Biocircuits engineering and bio-design automation: some recent results”

Newcastle University, June 15, 2015

Inteligencia Artificial Laboratorio

www.lia.upm.es

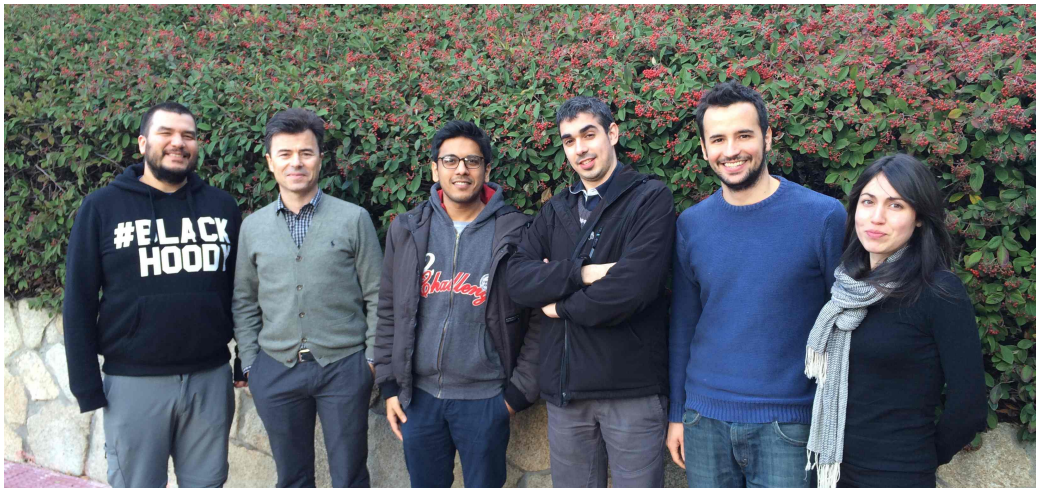
POLITÉCNICA

Alfonso Rodríguez-Patón
arpaton@fi.upm.es
@LIA_UPM

LIA members

Inteligencia Artificial Laboratorio
POLITÉCNICA
www.lia.upm.es

- Professors:
  - Andrei Paun
  - Petr Sosík
  - Iván Pau
  - Daniel Manrique
- PhD students:
  - Martín Gutiérrez
  - Paula Gregorio
  - Guillermo Pérez
  - Ismael Gómez
  - Vishal Gupta
  - Antonio García
  - Marcos Rodríguez
- Master students: Irene Serrano, Laura Puente, Luis Enrique Muñoz

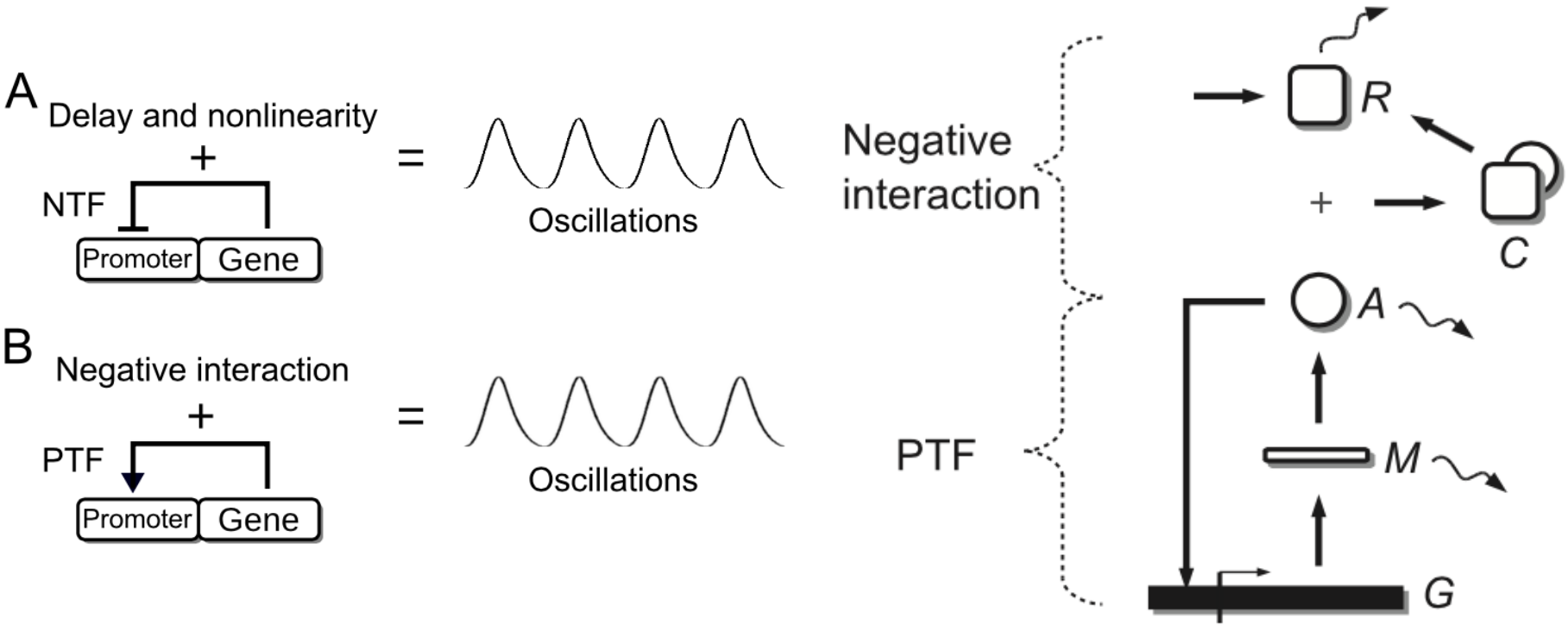


PLAS
WIRES
SEVENTH FRAMEWORK PROGRAMME
EVOPROG
GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMIA Y COMPETITIVIDAD

Outline

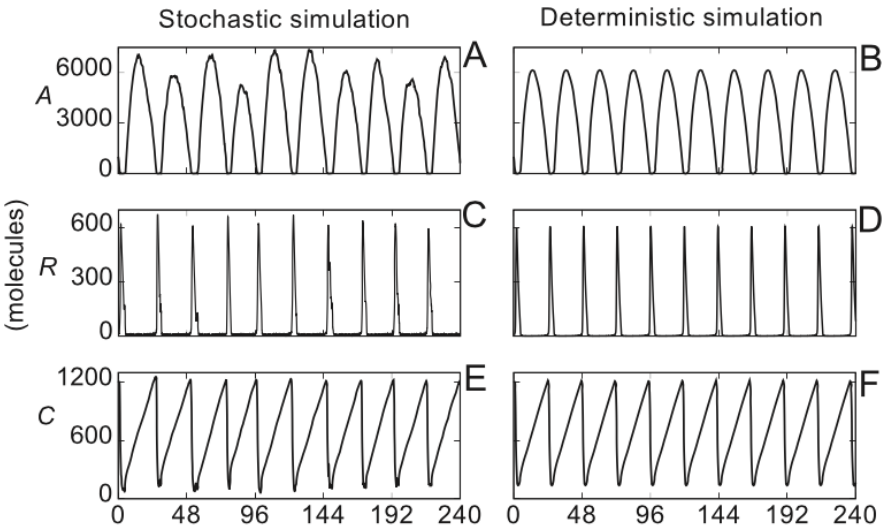
- Systems Biology: GRO simulator
- Synthetic Biology: PLASWIRES project, Directed Evolution.
- DNA Computing: Inference with DNA molecules.
- Lab automation: EVOPROG project

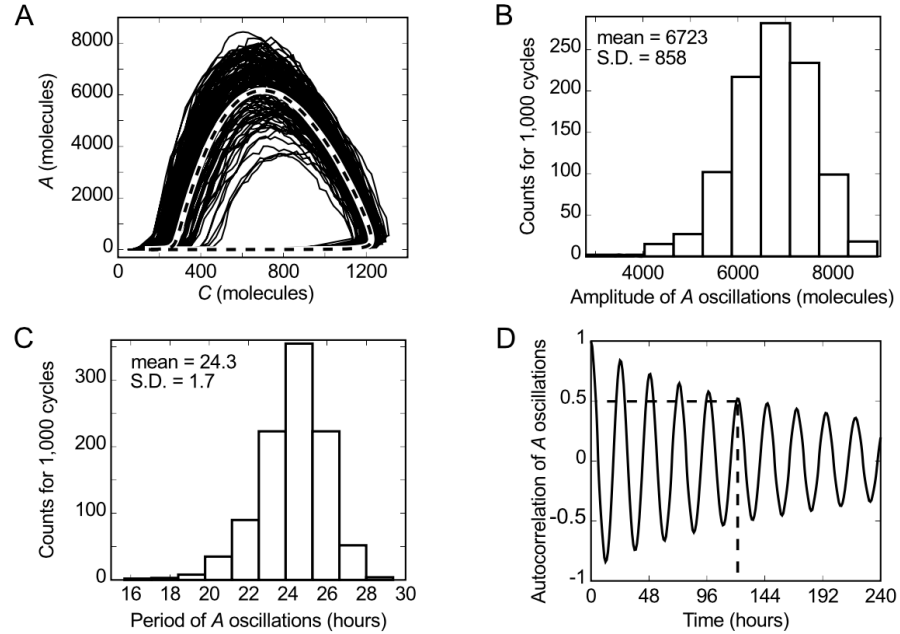
One-gene genetic oscillator with a positive feedback loop and a negative interaction

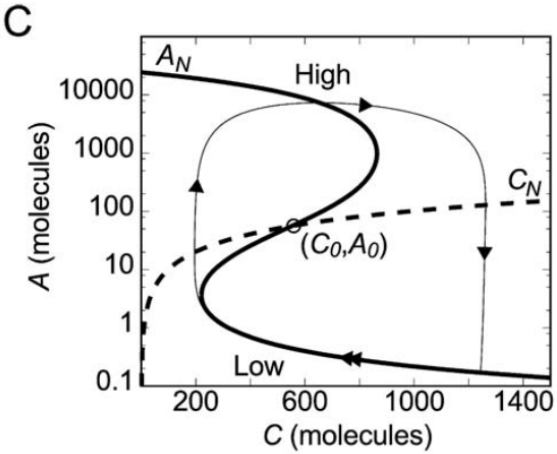


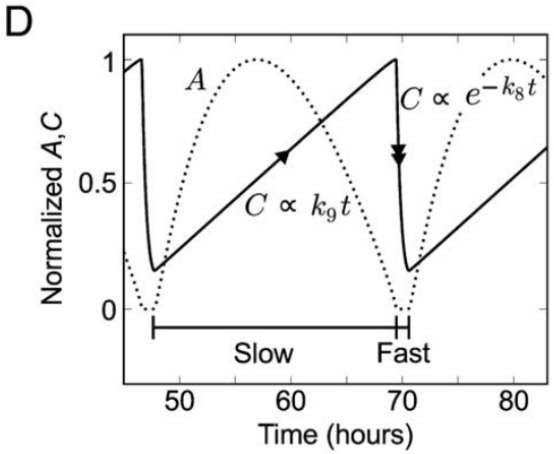
Miró-Bueno JM, Rodríguez-Patón A (2011) A Simple Negative Interaction in the Positive Transcriptional Feedback of a Single Gene Is Sufficient to Produce Reliable Oscillations. **PLoS ONE** 6(11): e27414. doi:10.1371/journal.pone.0027414

One-gene genetic oscillator with a positive feedback loop and a negative interaction









PLoS ONE 6(11): e27414.
doi:10.1371/journal.pone.0027414

DNA Computing in LIA group

Sainz de Murieta, I., & Rodriguez-Paton, A. (2014). Probabilistic reasoning with a Bayesian DNA device based on strand displacement. NATURAL COMPUTING, 13(4), 549-557.

Rodríguez-Patón, A., de Murieta, I. S., & Sosík, P. (2014). DNA strand displacement system running logic programs. Biosystems, 115, 5-12. Chicago

Computational modelling

The most frequent approaches for simulating genetic circuits are:

- Differential Equations (DEs)
  - High precision at single-cell level
  - Scale poorly to large-scale colonies and spatial component is not easily reproduced
- Agent/Individual based Models (IbMs)
  - Perform better at a large scale
  - Not as precise as DEs (most IbMs are rule based)
  - Provide a good spatial environment

LIA

- We are interested in designing, simulating and studying multicellular genetic circuits that run in bacterial colonies.
- We wan to simulate cell-cell communication based on conjugation: space is important

---

- Agent/Individual based Models (IbMs): are best suited for this purpose.

State of the Art: IbMs

- CellModeller
  - Python library for simulating bacterial colonies developed by U. Cambridge and Microsoft Research.
  - Simulates 2D or 3D colonies of rod-shaped bacteria.
  - Simulations are implemented through DEs or through rules.
  - Runs on OpenCL and reaches up to 32000 simulated bacteria in 30 mins.

State of the Art: IbMs (II)

- iDynoMiCS
  - Developed by J.U. Kreft lab at University of Birmingham
  - Simulates 2D and 3D bacterial colonies.
  - Simulations are implemented through rules (XML parametrization and Java).
  - Goal is to study biofilm formation.

State of the Art: IbMs (III)

- BactoSIM
  - Developed by LIA – UPM.
  - Simulates 2D spherical bacterial colonies.
  - Simulations are implemented through rules (based on Repast - Java).
  - Reaches 10^6 bacteria in 1 hour.
  - The goal of BactoSIM is to study bacterial conjugation.

State of the Art: IbMs (IV)

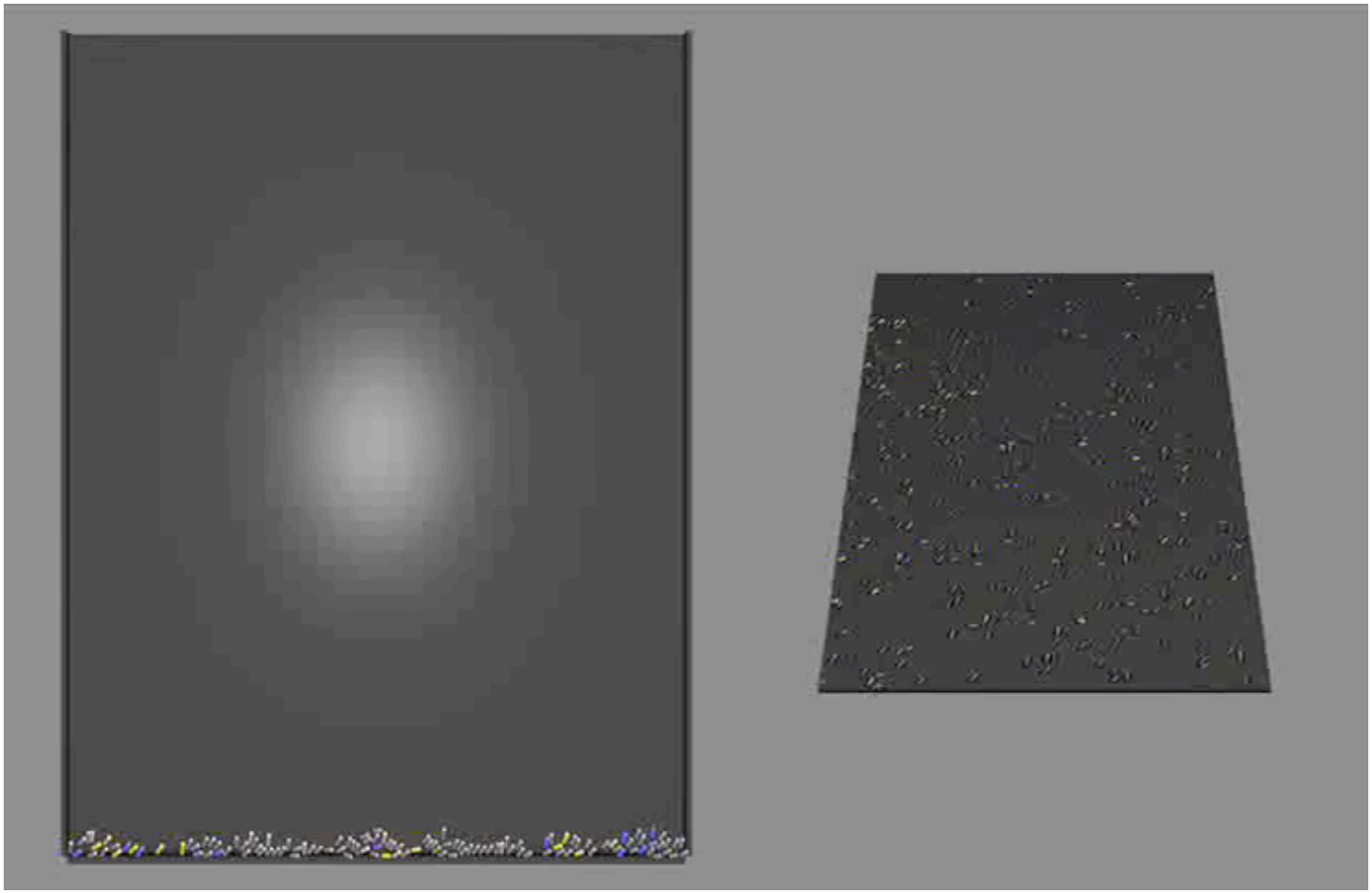
- GRO
  - Developed by Klavins lab at University of Washington.
  - Simulates 2D rod-shaped bacterial colonies.
  - Simulations are implemented through rules (gro language – based on guarded commands).
  - Reaches around 10^4 bacteria in 1 hour.
  - Aimed at simulating multicellular behaviors.

IbMs

- At LIA, we (have) work(ed) with these IbMs:
  - iDynoMiCS
  - BactoSIM
  - GRO
- A brief summary of our work will now be presented.

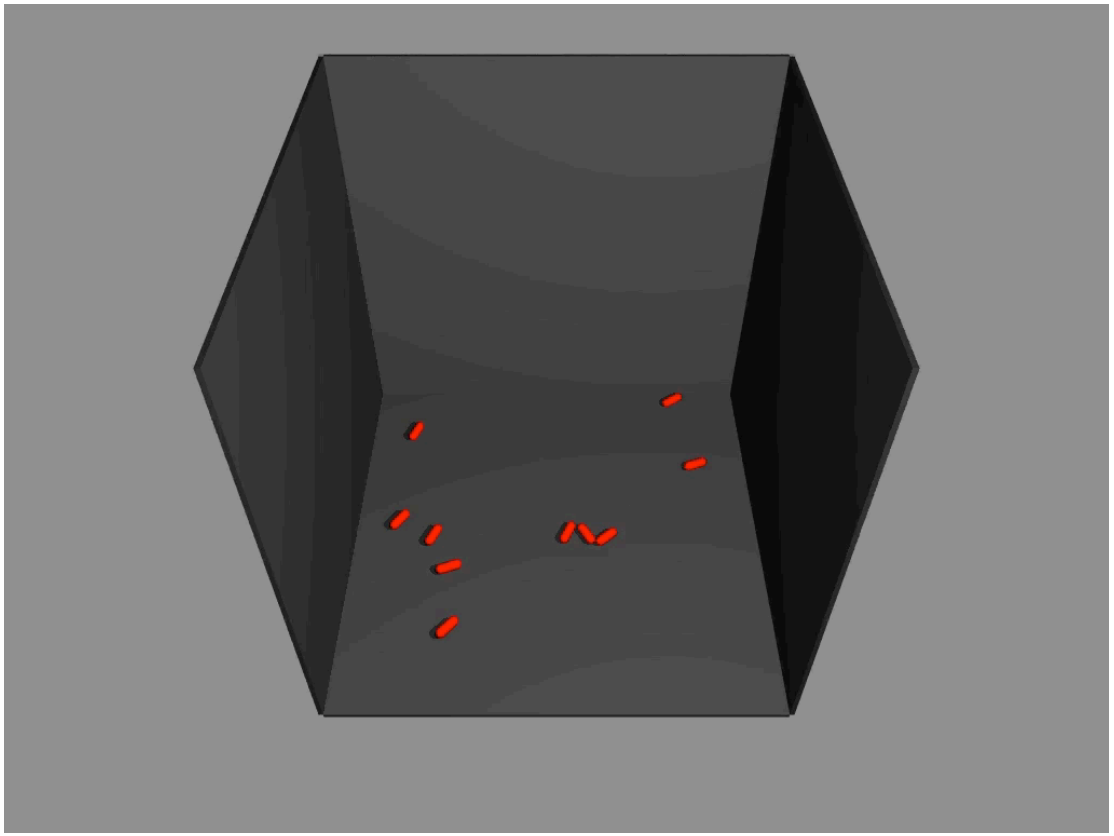
iDynoMiCS

iDynoMiCS - Conjugation



AND gate

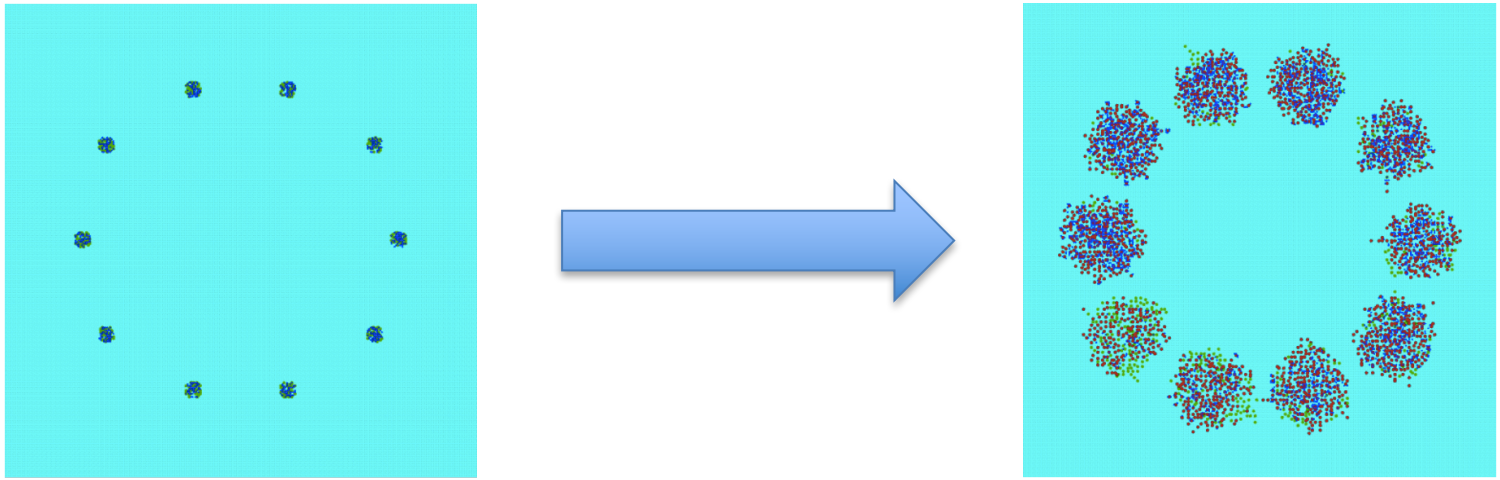
iDynoMiCS – Growth



Bacteria growing with rod-shaped bacteria and shoving

BactoSIM

BactoSIM – Measuring phage infectivity



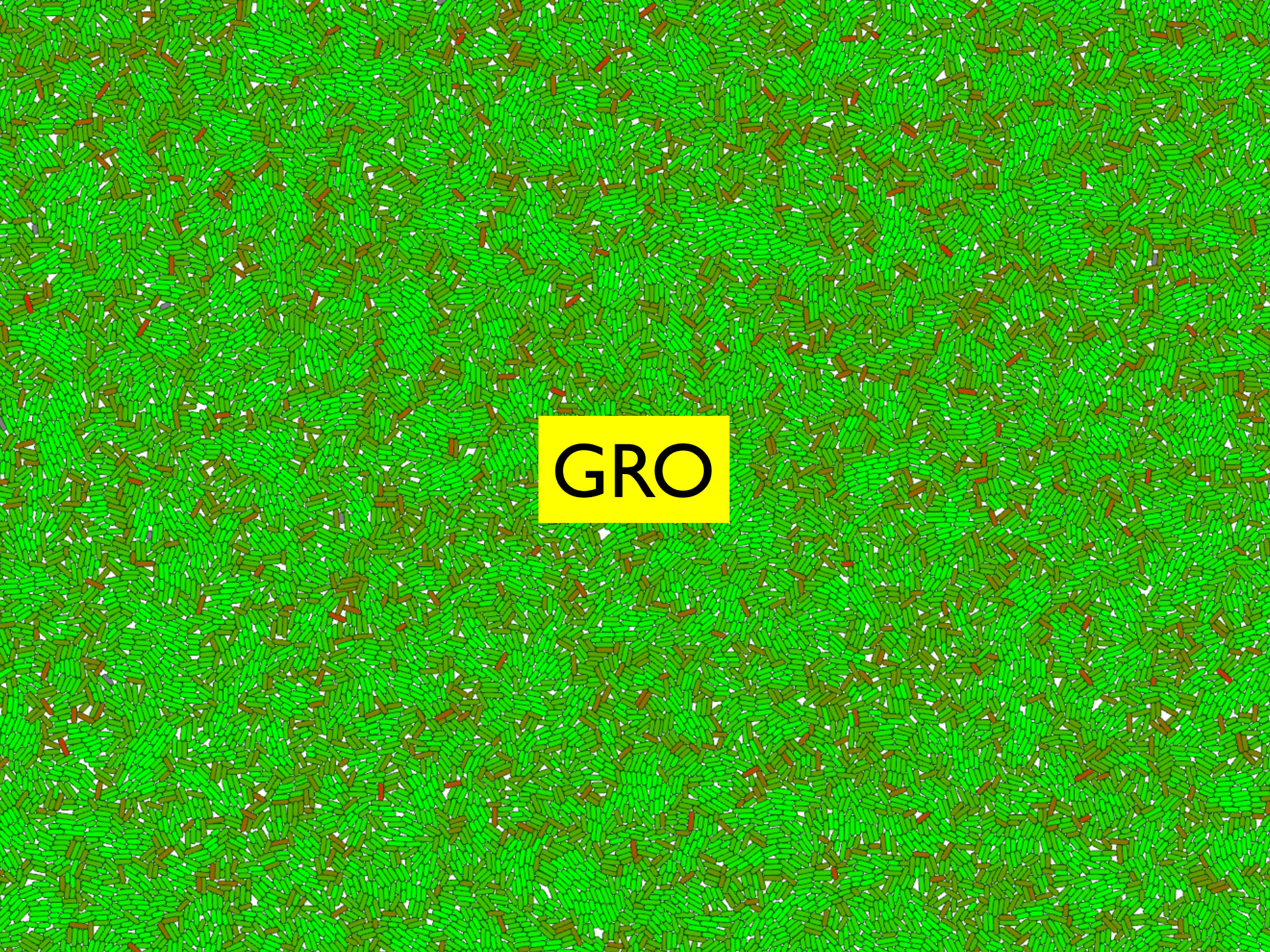
T = 0 mins

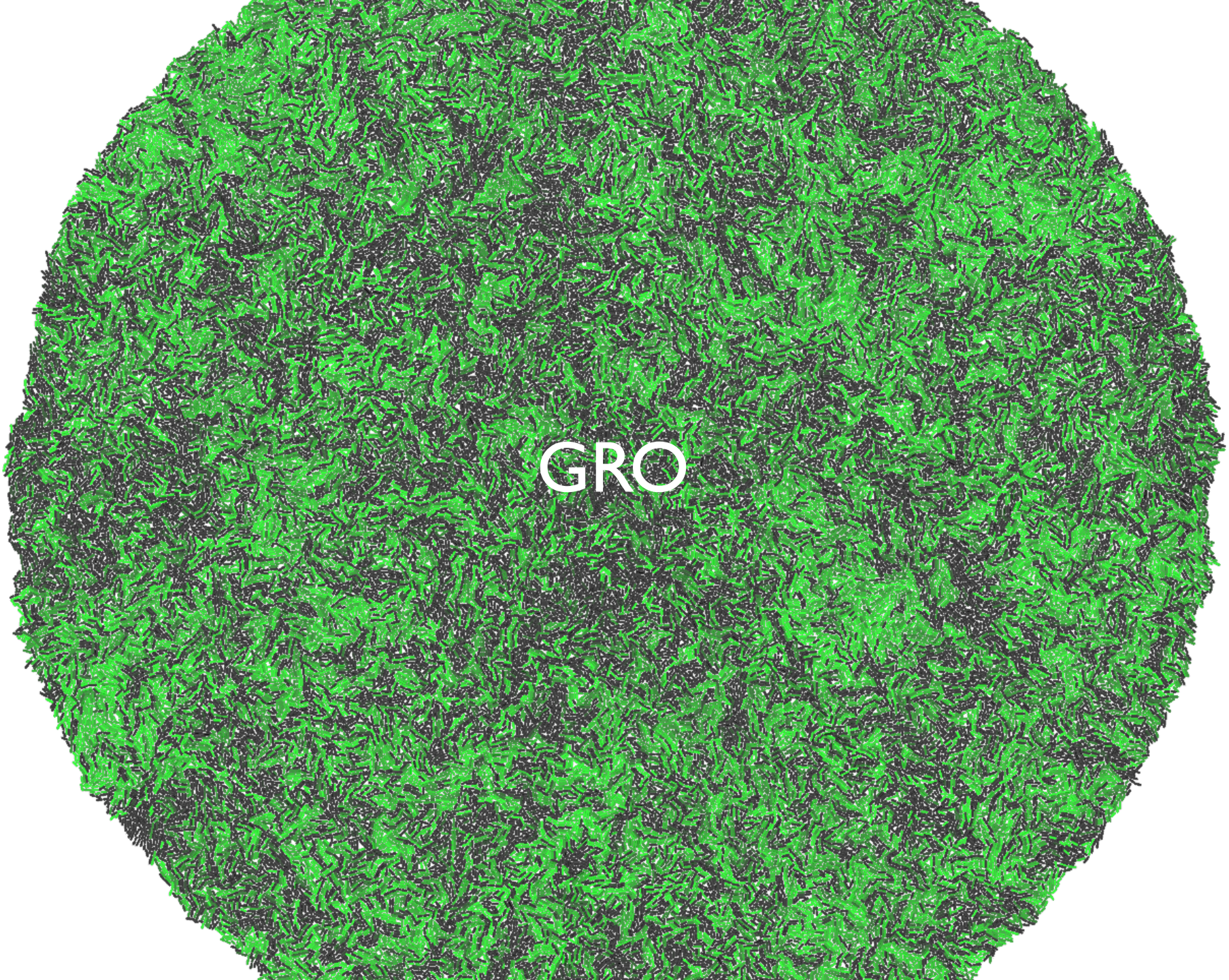
T = 125 mins

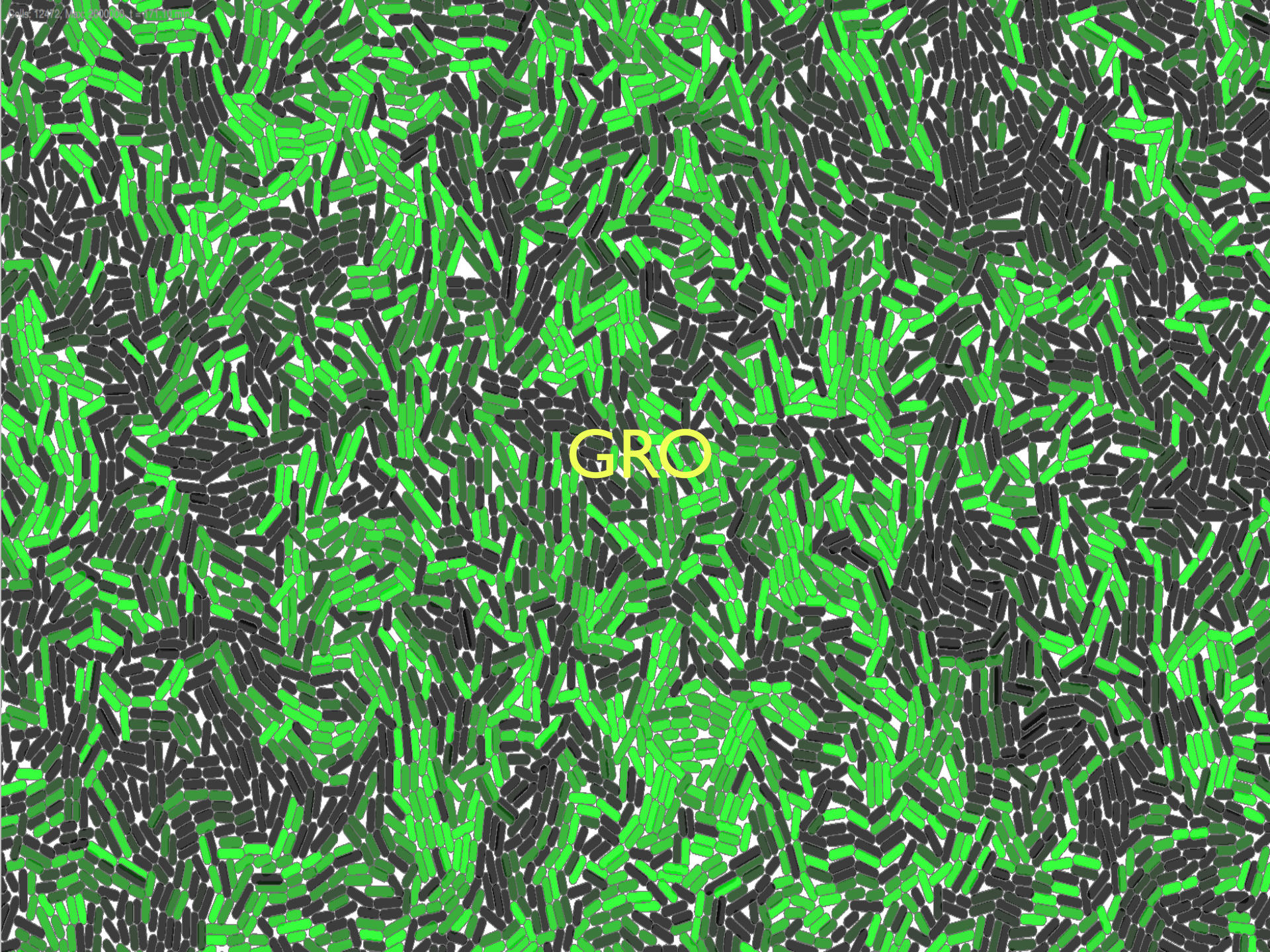
Phage

Uninfected bacterium

Infected bacterium

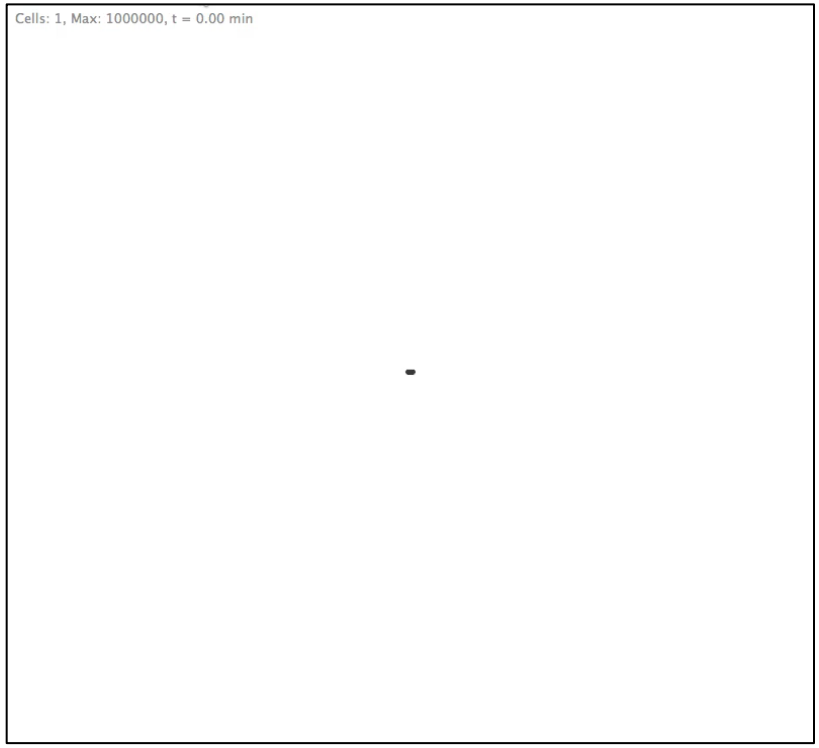

GRO


GRO


Cells: 12472, Max: 2000000, t = 07:31:00
GRO

GRO

- Developed at Klavins Lab (University of Washington)
- IbM based on guarded commands and functional programming.
- GRO is mainly concerned with studying bacterial colony growth and multicellular behaviors based on signals.
- Chipmunk acts as GRO's physics engine.



Some limitations of GRO

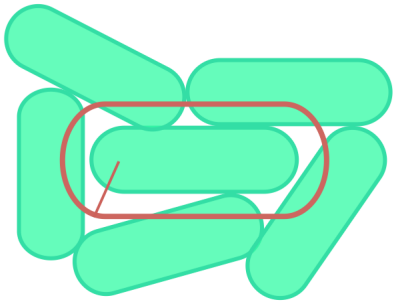
- Bacterial conjugation is not implemented.
- GRO is slow, it reaches about 20000 bacteria in 4 hours.
- Describing an experiment with guarded commands is unnatural for biologists.
- Colony growth does not take into account nutrient uptake.

GRO, improved by LIA

- We have implemented the following functionalities for GRO:
  - Bacterial conjugation
  - New shoving algorithm (CellEngine)
  - New genetic module
  - New nutrient uptake and growth module (CellSignal)

Implementation of conjugation process

- Conjugation was implemented atop the modified GRO (using CellEngine)

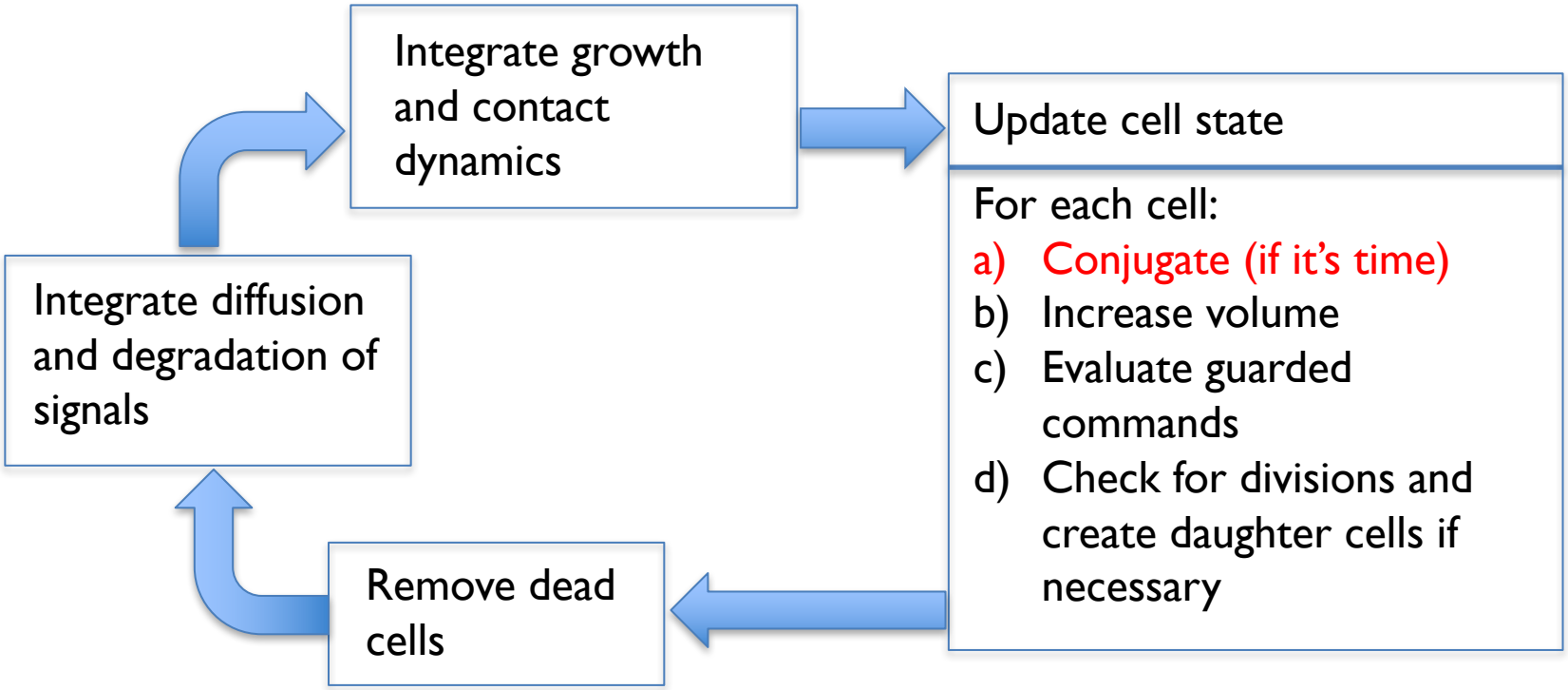


"Aura" calculation to retrieve a bacterium's neighbors

Conjugative plasmid: will you survive?

Implementation of conjugation process

- GRO's source code was modified to include conjugation process in its workflow



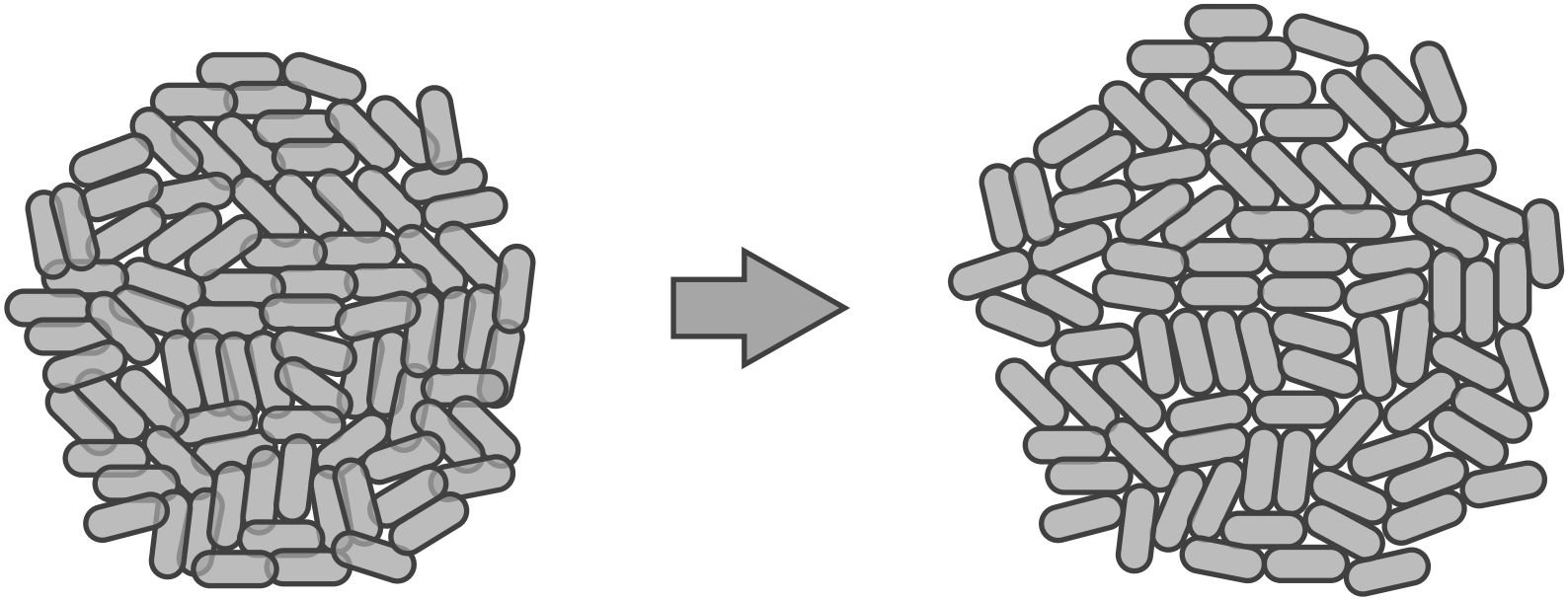
Adapted from Jang et al., 2012

GRO – Conjugation rules

- A bacterium conjugates in GRO under the following circumstances:
  - It has a conjugative plasmid
  - A probability of conjugation is calculated from a ratio between the # of average conjugations per life cycle, the # of timesteps and the # of neighbor bacteria.

CellEngine: a new fast shoving algorithm

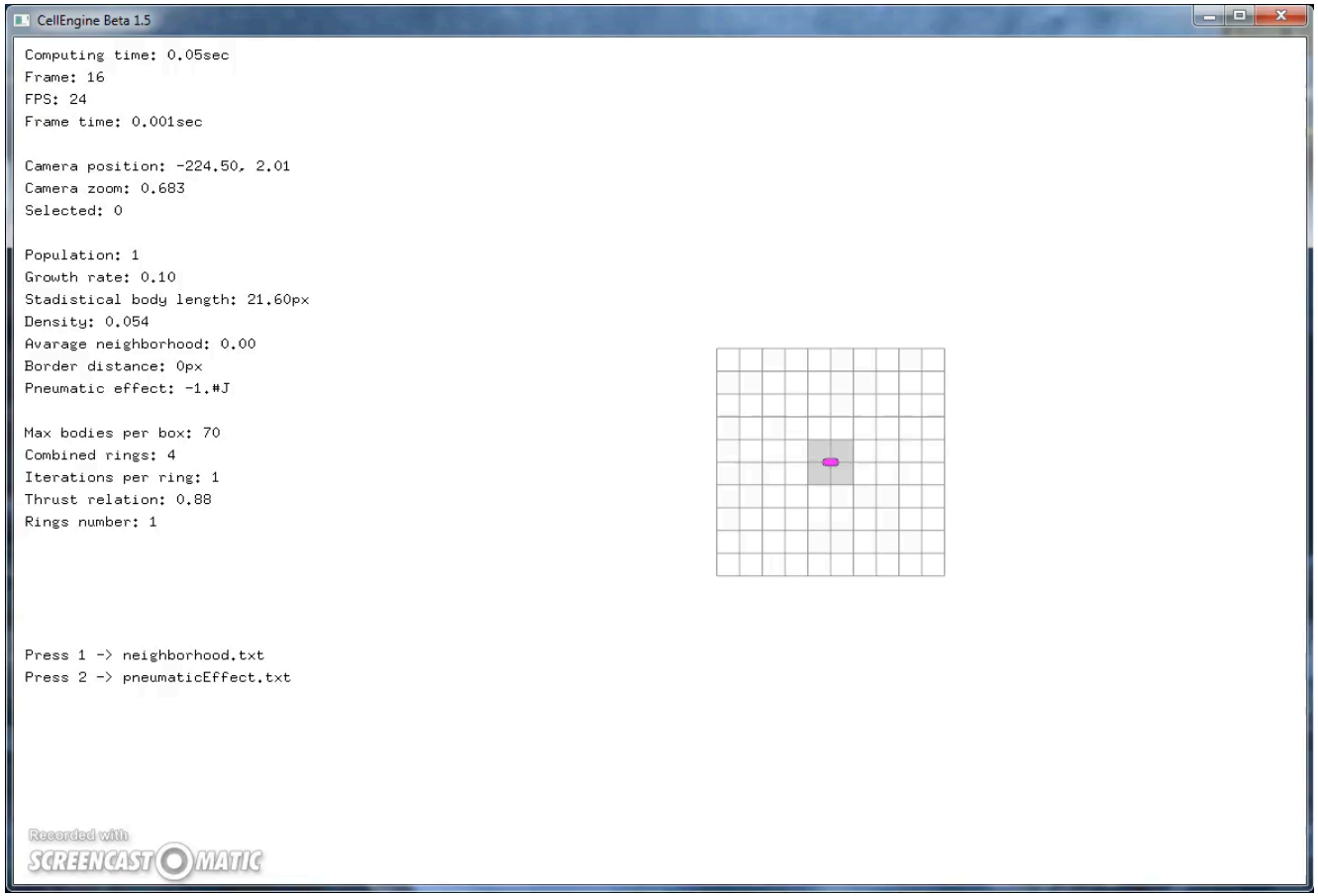
- CellEngine, a new shoving algorithm
- Based on the grouping of bacteria in rings



CellEngine: algorithm

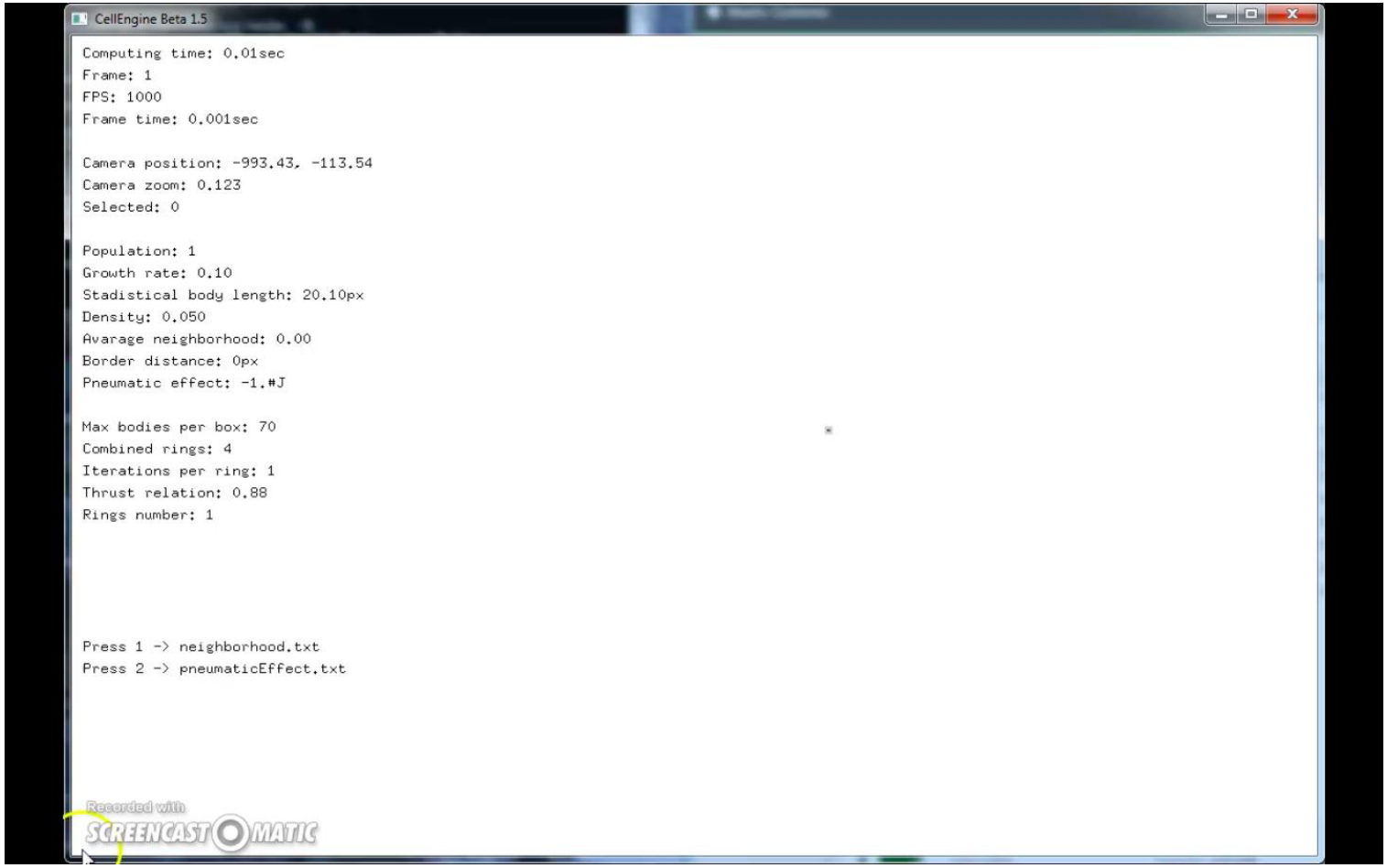
1. Find bacteria in the edge of the colony
2. Recursively create rings of bacteria of a certain width w inwards until reaching the center of the colony
3. Once the center is found, repeat for all rings starting at the center and moving outwards:
    1. Relax overlaps of ring i as if it were an independent colony. Assume the inner ring (i+1) as a wall and the outer ring (i-1) as non-existent.
    2. Relocate ring i-1 outwards around ring i.

CellEngine execution



Ring calculation

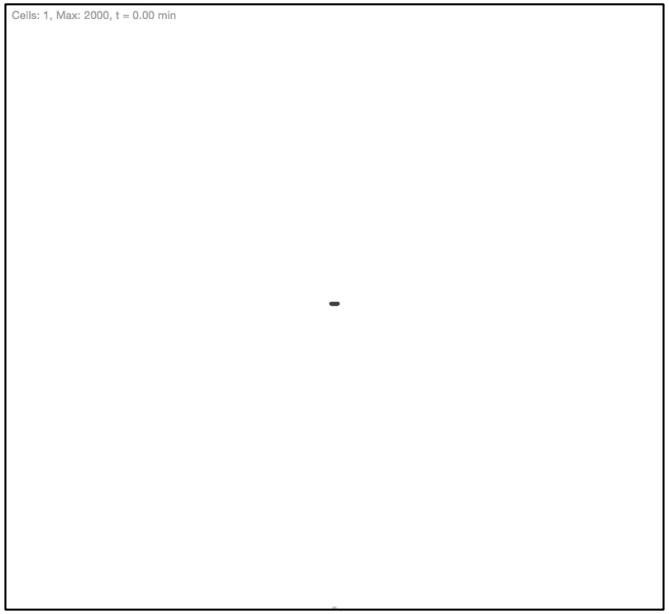
CellEngine execution



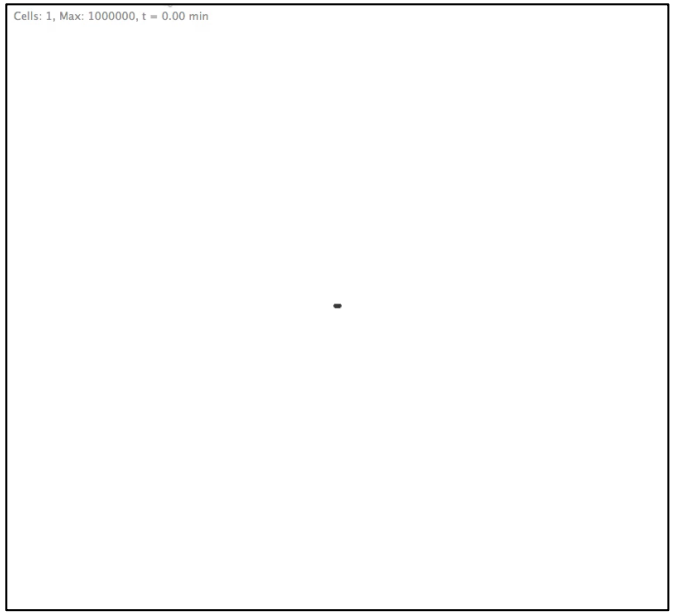
Ring calculation and growth (large scale)

CellEngine vs. Chipmunk

Growth test

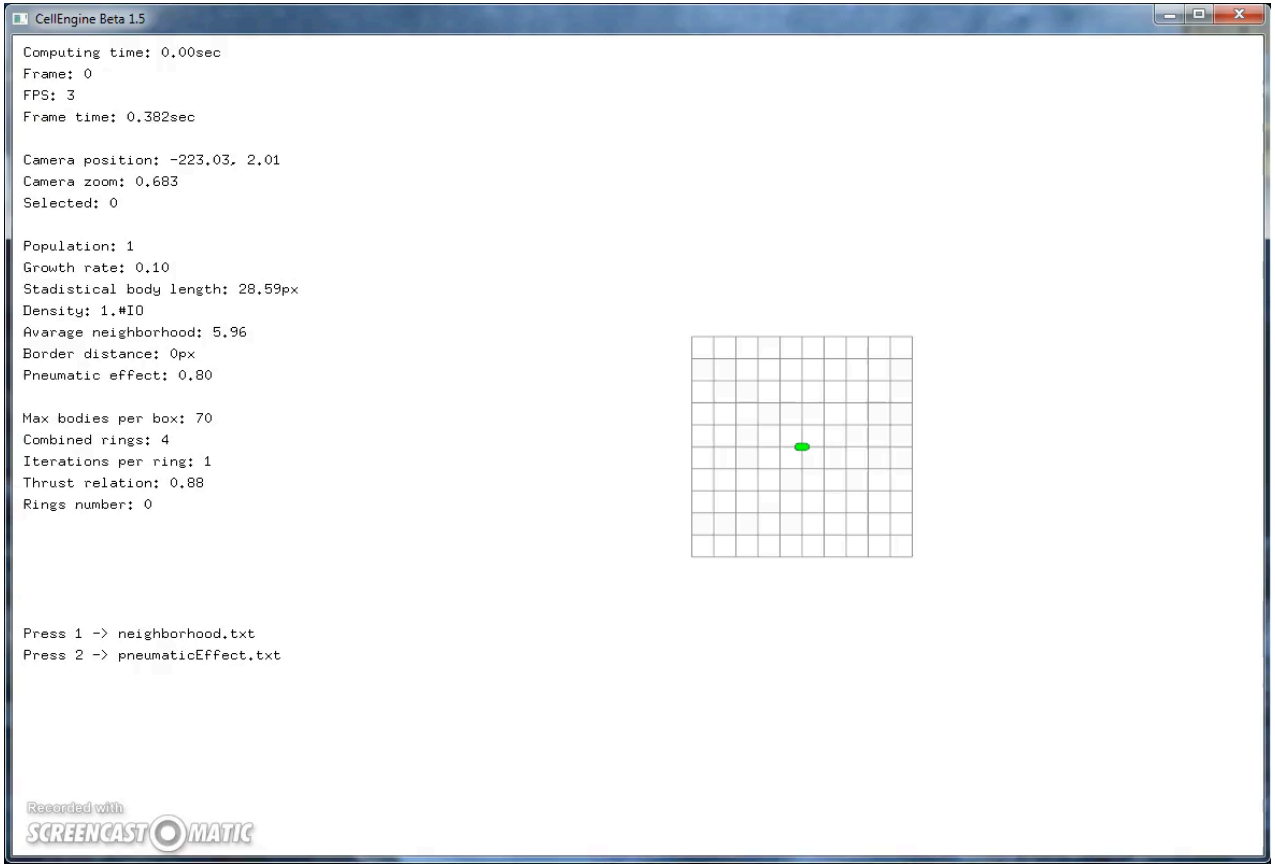


GRO (CellEngine)



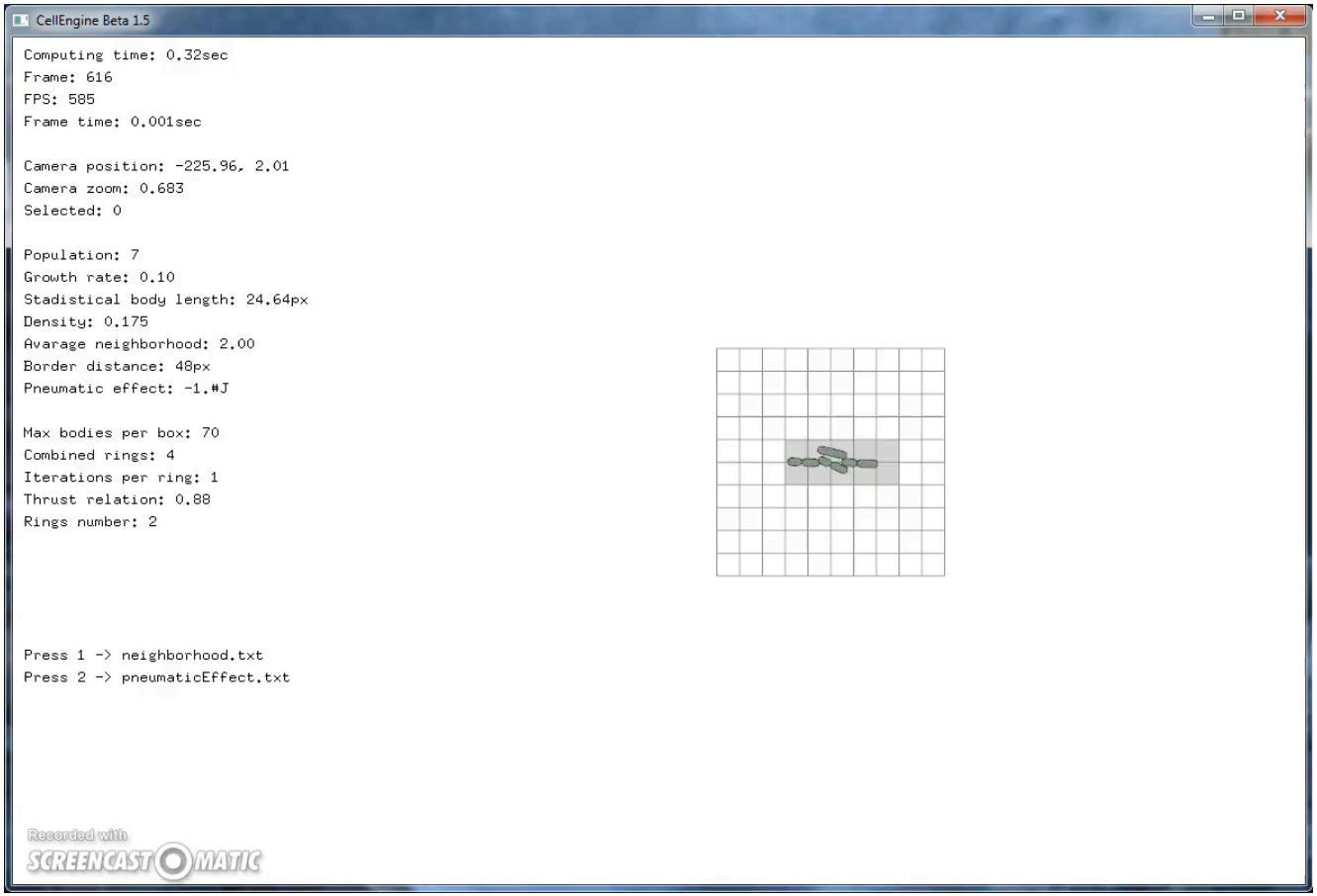
GRO (Chipmunk)

CellEngine execution



Bacterial colony growth

CellEngine execution



Cell alignment calculation

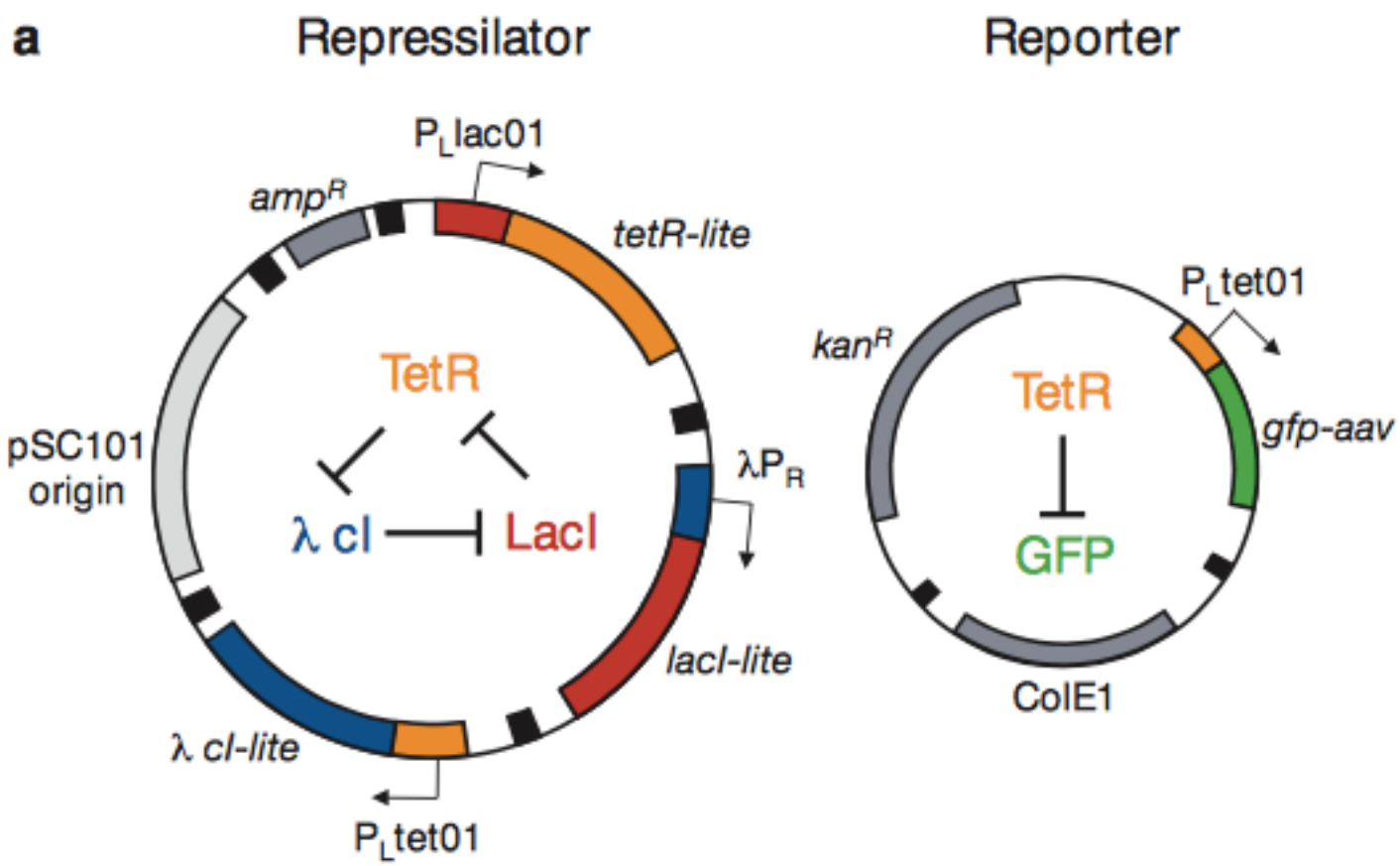
CellEngine – some numbers

	GRO (Chipmunk)	GRO (CellEngine)
Cells	Total time (hours)	Total time (hours)
1	0.00	0.00
500	0.00	0.00
1000	0.01	0.00
2000	0.02	0.00
4000	0.08	0.00
8000	0.41	0.00
16000	2.14	0.00
20000	4.02	0.01
100000	> 168	0.05

Genetic module

- Boolean values for the proteins: 0/1
- Boolean values for the state of the genes: ON/OFF
- How long must be a gene ON to produce enough protein to be considered as 1? Half-life activation time.
- How long must be a gene OFF to consider its associated protein takes value 0? Half-life degradation time.
- Promoters: Boolean logic gates: AND, OR, etc.
- Noise: Probability that a gene ON(OFF) switch to OFF(ON) respectively, without any change in the inputs
- Similar to a Probabilistic Asynchronous Boolean Network with delays. Or a piecewise-linear differential equation.

Example: Repressilator



Original design of the Repressilator

Example: Repressilator

```
include gro

set("dt",0.001);

period := 30;

t := 0;

program repressilator() :=
{
    delay := 0;
    yfp := 0;
    gfp := 0;
    rfp := 0;
    state := [s := rand(3) + 1];
    state.s = 1 & delay >= period : {rfp := 0, gfp := 0, state.s := 2, delay := 0}
    state.s = 2 & delay >= period : {rfp := 0, yfp := 0, state.s := 3, delay := 0}
    state.s = 3 & delay >= period : {yfp := 0, gfp := 0, state.s := 1, delay := 0}
    state.s = 1 & delay < period : {rfp := rfp + 1}
    state.s = 2 & delay < period : {yfp := yfp + 1}
    state.s = 3 & delay < period : {gfp := gfp + 1}
    true : {delay := delay + dt}
};

program main() :=
{
    true : {t := t + dt}
};

ecoli([x := 0, y := 0], program repressilator());
```

Guarded command based source code for the Repressilator

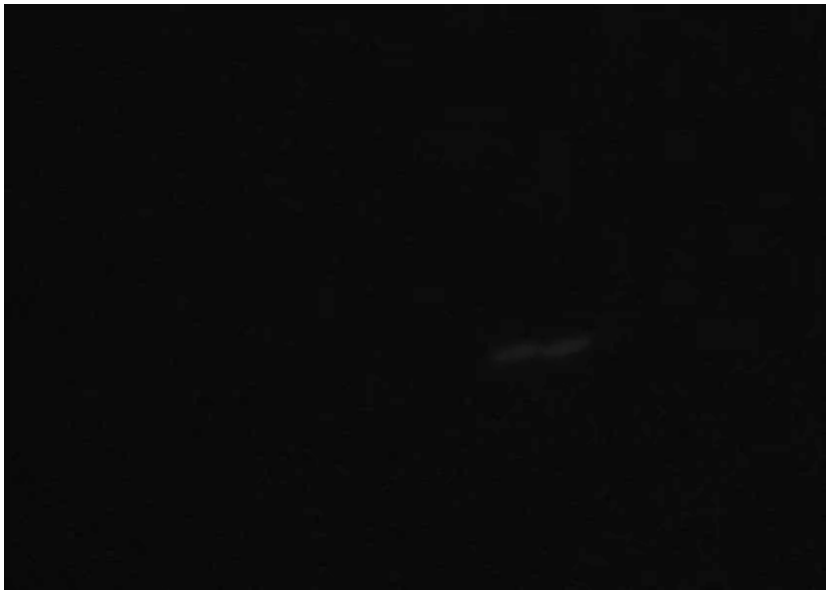
Example: Repressilator

```
include gro

set ( "dt", 0.1 );
set ( "population_max", 2000000 );

t := 0;

program p() :=
{
    skip();
};

set ("num_plasmids",2);
set ("num_proteins",4);

degradation_times({30.0170,32.2900,30.0170,33.8000},{2.0,2.0,2.0,2.0},
                  {0.0,0.0,0.0,0.0},{0.0,0.0,0.0,0.0});

operon ({true,false,false,false}, {false,false,false,false}, {0,-1,0,0}, {}, false,
        {32.2900,0.0,0.0,0.0}, {20.3,0.0,0.0,0.0}, {0.0,0.0,0.0,0.0}, {0.0,0.0,0.0,0.0},
         0, {0.0,0.0,0.0,0.0},{0.0038,0.9962,1,0});
operon ({false,false,true,false}, {false,false,false,false}, {-1,0,0,0}, {}, false,
        {0.0,0.0,30.0170,0.0}, {0.0,0.0,20.3,0.0}, {0.0,0.0,0.0,0.0}, {0.0,0.0,0.0,0.0},
         0, {0.0,0.0,0.0,0.0},{0.0088,0.9912,1,0});
operon ({false,true,false,false}, {false,false,false,false}, {0,0,-1,0}, {}, false,
        {0.0,30.0170,0.0,0.0}, {0.0,20.3,0.0,0.0}, {0.0,0.0,0.0,0.0}, {0.0,0.0,0.0,0.0},
         0, {0.0,0.0,0.0,0.0},{0.0003,0.997,1,0});
operon ({false,false,false,true}, {false,false,false,false}, {-1,0,0,0}, {}, false,
        {0.0,0.0,0.0,30.0170}, {0.0,0.0,0.0,20.3}, {0.0,0.0,0.0,0.0}, {0.0,0.0,0.0,0.0},
         0, {0.0,0.0,0.0,0.0},{0.0088,0.9912,1,0});

plasmids_matrix ({true,true,true,false,
                  false,false,false,true});

action({false,false,false,true},"d_paint",{"3","0","0","0"});
action({true,false,false,false},"d_paint",{"-1","0","0","0"});
action({false,true,false,false},"d_paint",{"-1","0","0","0"});

program main() :=
{
    c_ecolis(30, 200, {true, true}, {true,false,false,false}, {false,false,false,false}, program p());
    c_ecolis(1, 30, {false, false}, {false,false,false,false}, {false,false,false,false}, program movie());

    true:
    {
        t := t + dt;
    }
};
```

Genetic design based source code for the Repressilator

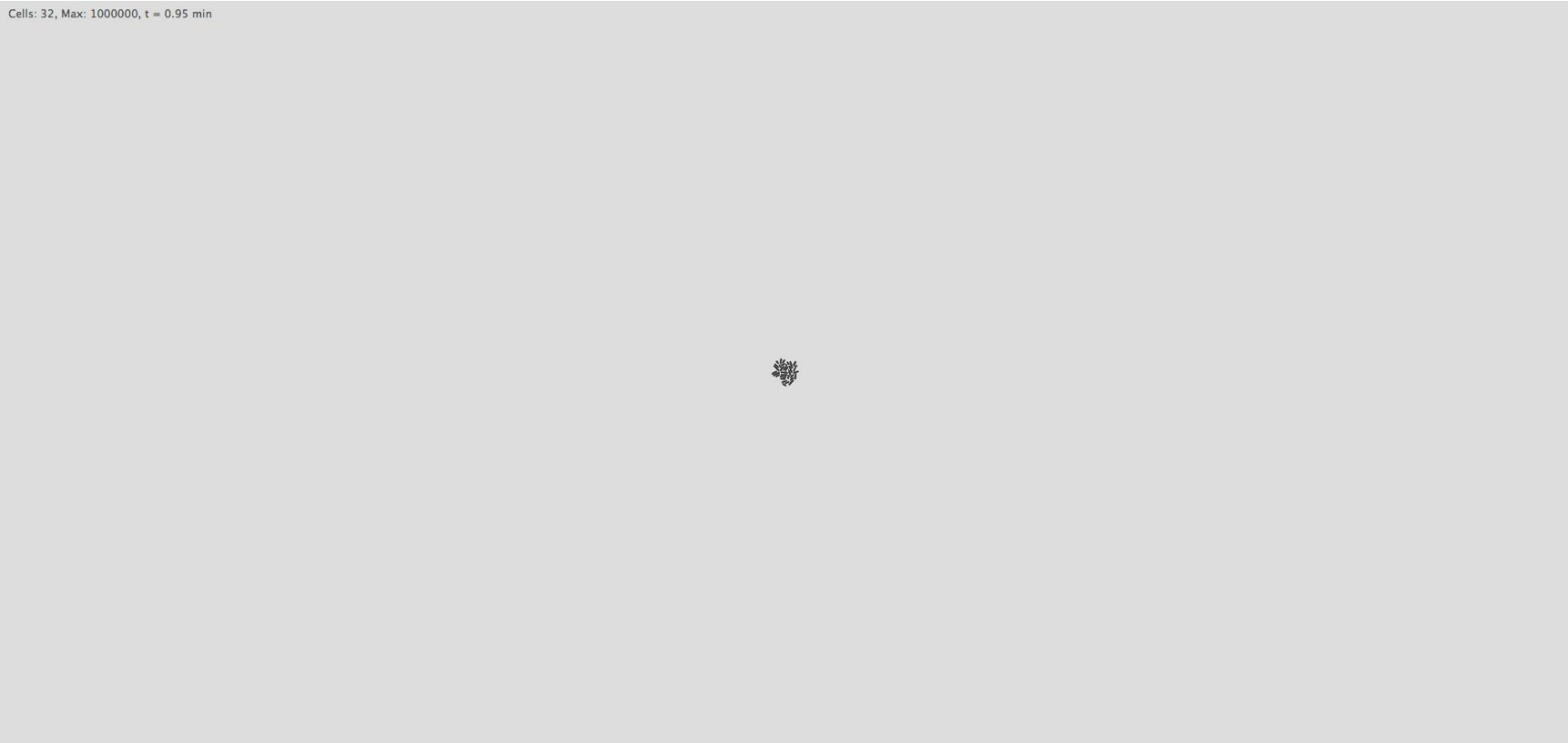
Example: Repressilator

Repressilator in GRO



Wet-lab Repressilator

Example: Edge detector



Plasmid-plasmid interaction

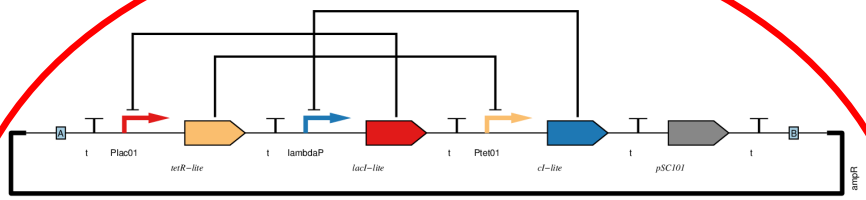
Nutrient uptake and growth module

We're working on more features…

- A simplified and faster version of signal diffusion
- Phage (lytic and non-lytic) dynamics.
- A new genetic module more "precise" with probabilities and threshold values of time for 0/1 protein states.
- 3D simulations: Morpheus, Biocellion

Perspectives

Quantitative data

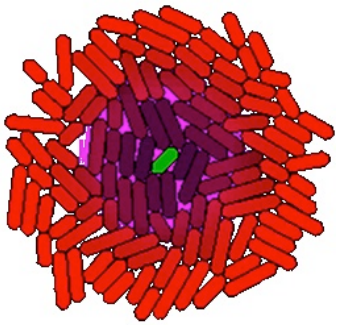



```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:sbol="http://sbols.org/v1#"
    xmlns:sublime="http://clarkparsia.com/sublime#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<sbol:DnaComponent rdf:about="urn:7914c028-828c-47c4-96d4-1af76c7ef044">
    <sbol:displayId>Unnamed</sbol:displayId>
    <sbol:annotation>
        <sbol:SequenceAnnotation rdf:about="urn:8198467f-c78f-474a-a014-1961ce5c72a8">
            <sbol:precedes rdf:resource="urn:9e5f6268-b09e-471b-9ac9-d1603d446fd9"/>
            <sbol:strand>+</sbol:strand>
            <sbol:subComponent>
                <sbol:DnaComponent rdf:about="urn:2939e2b4-f685-4e42-a8c3-9f63d7dea057">
                    <sbol:displayId>Plac01</sbol:displayId>
                    <sbol:name>Lac promoter</sbol:name>
                    <sbol:description></sbol:description>
```

SBOL

```
// Initial conditions
n_px := 3;
t_in_px := 30;
psx_to_none := 0.2;
delta_psx_to_none := 0.2;
n_psx_orig := 30;
t_in_psxo := 20;
```

GRO source code



GRO

Outline

- Systems Biology: GRO simulator
- **Synthetic Biology: PLASWIRES project**, Directed Evolution.
- DNA Computing: Inference with DNA molecules.
- Lab automation: EVOPROG project

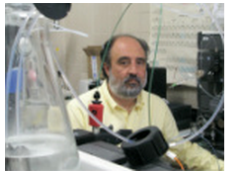
PLASWIRES project

PLASWIRES Project ID card

- Funded under: 7th FWP (Seventh Framework Programme)
- Area: FET Proactive: Evolving Living Technologies (EVLIT) (ICT-2013.9.6)
- Project reference: 612146
- Total cost: 2.62 million euro
- EU contribution: 2.01 million euro
- Execution: From 2013-10-01 to 2016-09-30
- Duration: 36 months

Alfonso Rodríguez-Patón (Coordinator)
Universidad Politécnica de Madrid (Sp
arpaton@fi.upm.es
www.lia.upm.es

Other partners:



Fernando de la Cruz
Universidad de Cantabria (Spain)
fernando.cruz@unican.es
http://grupos.unican.es/intergenomica/

Jim Haseloff
University of Cambridge (UK)
jh295@cam.ac.uk
http://www.haseloff-lab.org/

Didier Mazel
Institut Pasteur (France)
mazel@pasteur.fr
http://openwetware.org/wiki/Mazel

PLASWIRES

"Engineering Multicellular Biocircuits: Programming Cell-Cell Communication Using PLASmids as WIRES"

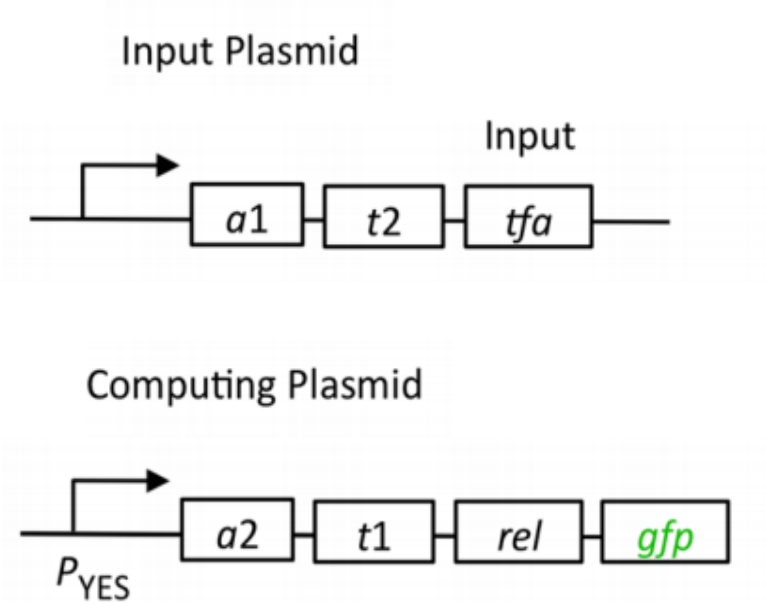
A Synthetic Biology FP7 European research project

PLASWIRES' main goal: To show how to program a parallel distributed living computer using conjugative plasmids as wires between cellular processors.

Outline

- Systems Biology: GRO simulator
- Synthetic Biology: PLASWIRES project, **Directed Evolution.**
- DNA Computing: Inference with DNA molecules.
- Lab automation: EVOPROG project

An Autonomous In Vivo Dual Selection Protocol for Boolean Genetic Circuits



	OFF selection		ON Selection	
Input	0	0	1	1
Output	0 (OFF)	1 (ON)	0 (OFF)	1 (ON)
	Correct (Survive)	Incorrect (killed by T1)	Incorrect (killed by T2)	Correct (Survive and conjugate)

Figure 2. YES-gate in vivo selection. Two plasmids needed: input plasmid (a conjugative plasmid with the activator input *tfa*) and computing plasmid (a mobilizable plasmid that contains the circuit P_{YES} to be selected). Selection genes: *t1*: toxin gene 1; *a1*: antitoxin gene 1; *t2*: toxin gene 2; *a2*: antitoxin gene 2; *rel*: relaxase gene (a conjugation gene). *tfa*: transcription factor (activator); *tfa* is an activator of promoter P_{YES}. *gfp*: green fluorescent protein gene.

Beneš, D., Sosík, P., & Rodríguez-Patón, A. (2015). An Autonomous In Vivo Dual Selection Protocol for Boolean Genetic Circuits. **Artificial Life** 21: 247–260 (2015) doi:10.1162/ARTL_a_00160

An Autonomous In Vivo Dual Selection Protocol for Boolean Genetic Circuits

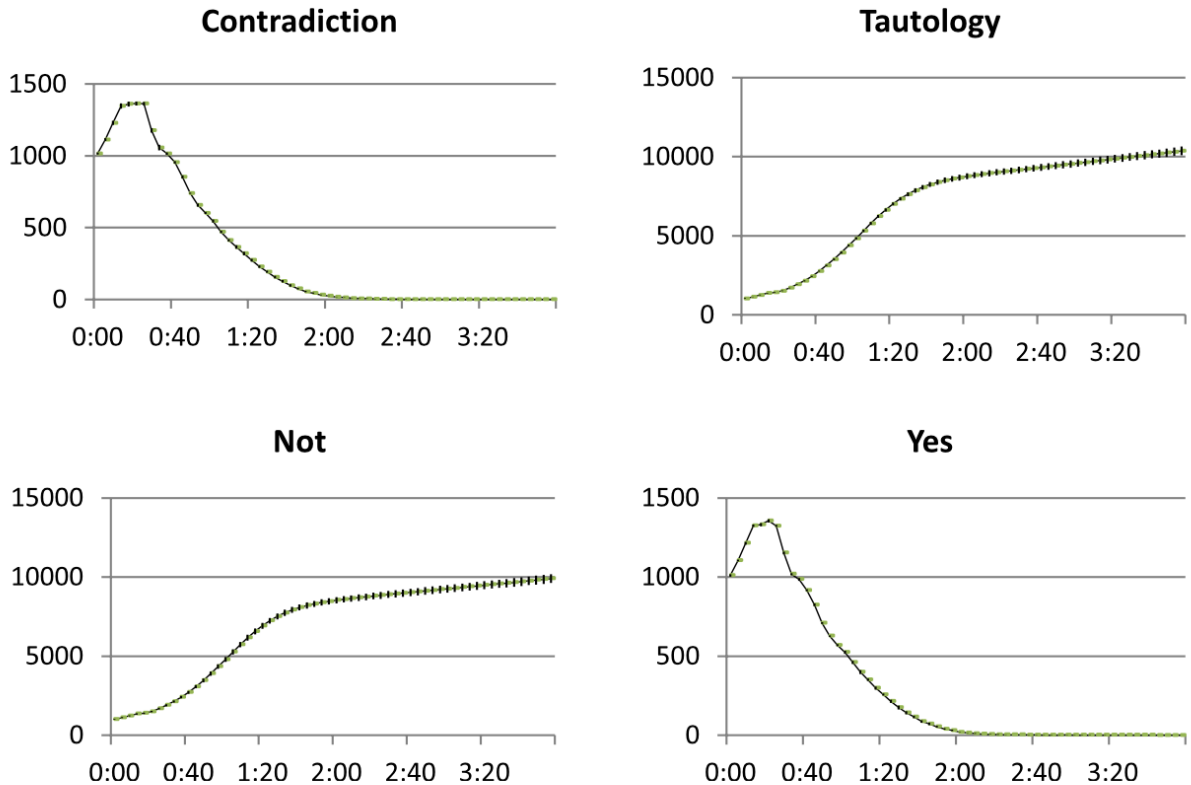


Figure 6. Dual selection of a NOT gate: ON selection step. Initially, there are 1000 bacteria (*y*-axis) with each type of one-input logic gate: YES, NOT, TAUT, and CONT. Input is 0; then only bacteria with output = 1 (ON state) should survive. Experiment finishes at 4 h (*x*-axis).

Beneš, D., Sosík, P., & Rodríguez-Patón, A. (2015). An Autonomous In Vivo Dual Selection Protocol for Boolean Genetic Circuits. **Artificial Life** 21: 247–260 (2015) doi:10.1162/ARTL_a_00160

Outline

- Systems Biology: GRO simulator
- Synthetic Biology: PLASWIRES project, **Directed Evolution.**
- **DNA Computing: Inference with DNA molecules.**
- Lab automation: EVOPROG project

Inference with DNA molecules

Basic Inference rules: modus ponens and modus tollens

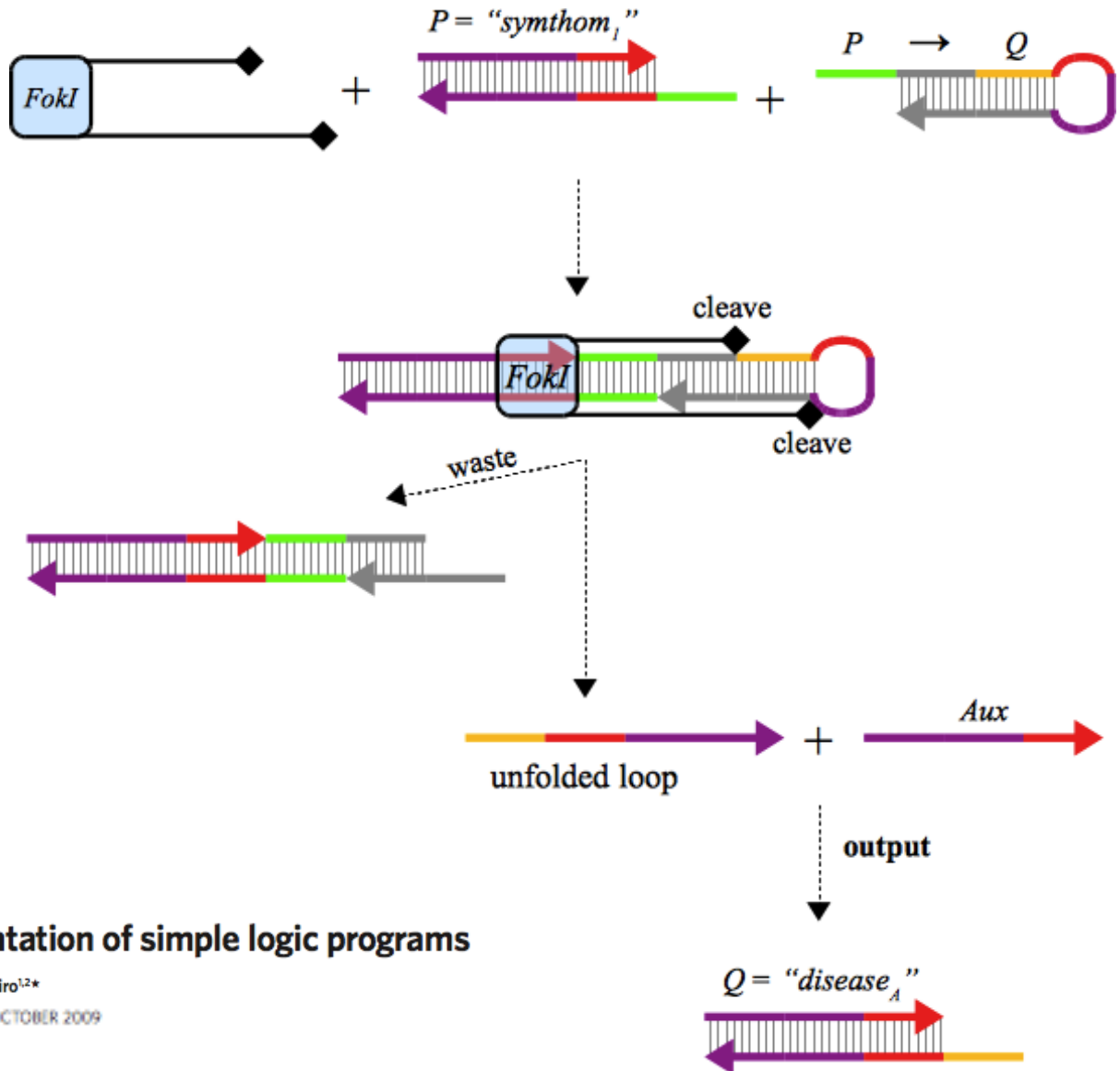
Modus Ponens states that from *P and the implication P → Q one can deduce Q.*

If *P*, then *Q*
P.
Therefore, *Q*

Modus Tollens states that from *NOT-Q and the implication P → Q one can deduce NOT-P.*

If *P*, then *Q*
Not-*Q*.
Therefore, Not-*P*

Previous works: Inference with DNA molecules



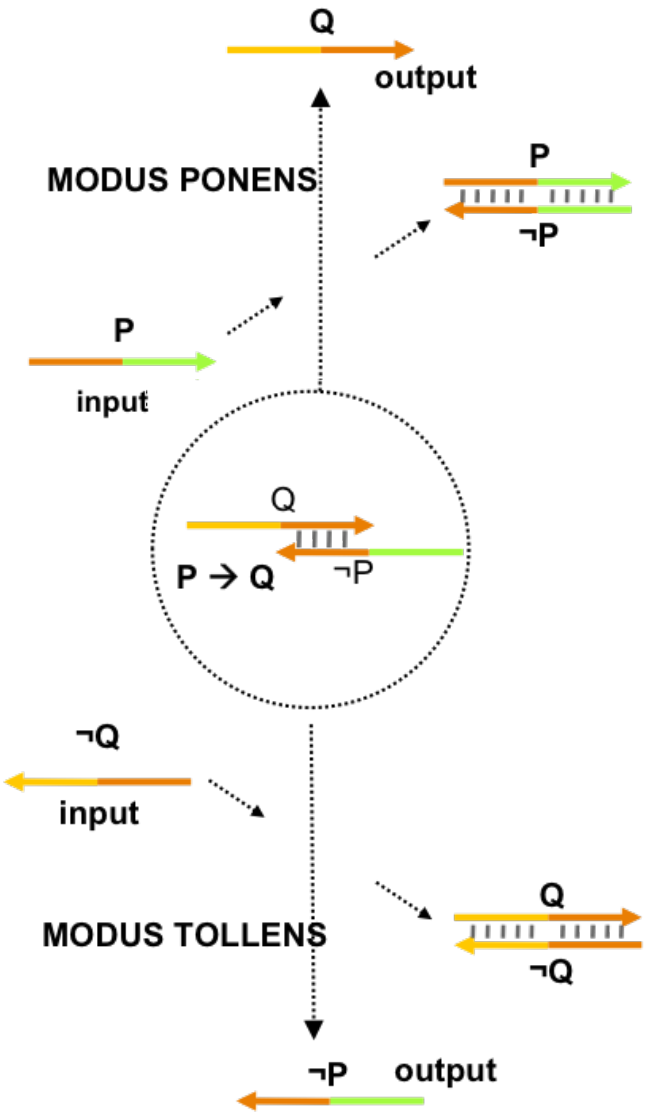
Molecular implementation of simple logic programs

Tom Ran[1], Shai Kaplan[2,3] and Ehud Shapiro[1,2]*

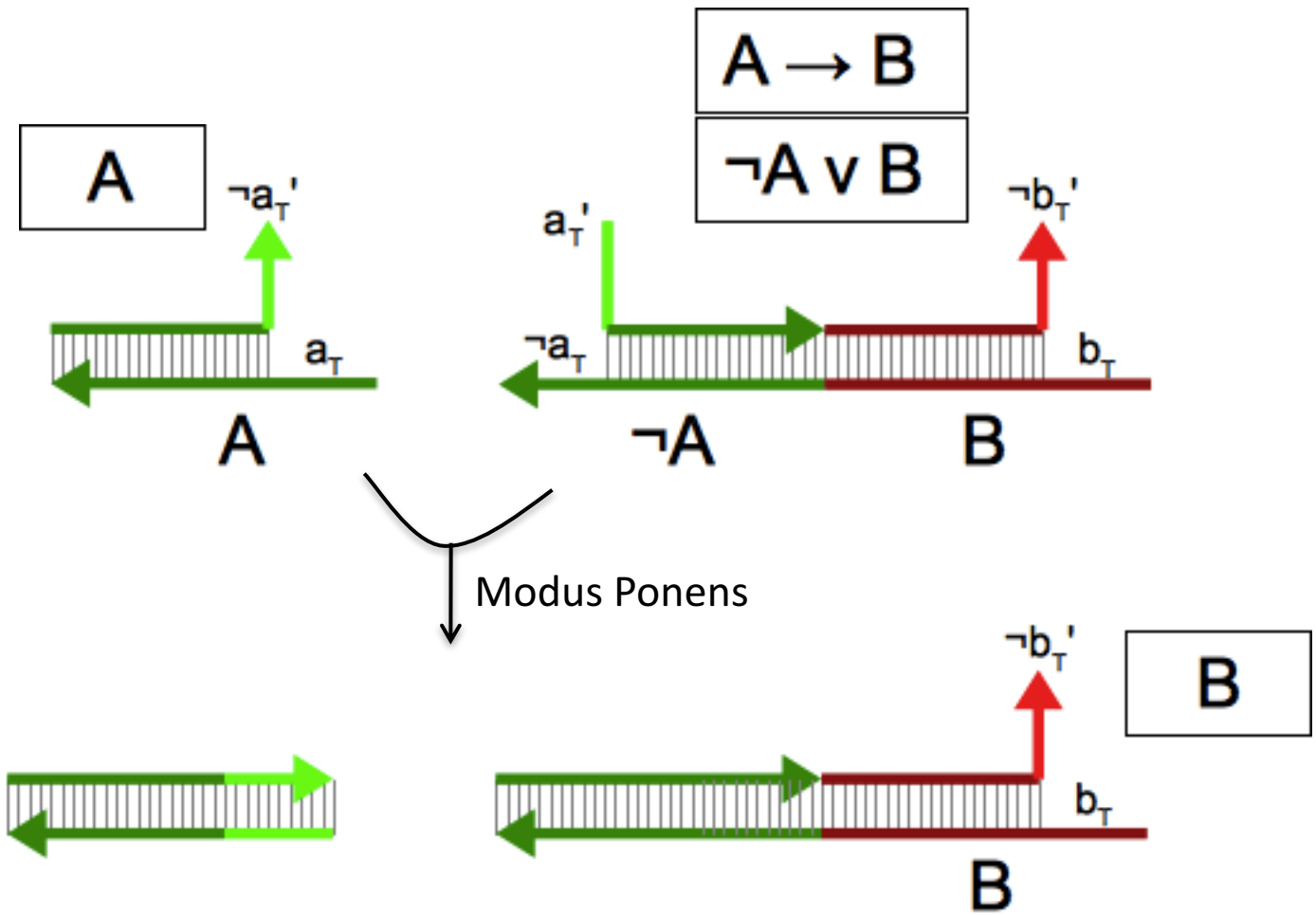
NATURE NANOTECHNOLOGY | VOL 4 | OCTOBER 2009

DNA strand displacement and competitive hybridization

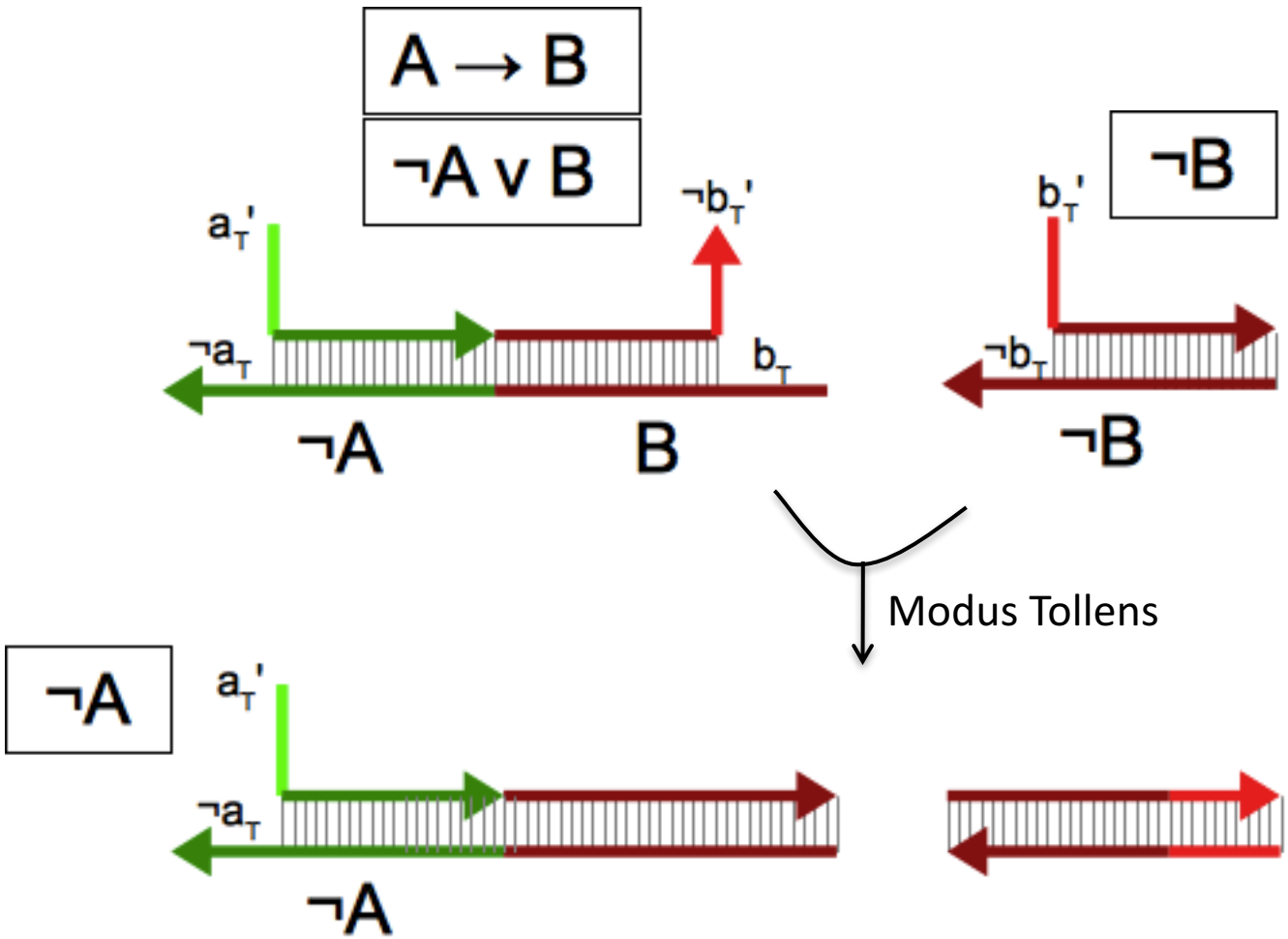
Inference with DNA strand displacement



Special case of resolution: Modus Ponens



Special case of resolution: Modus Tollens



Solving SAT applying resolution with autonomous 4-way branch migration

Examples:

$F_1 = B \wedge \neg B$ is unsatisfiable.

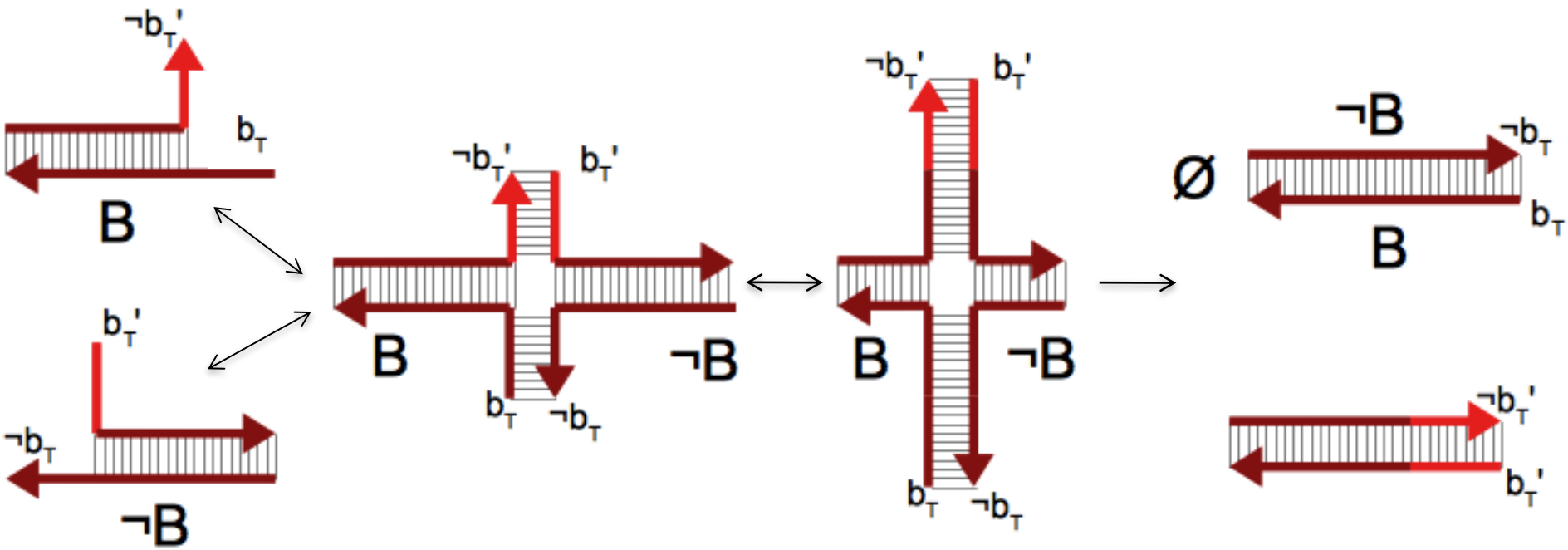
$F_2 = (A \vee B) \wedge (\neg B \vee C)$ is satisfiable.

$F_3 = \neg A \wedge (A \vee B) \wedge \neg B$ is unsatisfiable.

Applying resolution to all the clauses, if a refutation can be derived from the initial formula, then the formula is **unsatisfiable**. A refutation is a sequence of clauses obtained by iterated application of the resolution rule that finish in the **empty clause**.

In our experimental set-up the cover strands contains a fluorescent marker so the **empty clause** corresponds to a non-fluorescent double-stranded molecule with nicks between all parts encoding variables.

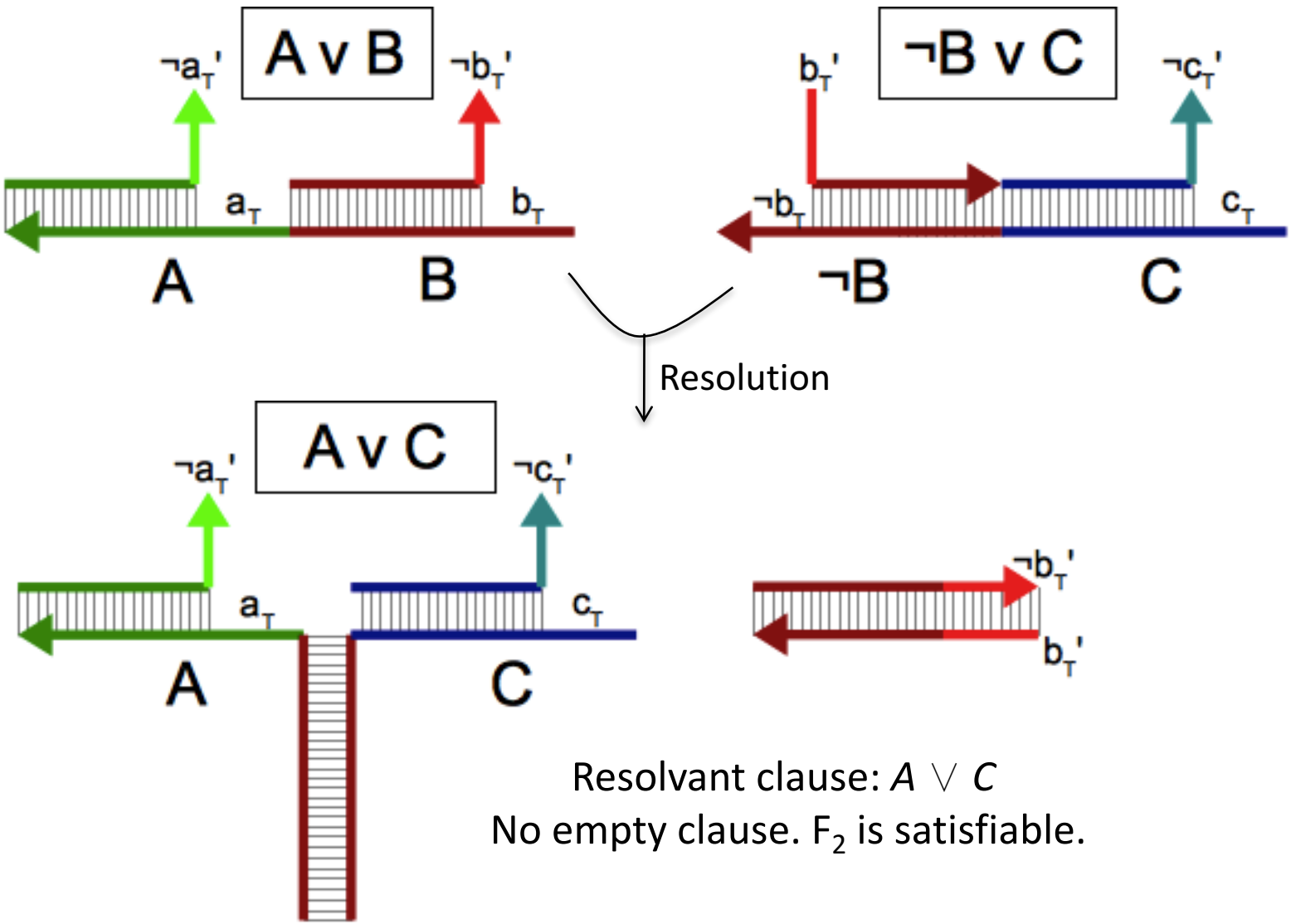
4-way DNA branch migration: $B \wedge \neg B$



Annealing 4-way branch migration Strand exchange

Autonomous resolution determines that $F_1 = B \wedge \neg B$ is unsatisfiable: we get the empty clause

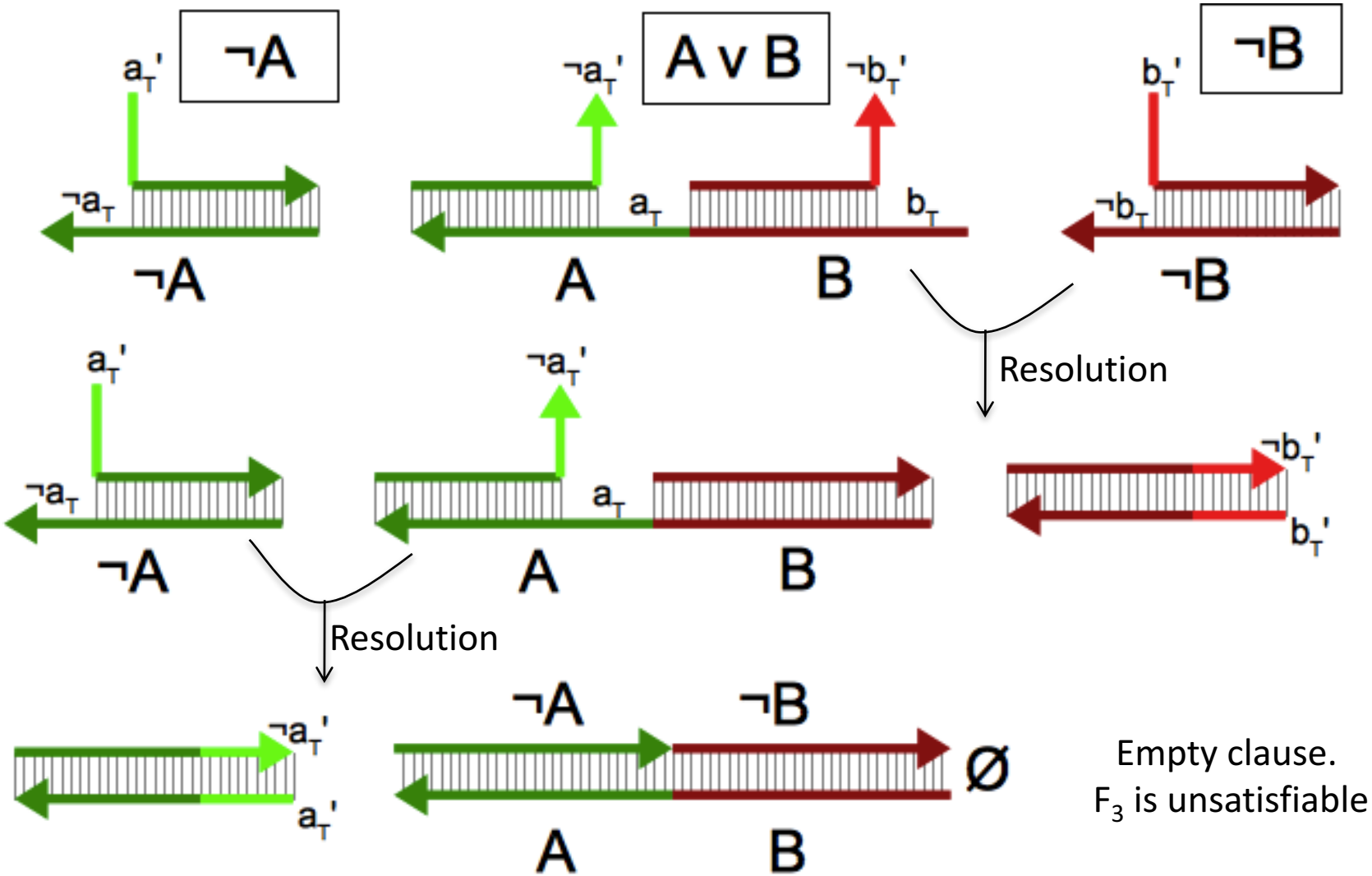
Solving SAT applying resolution with autonomous 4-way branch migration



Resolvant clause: $A \vee C$

No empty clause. F_2 is satisfiable.

Solving SAT applying resolution with autonomous 4-way branch migration



Empty clause.
F_3 is unsatisfiable

Outline

- Systems Biology: GRO simulator
- Synthetic Biology: PLASWIRES project, **Directed Evolution.**
- **DNA Computing: Inference with DNA molecules.**
- **Lab automation: EVOPROG project**

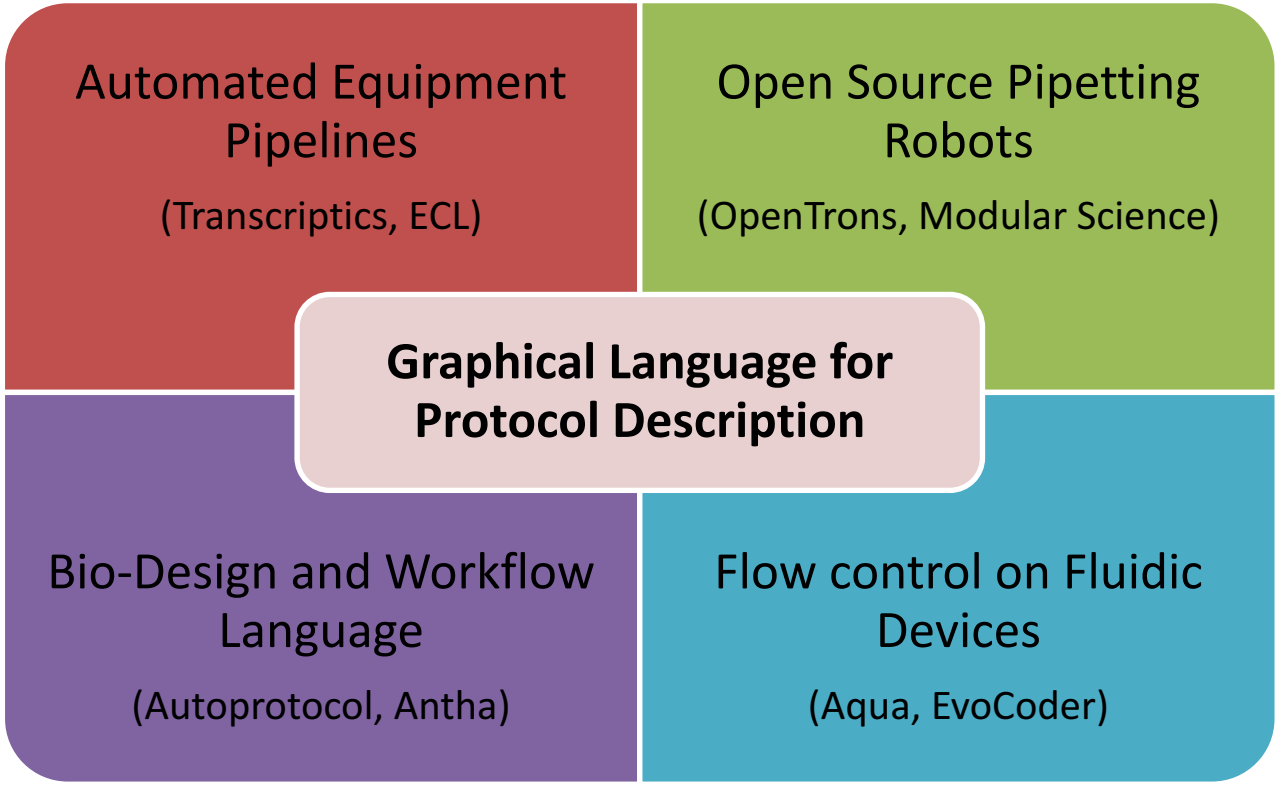
Applying ICT for biology Automation
A general view

Inteligencia Artificial
Laboratorio
POLITÉCNICA
www.lia.upm.es

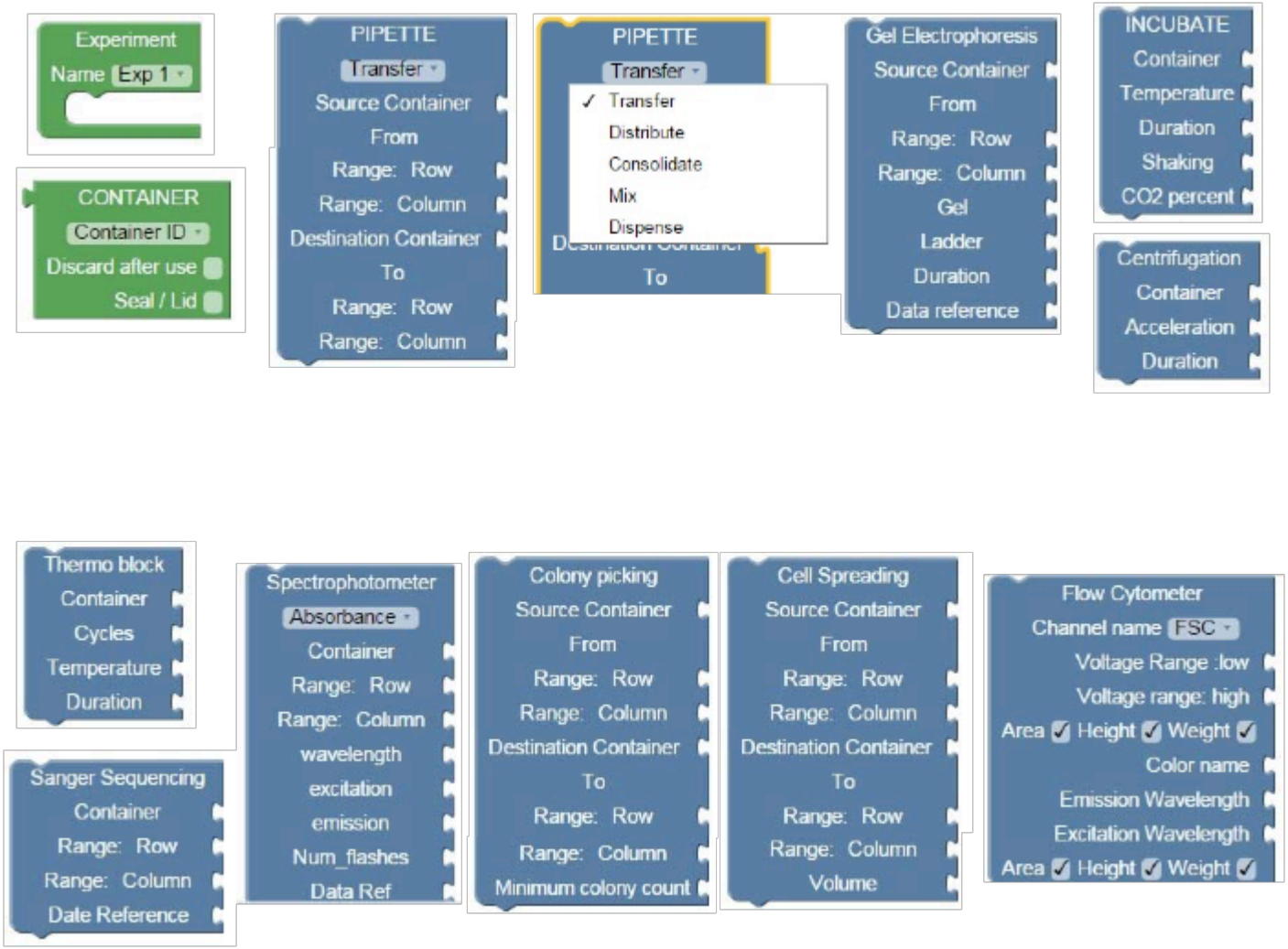
EVOPROG

SEVENTH FRAMEWORK PROGRAMME

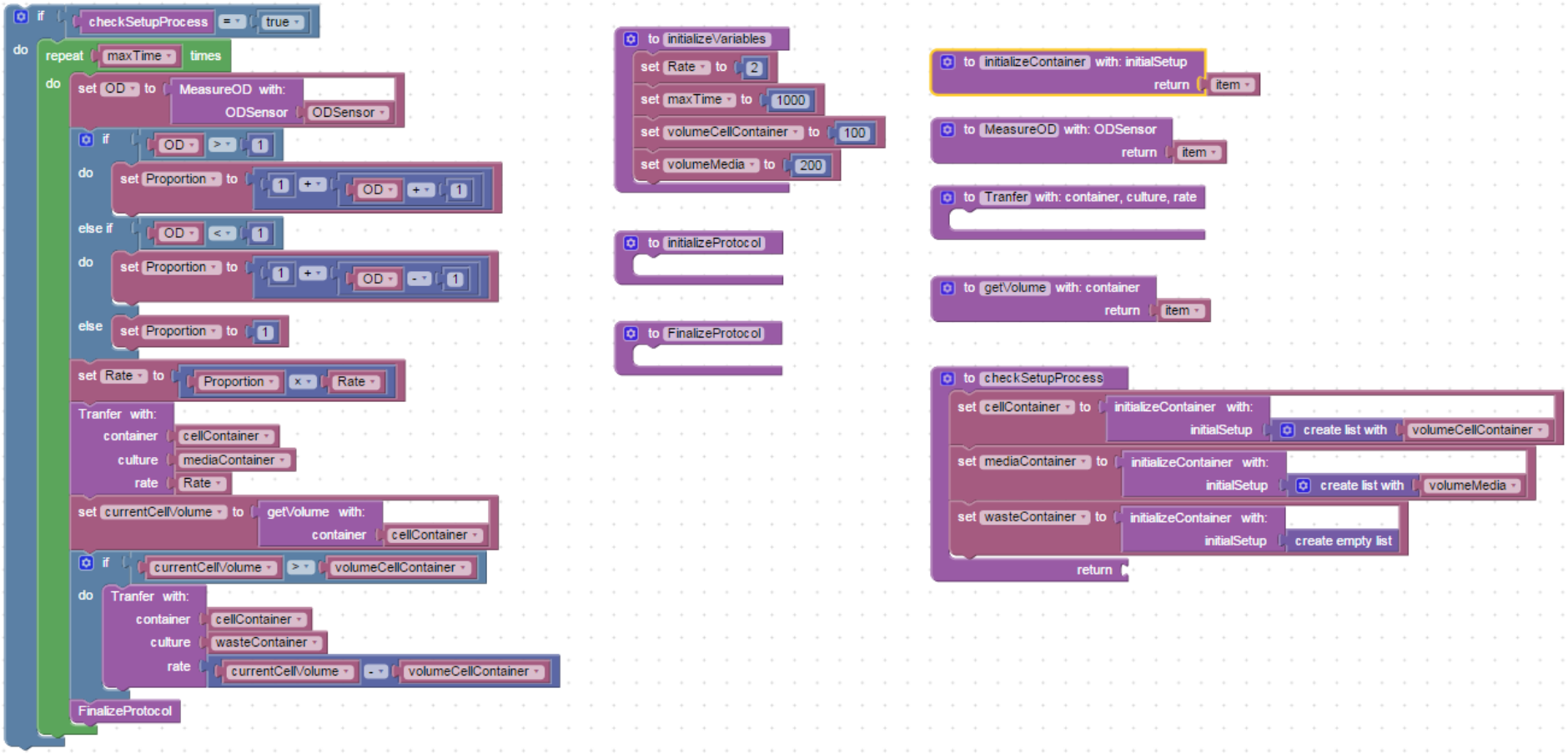
Automated Protocol Execution in Biology-



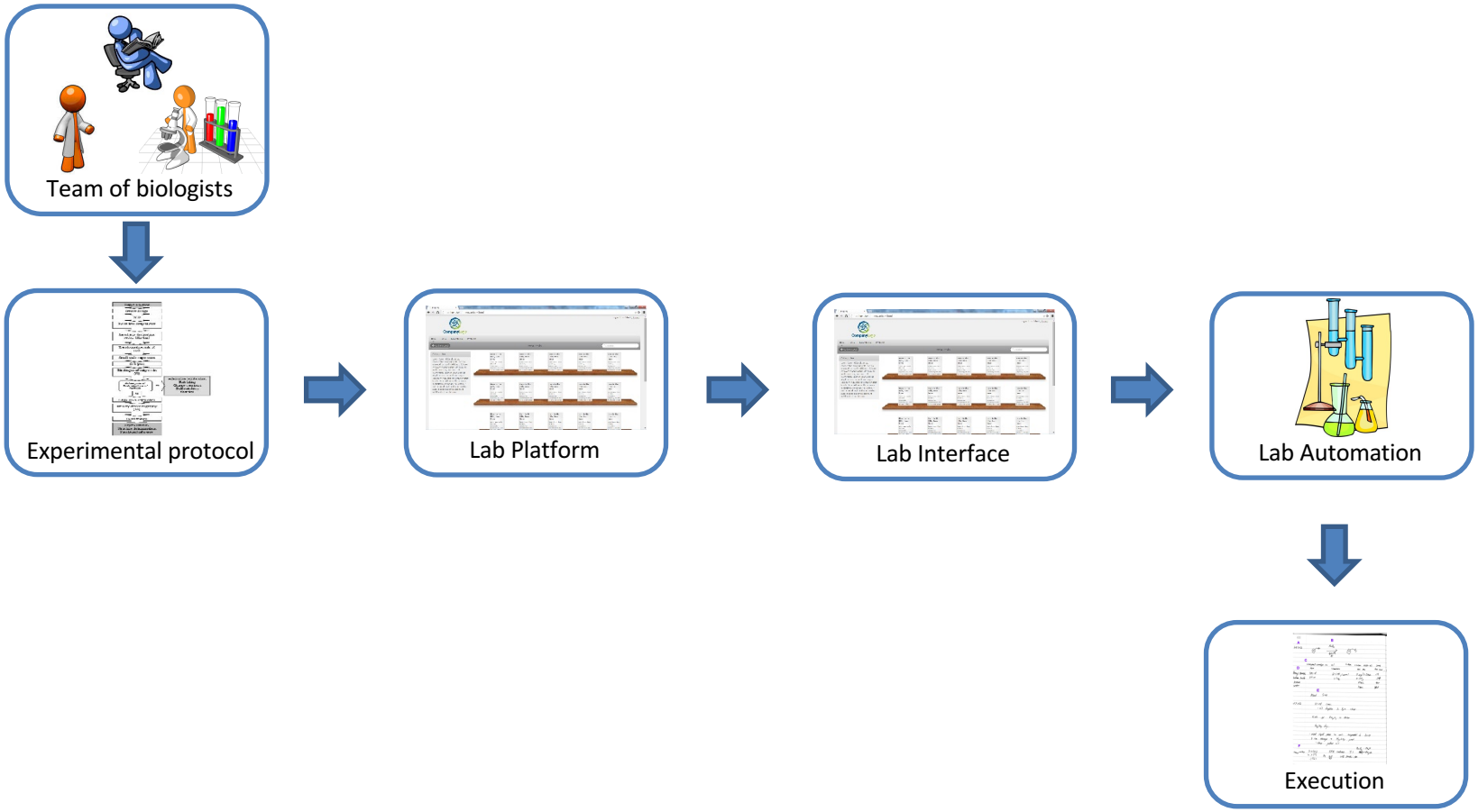
Graphical languages for Lab protocol description



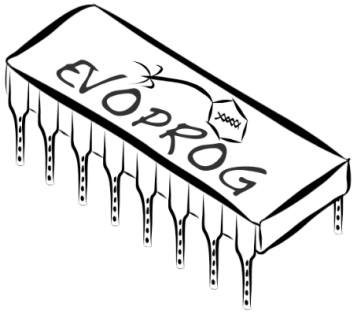
Graphical languages for Lab protocol description



Working on each stage of the process…



EVOPROG Project



EVOPROG

Project title	Project number	Call (part) identifier	Funding scheme
General-Purpose Programmable Evolution Machine on a Chip	610730	FP7-ICT-2013-10	Collaborative project

In our EVOPROG consortium we will program phage and bacteria to compute our combinatorial optimisation algorithms by constructing and using a high-throughput droplet device for the directed evolution of biomolecules de novo, integrating for the first time in silico and in vivo evolution. For this, we will develop a general-purpose 3D biochip utilizing computational and fluidics automation which could also be applied to perform in vivo molecular biology operations in high-throughput (including time-dependent characterisations of gene expression levels using fluorescent proteins).

University	**Involved People**
University of Warwick (UWAR)	Dr. Alfonso Jaramillo (Coordinator) Ms. Mariel Montesinos
Universidad Politecnica de Madrid (UPM)	Dr. Alfonso Rodríguez-Patón
Imperial College of Science, Technology and Medicine (IC)	Dr. Mark Isalan
Consejo Superior de Investigaciones Cientificas (CSIC)	Dr. Victor de Lorenzo
University of Glasgow (UoG)	Dr. Lee Cronin
University d'Evry-Val d'Essonne (UEVE)	Dr. Alfonso Jaramillo Dr. Shensi Shen Dr. Ilias Tagkopoulos

EVOPROG project: LIA's Tasks

Transforming biologists' thoughts and designs in real instructions and parameters for the Evoprog machine