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

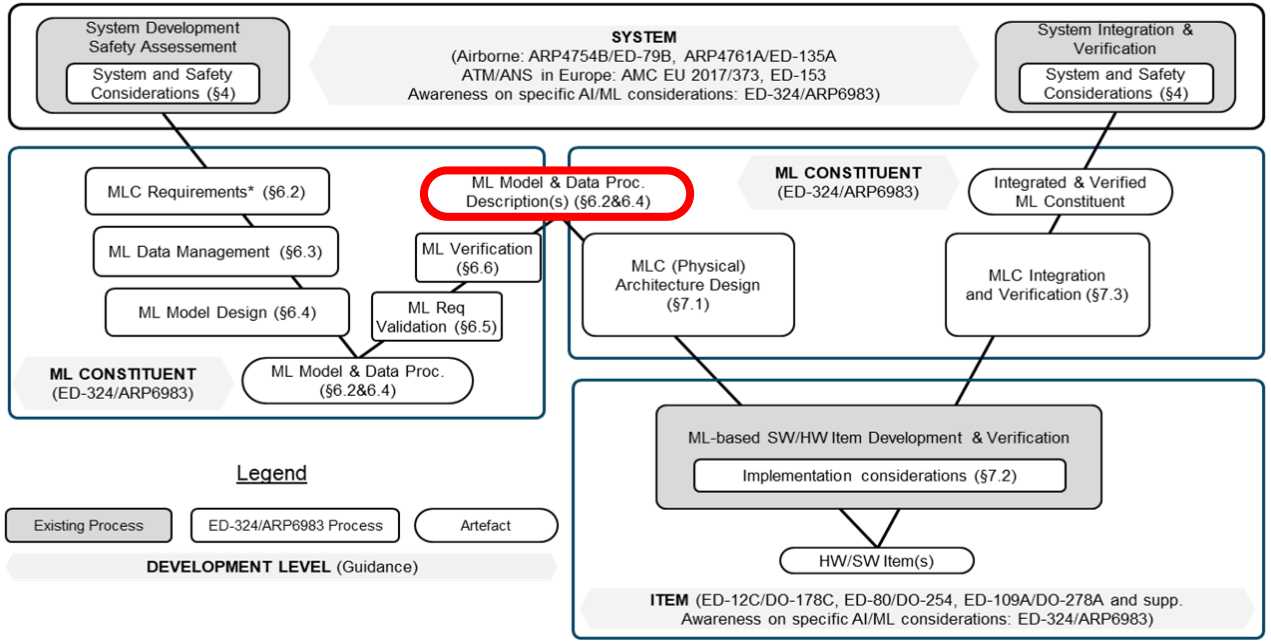
- Our objectives
- Our organization
- Some issues addressed by the WG
- Some results
- Next…

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

SAE INTERNATIONAL

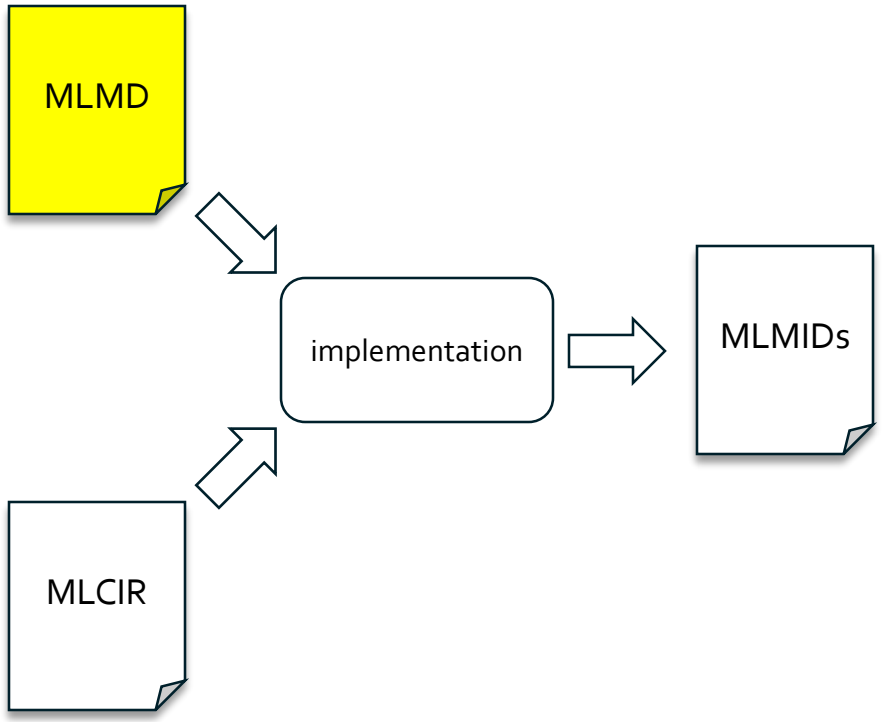
AEROSPACE RECOMMENDED PRACTICE	ARP6983™	REV. DRAFT 6-0
	Issued xxxx-xx Revised xxxx-xx Reaffirmed xxxx-xx Stabilized xxxx-xx Cancelled xxxx-xx Superseding XXXX	
Recommended Practice for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing ML Issue 1: Non-adaptive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

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			o	o	o	MLMD	7.4.7			①	①	①

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".



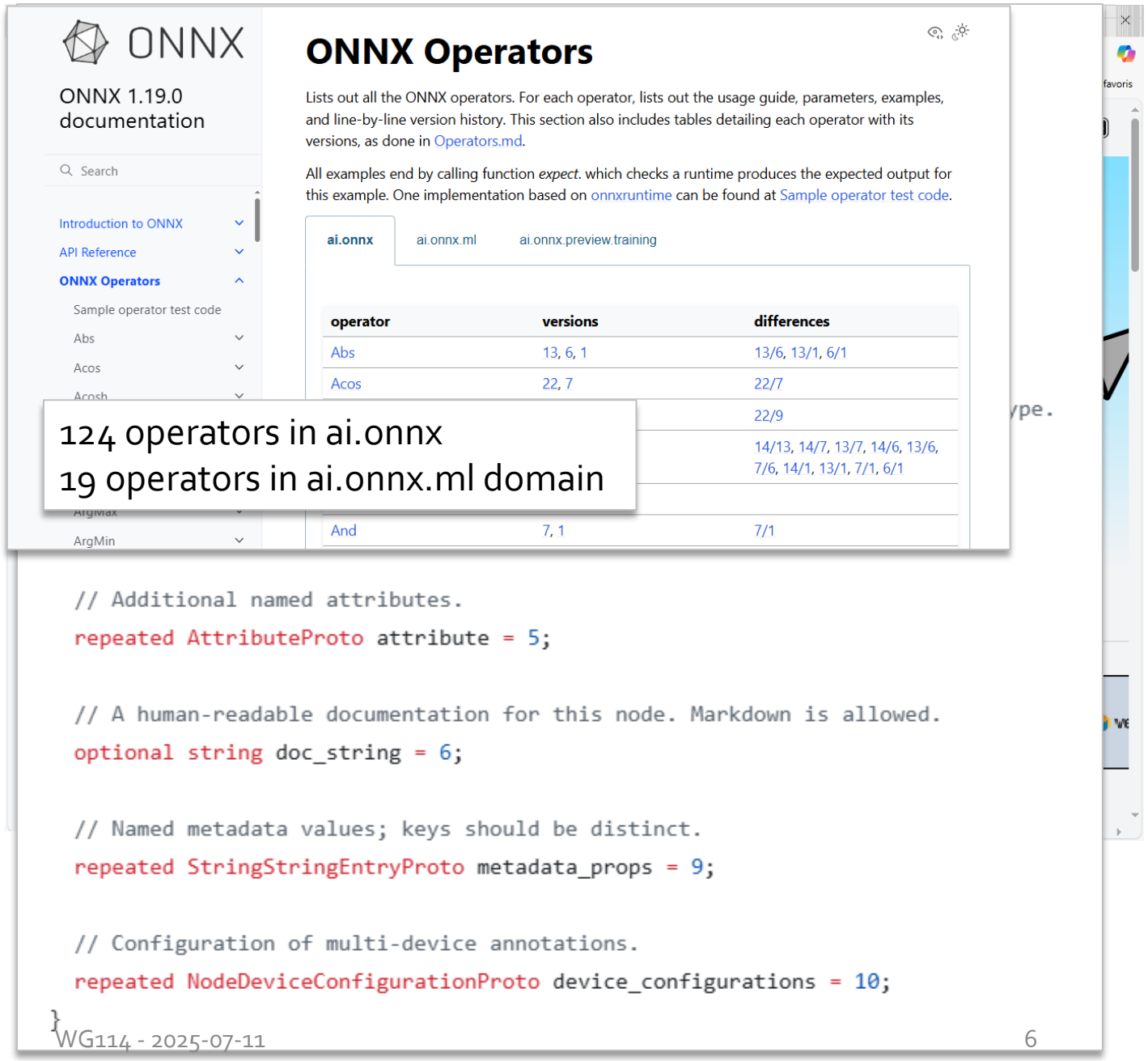
Why ONNX?

- 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)

What is ONNX?

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



SONNX

Meetings and attendance

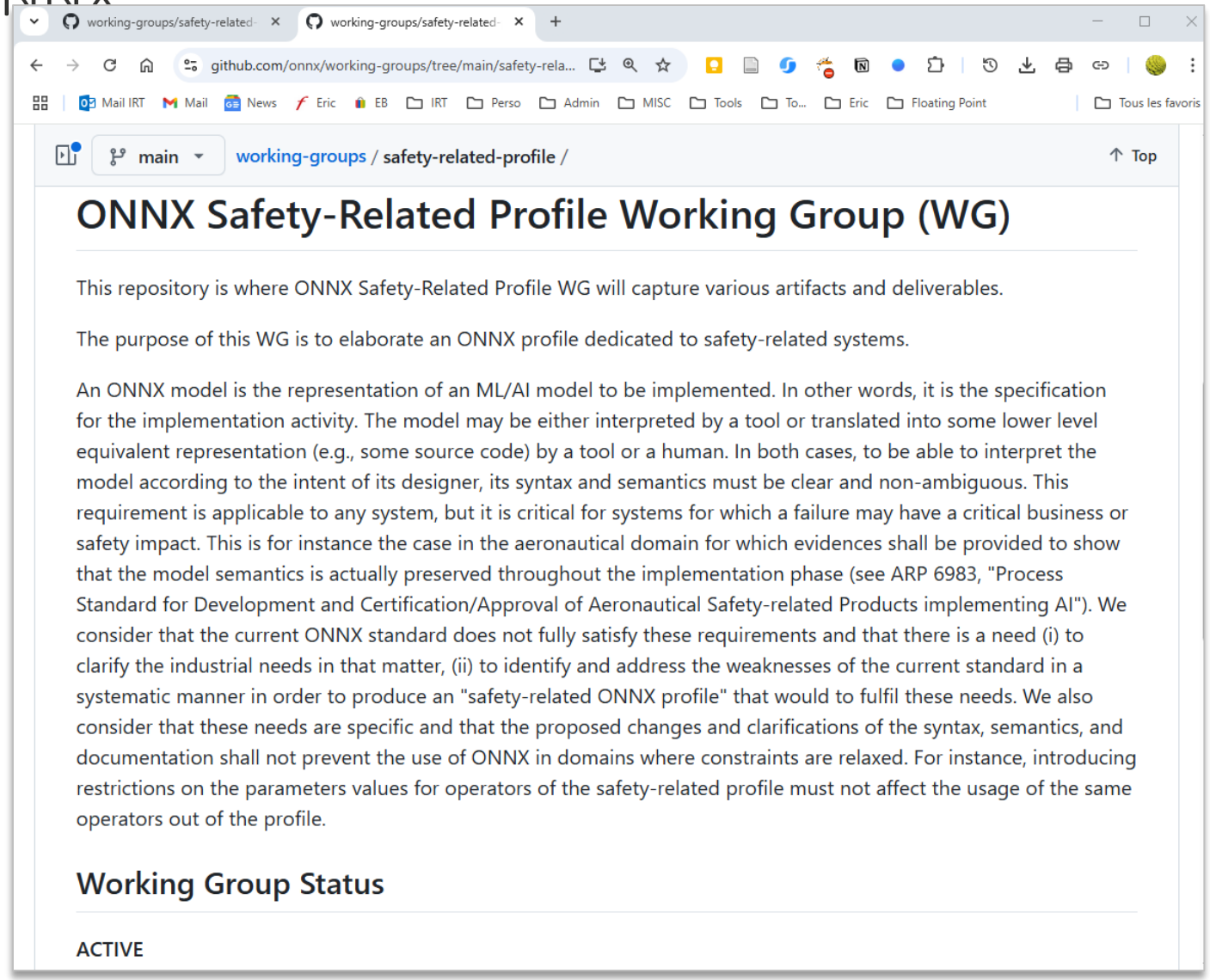
- 15 bi-weekly meetings (see minutes at https://github.com/ericjenn/working-groups/blob/ericjenn-srpwg-wg1/safety-related-profile/meetings/minutes.md)
- 2 workshops on formal methods
- Actual participation
  - Between 6-15 people per meeting

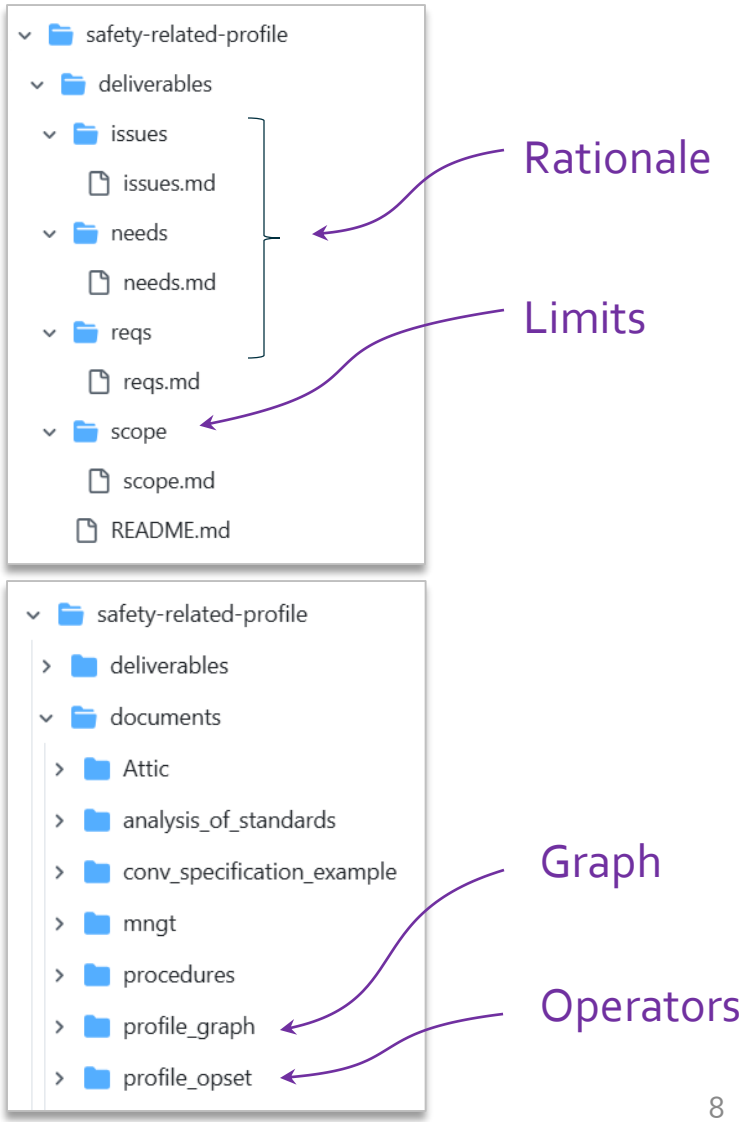
CEA, INRIA, IRT Saint-Exupery, ISAE SupAero, ONERA, TUM

- **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)

SONNX
Work area

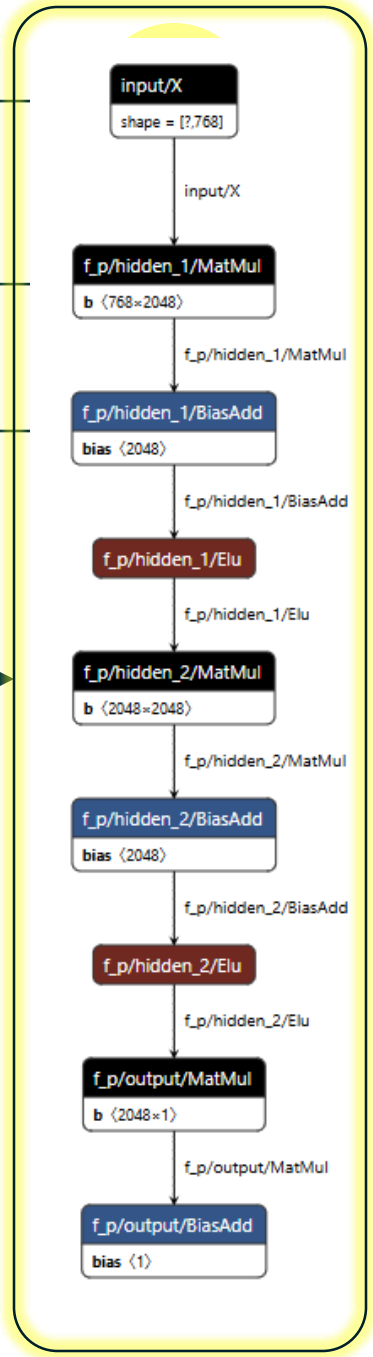
https://github.com/onnx/working-groups/tree/main/safety-related-profile
(latest) https://github.com/ericjenn/working-groups/tree/ericjenn-srpwg-wg1/safety-related-profile





Challenges for ONNX

- Provide an accurate description of the ML model, leaving no room to interpretation and approximations covering
  - The operator semantics (for all datatypes)
  - The graph semantics
  - The ONNX abstract (metamodel) and concrete (format) syntax



ONNX

Fine.

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 issues 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
            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 range of dtype?

The onnxrutime and reference implementation behave differently.

- Operator semantics
  - `RandomNormal`, `RandomUniform` (# 6408)
    - The operator mentions a seed 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 state full, it'll generate different values, but according to the same sequence.
    - The onnxruntime and reference implementation behave differently.

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.

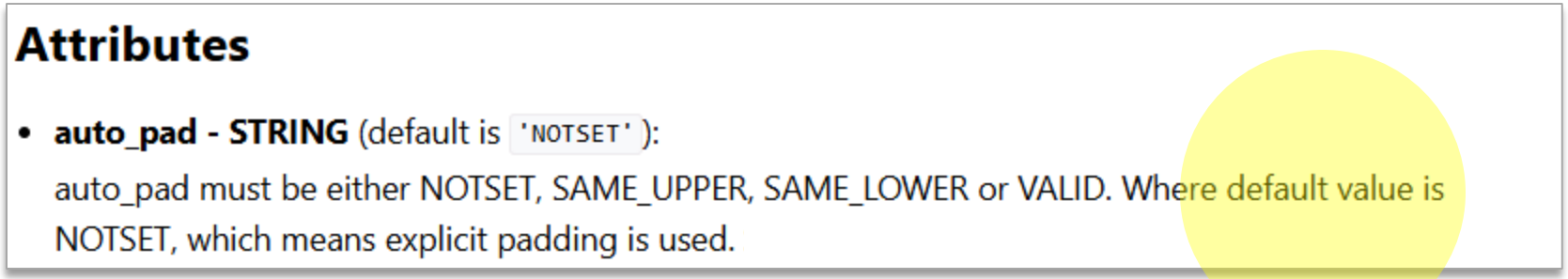
(Excerpt of ONNX doc.)

Problem: *What is the value used for padding in a convolution?*

- Uhhh… zero?

ONNX "issues"
Default values

Problem: ONNX operators use attributes that have default values



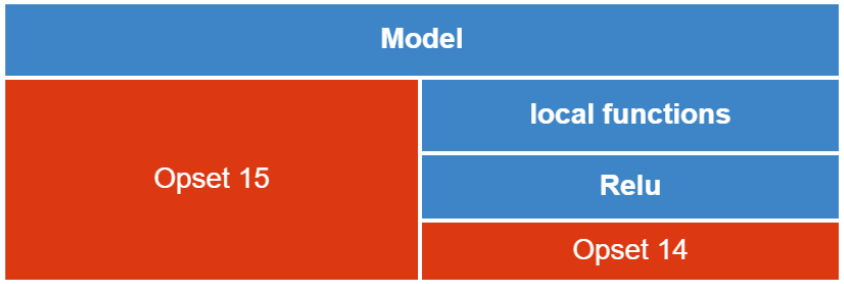
Conv operator

ONNX "issues"

Opset resolution, naming ambiguity

Problem: *ambiguity in opset resolution*

- 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 v15,
  Model local function Relu import Opset v14.



- 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.

From onnx/onnx/onnx-ml.proto at main · onnx/onnx · GitHub , line 498-501

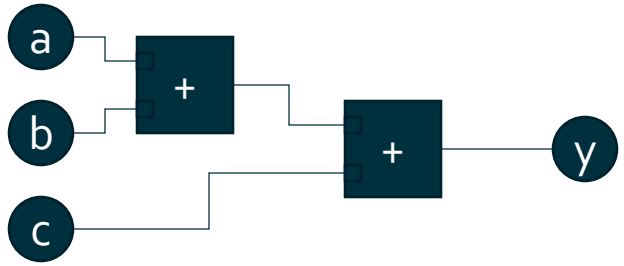
ONNX "issues"
Graph execution order

Problem: "ambiguity" in operator execution

- In what order are the operators of a graph executed?

$$y = a + b + c \stackrel{?}{=} (a + b) + c \stackrel{?}{=} a + (b + c)$$

Associativity is imposed by the graph



ONNX runtime

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

Other aspects

- 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

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)

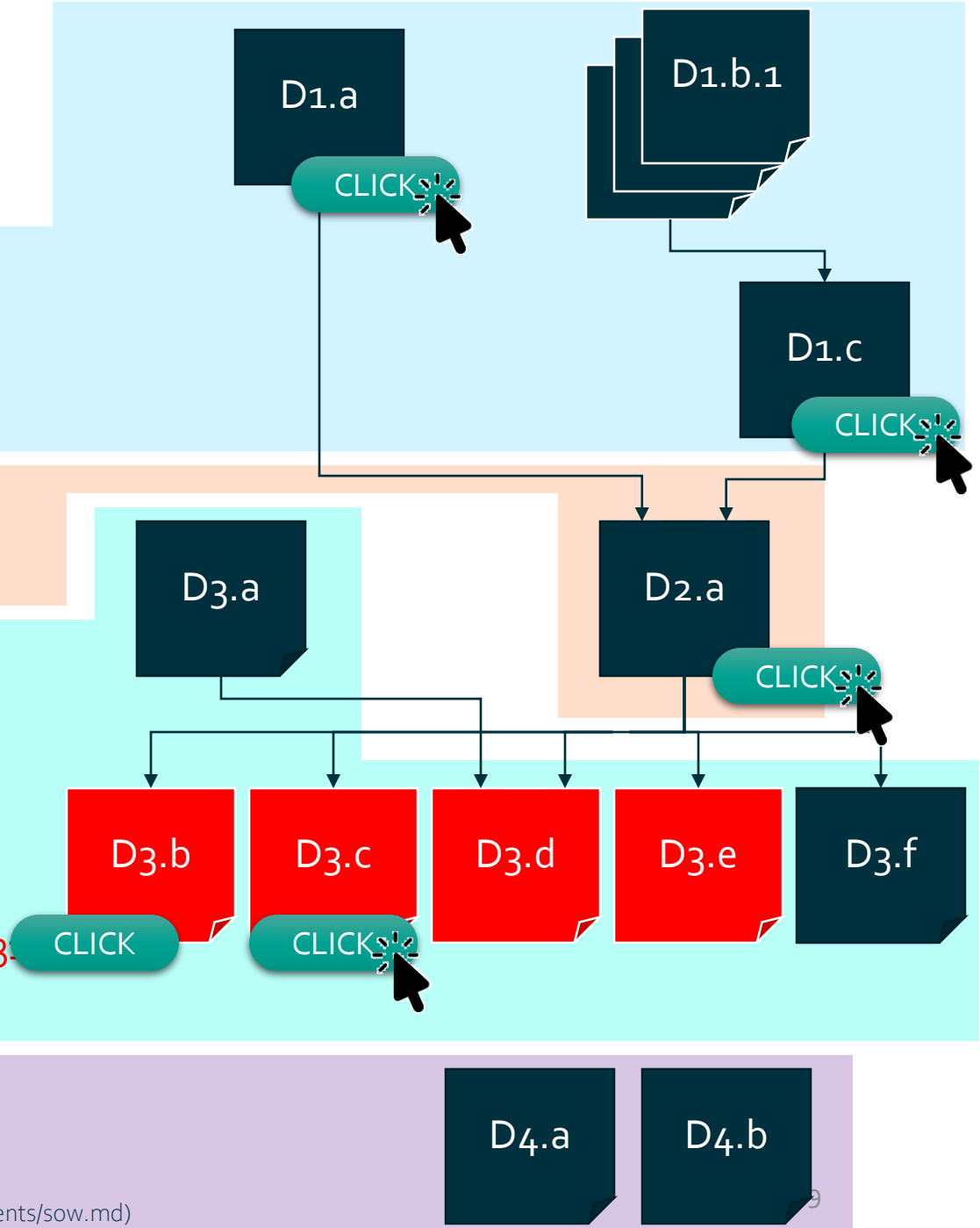
(D3.d) ONNX Safety-related profile – format (2025/12/31)

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

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

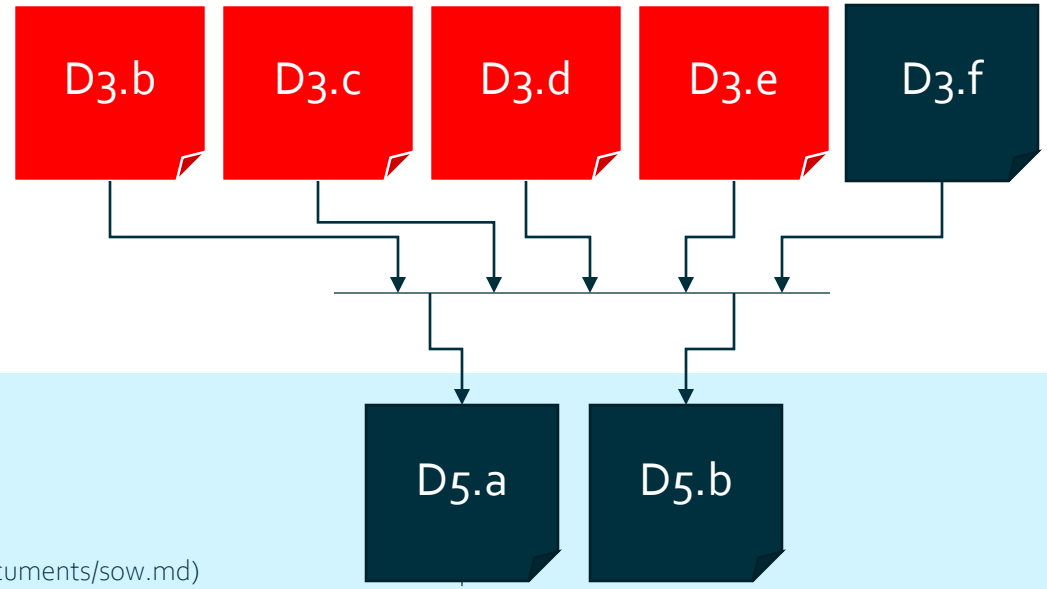
(D4.a) ONNX Safety-related profile **verification** report

(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)

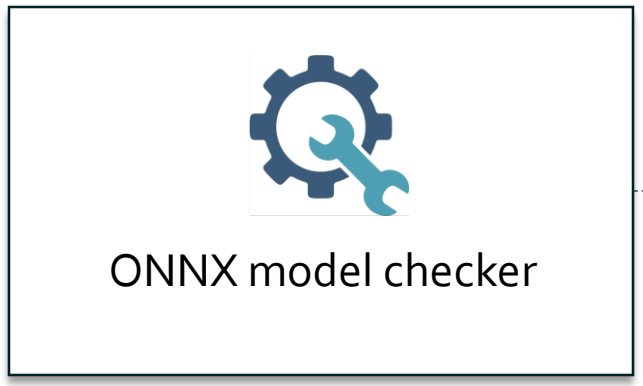
Deliverables
Status



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

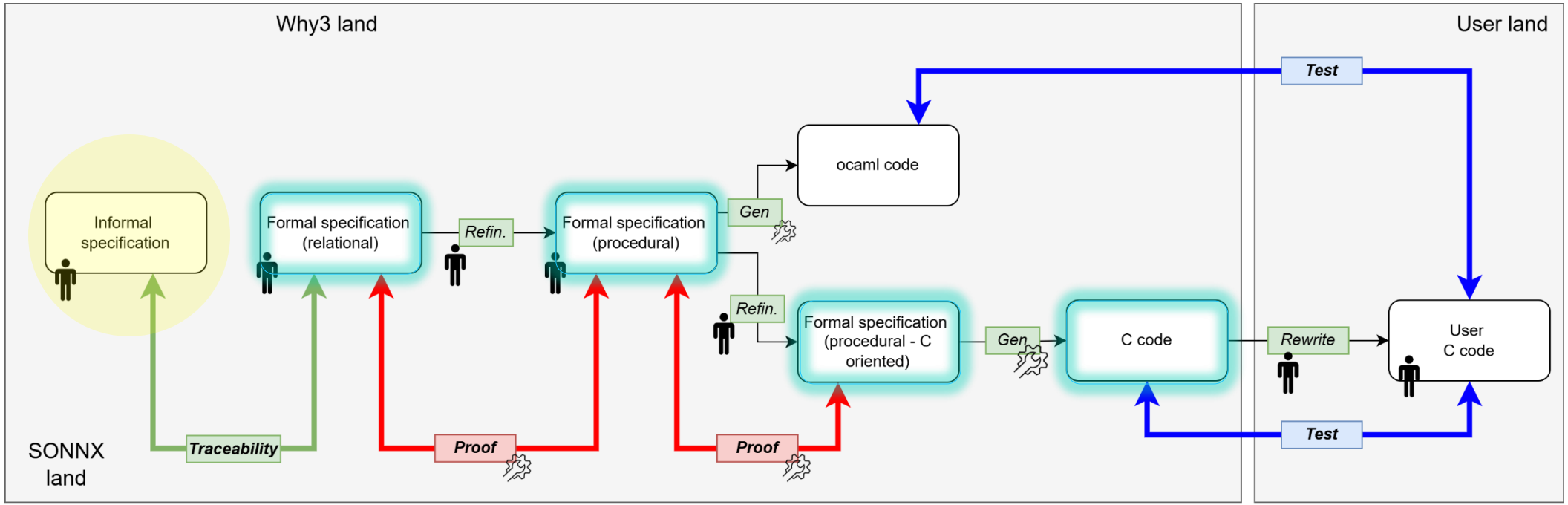
(D5.b) **Requirements** of tool <tool>(2025/12/31)

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



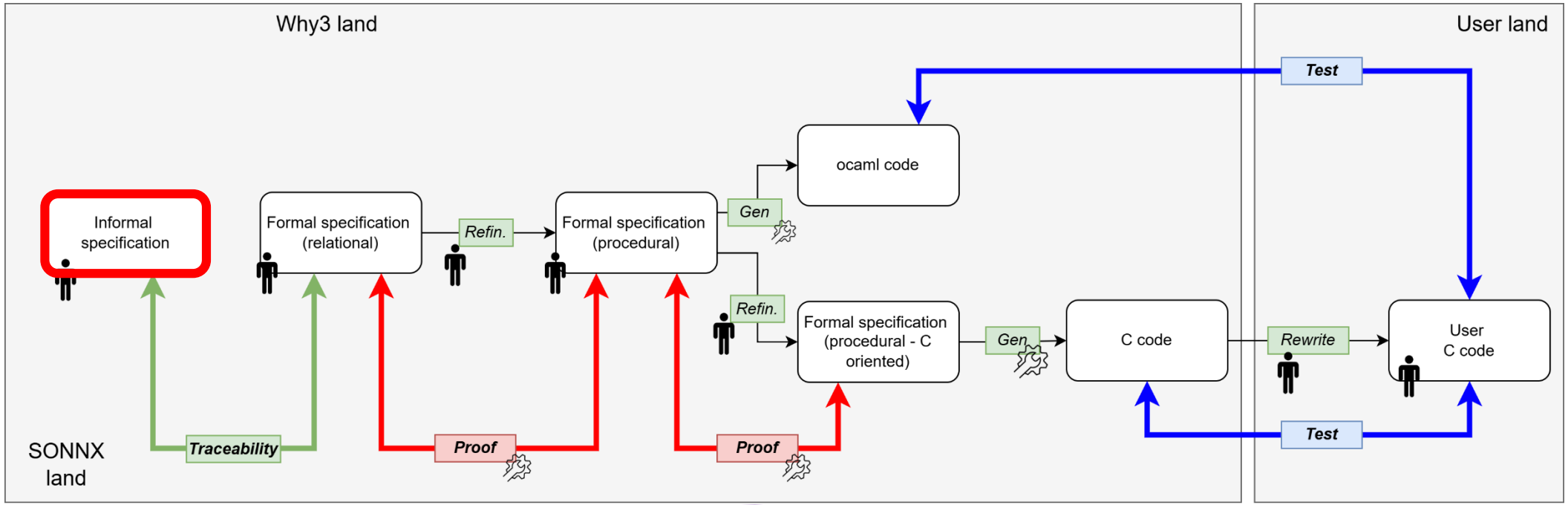
Formal specification and verification

Overview



Formal specification and verification

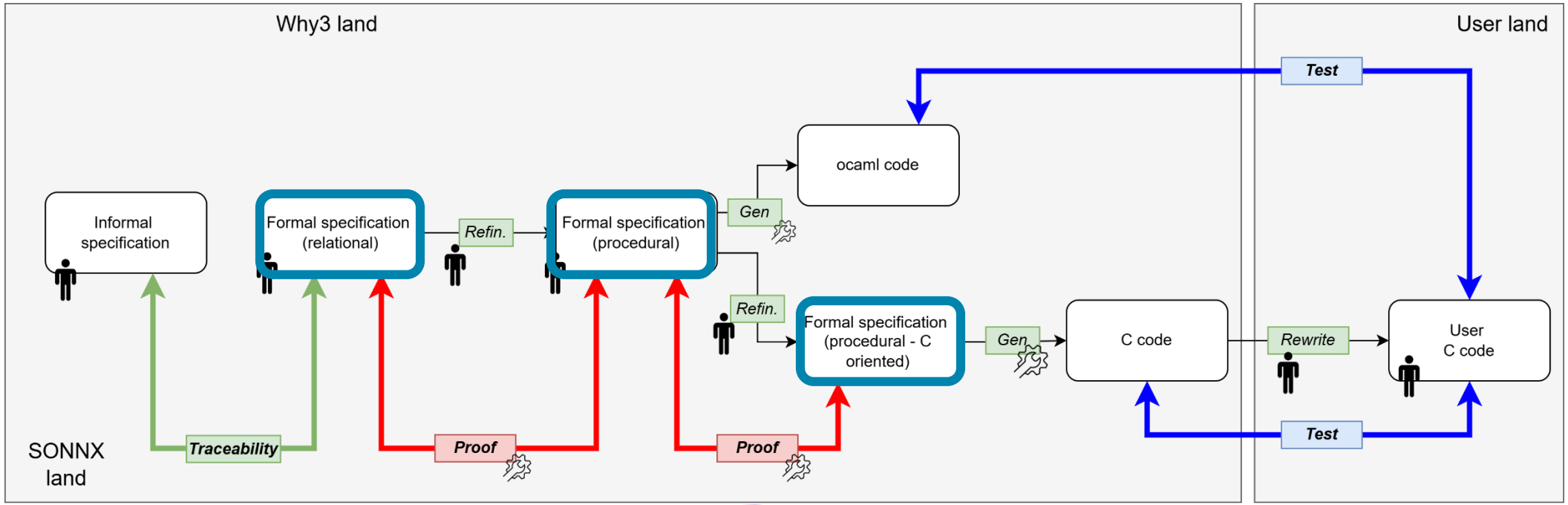
Information specification



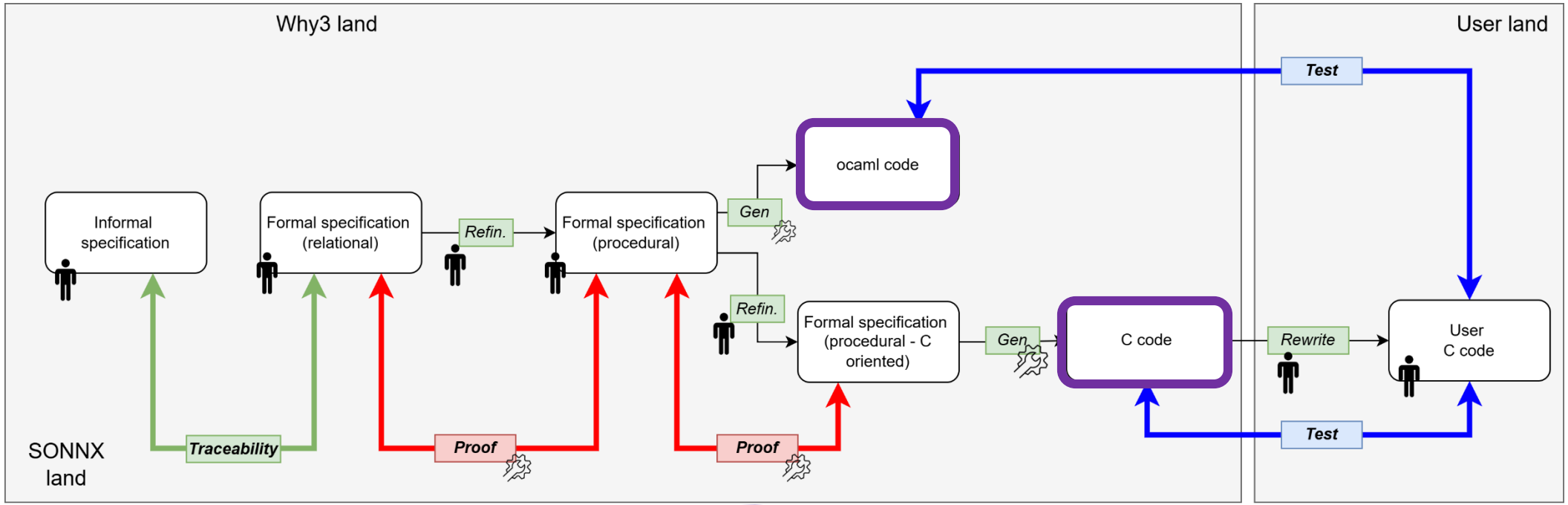
Formal specification and verification

Formal specification

traceable



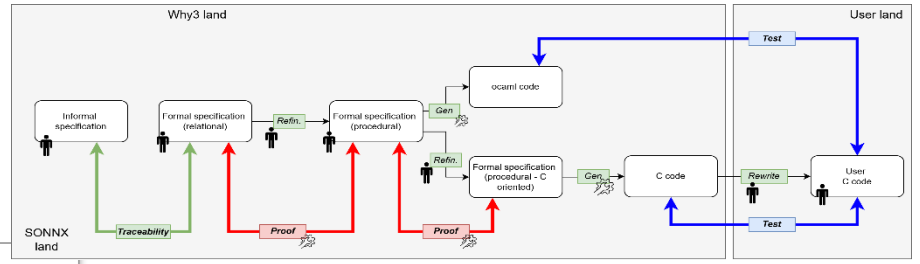
Deliverables
Reference implementation

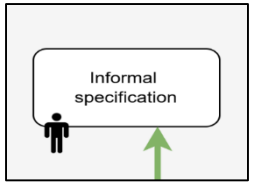


- Reference implementation
  - Interim ocaml code
  - Final C code

Deliverables

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





SONNX profile

Inputs

X

Tensor `X` is the input tensor on which convolution with kernel `W` is computed.

The shape of tensor `X` is $b(X) \times c(X) \times h(X) \times w(X)$.

Constraints

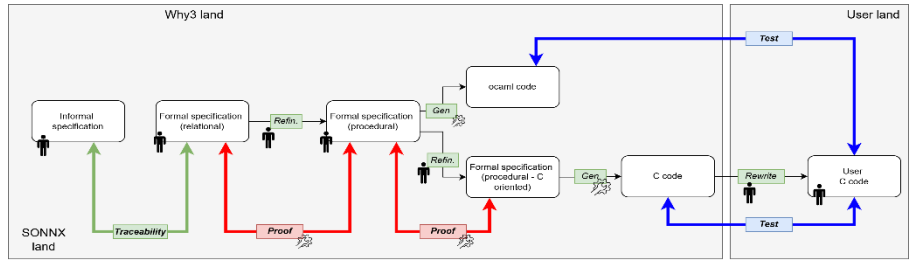
- (C1) Number of spatial axes of tensor `X`
  - Statement: The number of spatial axes of tensor `X` is 2. `[R1]`
  - Rationale: This restriction is introduced to simplify the implementation considering the actual industrial use cases.
- (C2) Consistency between the number of channels of `X` and `W`
  - Statement: $c(X) = fm(W)$
- (C3) Consistency between the shape of tensors `X`, `W`, `Y` and attributes `pads`, `dilations` and `strides`
  - Statement:
    - $\left\lfloor \frac{alpha - (dilations[0] \cdot h(W) - 1)}{strides[0]} \right\rfloor + 1 = h(Y)$ with $alpha = h(X) + pads[0] + pads[2]$

    and

    - $\left\lfloor \frac{beta - (dilations[1] \cdot w(W) - 1)}{strides[1]} \right\rfloor + 1 = w(Y)$ with $beta = w(X) + pads[1] + pads[3]$
  - Rationale: The size of the output is determined by the number of times the kernel can be applied on a given spatial axis.
- (C4) Axis denotations
  - Statement: If axis denotation is in effect, the operation expects input data tensor to have axis denotation [`DATA_BATCH`, `DATA_CHANNEL`, `DATA_FEATURE`, `DATA_FEATURE`].
  - Rationale: Denotation convention

Deliverables

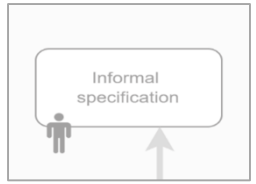
For informal to formal specification: **`tensors`**

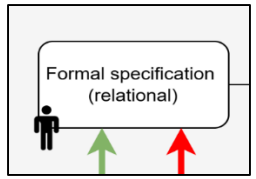


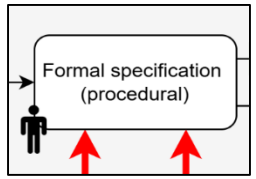

```
module Tensor
  use int.Int
  use map.Map
  use utils.Product
  use sequence.Seq

  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
```

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



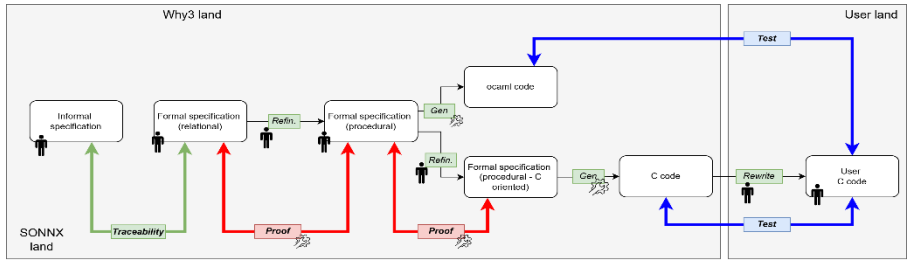


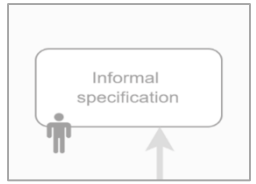


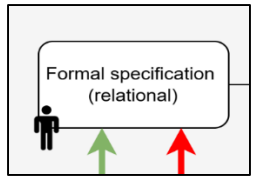
Deliverables

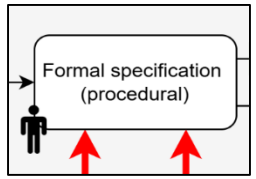
For informal to formal specification: the **conv** operator

```
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 }
```



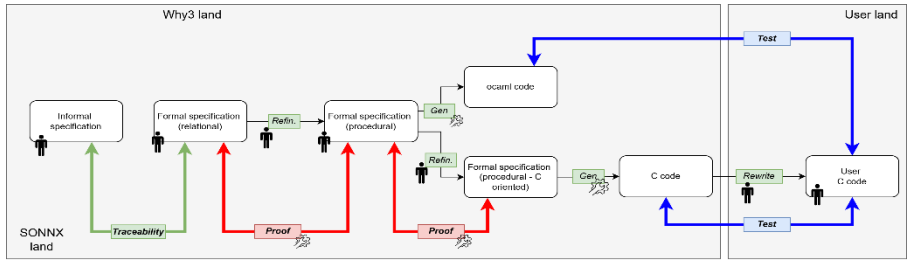


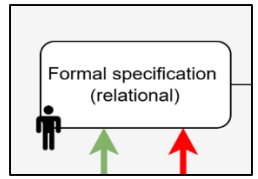


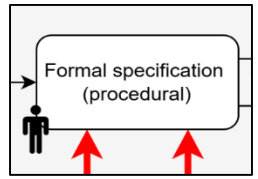


Deliverables

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







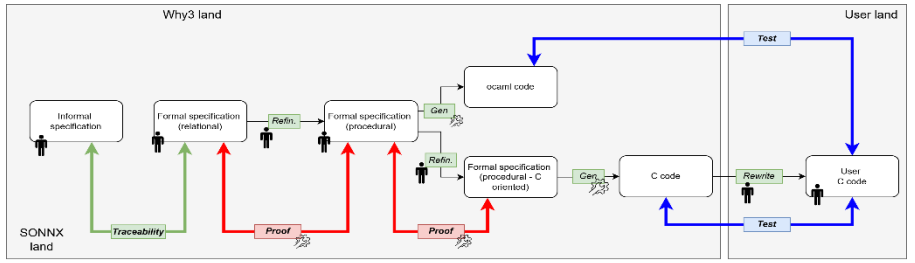

```
  requires {
            let h_out_calc = calculate_H_out (Ops4D.h_dim x) (Ops4D.h_dim w)
                                             (Ops4D.pad_h_begin pads) (Ops4D.pad_h_end pads)
                                             (Ops4D.dilation_h dilations) (Ops4D.stride_h strides) in
            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 }
```

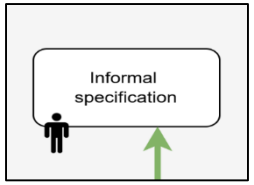
The shape

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:

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

Traceability link

Let i_a be the global index along dimension a, and let $i_a^{'}$ be the corresponding local index 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:

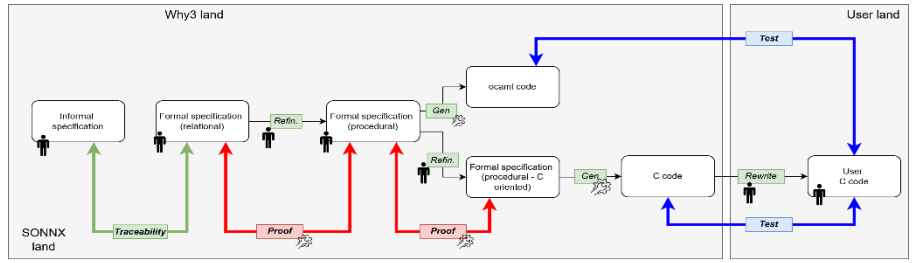
$$\text{T5: } \forall i_0, \ldots, i_{r-1}.\ Y[i_0, \ldots, i_{r-1}] = X_k[i_0, \ldots, i_a^{'}, \ldots, i_{r-1}]$$

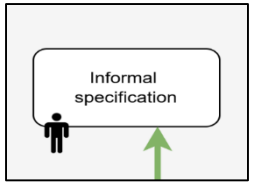
With i_0 and i_{r-1} are the indices which access respectively the first and last dimensions of a **r-dimensional** tensor. $i_0, \ldots, i_{r-1}$ represent a set of indices that uniquely identify an element within an **r-dimensional** tensor.

```
(* ==================== Concat shape ================================*)
(** T2: defining s_k which is the sum of the different tensors dimension along the axis considered **)
(*
      x_k: list (sequence) of input tensors
      axis: axis along which the concatenation is performed
*)
let rec function compute_s_k_at_axis (x_k: seq (tensor 'a)) (axis: int) : int =
  variant{x_k}  (* a_k has to decrease size *)
  match x_k with
  | L.Nil -> 0
  | L.Cons hd tl -> (hd.shape.dims)[axis] + compute_s_k_at_axis tl axis
  end
```

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

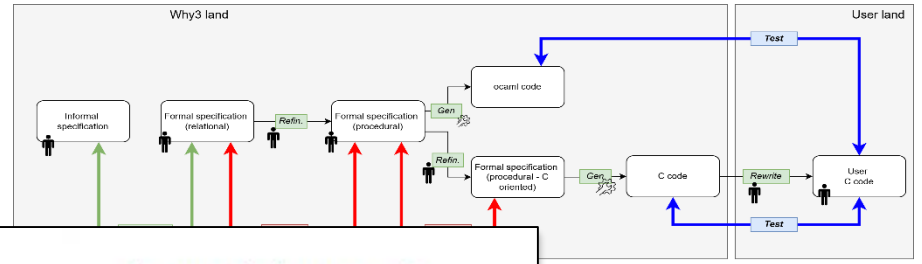
- `[T04a]` An operator specifies a relation (a function) between a set of input parameters and a set of outputs parameters.
  - Input and output parameters (resp. output) are free variables that can be bound to tensors using nodes
  - An operator has at least one output

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

```
type graph = {
    gi: list tensor_id;        (* graph inputs *)
    go: list tensor_id;        (* graph outputs *)
    gt: list tensor_id;    (* graph tensors *)
    gn: list node;          (* graph nodes *)
}
```

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.

```
type node = {
    ope: operator; (* The operator referred to by the node *)
    oi: list tensor_id; (* Input tensors, position-wise *)
    ou: list tensor_id; (* Output tensors, position-wise *)
}
    invariant {
        length oi = length ope.opi /\
        length ou = length ope.opo
    }
    by
    {
        ope= { name = ""; opi = Nil; opo = Nil };
        oi = Nil;
        ou = Nil;
    }
```

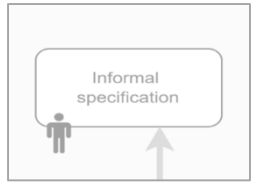
Tensors

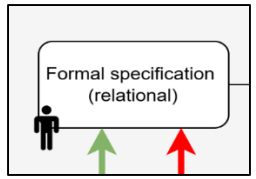
- `[T02b]` A tensor is an object that can hold a value or be uninitialized
- `[T02a]` A tensor is identified by a unique identifier within a graph

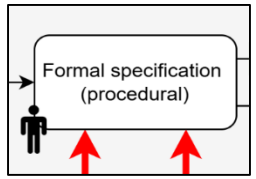
Operators

- `[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

```
type operator = {
    name: string; (* name of the operator *)
    opi: list shape;  (* input shapes *)
    opo: list shape;  (* output shapes *)
}
```

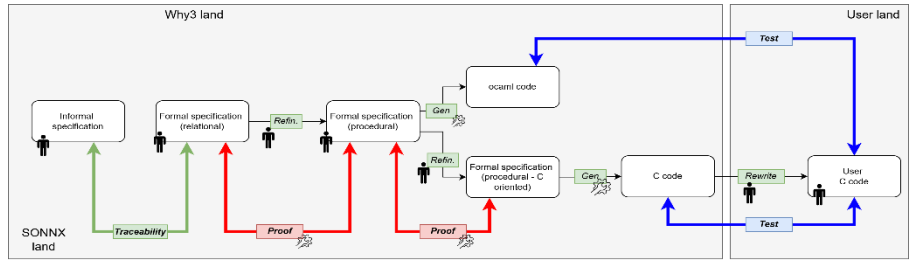




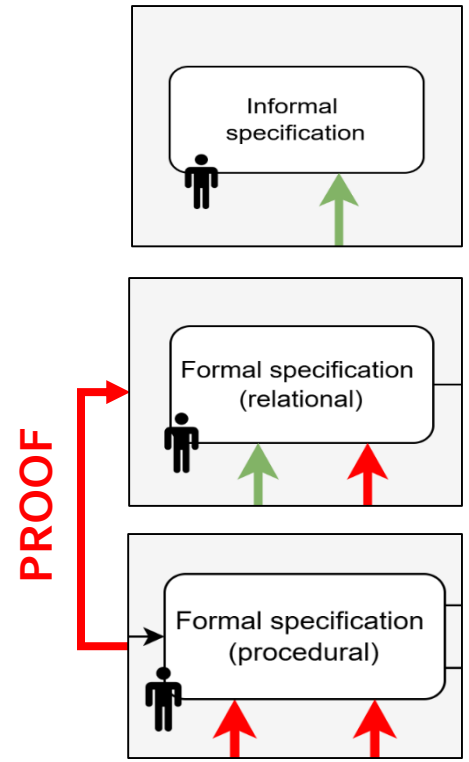


Deliverables

Other cases: the **graph**

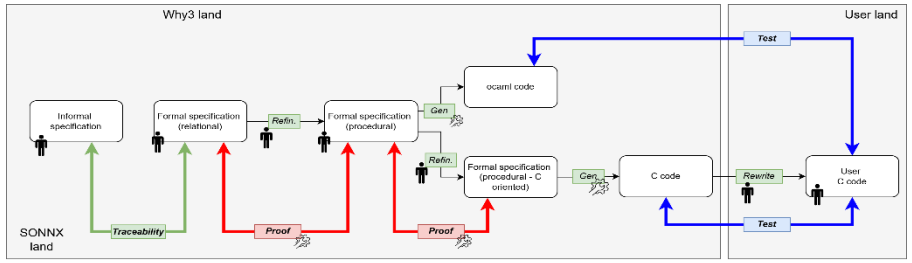



```
Partial (4/5) (23ms) (alt-ergo 4) (split_vc 2) (depth 2)
let rec assign_list (s: graph_state) (l: lis_map) : graph_state
    (* A tensor shall only appear once in the state *)

    (* A tensor shall only appear once in an assignment *)
    requires { forall t: tensor_id. t_appears_at_most_once t l }

    (* The assignment is correct  *)
    ensures { forall t, v . Mem.mem (t, v) l ->
        my_map_get_logic result t =  v }
    variant { l }
=
    match l with
    | Nil -> s  (* Nothing to assign, the state does not change *)
    | Cons (t, v) xs ->
        (* Assign the value and continue with the rest of the assignment list *)
        let s' = my_map_set s t v in
            (* Tensor t is correctly assigned *)
            assert {  my_map_get_logic s' t =  v } ;
            assume { NumOcc.num_occ t (project xs) = 0  } ;
            assume { forall t'. t_appears_at_most_once t' xs } ;
            let s'' = assign_list s' xs in
                assume { forall t'', v'' . Mem.mem (t'', v'') xs -> my_map_get_logic s' t'' =  v'' } ;
                s''
    end
```



Deliverables

Other cases: the **graph**



Why3 Interactive Proof Session

File Edit Tools View Help

Status Theories/Goals

- operator_one_output_ko
- all_operator_ins_and_outs_ok
- all_operator_ins_and_outs_ko
- all_node_with_one_output_ok
- all_node_with_one_output_ko
- filter'vc [VC for filter]
- no_dangling_tensor_ok
- no_dangling_tensor_ko_1
- no_dangling_tensor_ko_2
- graph_ins_are_node_ins_test_ok
- graph_ins_are_node_ins_test_ko
- graph_outs_are_node_outs_test_ok
- graph_outs_are_node_outs_test_ko
- graph_ins_outs_are_tensors_test_ok
- graph_ins_outs_are_tensors_test_ko
- is_initialized'vc [VC for is_initialized]
- are_initialized'vc [VC for are_initialized]
- node_ready'vc [VC for node_ready]
- make_list'vc [VC for make_list]
- fold_left2'vc [VC for fold_left2]
- apply'vc [VC for apply]
- exec_operator'vc [VC for exec_operator]
- equivalence_index_counting'vc [VC for equivalence_index_cou
  - split_vc
    - 0 [assertion]
    - 1 [postcondition]
      - split_vc
        - 0 [postcondition]

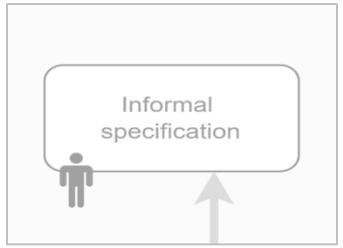
Task mappings-2.mlw

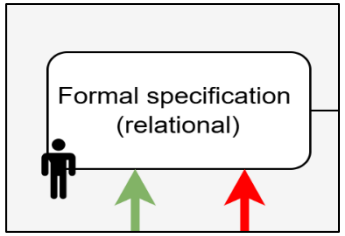
```
359         gi = Cons 1 (Cons 2 Nil);
360         go = Cons 3 (Cons 4 Nil);
361         gt = Cons 1 (Cons 2 (Cons 3 (Cons 4 Nil)));
362         gn = Cons { ope= add_op; oi=Cons 1 (Cons 2 Nil); ou=Cons 3 Nil}
363             (Cons { ope=sub_op; oi=Cons 1 (Cons 2 Nil); ou=Cons 4  Nil} Nil);
364     } in
365     no_dangling_tensor g
366
367
368 goal no_dangling_tensor_ko_1: (* Tensor 3 is the output of two nodes *)
369     let g : graph = {
370         gi = Cons 1 (Cons 2 Nil);
371         go = Cons 3 Nil;
372         gt = Cons 1 (Cons 2 (Cons 3 (Cons 4 Nil)));
373         gn = Cons { ope=add_op; oi=Cons 1 (Cons 2 Nil); ou=Cons 3 Nil}
374             (Cons { ope=sub_op; oi=Cons 1 (Cons 2 Nil); ou=Cons 3 Nil} Nil);
375     } in
376     not no_dangling_tensor g
377
378 goal no_dangling_tensor_ko_2: (* Tensor 4 is dangling *)
379     let g : graph = {
```

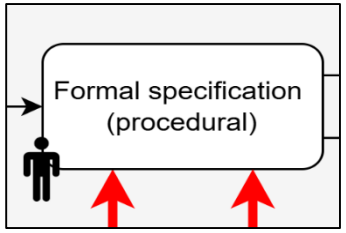
0/0/0 type commands here

Messages Log Edited proof Prover output Counterexample

```
Welcome to Why3 IDE
type 'help' for help

Session initialized successfully
File already in session: /home/eric/Work/Work/Code/why3/mappings-2.mlw
```







Numerical errors
Approach

- Thou shall not overspecify…

- Errors depend on method and computation error)
- No specification of errors

- Focus on error evaluation means
  - Empirical (incomplete)
  - Formal via abstract interpretation (e.g., fluctuat)
  - Formal via axiomatic proof

Numerical errors
Approach

- We propose a lower bound on the error for any value in the input domain, for any implementation complying with IEEE 754
  - This is not necessarily the smallest error but
    - The effort to express the formula remains acceptable
    - The complexity of the formula remains tractable
    - The verification of the property remains achievable
- For a restriction of the input domain, the error may be smaller
- The SONNX reference implementation will (?) comply with this constraint
- More efficient implementations may violate this constraint. In that case, the implementer has to provide its 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 $C_{\text{err}}^{\text{propag}}$ the propagated error of `Add` and $C_{\text{err}}^{\text{intro}}$ the introduced error of `Add`. Hence the numerical error of `C`, $C_{\text{err}} = C_{\text{err}}^{\text{propag}} + C_{\text{err}}^{\text{intro}}$.

Error propagation

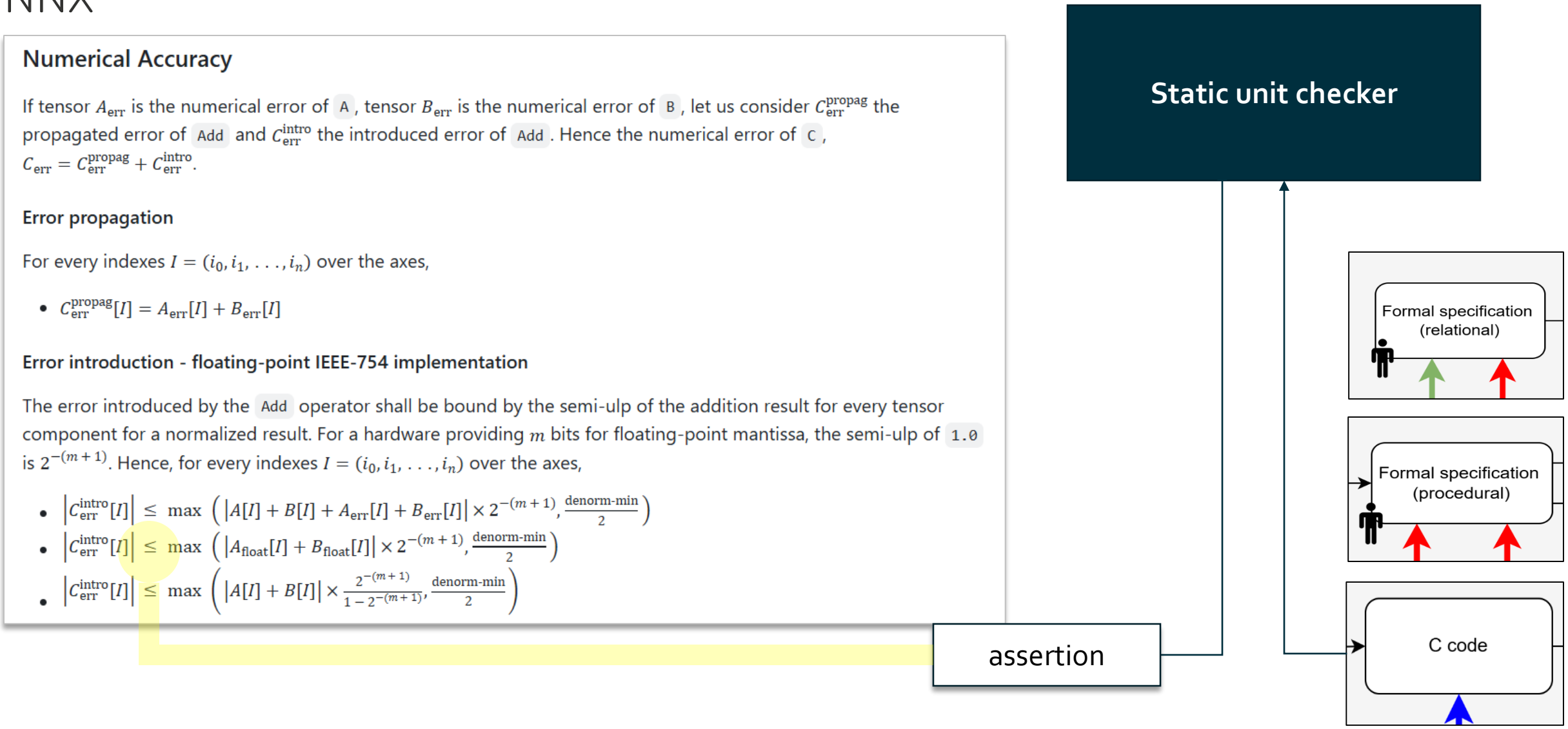
For every indexes $I = (i_0, i_1, \ldots, i_n)$ over the axes,

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

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,

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



Error conditions

- How to specify error conditions
- Examples

`y=Add(a: int32, b: int32)`

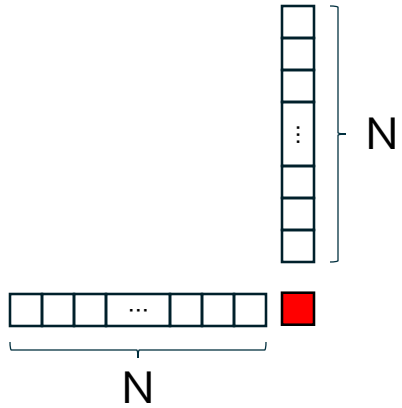
$$-2^{32} \leq a + b \leq 2^{32} - 1$$

Or, more conservatively,

$$-2^{31} + 1 \leq a < 2^{31} \text{ and } -2^{31} + 1 \leq b < 2^{31}$$

- For matrix multiplcation (e.g., **`MatMulInteger`**), a precondition can be expressed on the shape of the tensors

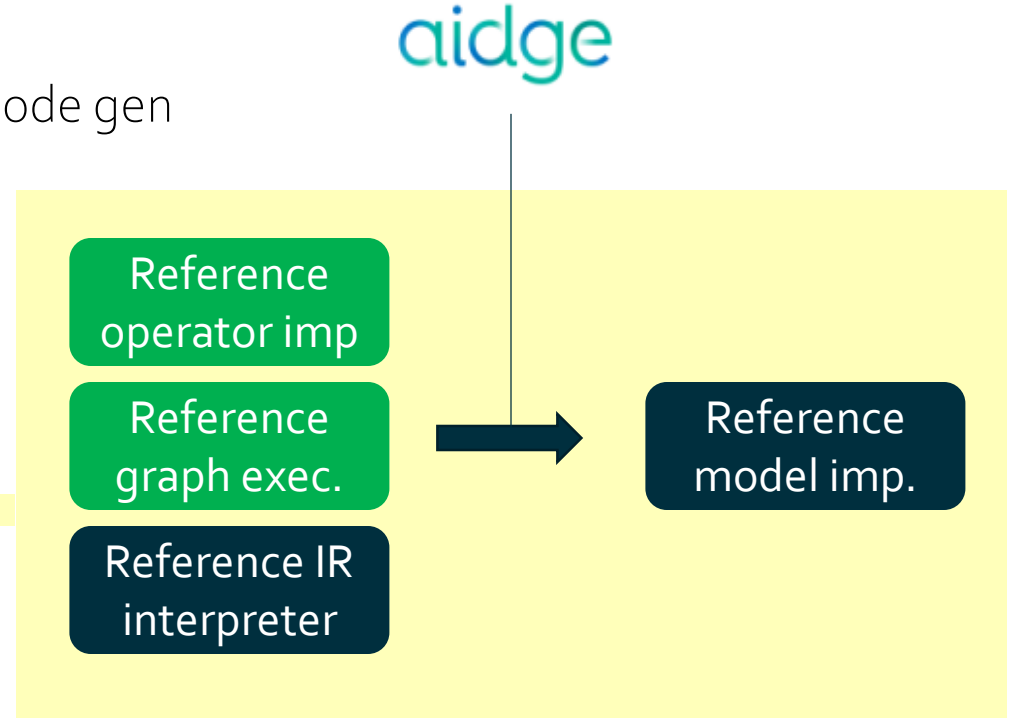
$$\mathrm{N} > \frac{2^{32} - 1}{128^2} \approx 133141.5$$



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
  - Compliance with ARP
  - Integration to the AIdge platform



  - Collaboration with ONNX infra on testing…
  - 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