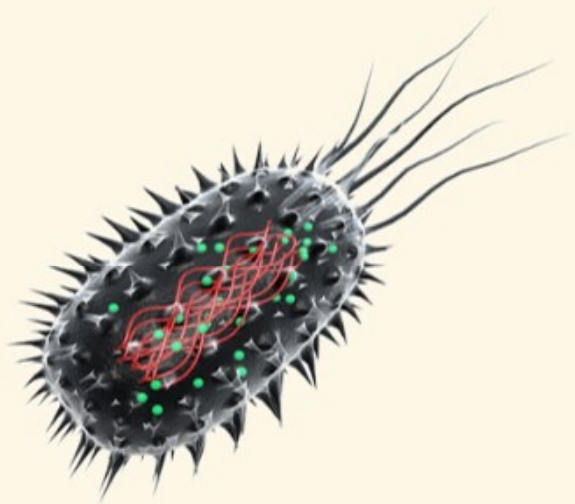
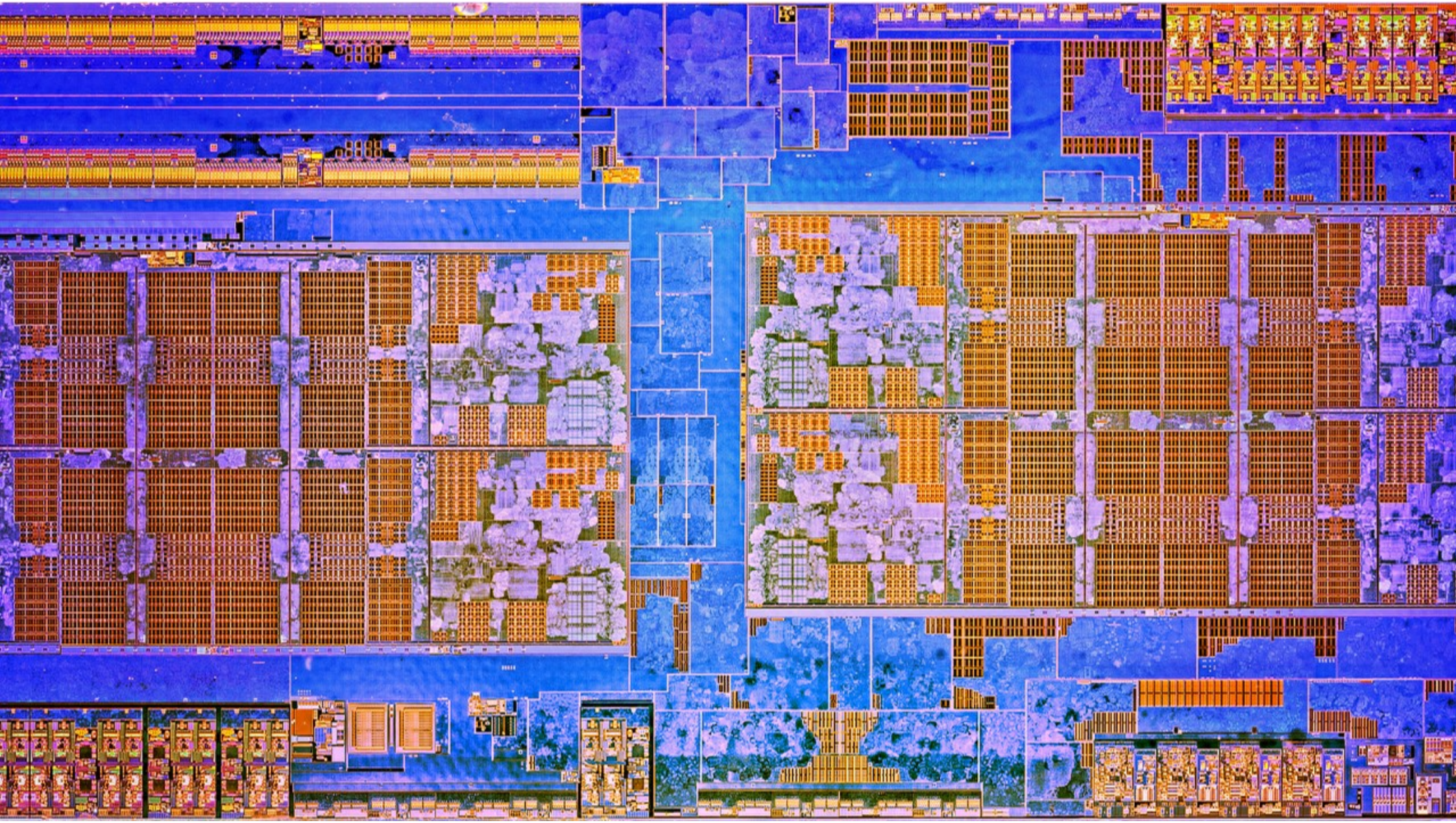
(RE)PROGRAMMING BIOLOGY

(BIOLOGICAL COMPUTING IN F#)

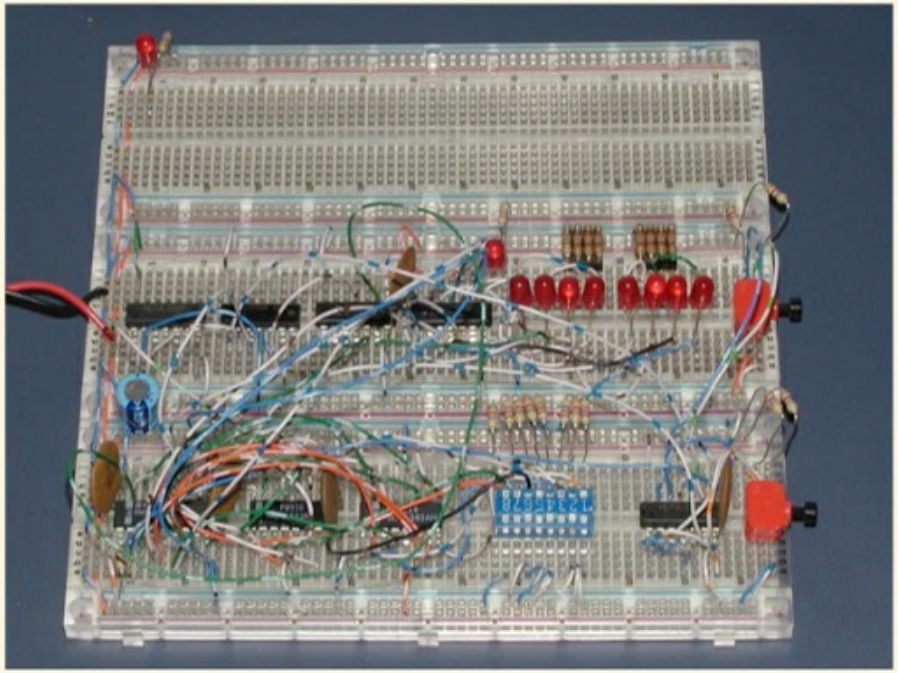


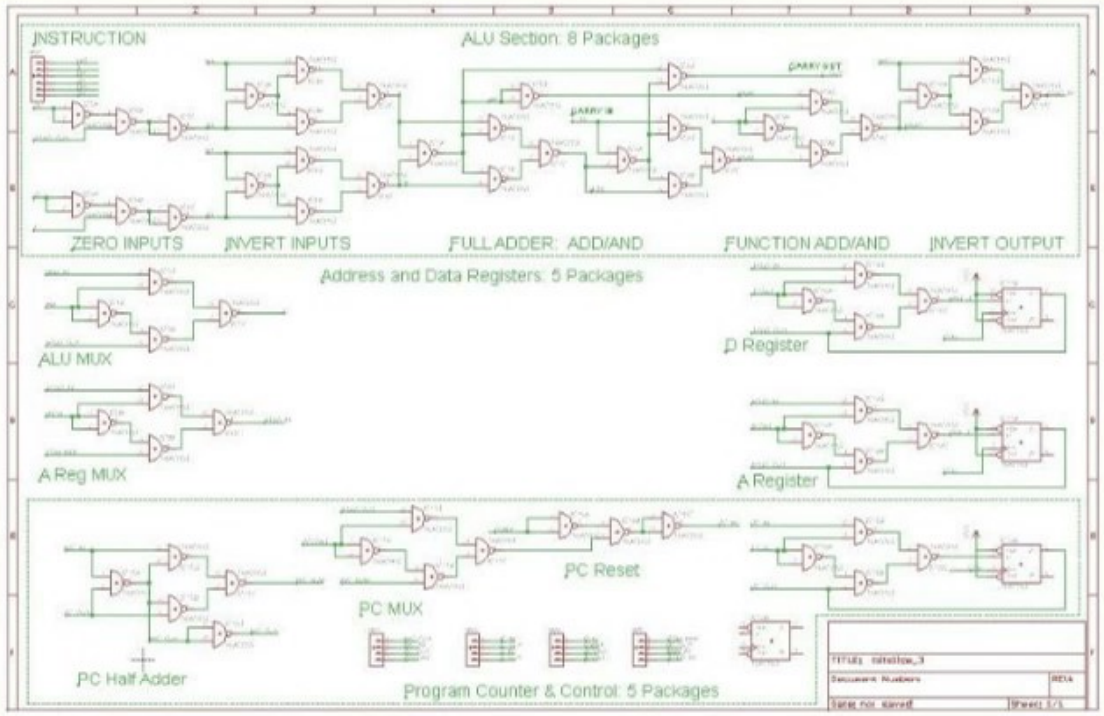
Colin Gravill
Biological Computation Group
Microsoft Research

PROGRAMMING SILICON (CHIPS)



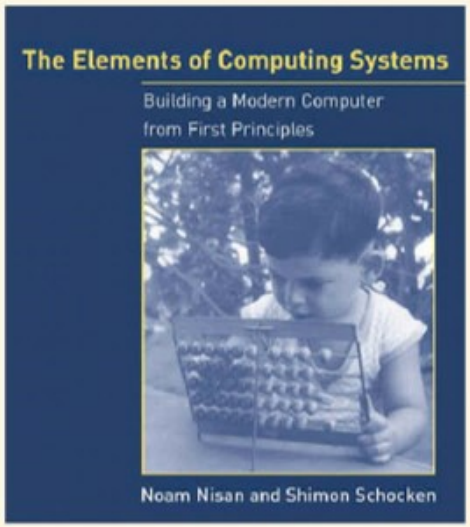
PROGRAMMING SILICON (SMALLER)

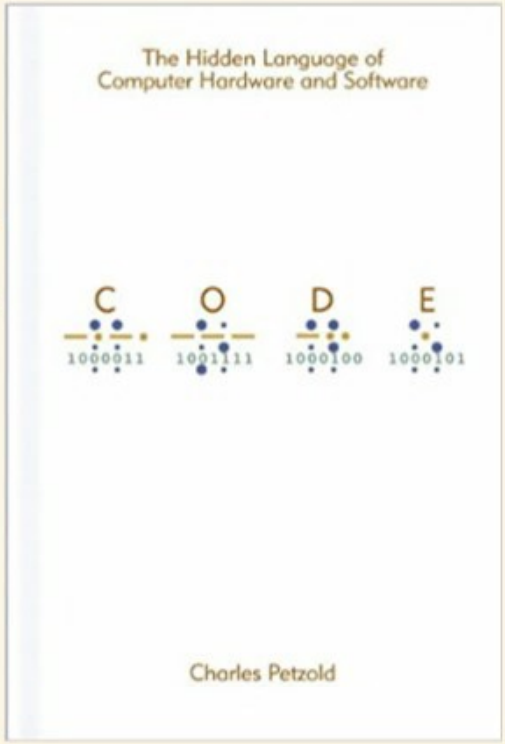




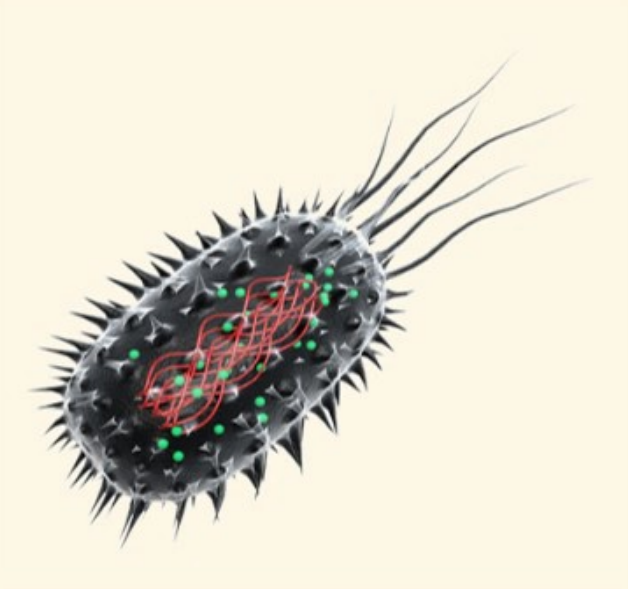
http://nand2tetris-questions-and-answers-forum.32033.n3.nabble.com/Physical-implementation-of-NAND-and-DFF-primitives-td4028219.html

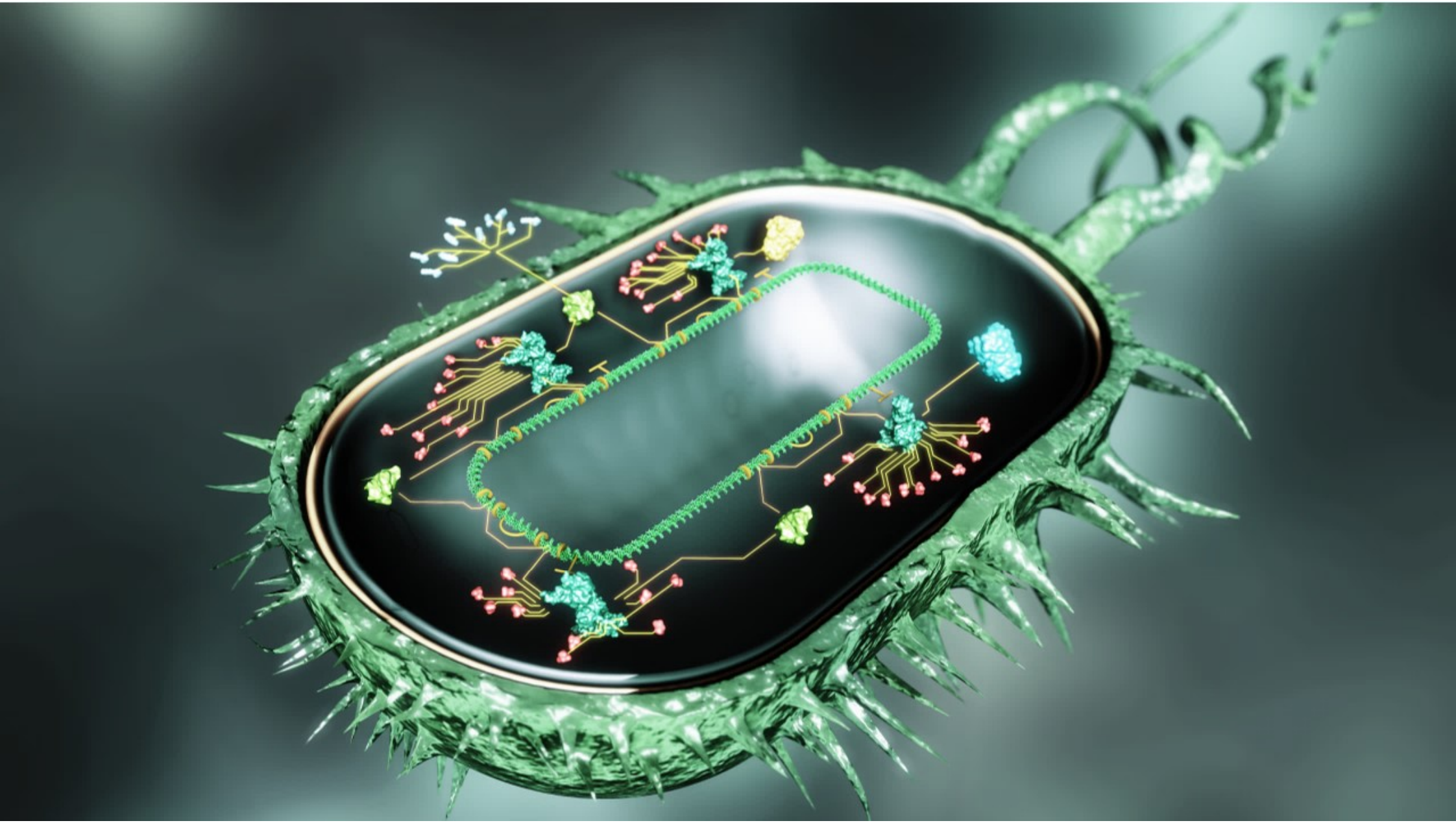
PROGRAMMING SILICON (SMALLER)



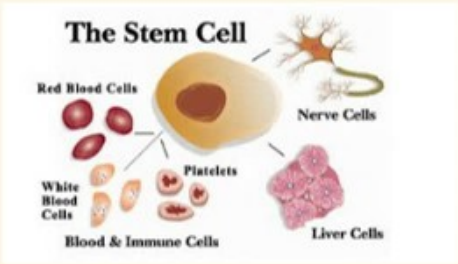


NOW BIOLOGY





BIOLOGICAL COMPUTATION

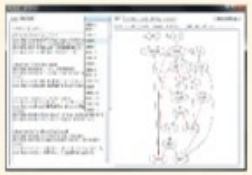


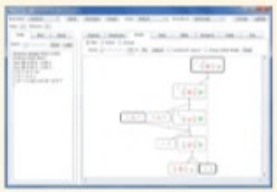
Development

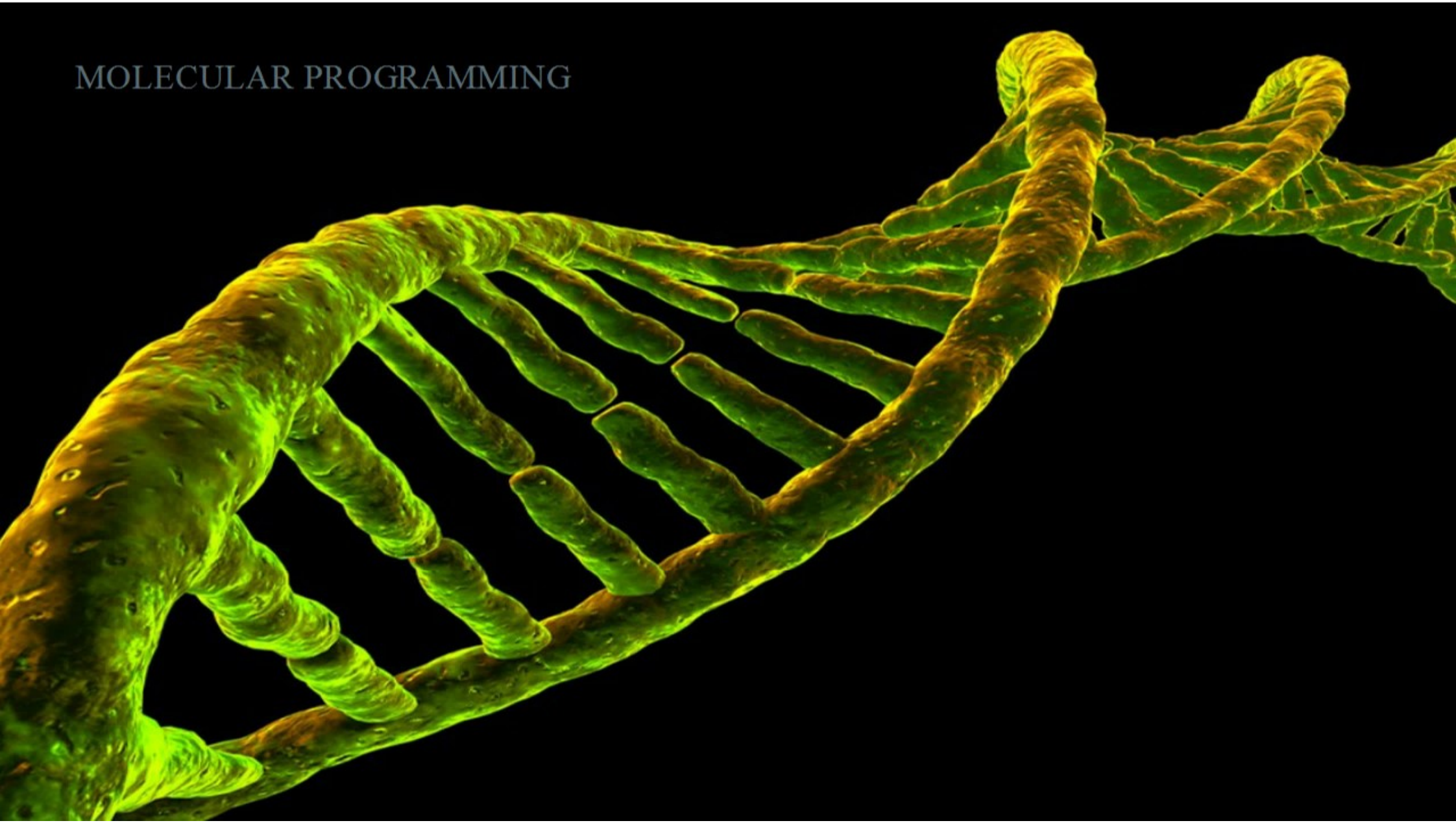
Molecular Programming

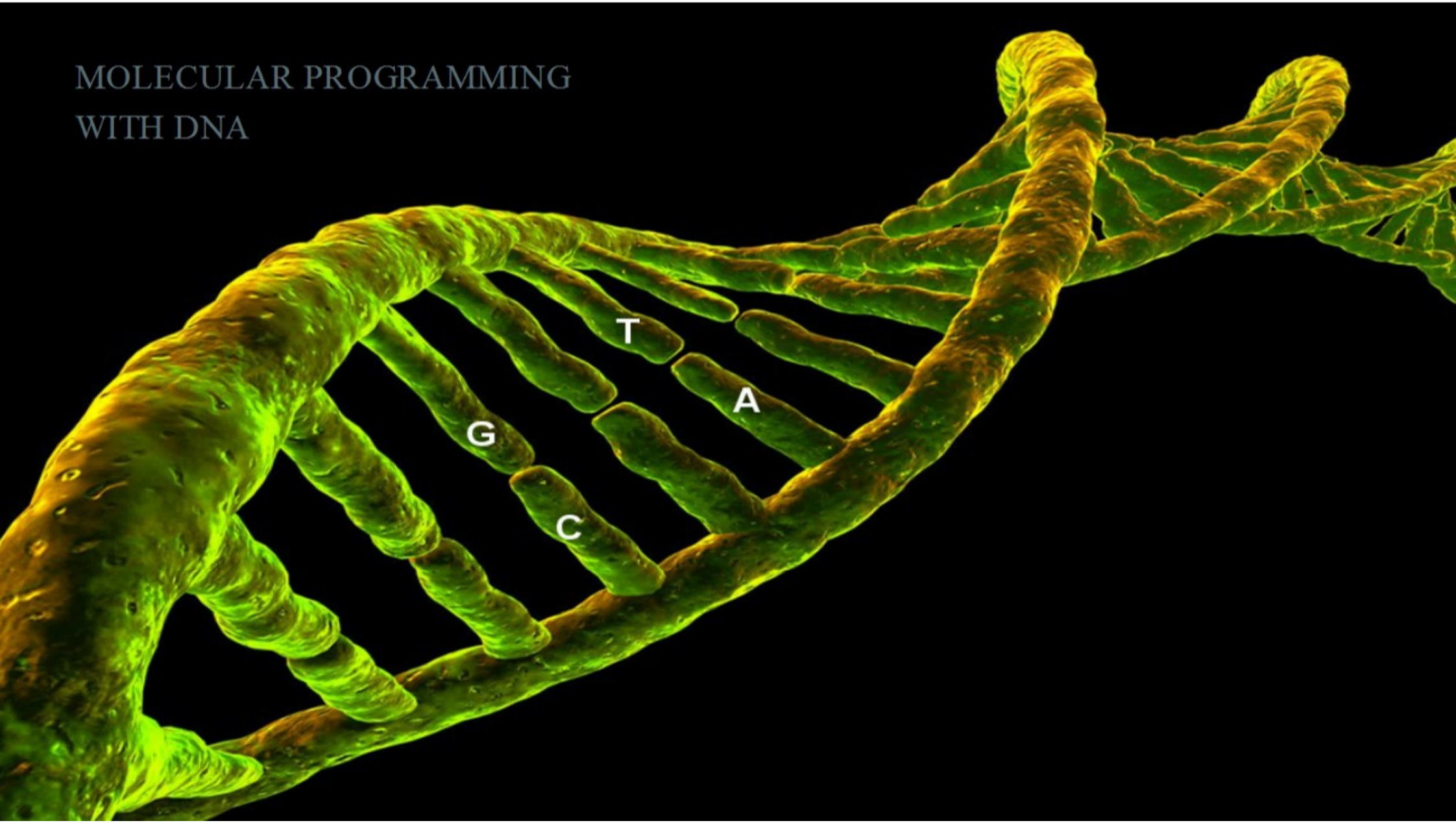
↓

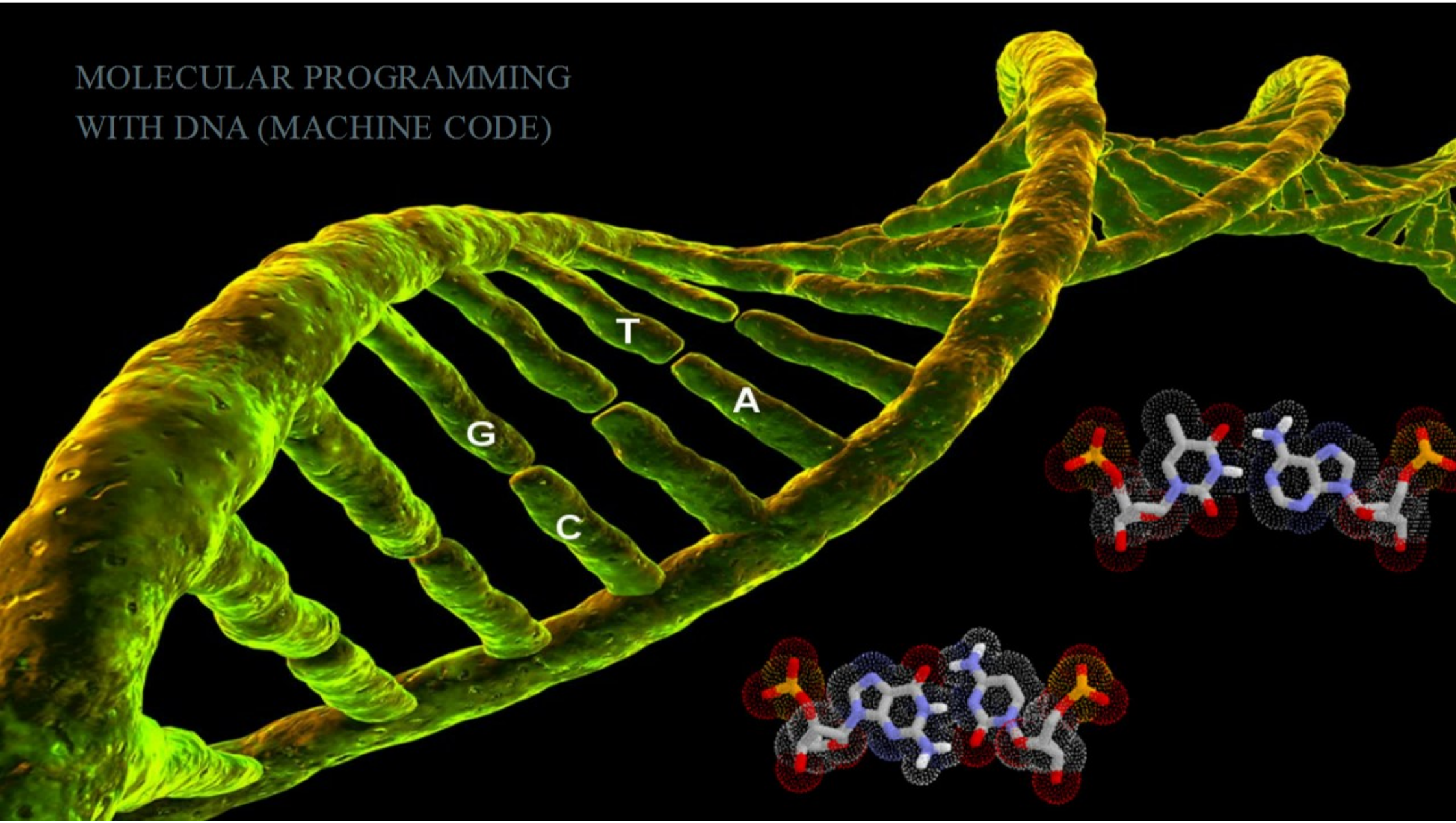
COMPUTER AIDED DESIGN TOOLS

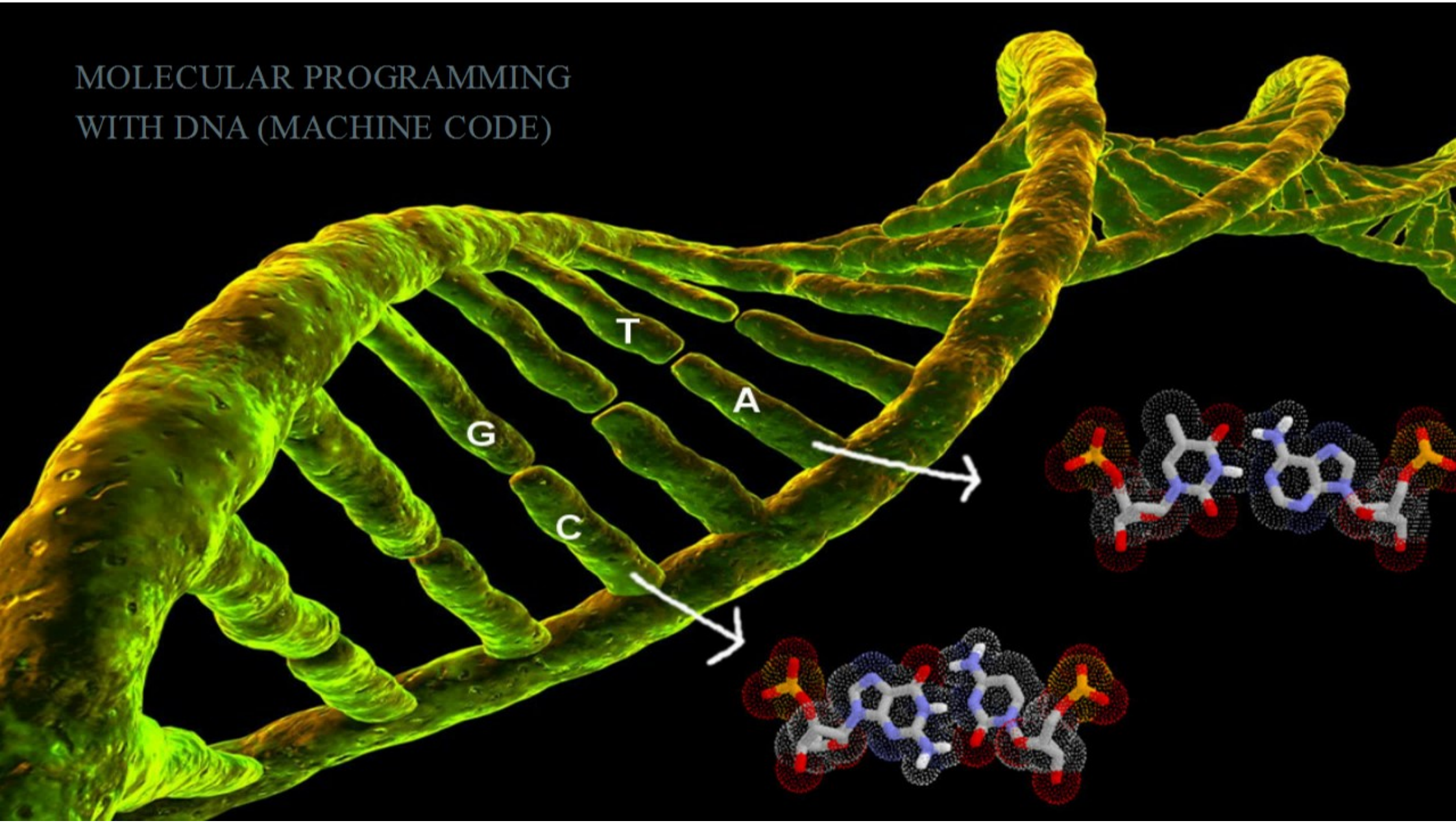




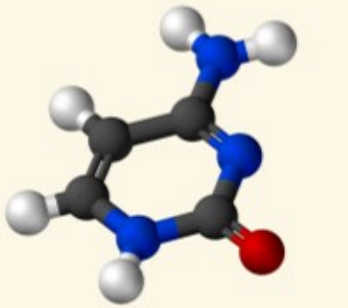

MOLECULAR PROGRAMMING


MOLECULAR PROGRAMMING
WITH DNA
T
A
G
C

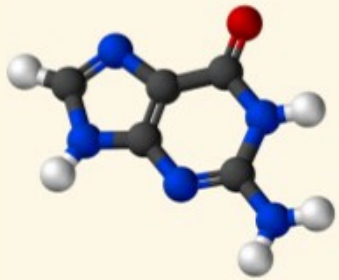

MOLECULAR PROGRAMMING
WITH DNA (MACHINE CODE)
T
A
G
C


MOLECULAR PROGRAMMING
WITH DNA (MACHINE CODE)
T
A
G
C

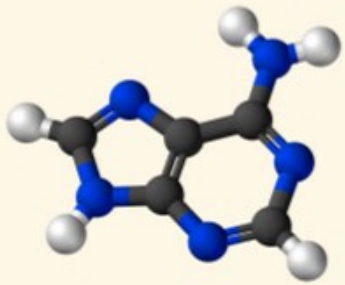
REPURPOSING DNA



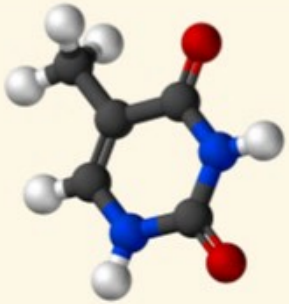
C - Cytosine



G - Guanine



A - Adenine



T - Thymine

APPEARS AS C-G, G-C, A-T, AND T-A, UNRAVELED:

GGCTTAAGTTTTGAATTGTT
CCGAATTCAAAACTTAACAA

DNA STRAND AFFINITY

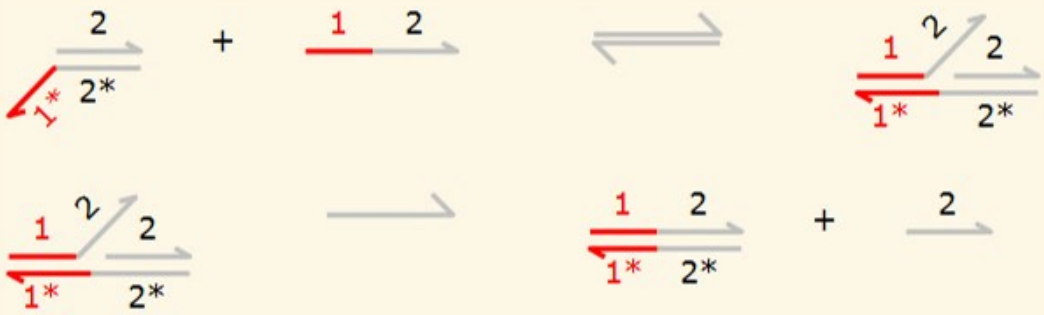
Short complementary segments bind *reversibly*

Long complementary segments bind *irreversibly*

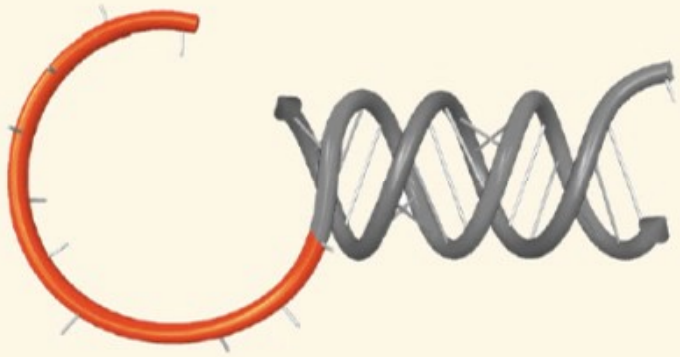
MOLECULAR DISPLACEMENT

Video not supported

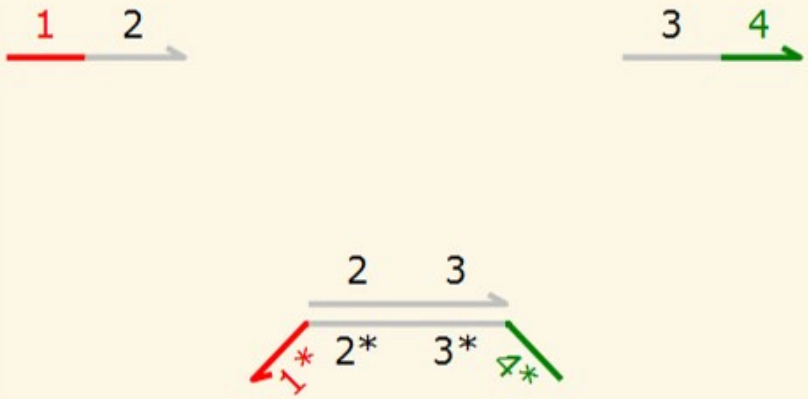
REACTIONS



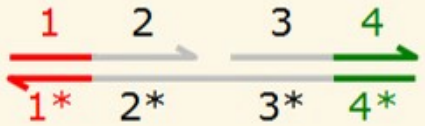
Describes strands like this interacting:



AND CIRCUIT



AND CIRCUIT



DOMAIN SPECIFIC LANGUAGE

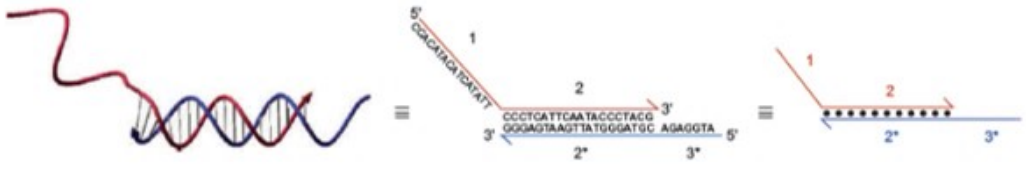
DYNAMIC STRAND DISPLACEMENT (DSD) TOOL

http://research.microsoft.com/dna
Search: Visual DSD

DSD RELATED PUBLICATIONS

Dynamic DNA nanotechnology using strand displacement reactions

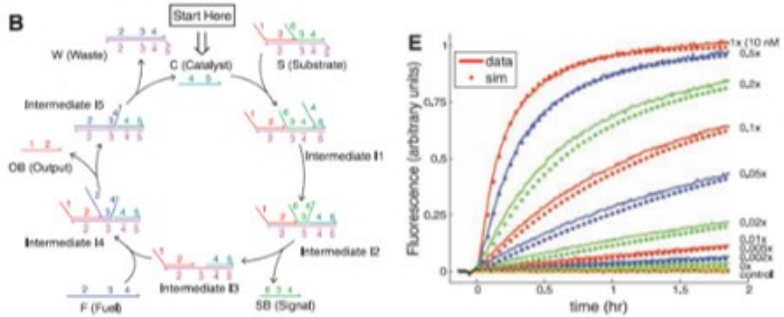
David Yu Zhang[1] and Georg Seelig[2]



NATURE CHEMISTRY | VOL 3 | FEBRUARY 2011

Engineering Entropy-Driven Reactions and Networks Catalyzed by DNA

David Yu Zhang,[1]† Andrew J. Turberfield,[2] Bernard Yurke,[3]* Erik Winfree[1]†



SCIENCE VOL 318 16 NOVEMBER 2007

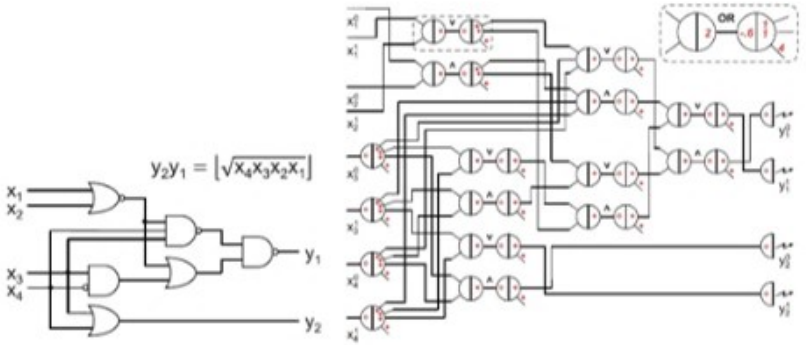
Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades

Lulu Qian[1] and Erik Winfree[1,2,3]*

CIRCUITS DESIGNED WITH DSD

Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades

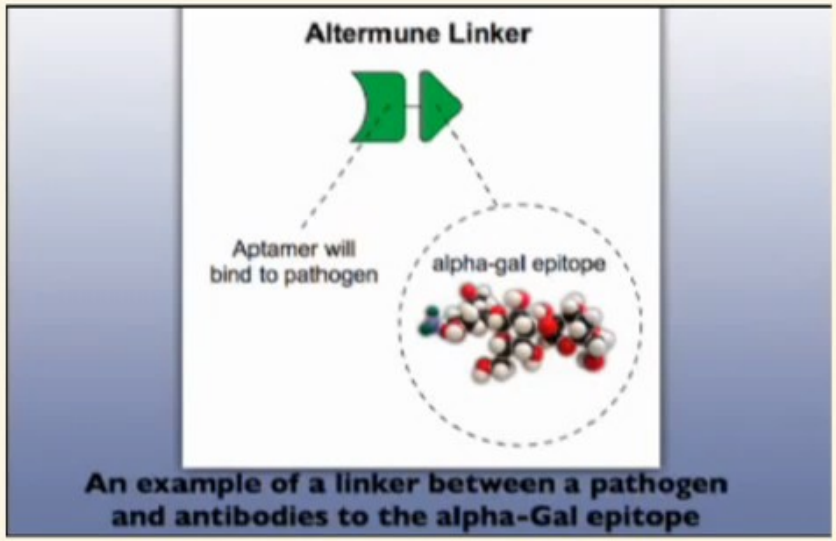
Lulu Qian[1] and Erik Winfree[1,2,3]



DNA APTAMERS

DNA aptamer binds to:

- a pathogen (something bad)
- a molecule our immune system already recognizes and immediately removes (eats) along with anything attached to it

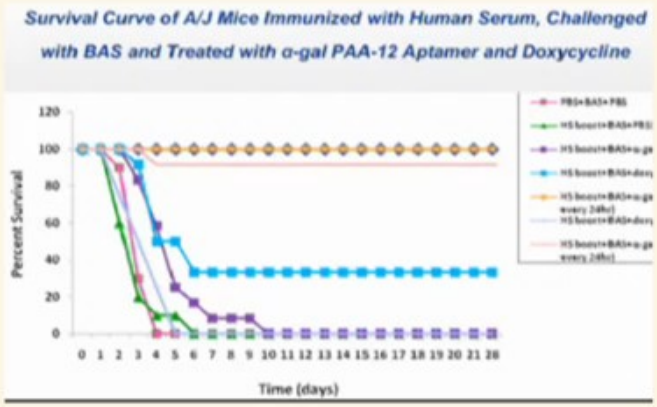


Patented (US20100285052) by Kary Mullis
(incidentally, also Nobel prize for inventing the Polymerase Chain Reaction)

DNA APTAMERS

Result: instant immunity

- Mice poisoned with Anthrax plus aptamer (100% survial)
- a molecule our immune system already recognizes and immediately removes (eats) along with anything attached to it

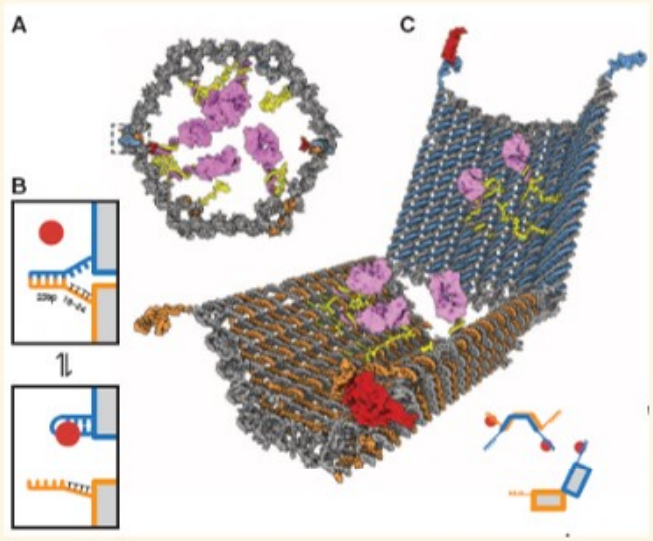


Patented (US20100285052) by Kary Mullis
(incidentally, also Nobel prize for inventing the Polymerase Chain Reaction)

DNA ORIGAMI BOX (DELIVERY)

A Logic-Gated Nanorobot for Targeted Transport of Molecular Payloads

Shawn M. Douglas,* Ido Bachelet,* George M. Church†



CORRECTNESS

At least two levels:

- The systems we describe
- The simulation of those systems

FSCHECK

- F# port of QuickCheck from Haskell
- Allows property based testing
- Example: List.rev is idempotent
  - ∀ l. rev (rev l) = l
- Testing:
  - Generate a large number of lists
  - Compute both sides
  - Compare

FSCHECK (GENERATORS)

To generate a large number of list of pairs:

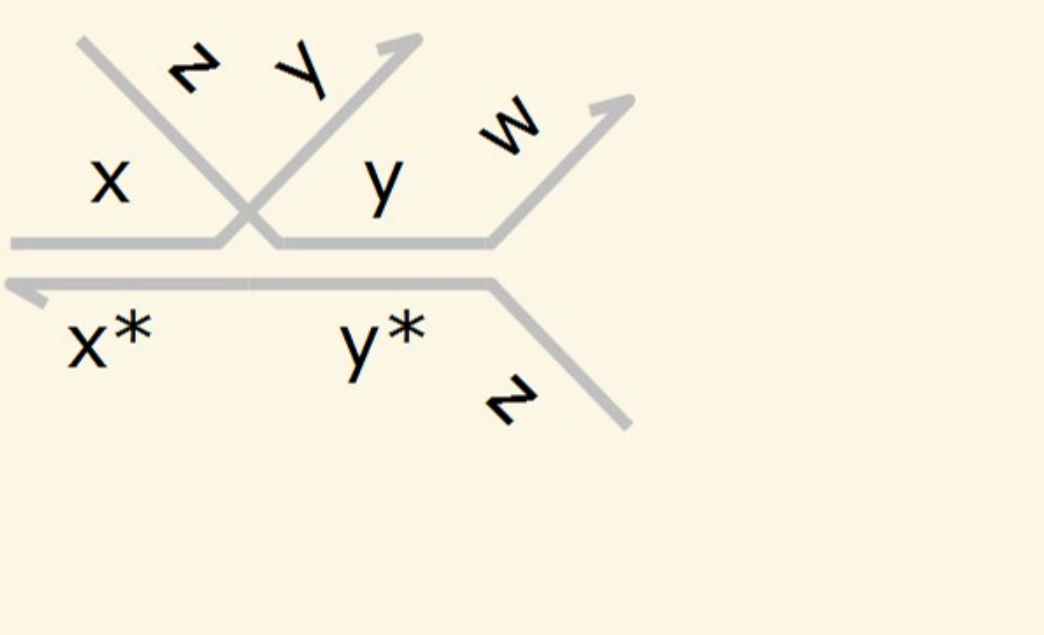
```
let gen_int = Arb.generate<int>

let gen_pair =
  Gen.map2 (fun a b -> (a,b)) gen_int gen_int

let gen_list = gen_int |> Gen.listOf
```

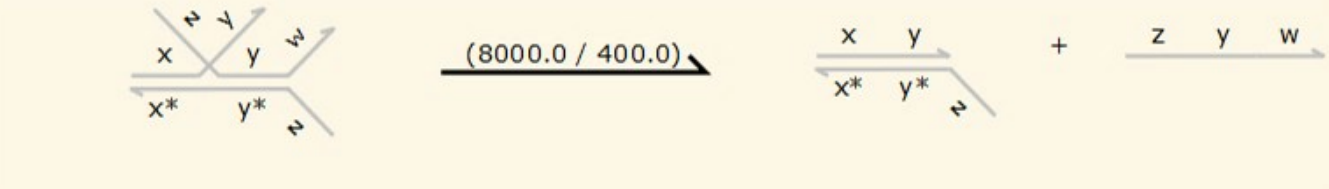
FSCHECK (MELTING)

Obtain the constituent strands of a DSD molecule



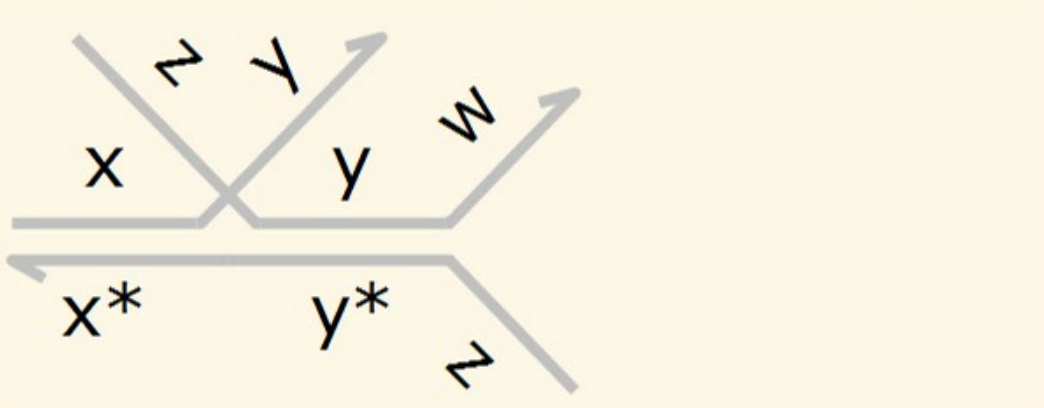
FSCHECK (MELTING PROPERTY)

- Reactions preserve strands and thus melting
  - $\forall a \rightarrow b.\ melt(a) = melt(b)$
- Testing:
  - Generate a large number of reactions
  - Melt both sides
  - Compare

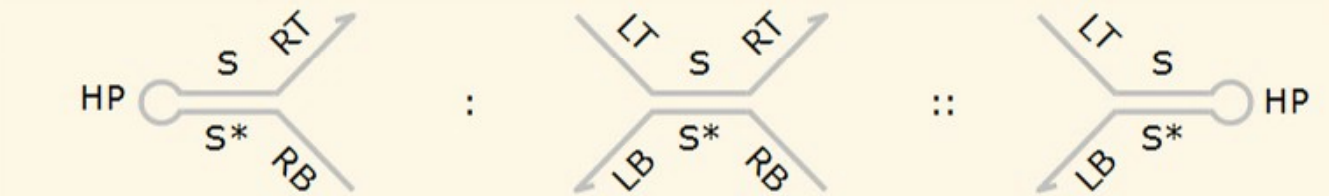


FSCHECK (DATA STRUCTURES)

We have a textual representation of species



[x]<y>:<z>[y]<w>{z}



<HP}[S]<RT>{RB} {LB}<LT>[S]<RT>{RB} {LB}<LT>[S]{HP>

This representation mirrors the data structures

FSCHECK (DATA STRUCTURES)

From segment.ml

```
type side = Left | Right
type t =
  | Double of (domain list * domain list * domain list * domain list * domain list)
  | Hairpin of (domain list * domain list * domain list * domain list * side * bool)
```

From gate.ml

```
type t = segment list list
```

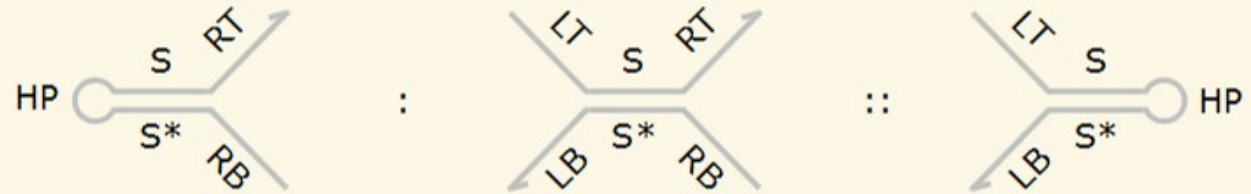
FSCHECK (DATA STRUCTURES)

From segment.ml

```
type side = Left | Right
type t =
  | Double of (domain list * domain list * domain list * domain list * domain list)
  | Hairpin of (domain list * domain list * domain list * domain list * side * bool)
```

From gate.ml

```
type t = segment list list
```



Hairpin(RB,RT,S,HP,Left,false)

Double(LB,LT,S,RT,RB)

Hairpin(LB,LT,S,HP,Right,true)

FSCHECK (GENERATORS)

Generating segments:

```
let dsd_segment_middle s =
    let ds = dsd_domain_list (s/5)
    Gen.map5 (fun lb lt s rt rb -> Segment.Double (lb, lt, s, rt, rb)) ds ds ds ds ds

let dsd_hairpin_left s =
    let ds = dsd_domain_list (s/4)
    Gen.map4 (fun ob ot s hp -> Segment.Hairpin (ob, ot, s, hp, Segment.Left, false)) ds ds ds d

let dsd_hairpin_right s =
    let ds = dsd_domain_list (s/4)
    Gen.map4 (fun ob ot s hp -> Segment.Hairpin (ob, ot, s, hp, Segment.Right, true)) ds ds ds

let dsd_segment_left s =
    Gen.oneof [ dsd_segment_middle s
                dsd_hairpin_left s ]

let dsd_segment_right s =
    Gen.oneof [ dsd_segment_middle s
                dsd_hairpin_right s ]
```

FSCHECK (GENERATORS)

Generating segments:

```
let dsd_segments_middle s =
    let s = s|>float|>sqrt|>int
    Gen.nonEmptyListOf (dsd_segment_middle s) |> Gen.resize s

let dsd_segments_left s =
    let x = Gen.eval s (Random.newSeed ()) (dsd_segment_left s)
    let s = s|>float|>sqrt|>int
    let xs = Gen.eval s (Random.newSeed ()) (dsd_segment_middle s |> Gen.listOf)
    gen { return x::xs }

let dsd_segments_right s =
    let x = Gen.eval s (Random.newSeed ()) (dsd_segment_right s)
    let s = s|>float|>sqrt|>int
    let xs = Gen.eval s (Random.newSeed ()) (dsd_segment_middle s |> Gen.listOf)
    gen { return xs@[x] }
```

FSCHECK (GENERATORS)

Generating segments:

```
let dsd_gate_left s =
    let x = Gen.eval s (Random.newSeed ()) (dsd_segments_left s)
    let s = s|>float|>sqrt|>int
    let xs = Gen.eval s (Random.newSeed ()) (dsd_segments_middle s |> Gen.listOf)
    gen { return x::xs }

let dsd_gate_right s =
    let x = Gen.eval s (Random.newSeed ()) (dsd_segments_right s)
    let s = s|>float|>sqrt|>int
    let xs = Gen.eval s (Random.newSeed ()) (dsd_segments_middle s |> Gen.listOf)
    gen { return xs@[x] }

let dsd_gate s =
    Gen.oneof [ dsd_gate_left s
                dsd_gate_right s ]
```

CORRECTNESS

At least two levels:

- The systems we describe
- The simulation of those systems

A DNA CONSENSUS ALGORITHM

Y (Minority)

X (Majority)

$$X + Y \rightarrow 2B$$
$$X + B \rightarrow 2X$$
$$Y + B \rightarrow 2Y$$

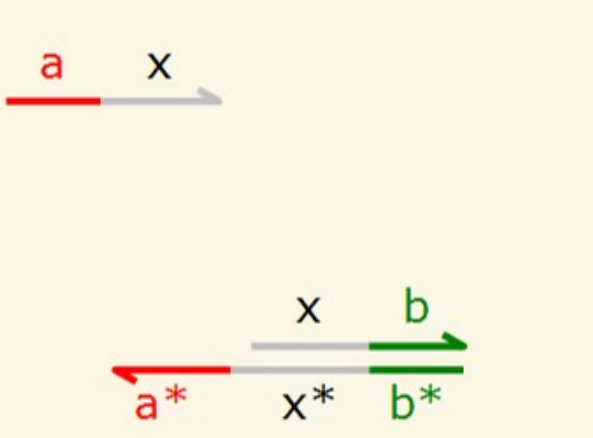
X (Majority)

A DNA CONSENSUS ALGORITHM

Video not supported

MOLECULAR COMPONENTS

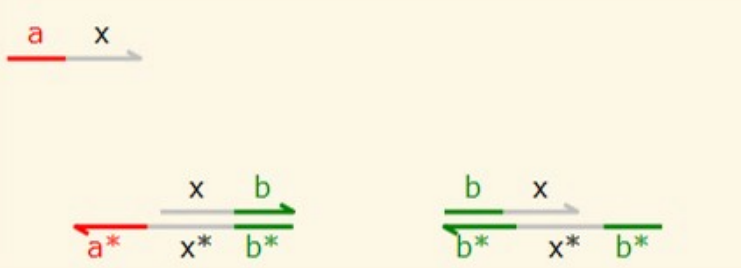
Possible to encode arbitrary reactions
A useful motif: (called seesaw)



This implements a ↔ b'

MOLECULAR COMPONENTS

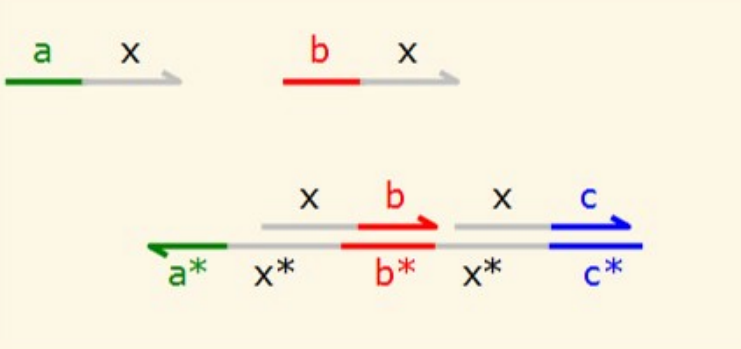
Possible to encode arbitrary reactions
A useful motif: (called seesaw)



This implements a ↔ b

MOLECULAR COMPONENTS

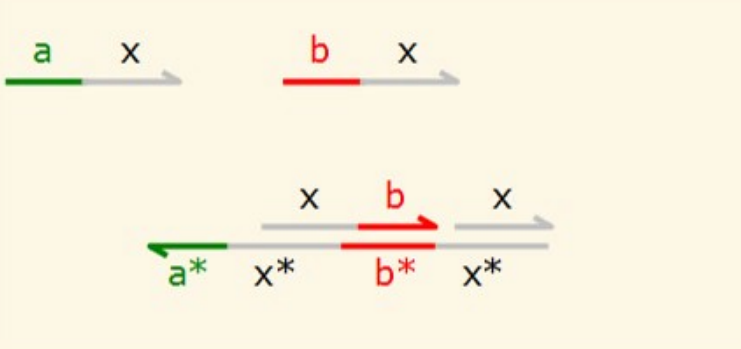
Possible to encode arbitrary reactions
Seesaws can be chained



This implements a + b ↔ b' + c'

MOLECULAR COMPONENTS

Possible to encode arbitrary reactions
Seesaws can be made irreversible



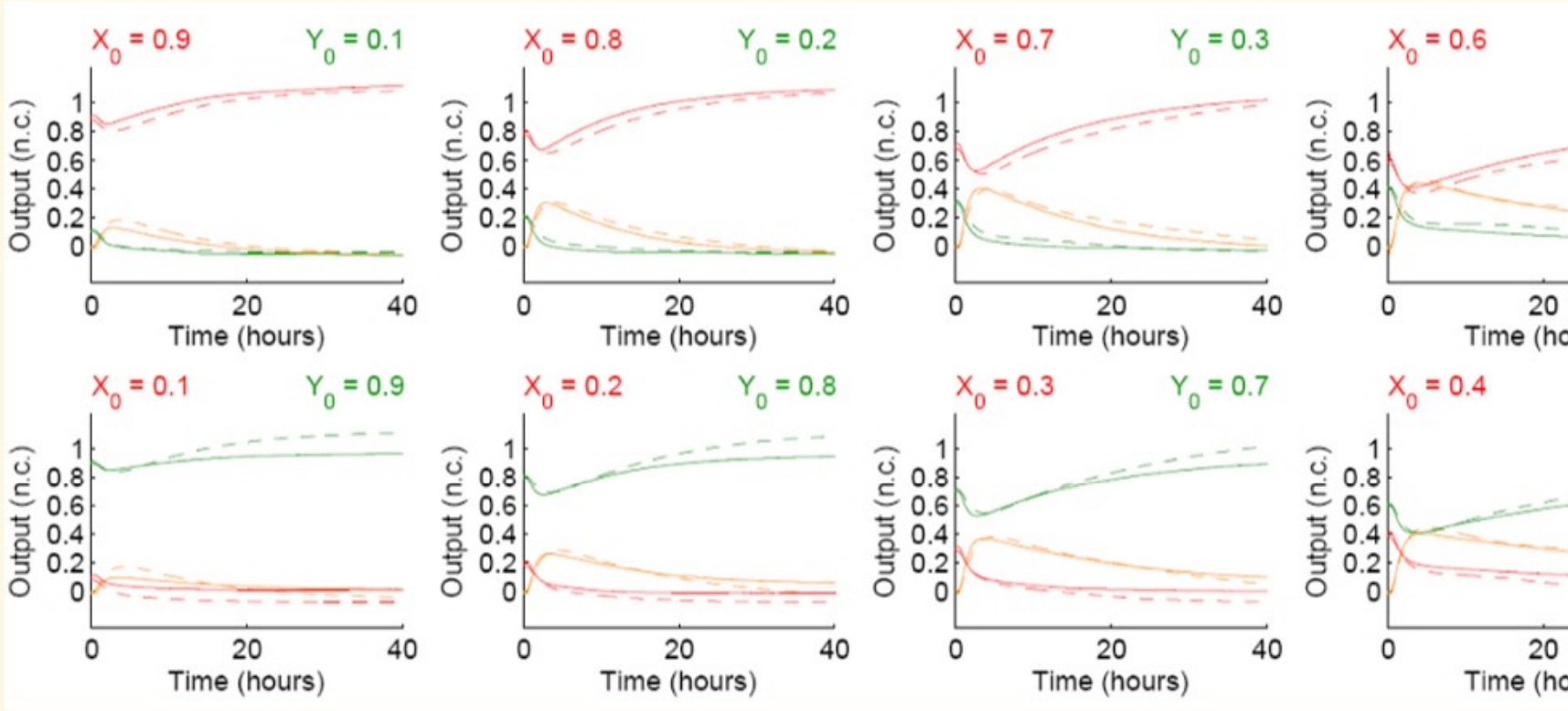
This implements a + b → b' + w

MOLECULAR COMPONENTS

- $X + Y \rightarrow 2B$
- $X + B \rightarrow 2X$
- $Y + B \rightarrow 2Y$

PREDICTING DYNAMICS OF THE FULL CIRCUIT

$$X + Y \rightarrow 2B \qquad X + B \rightarrow 2X \qquad Y + B \rightarrow 2Y$$



— Experimental Data
- - - Model prediction

ANALYSING FUNCTION: Z3 SMT SOLVER

Satisfiability Modulo Theories

FAQ (TO THE SMT SOLVER)

- Is there a trace of this program that leads to buffer overflow?
- Is there an execution of this protocol that reveals the secrets?
- Is there a command line argument that will crash this program?
- Is there a program implementing this function?
- Is there a program with this behaviour?
- Is there a program solving this problem?

FAQ (TO THE SMT SOLVER)

- Is there a trace of this program that leads to buffer overflow?
  - $\exists t.\ t \in \text{Traces}(P) \wedge \text{overflow}(t)$
- Need a theory rich enough to express Traces and overflow
- A negative answer is a proof of absence of buffer overflows

Execution trace $\leftrightarrow$ series of states of biological system

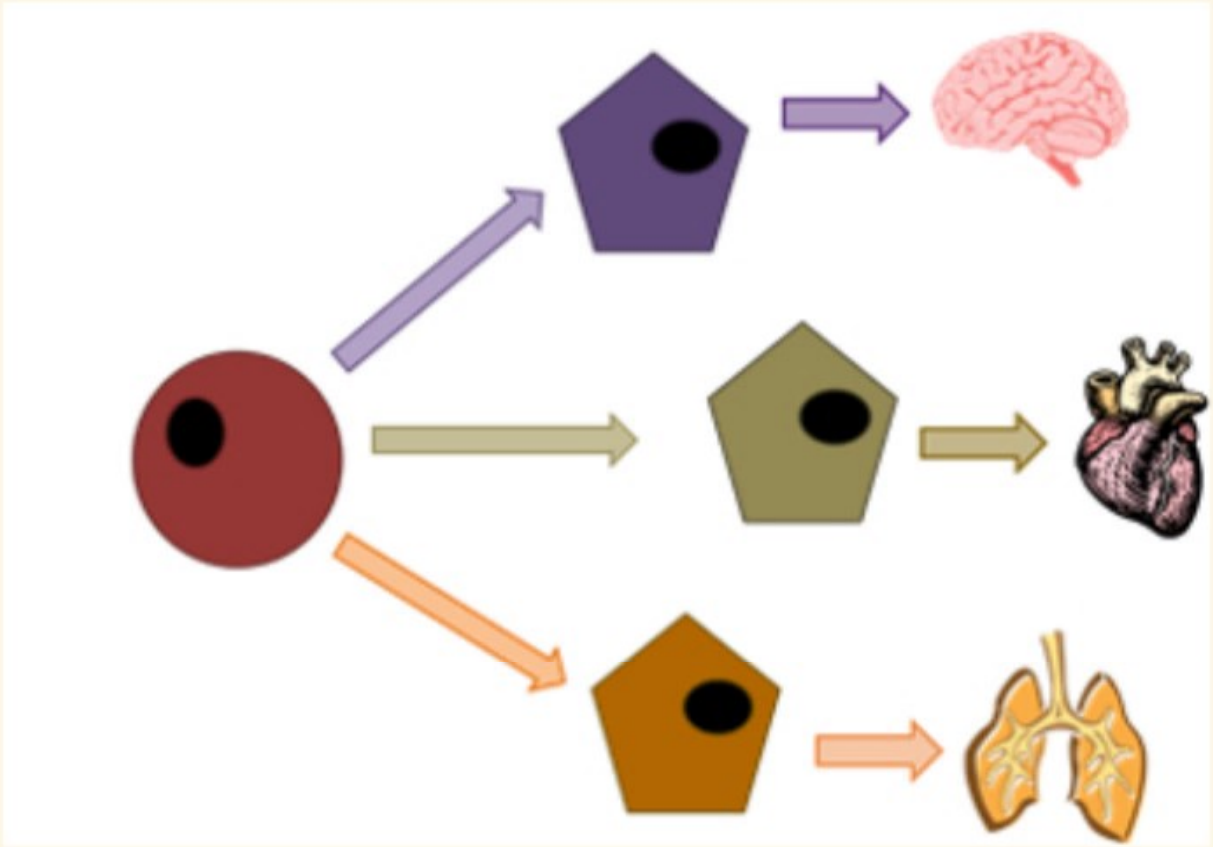
A CLASSICAL PUZZLE

A man wish to cross a river with his Cabbage, his Goat and his Wolf

- If left alone, goat eats cabbage
- If left alone, wolf eats goat
- Boat can only carry man plus 1 item

DEMO SMT/Z3

STEM CELLS



Stems cells can become any adult cell type. How they decide their fate is currently unknown.

GENETIC PROGRAMS

- Human DNA contain about 20000-25000 genes

- Genes act by being *transcribed* into a protein.
- Some of these proteins are transcription factors which perform the transcriptions.

GENE TRANSCRIPTION

Video not supported http://www.wehi.edu.au/education/wehitv

GENE NETWORK

Genes:

Genes affect the transcription rate of other genes.

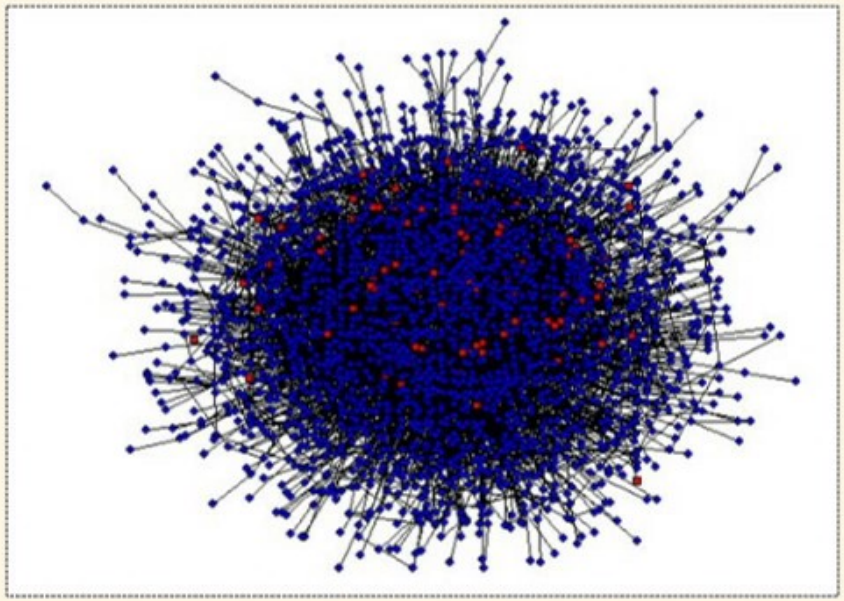
Upregulation (positive interaction):

→ black arrow

Downregulation (negative interaction):

⊣ red tack

FULL NETWORK (MOUSE)



INTERACTIONS

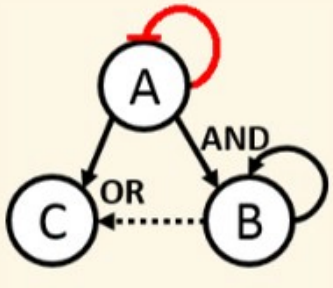
The interactions are from experimental results of gene expression levels.
Positive interaction:

→ black arrow

Possible positive interaction:

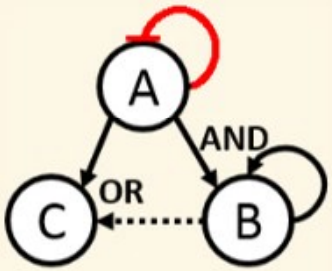
⇢ dotted arrow

TEXTUAL ENCODING

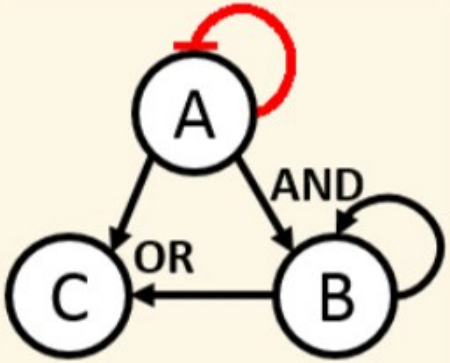


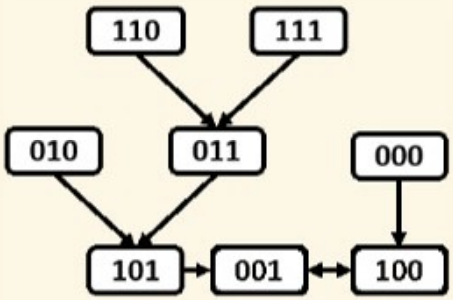
$$\forall b, b'.T(b,b') \leftrightarrow [b'(1) = \neg b(1)] \wedge$$
$$[b'(2) = b(1) \wedge b(2)] \wedge$$
$$[b'(3) = if\ c(1)\ then$$
$$b(1) \vee (2)$$
$$else$$
$$b(1)]$$

SMT ENCODING

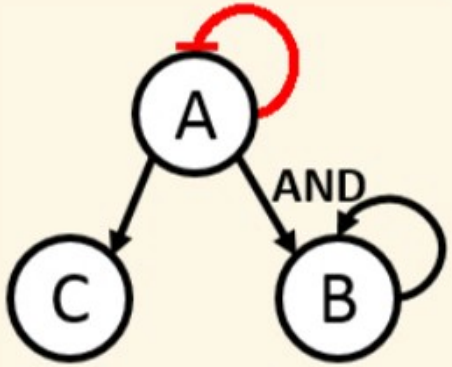


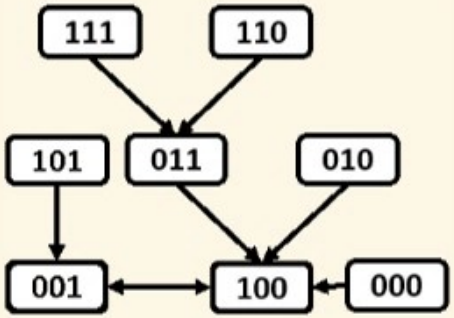
Try the optional B->C interaction



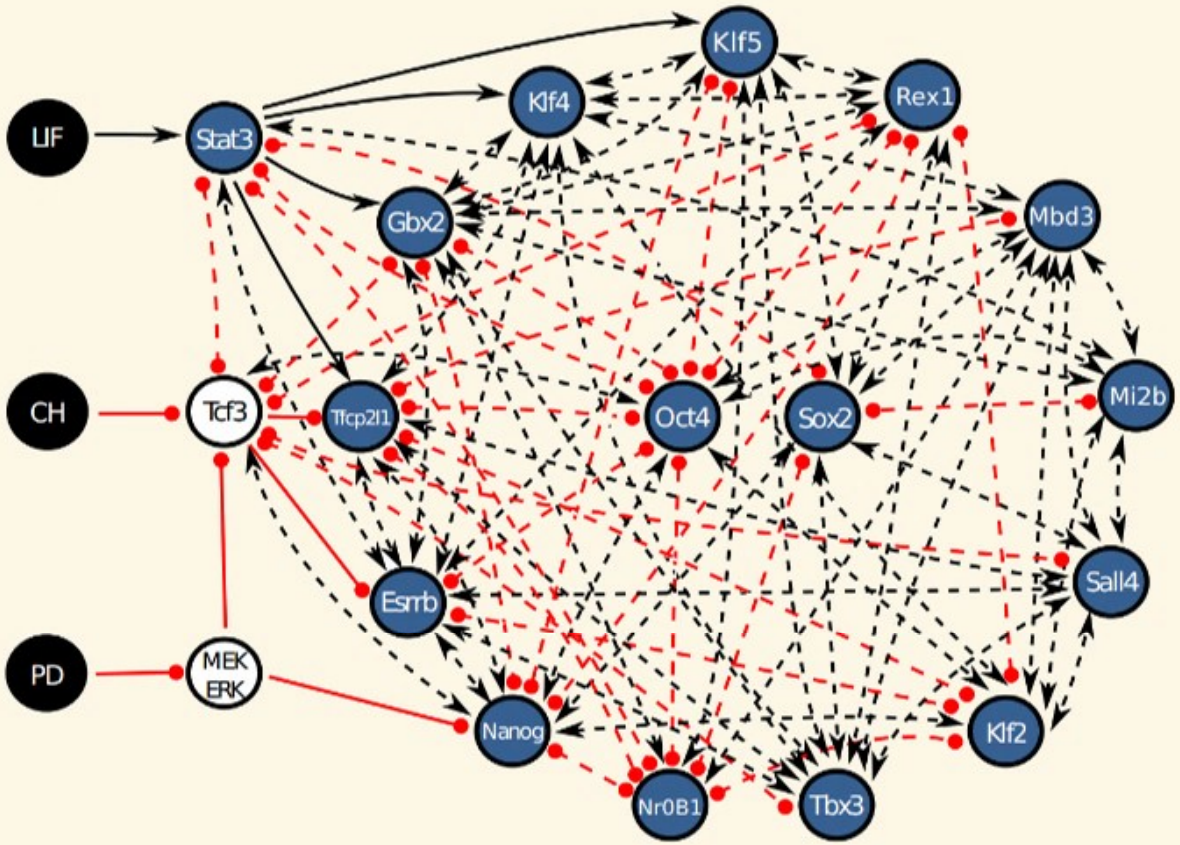


and with B-C missing



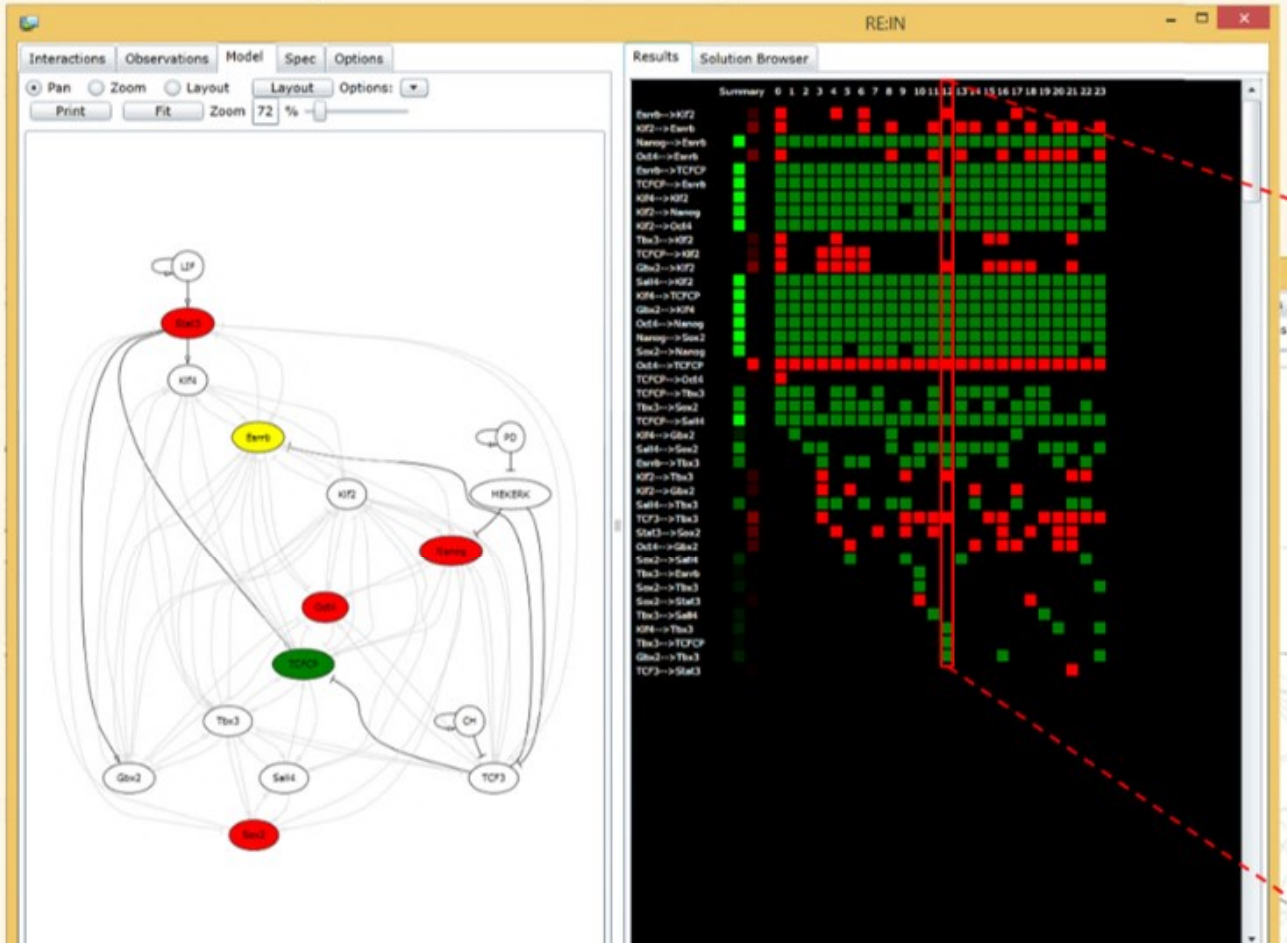


STEM CELL NETWORK

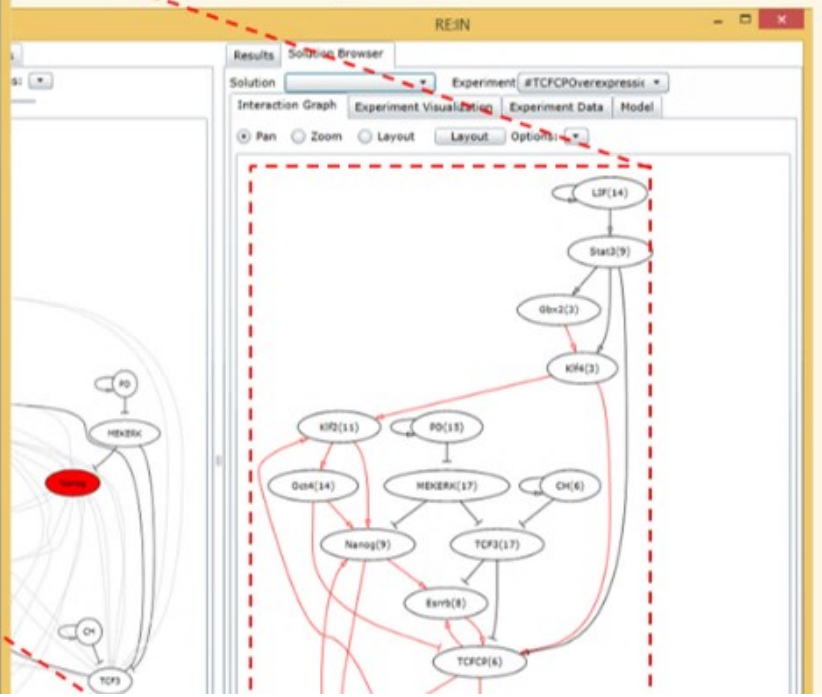


MODEL SYNTHESIS

Enumerating and comparing solutions



Examining individual solutions



REASONING ENGINE FOR INTERACTION NETWORKS (REIN) TOOL

http://research.microsoft.com/rein
Search:

COMPUTE RESOURCES

Despite the sophisticated methods each model and specification has substantial compute requirements

- 20 minutes compute on one core per model and spec
- One researcher easily easily saturates 100s cpu HPC cluster for days

Azure cloud computing

azure.microsoft.com/services/functions/

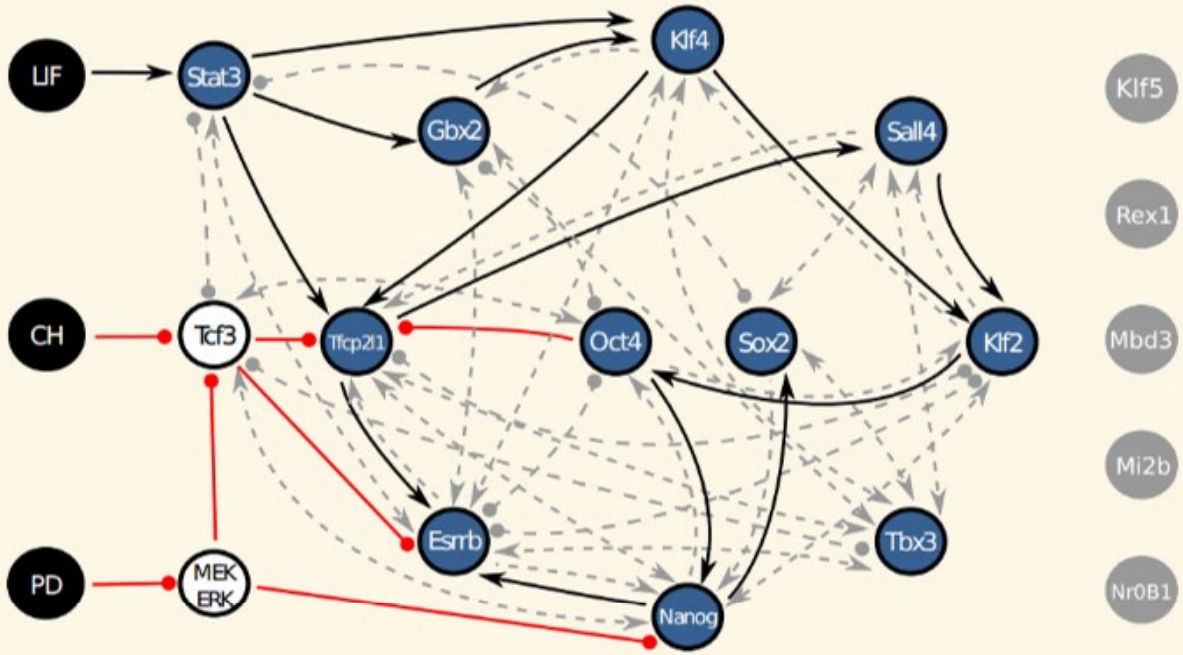
Loads are extremely bursty

- For the public version we use Azure
- Fairly reasonable process to share compute code
- Then stick an HTTP socket on with a web UI

Azure machine learning

azure.microsoft.com/services/machine-learning/

MINIMAL GENE NETWORK



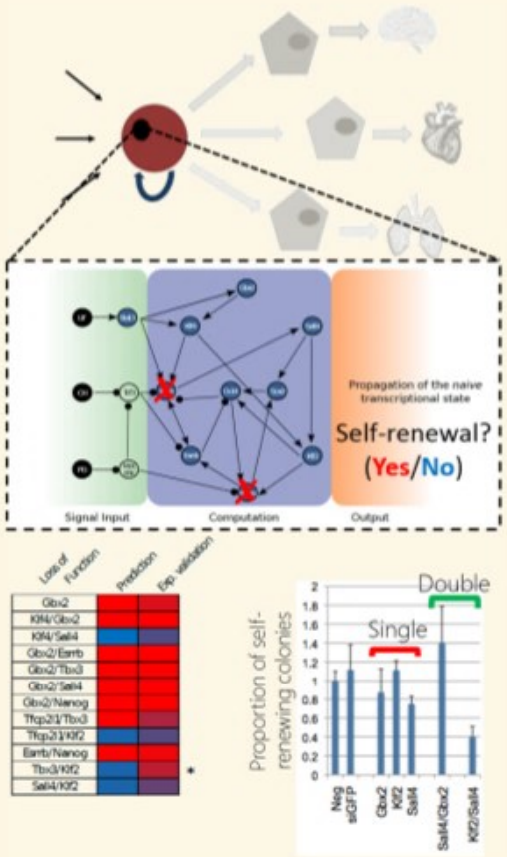
A MINIMAL SELF-RENEWAL PROGRAM

Identified a simple minimal network capable of reproducing the experimentally observed self-renewal behaviour

Reasoned about the effects of compound gene perturbations (double gene knockdowns) on self-renewal

Generated a number of predictions capturing complex compound effects

Validated predictions experimentally with high success rate



Defining an essential transcription factor program for naïve pluripotency
S-J. Dunn, G. Martello, B. Yordanov, S. Emmott, A. G. Smith (2014)

STEM CELLS IN THE CLINIC

Type 1 diabetes => embryonic stem cell derived pancreatic precursor cells

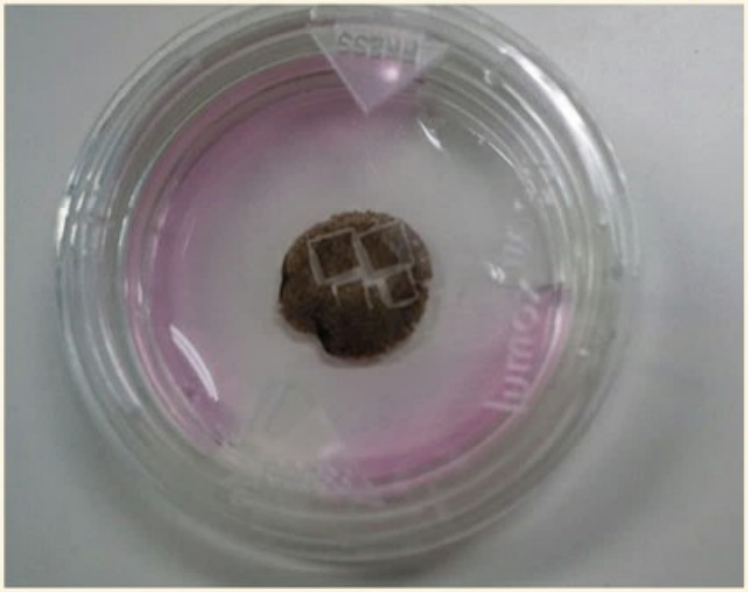
VIACYTE
Regenerating Health™

Severe myopia => transplantation of human retinal pigmented epithelium

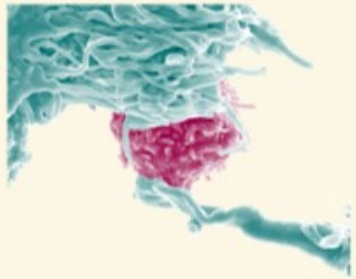
ocata
THERAPEUTICS™

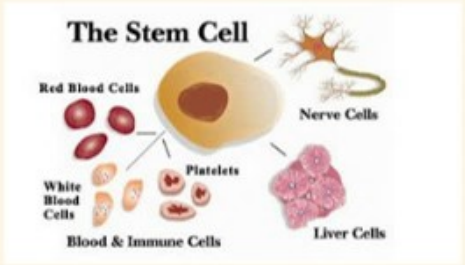
Macular degeneration => transplation of iPS grown retinal (September, 2014)

RIKEN



BIOLOGICAL COMPUTATION





Immunology Development Synthetic Biology Molecular Programming

↓

BIOLOGY DESIGN TOOLS

Simulation (Stochastic, ODE, PDE);
Analysis (probabilistic model-checking, symbolic verification with Z3);
Parameter estimation (Filzbach); Visualisation (DDD,MSAGL);
Unbounded computation
Interoperability (SBML,C#, PRISM, Matlab, Chaste, CellModeller)

COMPUTIONAL SCIENCE GROUP & COLLABORATORS

Andrew Phillips

Boyan Yordanov

Filippo Polo

Hillel Kugler

Luca Cardelli

Neil Dalchau

Paul Grant

Sara-Jane Dunn

Microsoft Research
Christoph Wintersteiger,
Youssef Hamadi,
Simon Youssef

University of Cambridge
Graziano Martello,
Austin Smith
Jim Haseloff,
Tim Rudge,
PJ Steiner,
Fernan Federici,
James Brown

University of New Mexico
Matthew Lakin,
Darko Sefanovic

University of Washington
Georg Seelig,
Yuan Chen