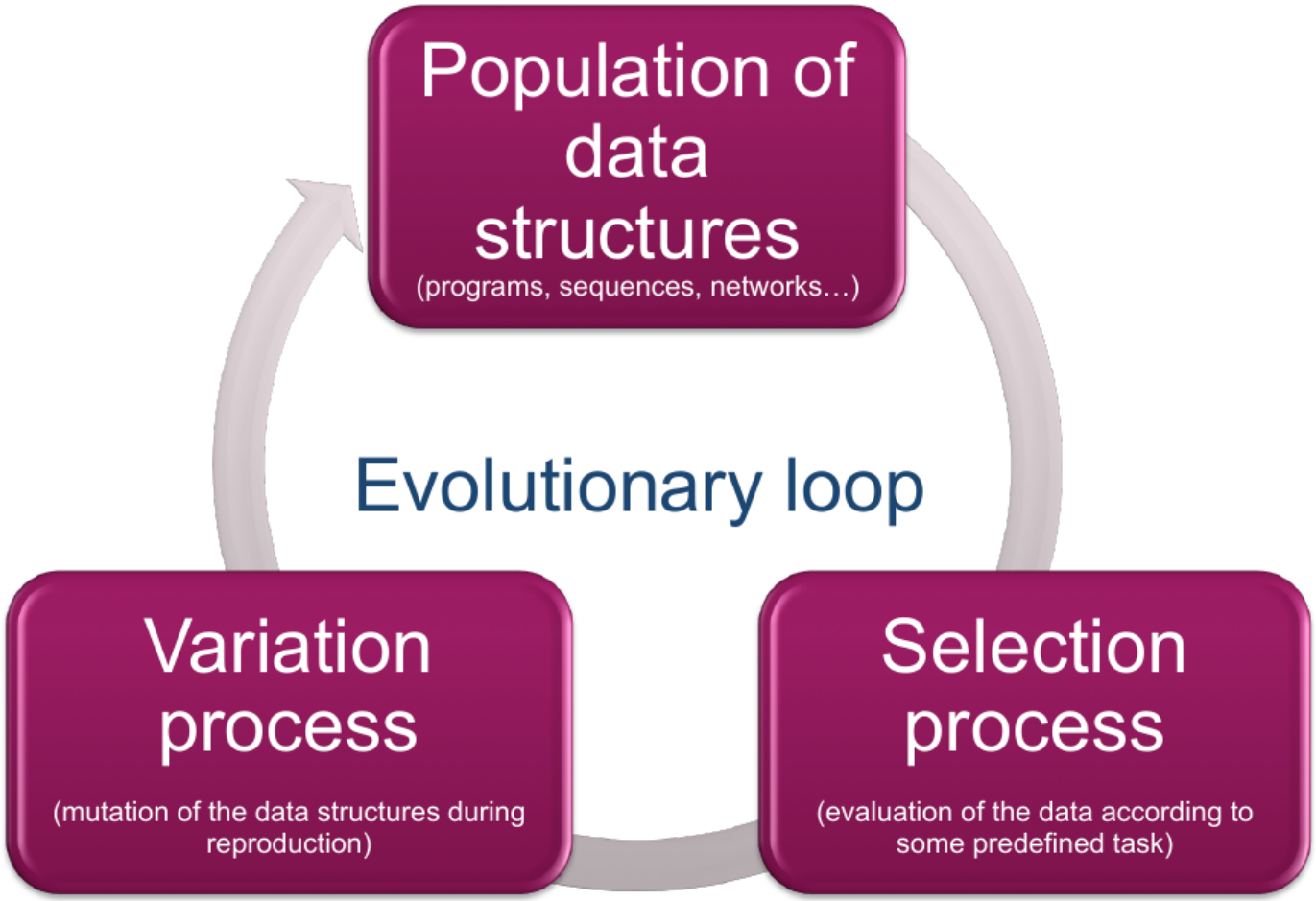
Introducing micro-Aevol: Underlying models, implementation and possible optimizations and parallelization

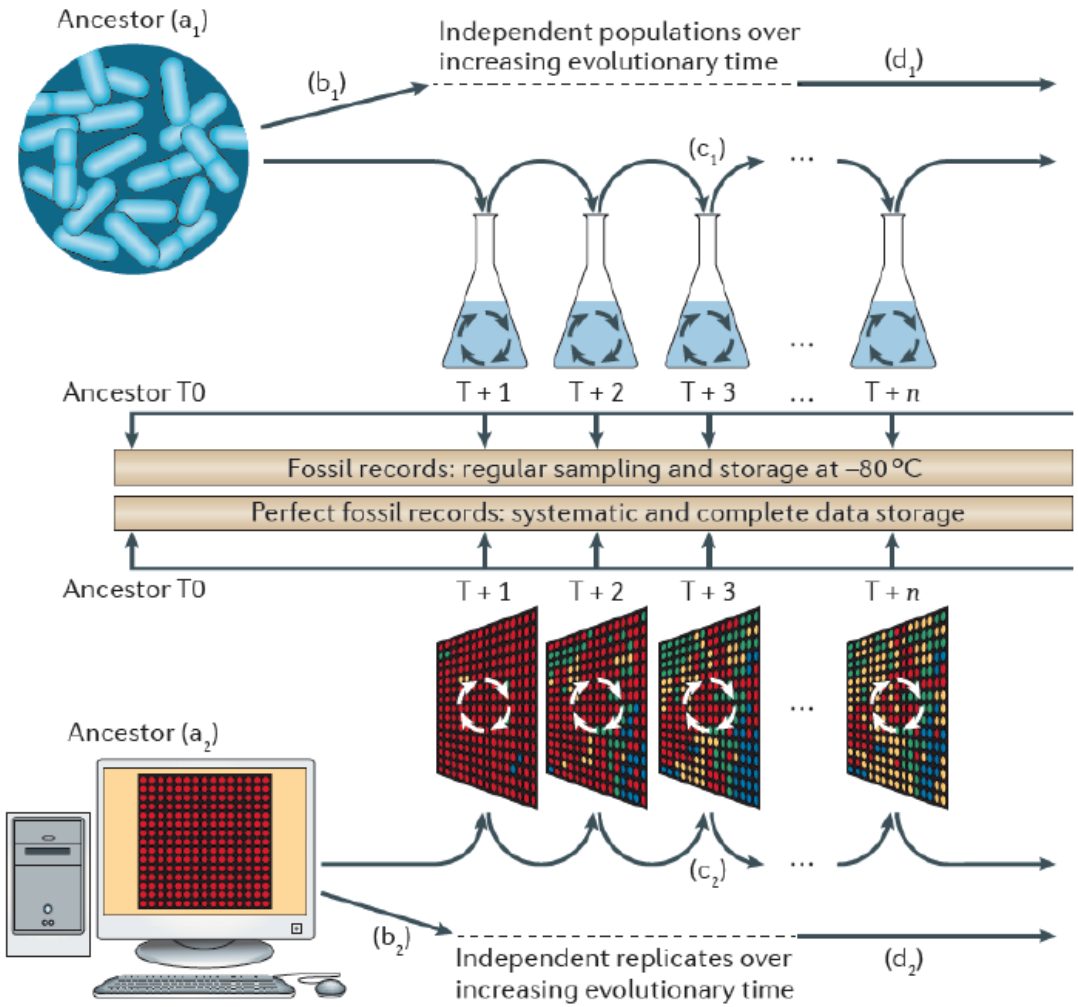
Jonathan Rouzaud-Cornabas

LIRIS / Insa de Lyon – Inria Beagle

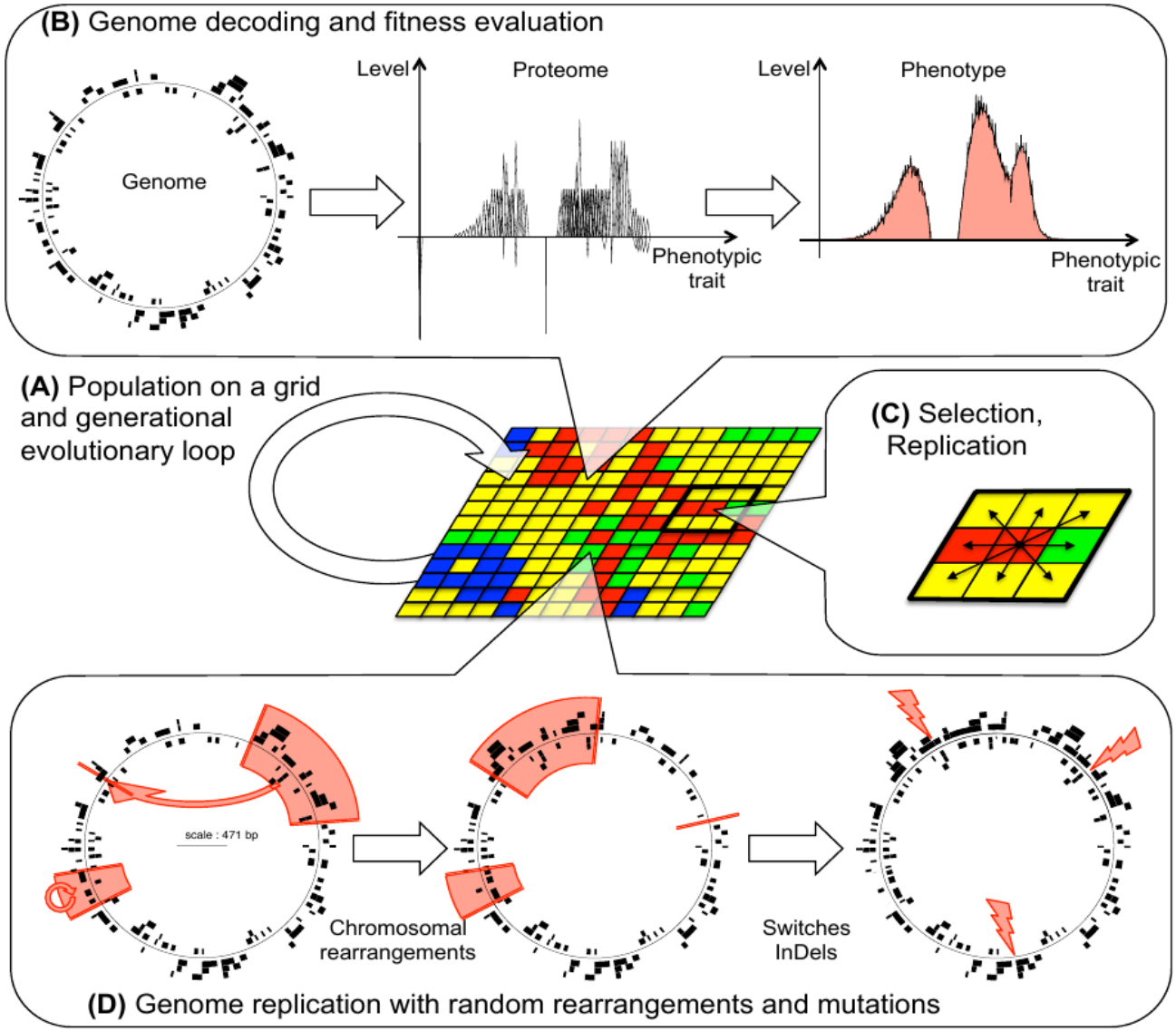
Evolutionary Loop



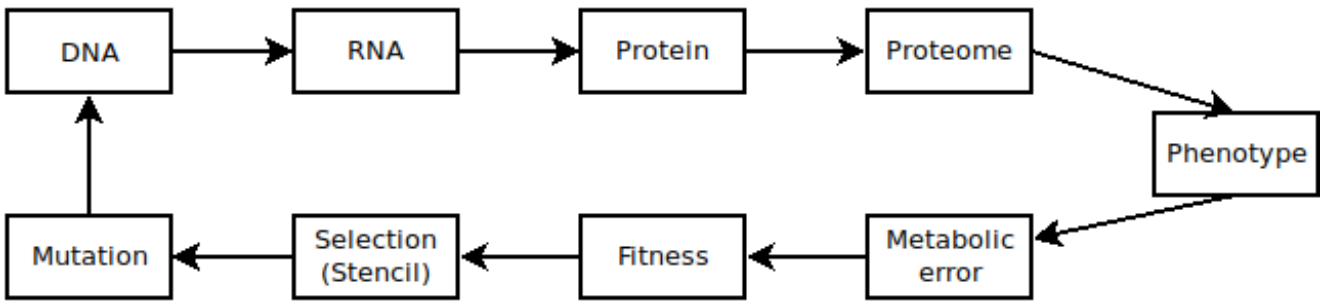
In-silico experimental evolution



The biological model of Aevol

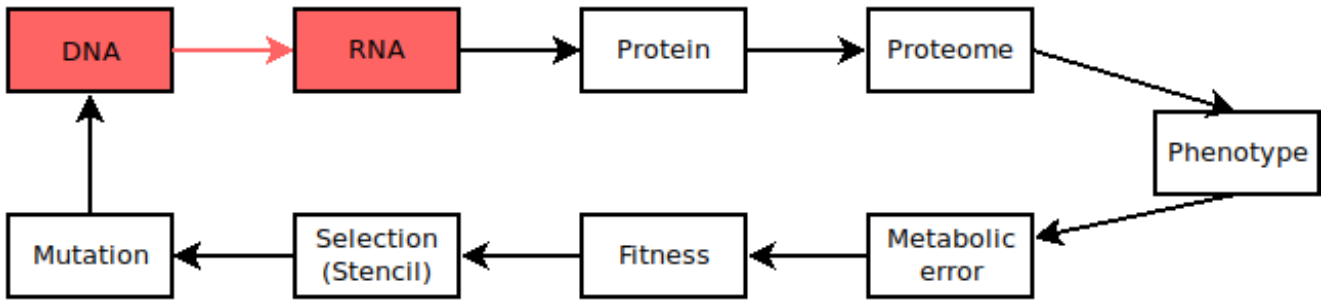


The biological model: A computer science view



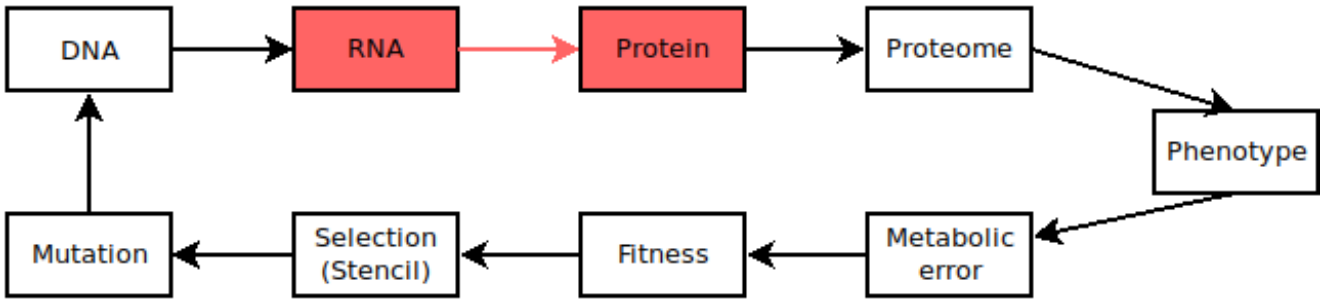
- The workflow is repeated for every cell of the grid
- Each cell contains one and only one organism

The biological model: A computer science view



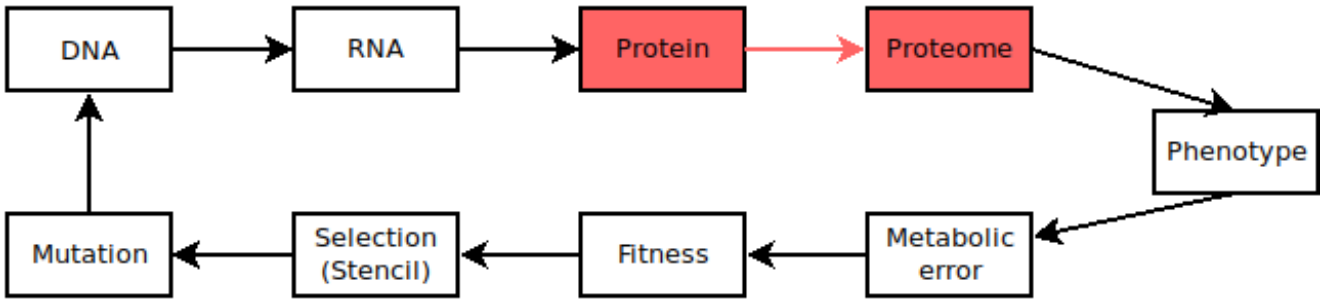
- DNA $\rightarrow$ RNA : **Transcription** : Looking for 2 motifs
  1. Promoter (size = 22) : Hamming distance (predefined motif, max. error = 4)
  2. Terminator (size = 4) : Exact match ($DNA[pos : pos + 4) == NOTDNA[pos + 9 : pos + 5)$)

The biological model: A computer science view



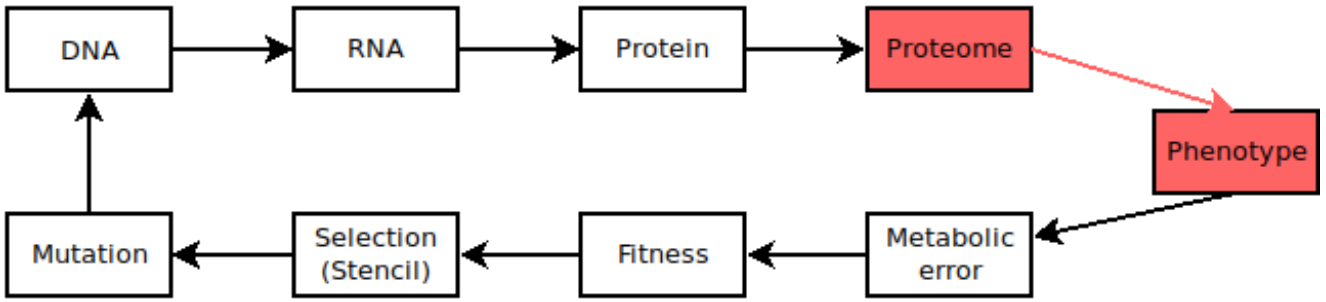
- DNA → RNA : **Transcription** : Looking for 2 motifs
- RNA → Protein : **Translation** : Looking for 2 motifs
  1. Shine Dalgarno (size = 6) + space (size = 4) + Start (size = 3) : Exact match
  2. Stop (size = 3) : Exact match

The biological model: A computer science view



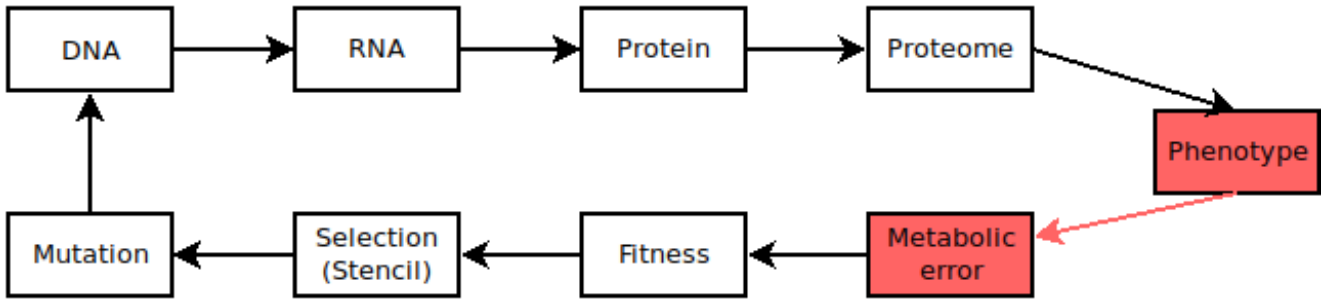
- DNA $\rightarrow$ RNA : **Transcription** : Looking for 2 motifs
- RNA $\rightarrow$ Protein : **Translation** : Looking for 2 motifs
- Protein $\rightarrow$ Proteome : **Folding** :
  - Looking for $N = \frac{Protein\_Length}{3}$ motifs
  - Decoding each codon into M, W, H Amino Acid
  - Translate M, W, H into triangle and approximate them into 300 double array

The biological model: A computer science view



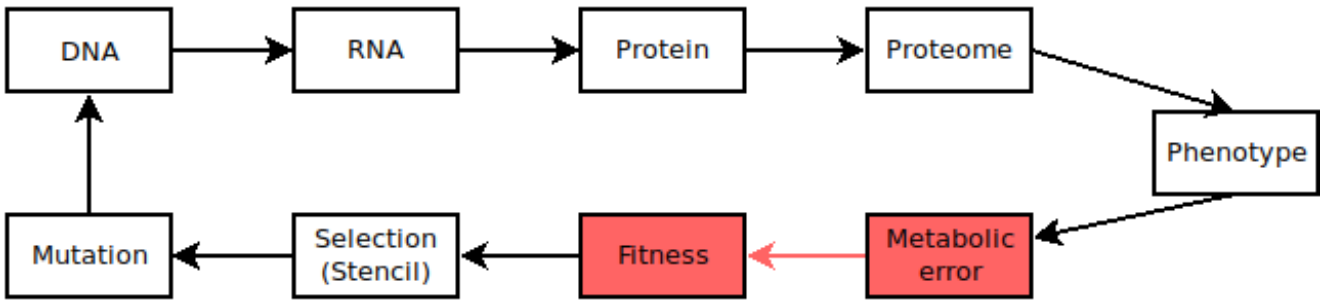
- DNA → RNA : **Transcription** : Looking for 2 motifs
- RNA → Protein : **Translation** : Looking for 2 motifs
- Protein → Proteome : **Folding** :
- Proteome → Phenotype : **Summing arrays** :
  - Looking for $N = \frac{Protein\_Length}{3}$ motifs
  - Decoding each codon into M, W, H Amino Acid
  - Translate M, W, H into triangle and approximate them into 300 double array
  - Summing into a global 300 double array

The biological model: A computer science view



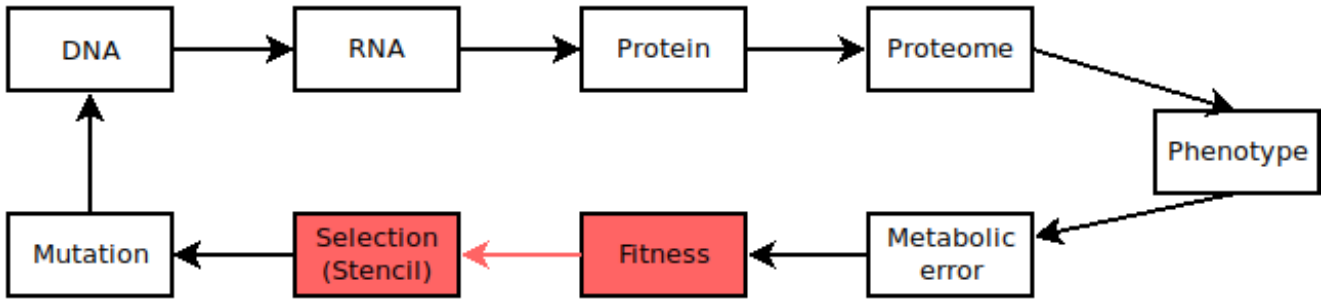
- DNA → RNA : **Transcription** : Looking for 2 motifs
- RNA → Protein : **Translation** : Looking for 2 motifs
- Protein → Proteome : **Folding** :
- Proteome → Phenotype : **Summing arrays** :
- Phenotype → Metabolic error: **Difference between 2 arrays**
  - Phenotypical Target : Pre-defined array built from a sum of gaussian curves
  - Compute the difference between the phenotype and the phenotypical target

The biological model: A computer science view



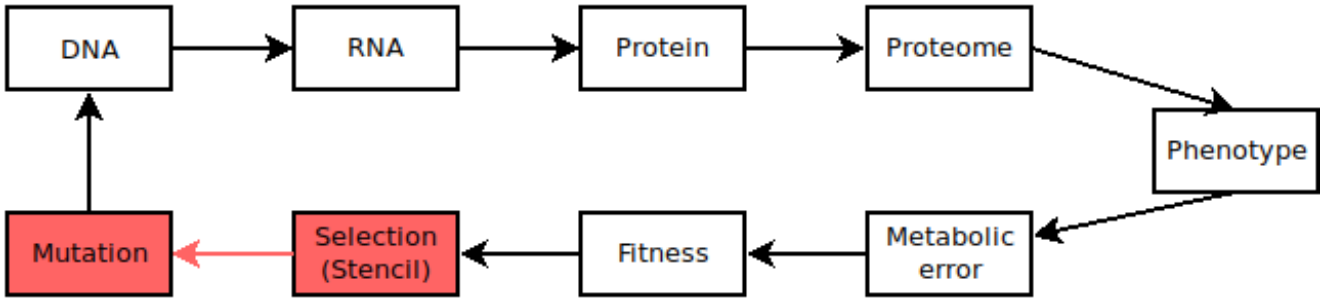
- Metabolic error $\rightarrow$ Fitness:
  - $\exp(-selection\_pressure \times metabolic\_error)$

The biological model: A computer science view



- Metabolic error $\rightarrow$ Fitness:
- Fitness $\rightarrow$ Selection : **Stencil 9-point 2D**
  - Only kernel requiring data coming from other cells of the grid
  - Fetch fitness of the neighboring cells
  - Use the fitness proportionate algorithms to select the organism that will reproduce in the cell
  - Copy the DNA of the selected organism into the cell
  - Delete the old organism

The biological model: A computer science view



- Metabolic error $\rightarrow$ Fitness:
- Fitness $\rightarrow$ Selection : **Stencil 9-point 2D**
- Selection $\rightarrow$ Mutation : **Modify the DNA**
  - Random variation base on different type of mutations

Aevol VS mini-Aevol

- Aevol VS micro-Aevol SLOC : 87,000 VS 2,800
- Simplified model
  - Only one strand of DNA (and not two) *i.e.* the DNA is read one-way and not both-way
  - A reduce mutation sets: no mutation that change the size
  - A lot of advances features are missing: no plasmid, 4bp DNA, ...
- Implementation
  - Lot less robust to input errors
  - Cannot change model parameters during an evolution
  - No phylogenetic tree
  - No postprocessing tool

Optimizations

- Working only on diff (where the DNA has changed during the mutation) *i.e.* do not reparse the whole DNA at each generation (available in the git repository for sequential CPU)
- Implement a new version of the algorithm that reduce the memory reads
- Implement and test different algorithms to evaluate the NUMA effect
- Remove the "modulo" that allows to simulate circular DNA (copy a part of the beginning of the DNA at the end to do it)
- Change the coding of the DNA from a char array to a bitset
- Vectorization of the comparison between DNA and motifs
- Search multiple patterns at once through vectorization
- Evaluate classic mutation operator (malloc+memcpy+mutation) and backtrace mutation operator (malloc+copy)
- One large 1D structure (merge all DNAs into a big DNA) to ease load balancing
- Include metadata (promoter, terminator, start, stop) into a bitset

Parallelization

- OpenMP (parallel for and/or tasks with or without dependencies)
- CUDA
- Kokkos
- MPI
- Hybrid *e.g.* Kokkos CPU+GPU, OpenMP+MPI, ...

- Implement the stencil (*i.e.* asynchronous time between grid cells)

Methodology

1. Use GIT (or another versioning software)
2. For each optimization/parallelization, tests its performance, scalability (weak and strong) and create visualization of them (*e.g.*, through one or multiple Jupyter notebook)
3. You need to verify the reproducibility of your code *i.e.*, same parameters must lead to same results

Simulation parameters:

- Size of the DNA ($-g$): 500, 5000, 50,000 (you can go up to 500,000)
- Population size ($-w$ and $-h$) : 32x32, 128x128, 256x256 (you can up to 1024x1024)
- Mutation rate ($-m$) : 0.0001, 0.00001, 0.000001