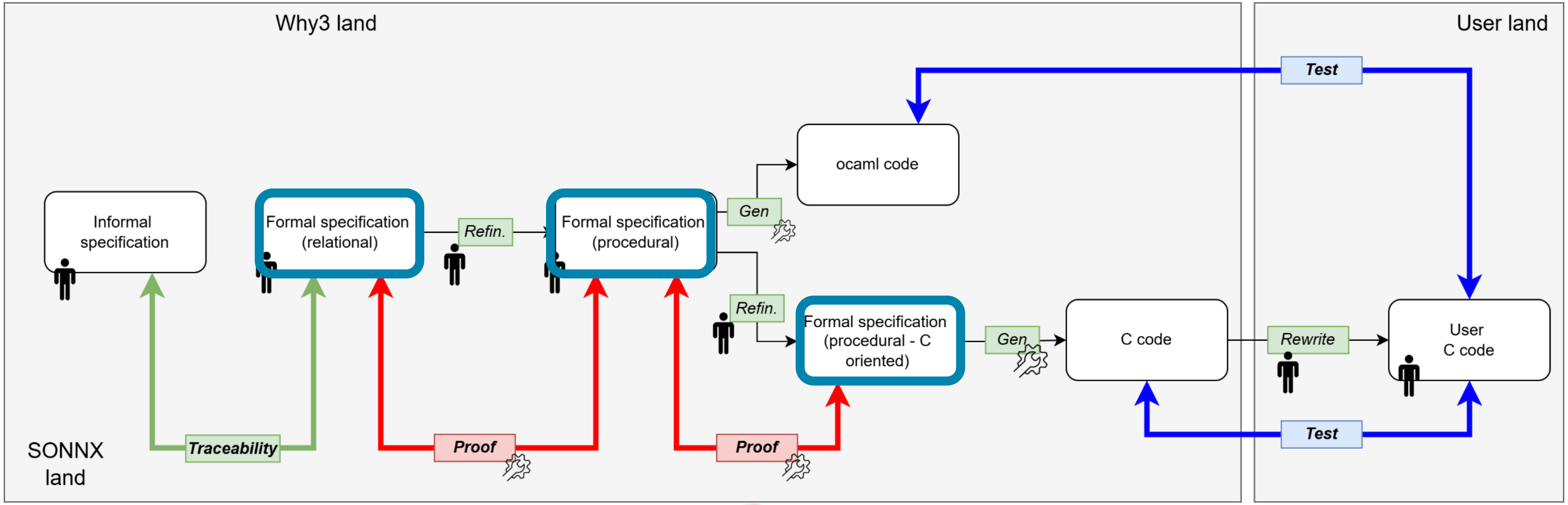
Overview



Informal specification:

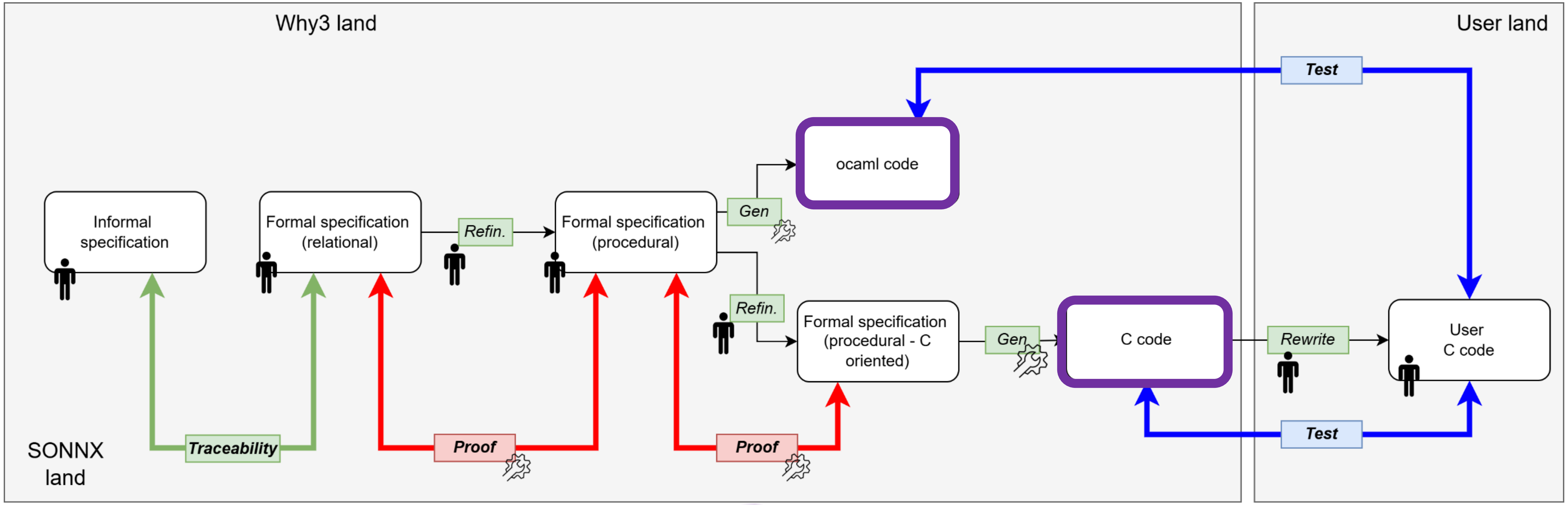
Deliverables
Overview

traceable



- Formal specification
  - Relational
  - Operational
    - Generic
    - C-bounded

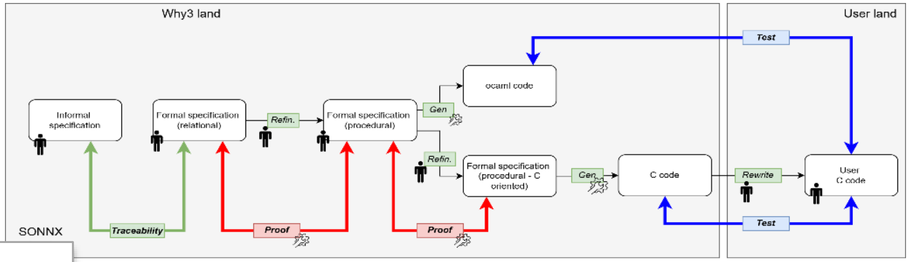
Deliverables
Overview

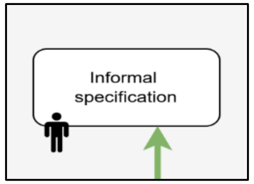


- Reference implementation
  - Interim caml code
  - Final C code

Deliverables

For informal to formal specification: the **conv** operator





Attributes

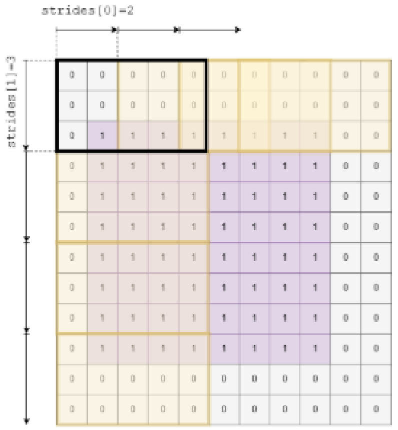
`strides` : list of int

Attribute `strides` determines how the kernel is applied on tensor `X` during the convolution.

For instance, with stride[0] = 2 and stride[1] = 3, the kernel is applied to data 2 units on right in the first spatial axis and to data 3 units down in the second spatial axis at each step of the convolution.

> The previous sentence is not clear...

The effect of the `strides` attribute is illustrated on the following figure. In this example, `strides` =(2,3).



strides=(2,3)

Constraints

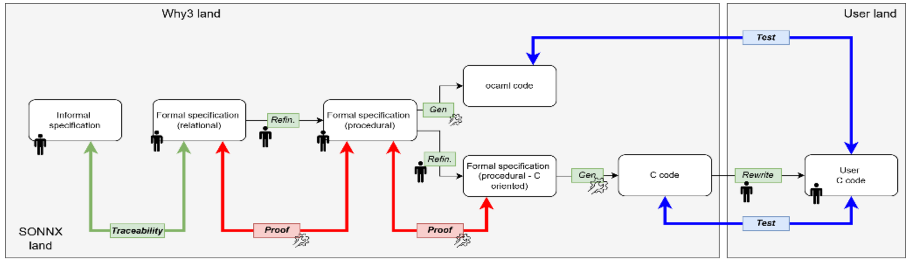
- (C1) Value domain
  - Statement: `strides` is a list of strictly positive integers.
  - Rationale: Stride values are used in the denominator of expression in [constraint (C3) of X]
- (C2) Consistency between the shape of tensors `X`, `W`, `Y` and attributes `pads`, `dilations` and `strides`.
  - Statement: [See constraint (C3) of X]

xis.

, `DATA_CHANNEL` ,

Deliverables

For informal to formal specification: the **conv** operator

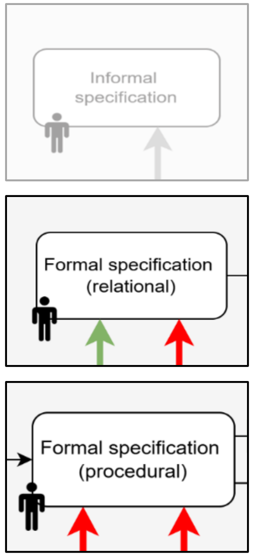



```
module Tensor
  use int.Int
  use map.Map
  use utils.Product
  use sequence.Seq

  type shape = { dims : seq int }
    invariant { forall i. 0 <= i < length dims -> 0 < dims[i] }
    meta coercion function dims

  function sizeof (s : shape) : int = product 0 (length s) (fun i -> s[i])

  val sizeof (s : shape) : int
    ensures { result = sizeof s }

  type index = seq int

  predicate valid (idx : index) (s : shape) =
    length idx = length s /\
    forall i. 0 <= i < length s -> 0 <= idx[i] < s[i]

   type tensor 'a = {
     shape : shape ;
     value : map index 'a ;
   }

   meta coercion function value

   function dim (t : tensor 'a) : int = length t.shape

end
```


```
type shape = { dims : seq int }
  invariant { forall i. 0 <= i < length dims -> 0 < dims[i] }
  meta coercion function dims
```

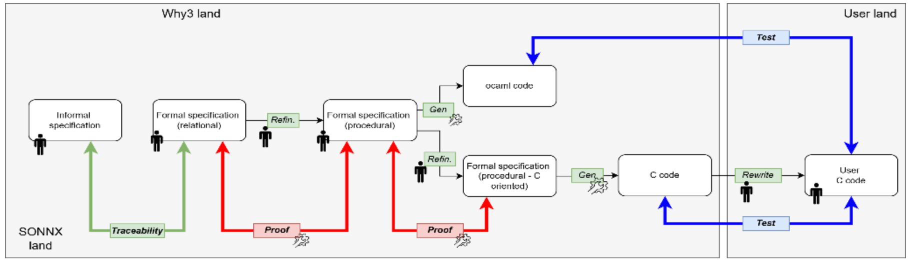

XXXXX


```
predicate valid (idx : index) (s : shape) =
  length idx = length s /\
  forall i. 0 <= i < length s -> 0 <= idx[i] < s[i]
```

Deliverables

For informal to formal specification: the **conv** operator

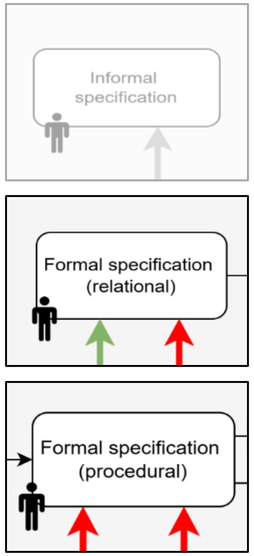



```
let function conv2d_int (x: tensor int) (w: tensor int) (b: option (tensor int))
                        (strides pads dilations: seq int)
                        (group_val: int)
                        (auto_pad_is_not_set: bool)
                        : tensor int

  (* --- Core Tensor Dimension Requirements --- *)
  requires { dim x = 4 /\ dim w = 4 }
  requires { Ops4D.c_dim x = Ops4D.c_dim w }
  requires { Ops4D.c_dim x > 0 }
  requires { Ops4D.h_dim w > 0 /\ Ops4D.w_dim w > 0 }
  requires { Ops4D.n_dim w > 0 }
  requires { Ops4D.n_dim x > 0 }

  (* --- Attribute Sequence Length Requirements --- *)
  requires { Seq.length strides = 2 }
  requires { Seq.length pads = 4 }
  requires { Seq.length dilations = 2 }

  (* --- Attribute Value Domain Requirements --- *)
  requires { Ops4D.stride_h strides > 0 /\ Ops4D.stride_w strides > 0 }
  requires { Ops4D.pad_h_begin pads >= 0 /\ Ops4D.pad_w_begin pads >= 0 /\
             Ops4D.pad_h_end pads >= 0 /\ Ops4D.pad_w_end pads >= 0 }
  requires { Ops4D.dilation_h dilations > 0 /\ Ops4D.dilation_w dilations > 0 }

  (* --- ONNX Profile Restrictions --- *)
  requires { group_val = 1 }
  requires { auto_pad_is_not_set }
```


```
(* --- Core Tensor Dimension Requirements --- *)
requires { dim x = 4 /\ dim w = 4 }
requires { Ops4D.c_dim x = Ops4D.c_dim w }
requires { Ops4D.c_dim x > 0 }
requires { Ops4D.h_dim w > 0 /\ Ops4D.w_dim w > 0 }
requires { Ops4D.n_dim w > 0 }
requires { Ops4D.n_dim x > 0 }
```

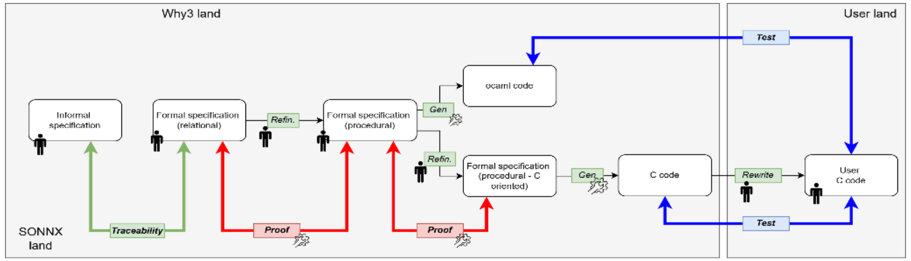
```
(* --- Attribute Sequence Length Requirements --- *)
requires { Seq.length strides = 2 }
requires { Seq.length pads = 4 }
requires { Seq.length dilations = 2 }
```

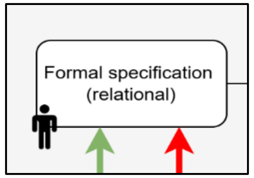
```
(* --- ONNX Profile Restrictions --- *)
requires { group_val = 1 }
requires { auto_pad_is_not_set }
```

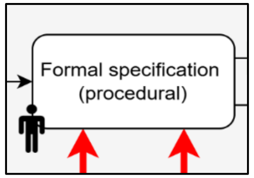



Deliverables

For informal to formal specification: the **conv** operator







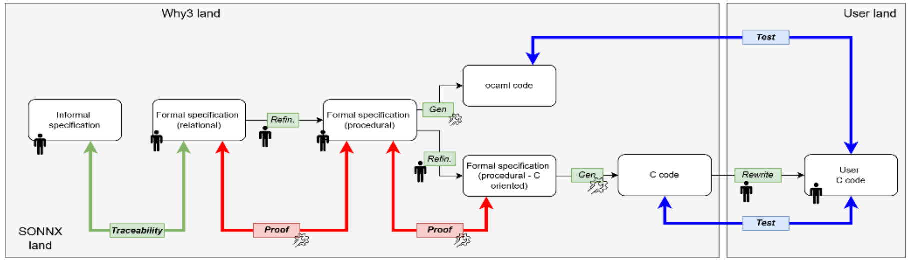

```
  requires {
           let h_out_calc = calculate_H_out (Ops4D.h_dim x) (Ops4D.h_dim w)
                                            (Ops4D.pad_h_begin pads) (Ops4D.pad_h_end pads)
                                            (Ops4D.dilation_h dilations) (Ops4D.stride_h strides) in
           let w_out_calc = calculate_W_out (Ops4D.w_dim x) (Ops4D.w_dim w)
                                            (Ops4D.pad_w_begin pads) (Ops4D.pad_w_end pads)
                                            (Ops4D.dilation_w dilations) (Ops4D.stride_w strides) in
           h_out_calc > 0 /\ w_out_calc > 0
         }
=
  let res_shape = conv2d_output_shape x w strides pads dilations in
  let res_value_func = conv2d_output_value x w b strides pads dilations res_shape in
  { shape = res_shape; value = res_value_func }
```

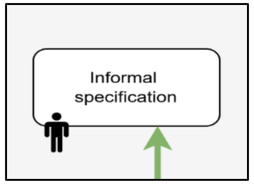
The shape

The value

Deliverables

For informal to formal specification: the **concat** operator





Let a be the concatenation axis and $d_{k,a}$ **(T2)** the dimension of the X_k input tensor k along the axis a.

Let s_k be the cumulative offset along axis before input X_k as:

$$\text{T2: } s_k = \sum_{j=0}^{k-1} d_{j,a}$$

Let i_a be the global index along dimension a, and let i'_a be the corresponding local within a local tensor X_k. This relationship can be defined as follows:

$$\text{T3: } i'_a = i_a - s_k$$

If the global index i_a satisfies the condition:

$$\text{T4: } s_k \leq i_a < s_k + d_{k,a}$$

then the relationship holds:

$$\text{T5: } \forall i_0, \ \dots, i_{r-1} .\ Y[i_0, \ \dots, i_{r-1}] = X_k[i_0, \ \dots, i'_a, \ \dots, i_{r-1}]$$

With i_0 and i_{r-1} are the indices which access respectively the first and last dimensi **r-dimensional** tensor. $i_0, \ \dots, i_{r-1}$ represent a set of indices that uniquely identify a element within an **r-dimensional** tensor.

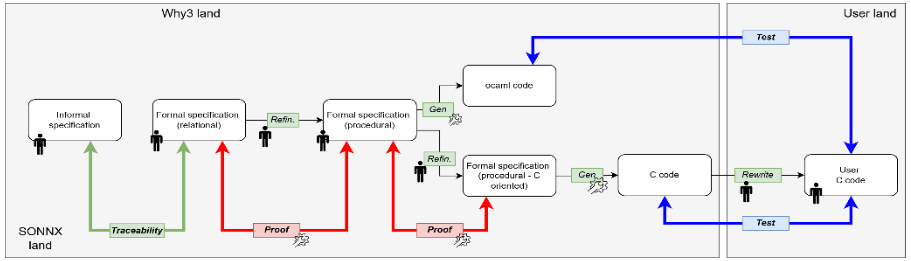
```
(** T3: defining the local index i' for a i (global index) given **)
(*
  seq_D_axis: seq_D_axis is the sequence of all the d_k,axis for each k tensor
  i_axis: value on the axis along which the concatenation is performed
  k: Current index in sequence seq_D_axis. Must be 0 at the first call of the
  function
  s_k: Current sum of the previous dimensions in seq_D_axis. Must be 0 at the
  first call of the function
*)
let rec rec_find_k_and_i_prime (seq_D_axis: seq int) (i_axis: int) (k: int)
(s_k: int) : (int, int)
 variant { length seq_D_axis - k }  (* Termination measure *)
=
  if i_axis < s_k + seq_D_axis[k] then (* Inequality (T5) to define to in which
  k tensor the local index i' is defined *)
    (k, (i_axis - s_k)) (* Definition (T5) by keeping the upper part of the
    inequality: i' = i - s_k *)
  else
    (* The global index i is superior to s_k + seq_D_axis[k] so the k tensor to
    define the local index i' is not the current one but next one k+1. Update
    the offset (s_k) by adding the length of the tensor we just checked
    (current tensor). *)
    rec_find_k_and_i_prime seq_D_axis i_axis (k + 1) (s_k + seq_D_axis[k])
```

```
considered **)
```

The concat operator

Deliverables

Test oracle: the `where operator`

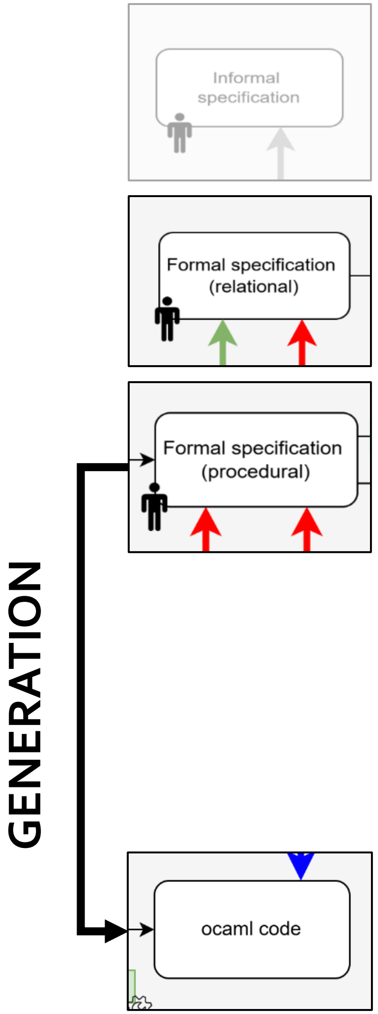



```
module Where
  use int.Int
  use map.Map
  use utils.Same
  use tensor.Shape
  use tensor.Tensor

  let function where (cond : tensor bool) (a b : tensor 'a) : tensor 'a =
  {
    shape = same cond.shape (same a.shape b.shape) ;
    value = fun i -> if cond.value[i] then a.value[i] else b.value[i] ;
  }

end
```

where.mlw (specification)

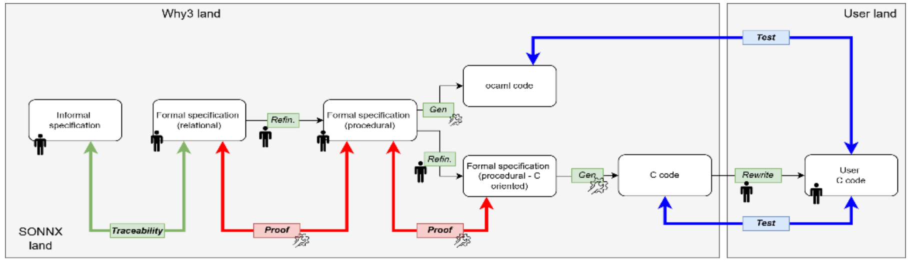
```
let where :
  type a. ((bool) Tensor__Tensor.tensor) -> (a Tensor__Tensor.tensor) ->
          (a Tensor__Tensor.tensor) ->  (a Tensor__Tensor.tensor) =
  fun cond a b -> { Tensor__Tensor.shape =
                     ((ignore ((ignore (b.Tensor__Tensor.shape) ; (a.Tensor__Tensor.shape))) ;
                     (cond.Tensor__Tensor.shape))); Tensor__Tensor.value =
                     (fun (i: (int) list) ->
                        if ((cond.Tensor__Tensor.value) i)
                        then ((a.Tensor__Tensor.value) i)
                        else ((b.Tensor__Tensor.value) i)) }
```

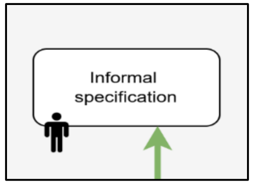
opwhere_Where.ml (implementation)



Deliverables

Other cases: the `graph`





Graph

- `[T01a]` A graph contains a set of nodes
- `[T01b]` A graph contains a set of tensors that are inputs and outputs of the nodes
  - Some of those tensors are inputs (resp. outputs) of the graph, i.e, their values are set (resp. returned) before (resp. after) executing the graph

Nodes

- `[T03a]` A node refers to an operator
  - An operator may be referred to by multiple nodes
- `[T03b]` There is a 1-to-1 mapping between the set of inputs and outputs of a node and the set of inputs and outputs of its associated operator `[R1]`.
  - Note that is is a restriction with respect to the ONNX standard that allows fewer inputs or outputs when the omitted input or output is optional.

Tensors

- `[T02b]` A tensor is an object that can hold a value or be uninitialized
- `[T02a]` A tensor is identified by a unique identifier within a graph

Operators

- `[T04a]` An operator specifies a relation (a function) between a set of input parameters and a set of outputs parameters.
  - Input and output parameters (resp. output) are free variables that can be bound to tensors using nodes
  - An operator has at least one output

Execution Semantics

- `[T05a]` A node is executable if all its input tensors are initialized
- `[T05b]` Executing a node means assigning values to output tensors such that the inputs-outputs relation specified by the operator holds
- `[T05c]` All executable nodes are executed
- `[T05d]` An executable node is executed only once
- `[T05e]` A tensor is assigned at most once (Single Assignment)

Deliverables
Other cases: the **`graph`**

Graph

- `[T01a]` A graph contains a set of nodes
- `[T01b]` A graph contains a set of tensors that are inputs and outputs of the nodes
  - Some of those tensors are inputs (resp. outputs) of the graph, i.e, their values are set (resp. returned) before (resp. after) executing the graph

Nodes

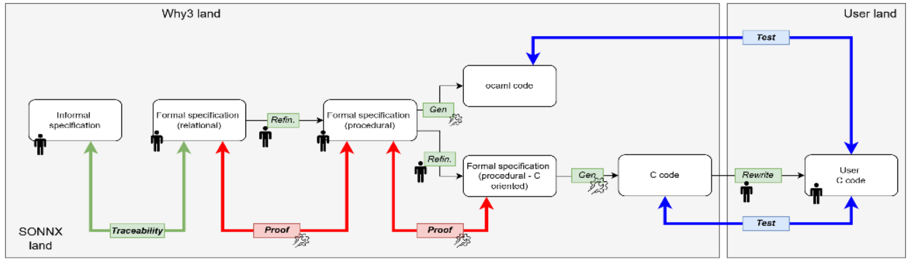
- `[T03a]` A node refers to an operator
  - An operator may be referred to by multiple nodes
- `[T03b]` There is a 1-to-1 mapping between the set of inputs and outputs of a node and the set of inputs and outputs of its associated operator `[R1]`.
  - Note that is is a restriction with respect to the ONNX standard that allows fewer inputs or outputs when the omitted input or output is optional.

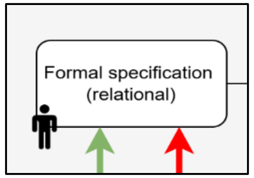
Tensors

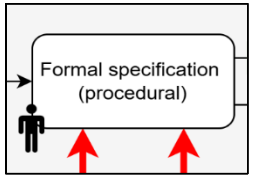
- `[T02b]` A tensor is an object that can hold a value or be uninitialized
- `[T02a]` A tensor is identified by a unique identifier within a graph

Operators

- `[T04a]` An operator specifies a relation (a function) between a set of input parameters and a set of outputs parameters.
  - Input and output parameters (resp. output) are free variables that can be bound to tensors using nodes
  - An operator has at least one output

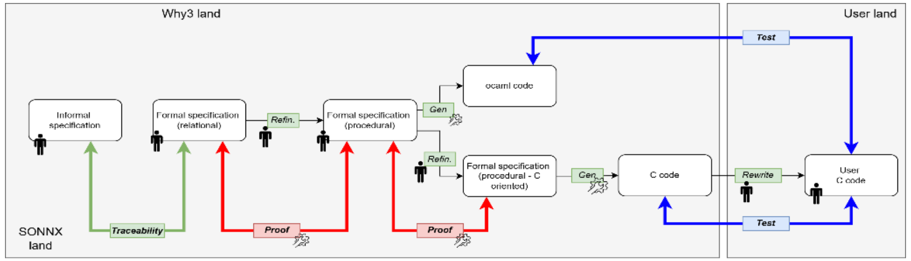


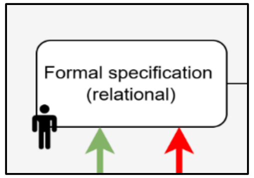


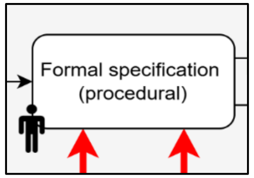


Deliverables

Other cases: the `graph`








```
let exec_graph (s: graph_state) (g: graph) : graph_state
    (* [TXX] The graph can only be executed if its inputs are initialized *)
    requires { forall t. Mem.mem t g.gi -> tensor_is_initialized s t }
     (* [TXX] After execution, all output tensors are initialized *)
    ensures { forall t. Mem.mem t g.go -> tensor_is_initialized result t  }
    =
        exec_nodes_until_completion s g.gn
```

```
let rec exec_nodes_until_completion (s: graph_state) (ns: list node) : graph_state =
    (* All outputs of the nodes are initialized *)
    ensures { forall n: node. Mem.mem n ns ->
                 forall t: tensor_id. Mem.mem t n.ou ->
                     tensor_is_initialized result t }
```

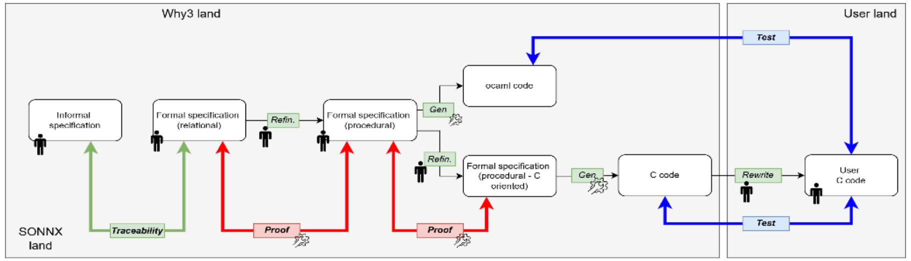
```
let rec exec_nodes (s: graph_state) (ns: list node) : graph_state =
    (* All nodes in the list are ready to be executed *)
    requires { forall n. Mem.mem n ns -> node_is_ready s n }
    (* All outputs of the nodes are initialized *)
    ensures { forall n : node. Mem.mem n ns ->
                 forall t: tensor_id. Mem.mem t n.ou ->
                    tensor_is_initialized result t }
```

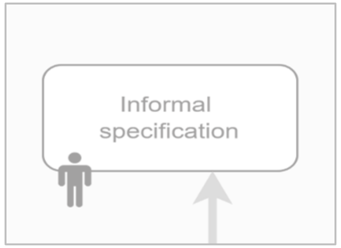
```
let exec_node (s: graph_state) (n: node) : graph_state
    (* [T05a] The node is ready to be executed *)
    requires { node_is_ready s n }
    (* [T05a] The number of inputs matches the number of operator's inputs *)
    requires { length n.oi = length n.ope.opi }
    (* [T03b] The number of outputs must match the number of operator's outputs *)
    requires { length n.ou = length n.ope.opo }
    (* After execution, all output tensors are set *)
    ensures  { forall t: tensor_id. Mem.mem t n.ou -> tensor_is_initialized result t }
=
    (* the values of tensors that are inputs to a node *)
    let inputs = apply (fun t -> my_map_get s t) n.oi in
        assert { forall v.  Mem.mem v inputs -> v <> None  };
    (* the values of all outputs after evaluation *)
    let outputs = exec_operator n.ope inputs in
        assert { forall v.  Mem.mem v outputs -> v <> None  };
        (* the updated state *)
        assign_list s (zip n.ou outputs)
```

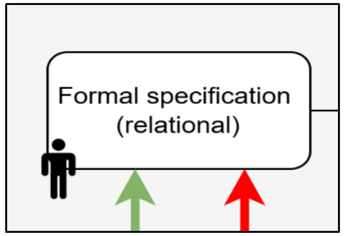
```
let exec_operator (op: operator) (inputs: list (option value)) : list (option value)
    (* The node provides as any iputs as needed by the operator *)
    requires { length inputs = length op.opi }
    (* All inputs are initialized before execution *)
    requires { forall i. Mem.mem i inputs -> i <> None }
    (* All outputs are initialized after execution*)
    ensures { forall i. Mem.mem i result -> i <> None }
    (* There is one value per output tensor *)
    ensures { length result = length op.opo }
=
    (* This is a dummy implementation that returns the appropriate number of values *)
    make_list (Some (any value)) (length op.opo)
```

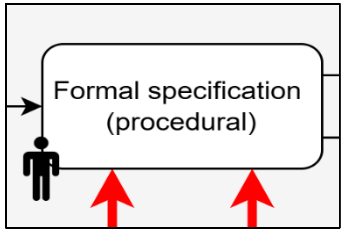
Deliverables

Other cases: the **graph**





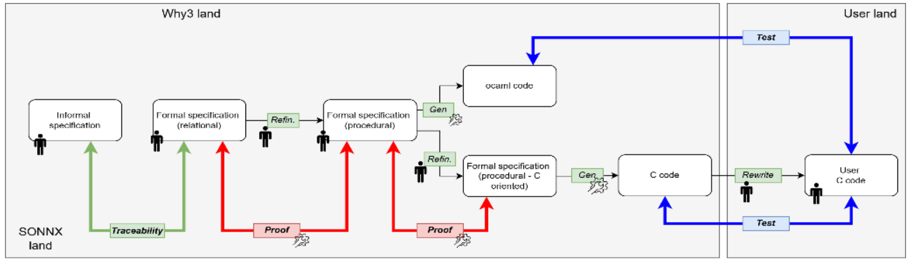


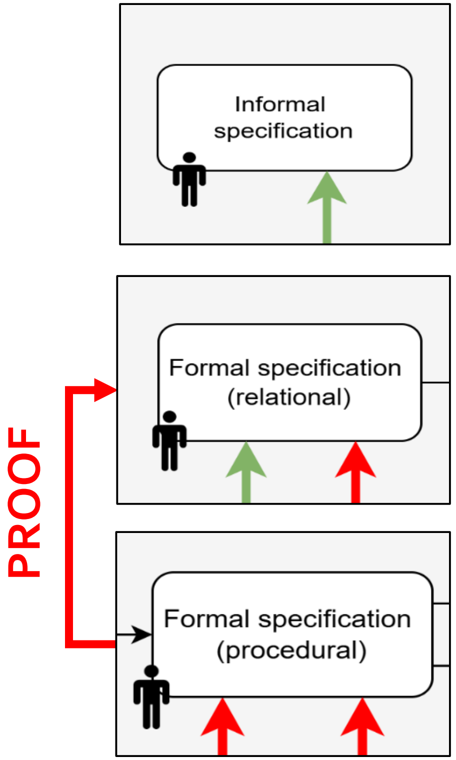


The graph

Deliverables

Other cases: the **`graph`**





Numerical errors
Approach

- Handling of numerical errors
  - **No specification of errors** (depend on method and computation error)
  - We shall not overspecify
  - We propose to specify
    - the means to evaluate errors

  - Provision of error estimation methods :
    - Empirical (incomplete)
    - Formal via abstract interpretation (e.g., fluctuat)
    - Formal via axiomatic proof

Numerical errors
Approach

- A1 : We propose a lower bound on the error (the smallest error) for any value in the input domain, for any implementation complying with IEEE 754
  - This is not necessarily the smallest error (this is actually an upper bound)so that
    - The effort to express the formula remain acceptable
    - The complexity of the formula remain tractable
    - The verification of the property remains achievable
- A2: For a restriction of the input domain, the error may be smaller
- A3: The SONNX reference implementation will (?) comply with this constraint
- A4: Actual, more efficient implementations may violate this constraint. In that case, the implementer has to provide its own express their own precision requirement, following the structure of the provided formula.
- A tool will be used to demonstrate that the accuracy constraint is satisfied

Numerical errors

Example: the **`add`** operator

Numerical Accuracy

If tensor A_{err} is the numerical error of `A`, tensor B_{err} is the numerical error of `B`, let us consider $C_{\mathrm{err}}^{\mathrm{propag}}$ the propagated error of `Add` and $C_{\mathrm{err}}^{\mathrm{intro}}$ the introduced error of `Add`. Hence the numerical error of `C`, $C_{\mathrm{err}} = C_{\mathrm{err}}^{\mathrm{propag}} + C_{\mathrm{err}}^{\mathrm{intro}}$.

Error propagation

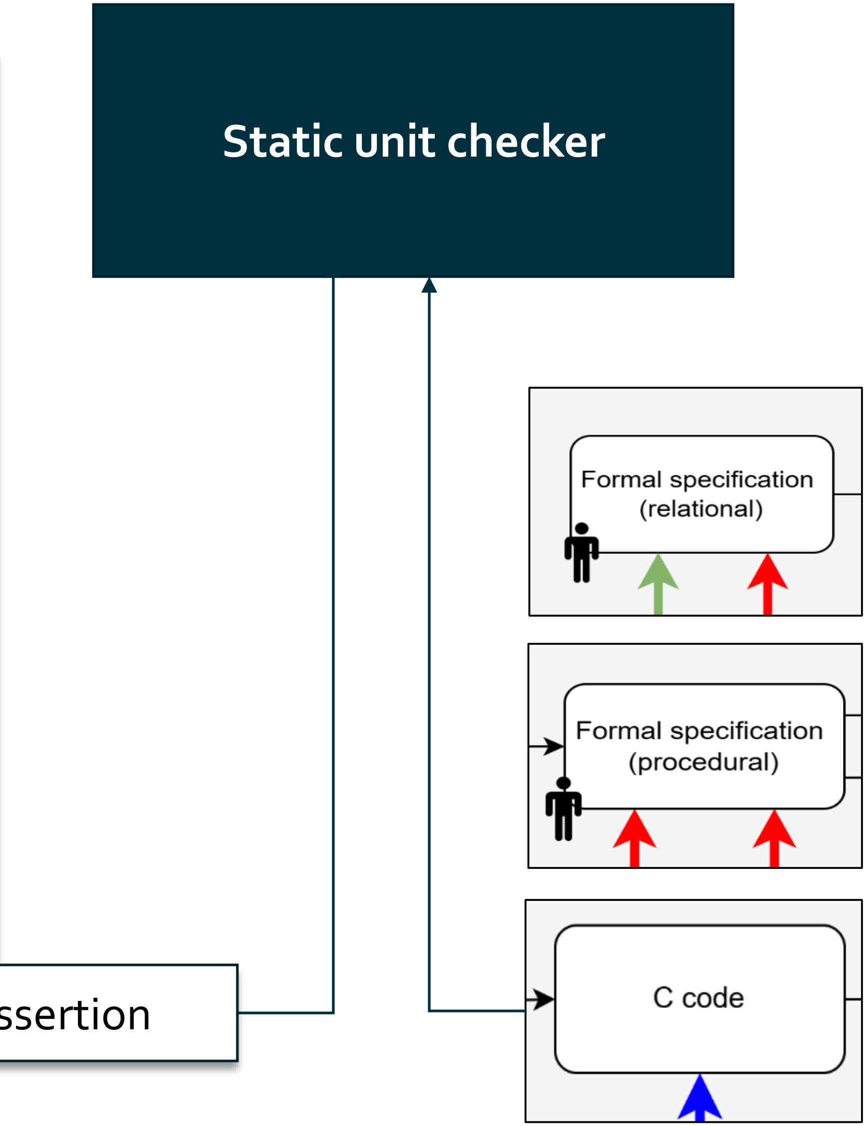
For every indexes $I = (i_0, i_1, \ldots, i_n)$ over the axes,

- $C_{\mathrm{err}}^{\mathrm{propag}}[I] = A_{\mathrm{err}}[I] + B_{\mathrm{err}}[I]$

Error introduction - floating-point IEEE-754 implementation

The error introduced by the `Add` operator shall be bound by the semi-ulp of the addition result for every tensor component for a normalized result. For a hardware providing m bits for floating-point mantissa, the semi-ulp of `1.0` is $2^{-(m+1)}$. Hence, for every indexes $I = (i_0, i_1, \ldots, i_n)$ over the axes,

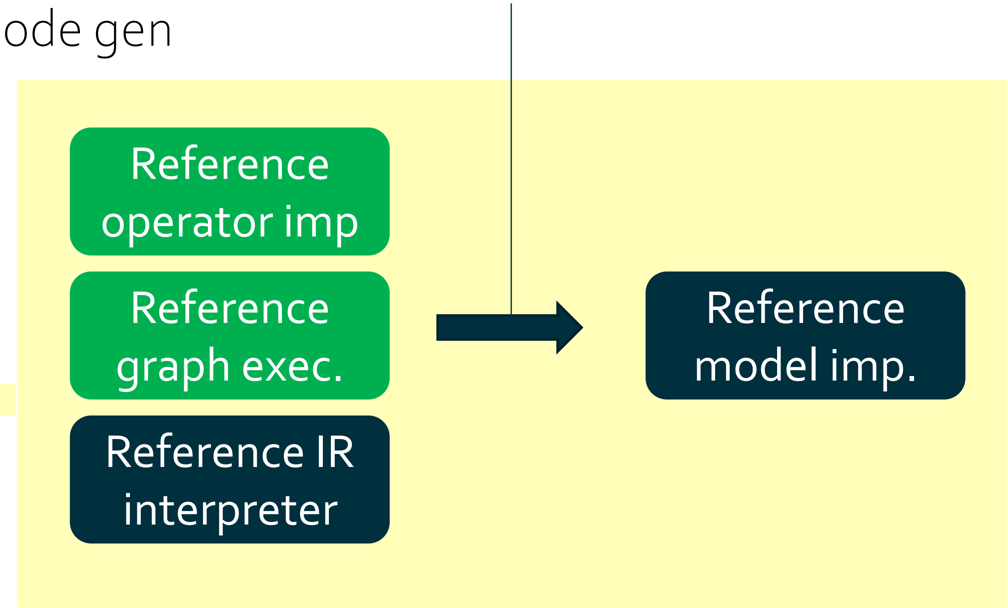
- $\left|C_{\mathrm{err}}^{\mathrm{intro}}[I]\right| \leq \max\left(\left|A[I] + B[I] + A_{\mathrm{err}}[I] + B_{\mathrm{err}}[I]\right| \times 2^{-(m+1)}, \frac{\text{denorm-min}}{2}\right)$
- $\left|C_{\mathrm{err}}^{\mathrm{intro}}[I]\right| \leq \max\left(\left|A_{\mathrm{float}}[I] + B_{\mathrm{float}}[I]\right| \times 2^{-(m+1)}, \frac{\text{denorm-min}}{2}\right)$
- $\left|C_{\mathrm{err}}^{\mathrm{intro}}[I]\right| \leq \max\left(\left|A[I] + B[I]\right| \times \frac{2^{-(m+1)}}{1 - 2^{-(m+1)}}, \frac{\text{denorm-min}}{2}\right)$



Conclusion
Where are we? What's next?

- **Where are we:**
  - First drafts to be consolidate / completed…

- **What's next:**
  - Completion of operator informal and formal spec + proof + code gen
  - Completion graph spec + proof + code gen
  - IR
  - Generation of C implementations
  - Integration to the AIdge platform



  - Actual integration in the ONNX ecosystem…

Contacts

- Eric JENN (eric.jenn@irt-saintexupery.com)
- Jean SOUYRIS (jean.souyris@airbus.com)

- To join the mailing list, send a message to:
  onnx-sonnx-workgroup+subscribe@lists.lfaidata.foundation