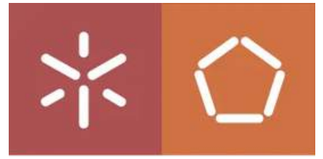
Verifying the POCKET+ lossless compression algorithm

Milestone 1

March 2026 - Project in Formal Methods of Programming

Cláudia Faria PG60240
Patrícia Bastos PG60287

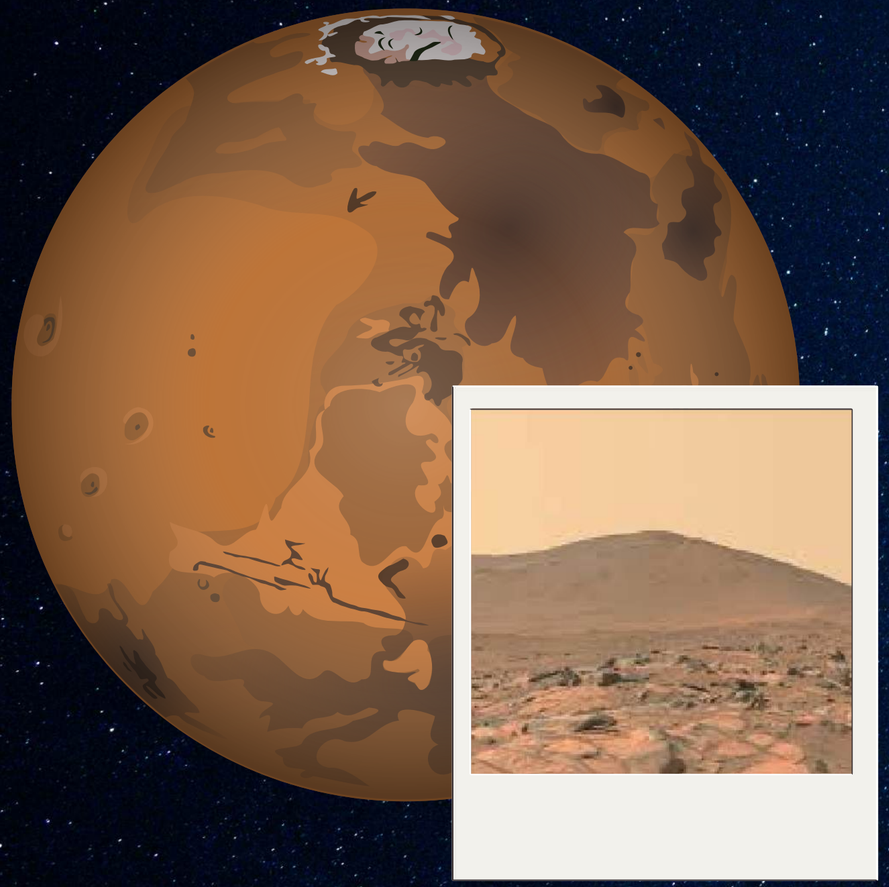


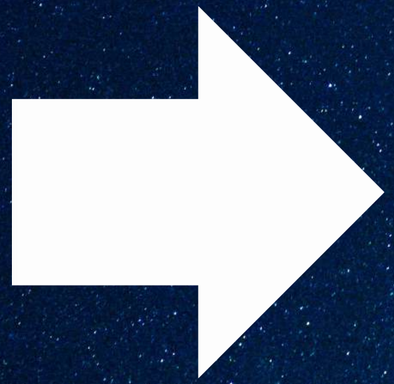
TELESPAZIO
a LEONARDO and THALES company

This project is an international collaboration with the space company **Telespazio**.

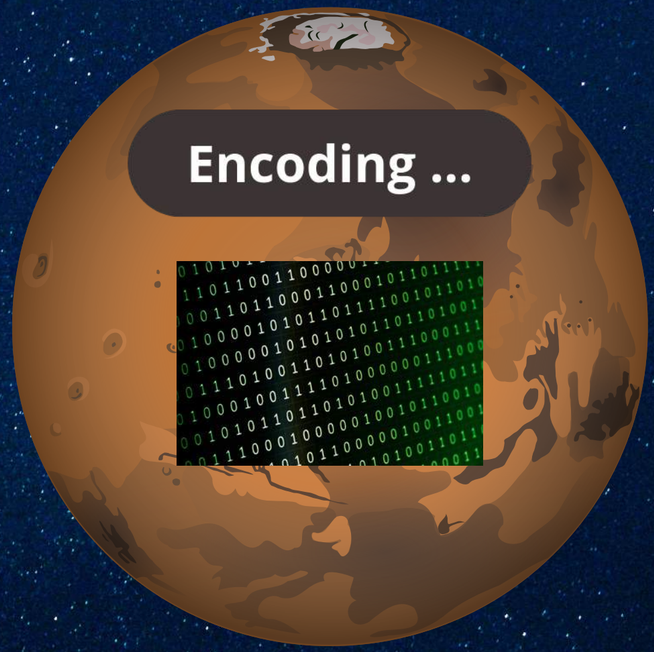
Milestone 1

What is Pocket+?






Decoding ...


Encoding ...

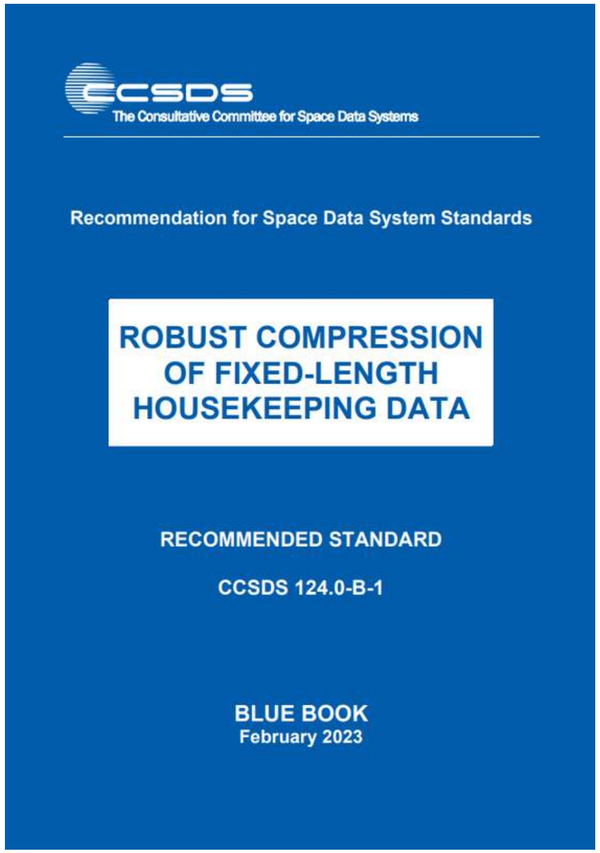
Main Objective

Formal verification of the Pocket+ protocol

Milestone 1

The Recommended Standard

The Recommended Standard



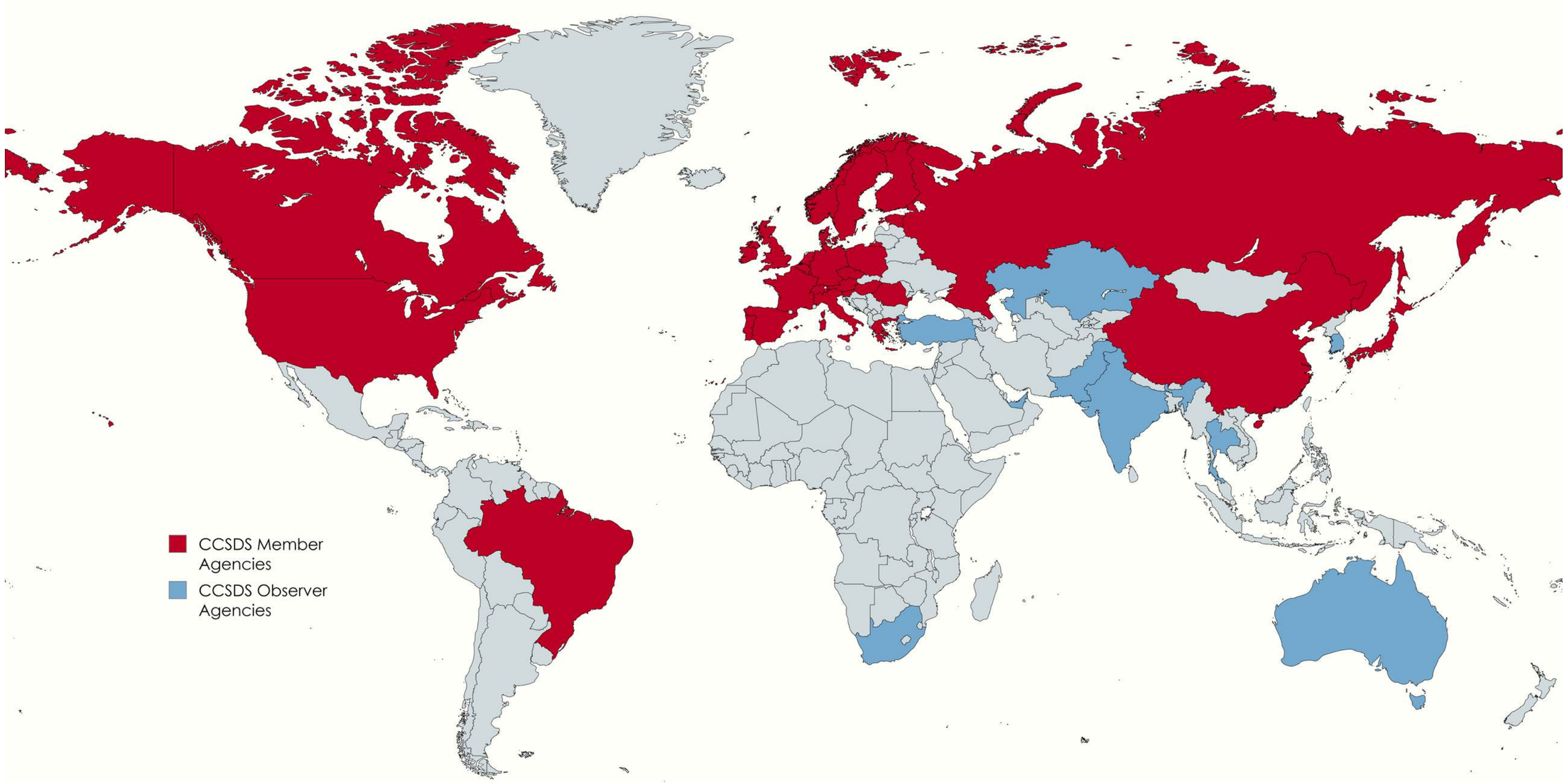
Consultative Committee

CCSDS

for Space Data Systems

Milestone 1

March 2026

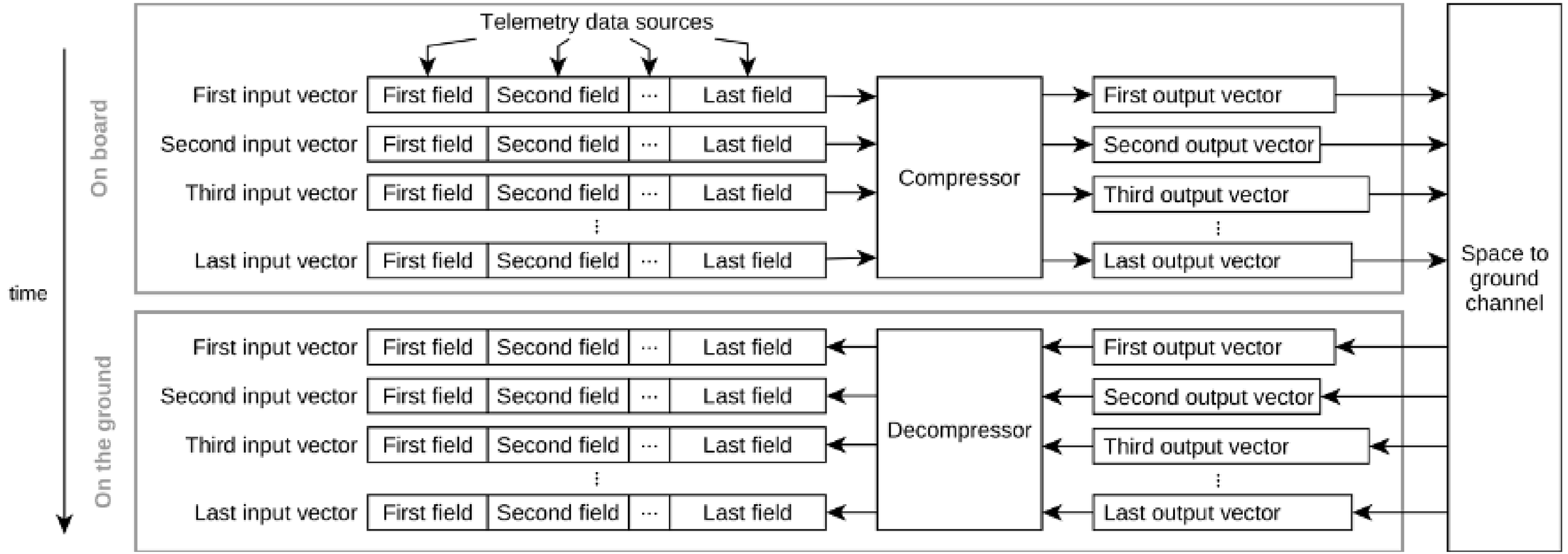


Existing Implementation

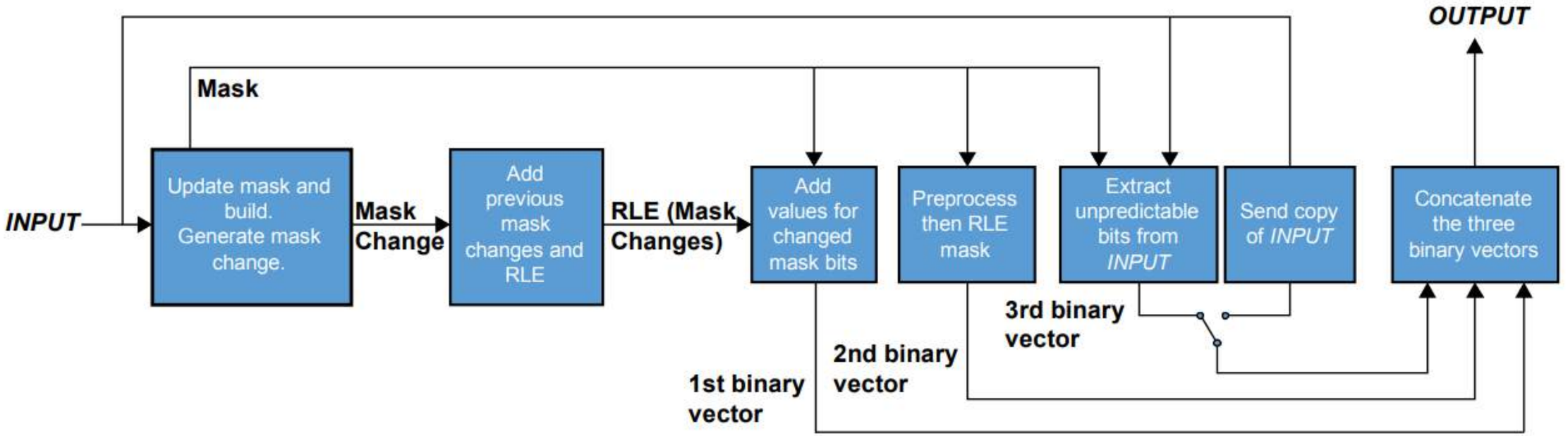
C++ code implemented by

VISIONSPACE

Full protocol



Encoder

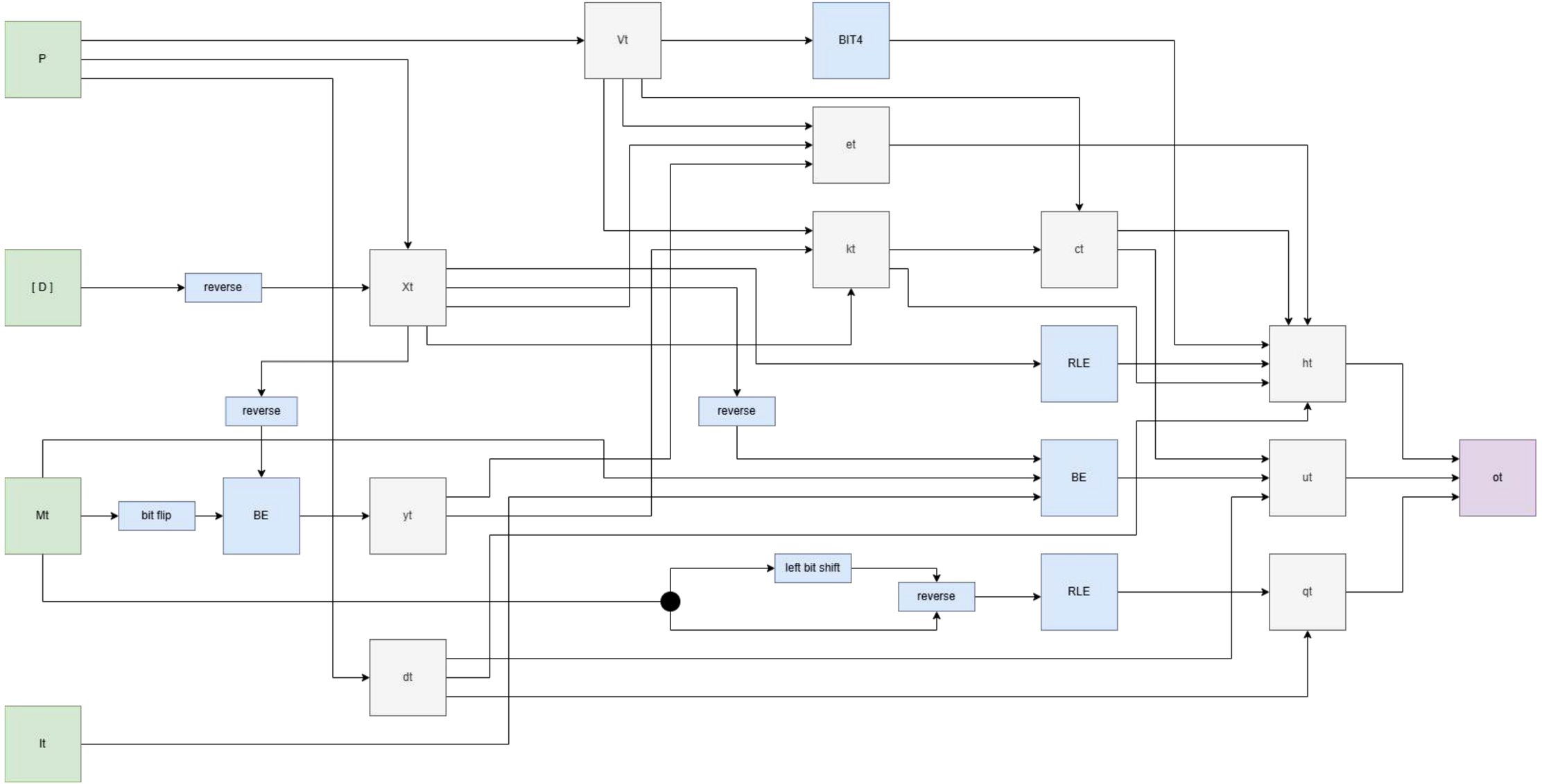

OUTPUT
Mask
INPUT
Update mask and build. Generate mask change.
Mask Change
Add previous mask changes and RLE
RLE (Mask Changes)
Add values for changed mask bits
Preprocess then RLE mask
Extract unpredictable bits from INPUT
Send copy of INPUT
Concatenate the three binary vectors
3rd binary vector
2nd binary vector
1st binary vector

Strategy

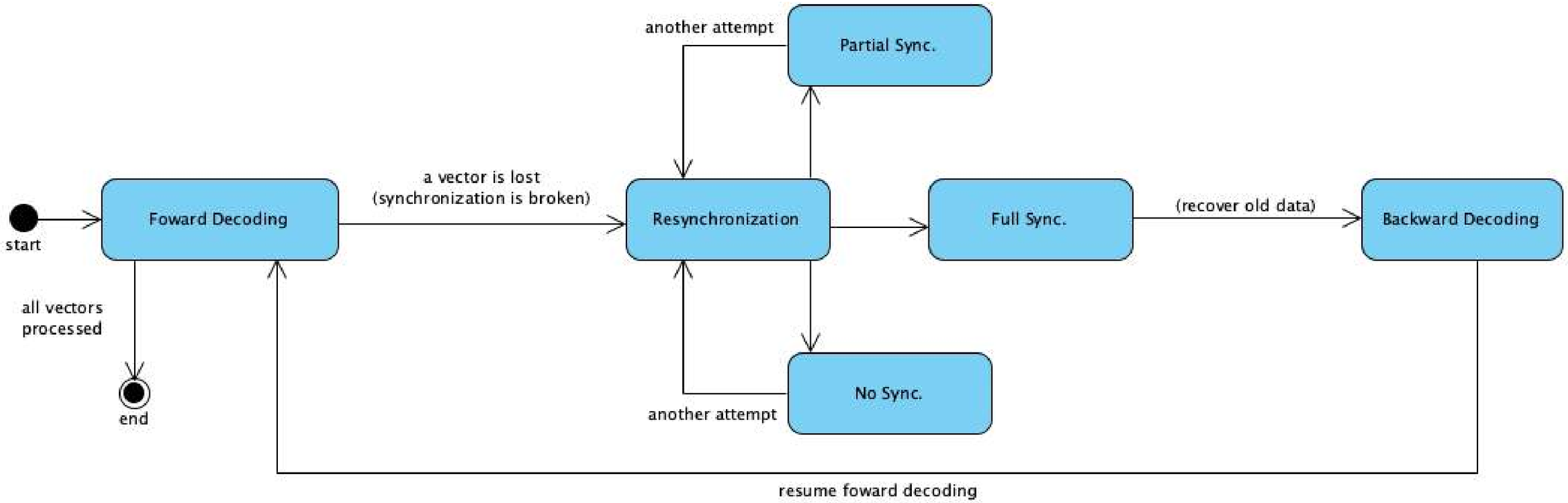
Functional implementation in Haskell + its formal verification

Milestone 1

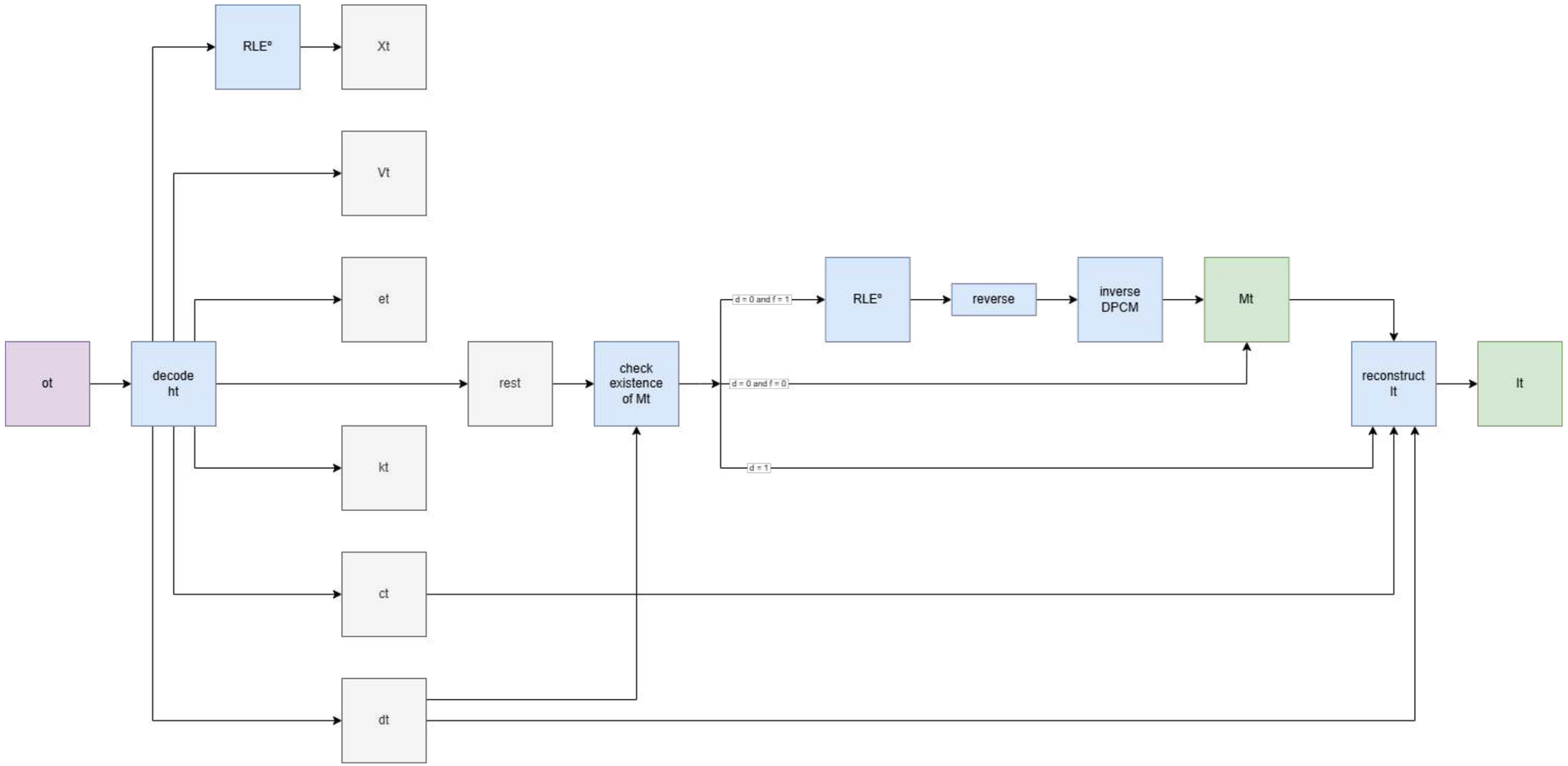
March 2026



Decoder


another attempt
Partial Sync.
a vector is lost
(synchronization is broken)
Foward Decoding
Resynchronization
Full Sync.
(recover old data)
Backward Decoding
start
all vectors
processed
end
No Sync.
another attempt
resume foward decoding

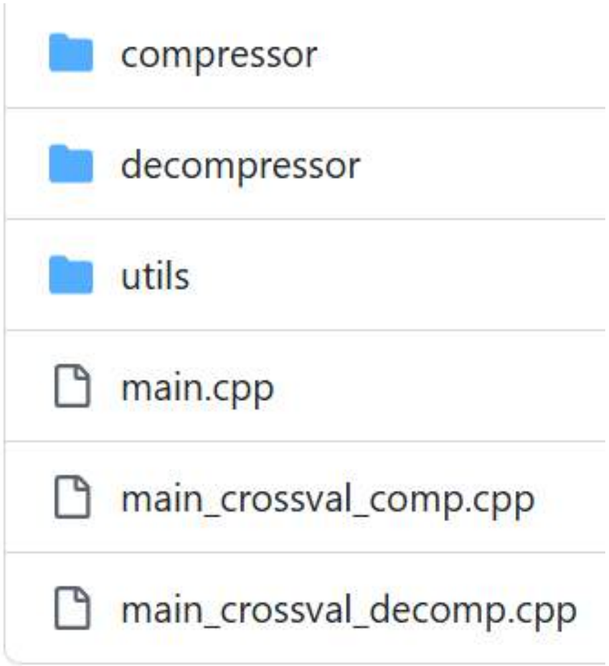
Foward Decoding



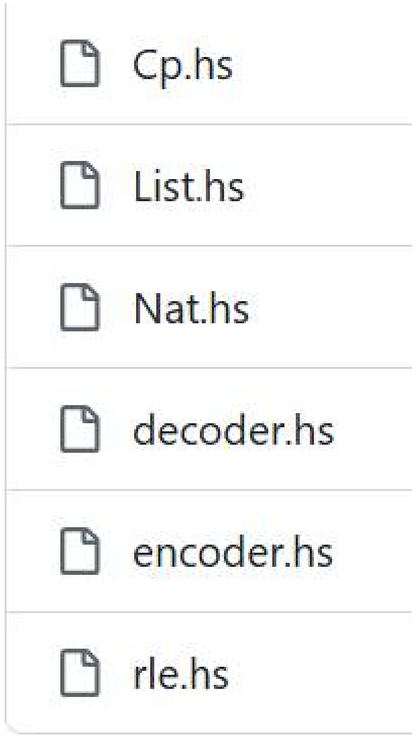
Milestone 1

Implementation and Verification

Implementation

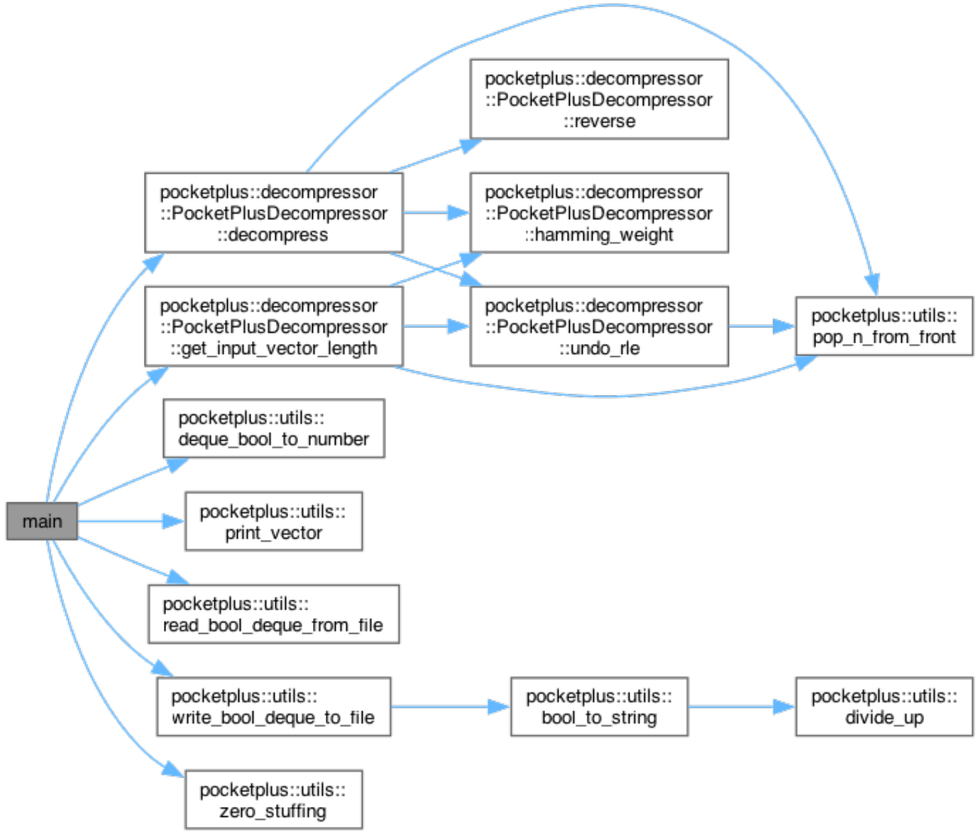


C++ (Vision Space)

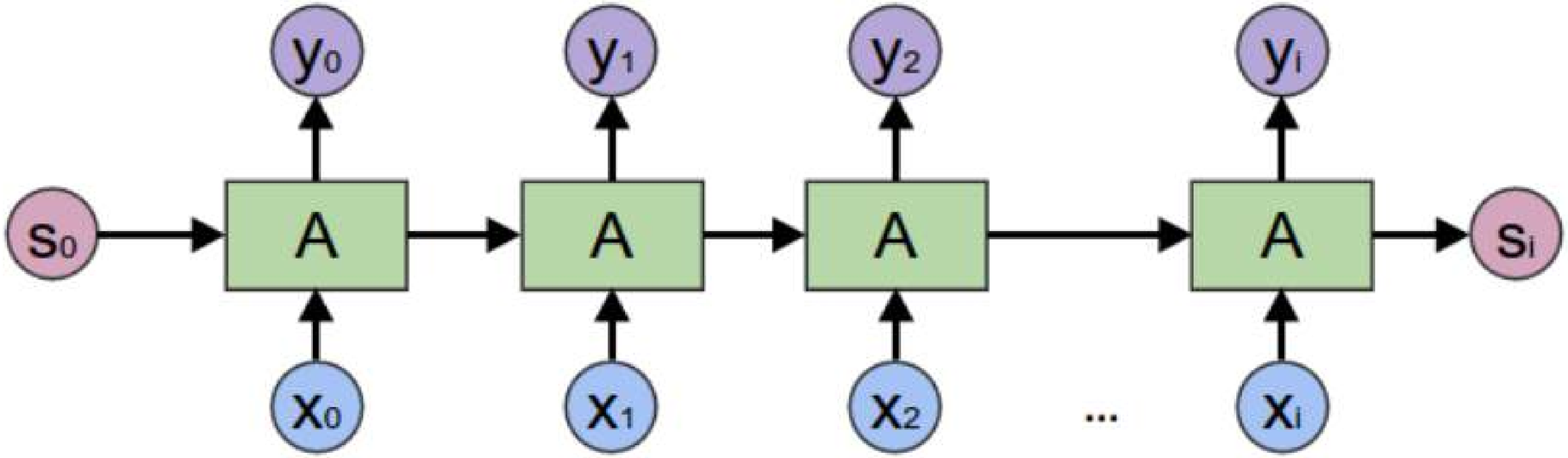


Haskell (UMinho)

Call Graph


pocketplus::decompressor ::PocketPlusDecompressor ::reverse
pocketplus::decompressor ::PocketPlusDecompressor ::decompress
pocketplus::decompressor ::PocketPlusDecompressor ::hamming_weight
pocketplus::decompressor ::PocketPlusDecompressor ::get_input_vector_length
pocketplus::decompressor ::PocketPlusDecompressor ::undo_rle
pocketplus::utils:: pop_n_from_front
pocketplus::utils:: deque_bool_to_number
main
pocketplus::utils:: print_vector
pocketplus::utils:: read_bool_deque_from_file
pocketplus::utils:: write_bool_deque_to_file
pocketplus::utils:: bool_to_string
pocketplus::utils:: divide_up
pocketplus::utils:: zero_stuffing

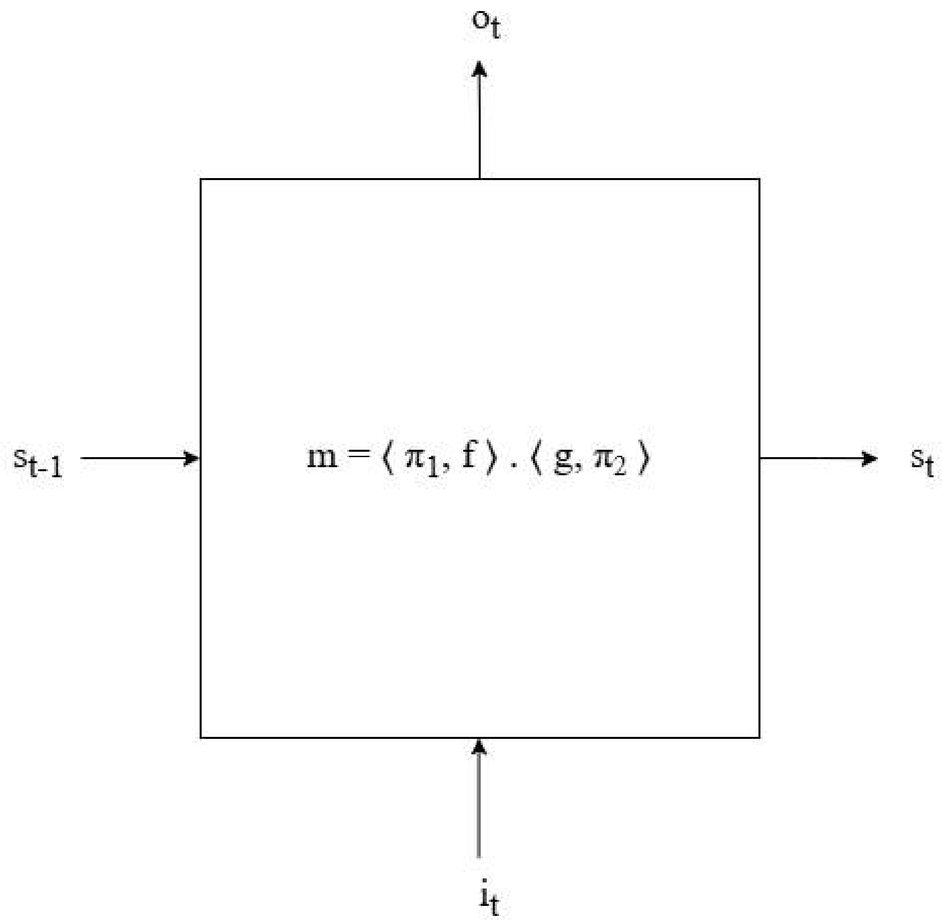
Encoder - Mealy Machine chain



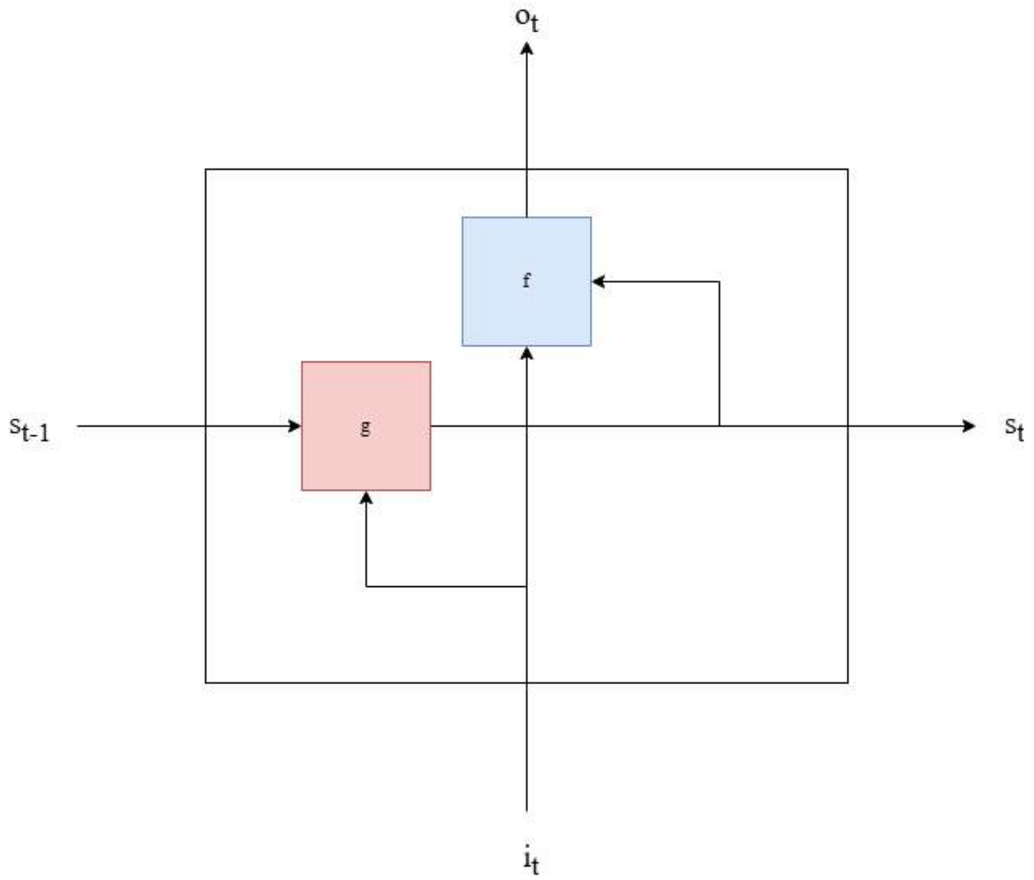
encoder = mapAccumL (curry m g f)

Image: https://colah.github.io/posts/2015-09-NN-Types-FP/

Encoder - Mealy Machine chain



Encoder - Mealy Machine chain



Verification

The Decoder should be the
left inverse of the Encoder

Decoder . Encoder = id

Verification

Using Calculus of Information Systems, we will

- Verify that the Encoder is **injective**
- Get the converse of functions using **least complements**

Example

$$RLE = fRLE \cdot \mathcal{F}RLE \cdot gRLE$$
$$RLE^{\circ} = gRLE^{\circ} \cdot \mathcal{F}RLE^{\circ} \cdot fRLE^{\circ}$$

To prove that ***RLE · RLE° = id***
there needs to be **injectivity**

RLE Verification

```
rle = sentinel . (>>= count) . run2count . dropZeros
   where run2count = anaList gRLE
         sentinel = (++ [1,0])
```

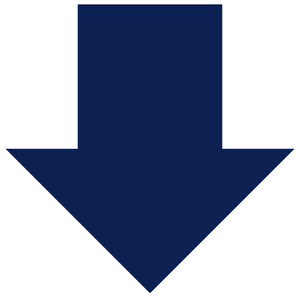
not **injective**

RLE is not injective

Least Complement

```haskell
dropZeros = reverse .  dropWhile(==0) . reverse
```

Not Injective !



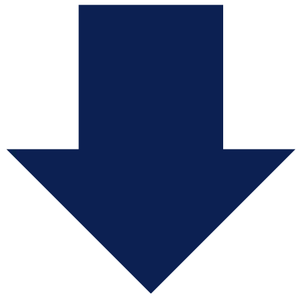
Injective !

```haskell
dropZeros :: [Bit] -> ([Bit],Int)
dropZeros = (reverse >< length) . swap . cspan (==0) . reverse
```

Least Complement

```
rle = sentinel . (>>= count) . run2count . dropZeros
   where run2count = anaList gRLE
         sentinel = (++ [1,0])
```

Not Injective !



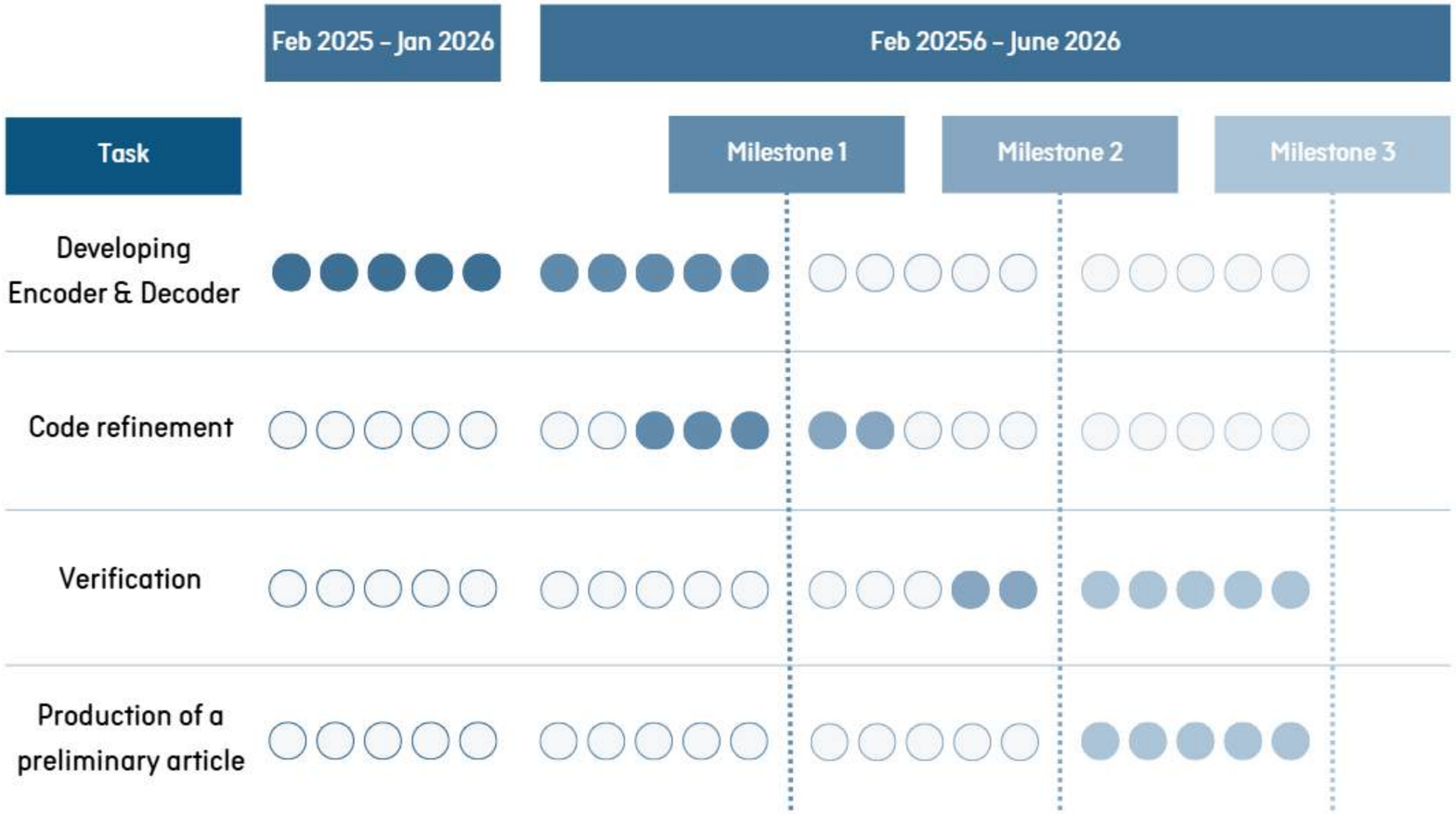
```
rle = (split (sentinel . (>>= count) . run2count . fst) snd) . dropZeros
   where run2count = anaList gRLE
         sentinel = (++ [1,0])
```

Injective !

Milestone 1

What’s next?

Plan


Feb 2025 – Jan 2026
Feb 20256 – June 2026
Task
Milestone 1
Milestone 2
Milestone 3
Developing Encoder & Decoder
Code refinement
Verification
Production of a preliminary article


Thank you!