UMR IRISA

Institut de Recherche en Informatique et Systèmes Aléatoires

Programmation par ensembles-réponses, application à l'étude de systèmes biologiques à grande-échelle

Anne SIEGEL, CNRS, IRISA (Rennes)

PRESENTATION

Scientific background

- Math: discrete dynamical systems & fractals
- Modelling : systems biology
- Computer science: Knowledge representation

IRISA Laboratory

- Computer science lab of Rennes
- 800 employees, > 35 teams
- Seven departments, among which « data and knowledge management »

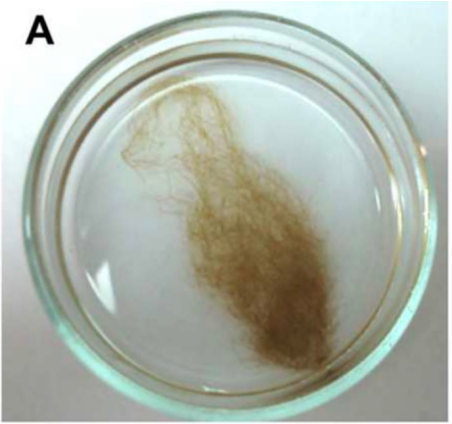
Bioinformatics & systems biology group

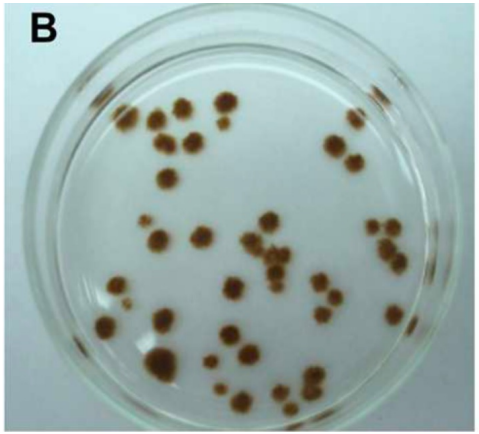
- **2 teams and 1 platform (45 in total)**
- **Dyliss team**: symbolic methods for integrating and reasoning over large-scale heterogeneous data

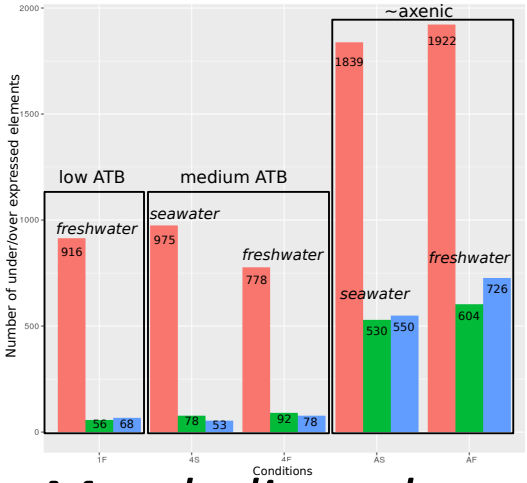
A SIMPLE QUESTION FROM BIOLOGY: WHY ALGAE NEED BACTERIA ?

CNRS • SORBONNE UNIVERSITÉ
Station Biologique de Roscoff

S. Dittami







E. siliculosus

[Dittami2014, Tapia2016]

In axenic condition…

Metabolic scale

Fremy, Frioux, unpublished

Question: which bacteria (and how) allow algae to grow ?

C. Frioux, S. Dittami …

UMR IRISA

OUTLINE

Underlying life science question: elucidate functions at the microbiological scale

Methodological question: How computer science can be useful ?

1. Short historical reminder: from observations to discoveries in life sciences
2. (Identification of) dynamical systems
3. Focus on the metabolic scale.
4. Reasoning over discrete dynamical systems
5. Knowledge representation : introduction to ASP
6. More insights on ASP
7. Back to biology towards more tricky problems
8. Still new issues

1. From observations to discoveries in life sciences

1. Short historical reminder: from observations to discoveries in life sciences
2. (Identification of) dynamical systems
3. Focus on the metabolic scale.
4. Reasoning over discrete dynamical systems
5. Knowledge representation : introduction to ASP
6. More insights on ASP
7. Back to biology towards more tricky problems
8. Still new issues

DEALING SCIENTIFICALLY WITH NATURE

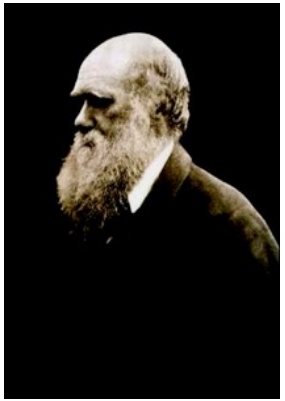
- Experimental science viewpoint (Starting with Plato, astronomy): the aim of science is to **propose hypotheses intended to save the appearances**.
    - The task of hypothesis building is provided by abduction.

- Demonstrative science viewpoint (Aristotle): the aim of science is to **discover the principles which would serve as explanations for the observed phenomena (Aristotle)**.
    - The task of discovery is provided by induction.

Source: https://www.ucm.es/data/cont/docs/481-2013-10-14-abductive.pdf

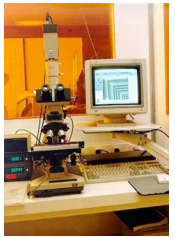
OBSERVING (19TH CENTURY)

Use observations (and comparisons) to exhibit biological theories (=dogma)



Charles Darwin,
Theory of evolution

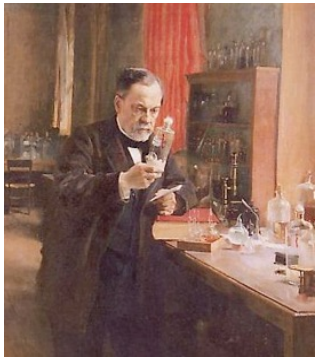
Jeanne Villepreux-Power,
Founder of Marine Biology



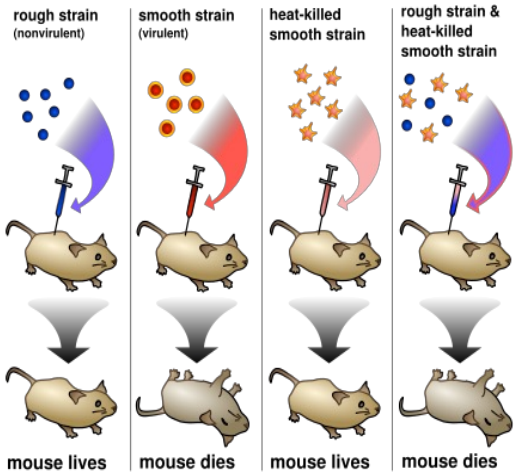
Abduction is the process of inferring certain (…) laws and hypotheses (...) that explain or discover some (…) phenomenon or observation; it is the process of reasoning in which explanatory hypotheses are formed and evaluated. (Magnani, 2001)

EXPERIMENTING (20TH CENTURY)

Use experiments to prove hypotheses and generate discoveries



Louis Pasteur,
microbiology

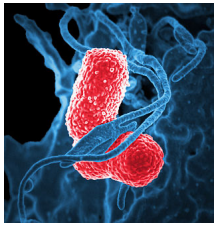


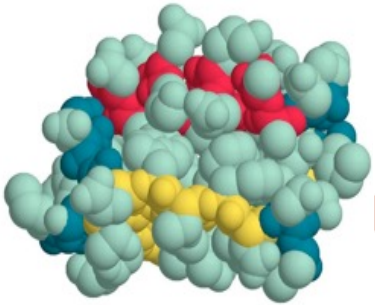
Germaine Benoit,
Infectious diseases
(therapeutic molecules)

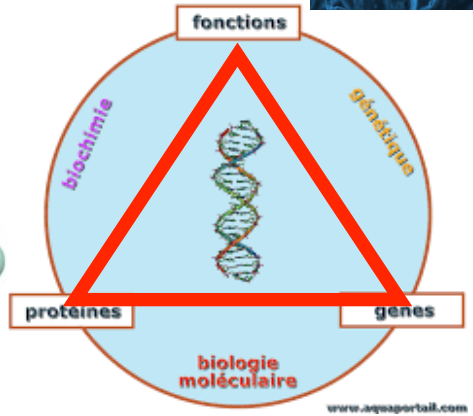
Deduction is used to take advantages of existing theories
→ New theories emerge from incompleteness between former theories and experimental observations

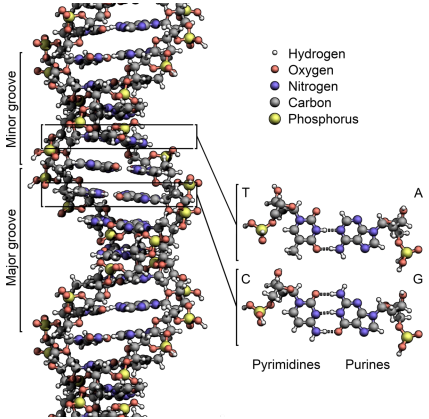
INCREASE OBSERVATIONS : MOLECULAR BIOLOGY (2ND HALF OF 20TH CENTURY)

Develop new technological devices to observe and experimental at smaller scales





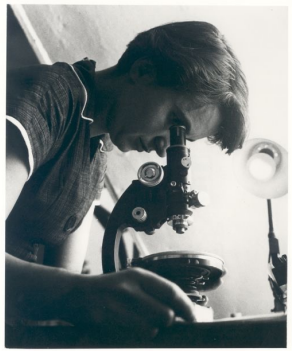




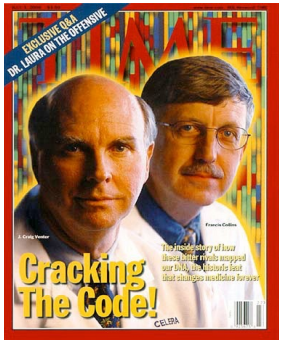
→ Acquiring data towards induction processes

GENOMES

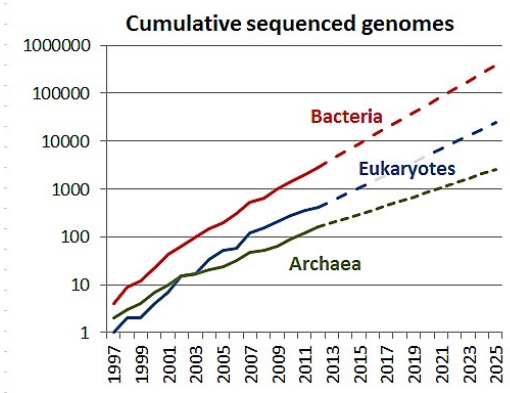
We now have the capability to sequence the genome of organisms and samples in a few time.



ADN (Watson, Creek, Franklin)



Genome Sequencing
(Venter, Collins)

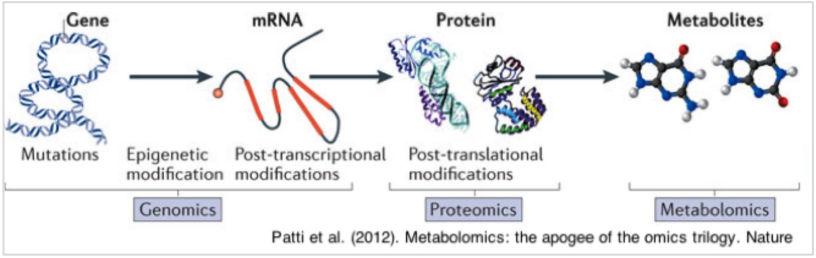


https://doi.org/10.6084/m9.figshare.723384.v1

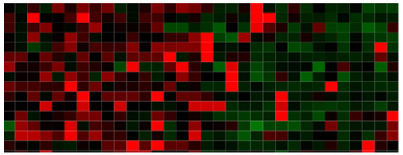
→ Massive data to explore

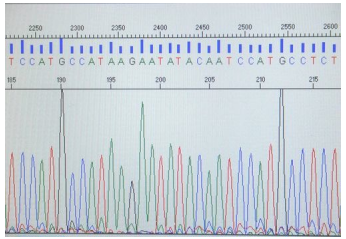
TRANSCRIPTOMICS – PROTEOMICS – (…)

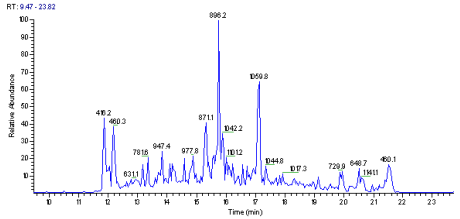
Measure the consequences of the presence of genes in species genomes



Patti et al. (2012). Metabolomics: the apogee of the omics trilogy. Nature

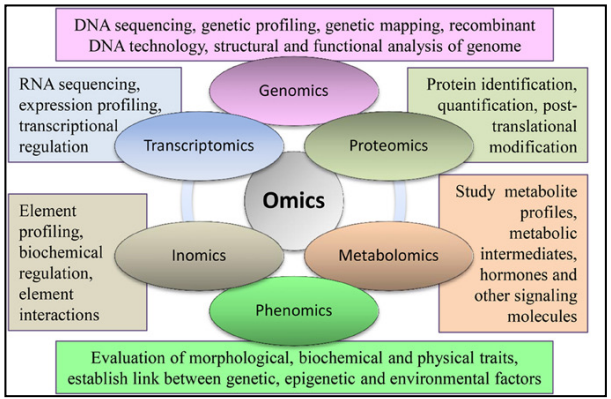




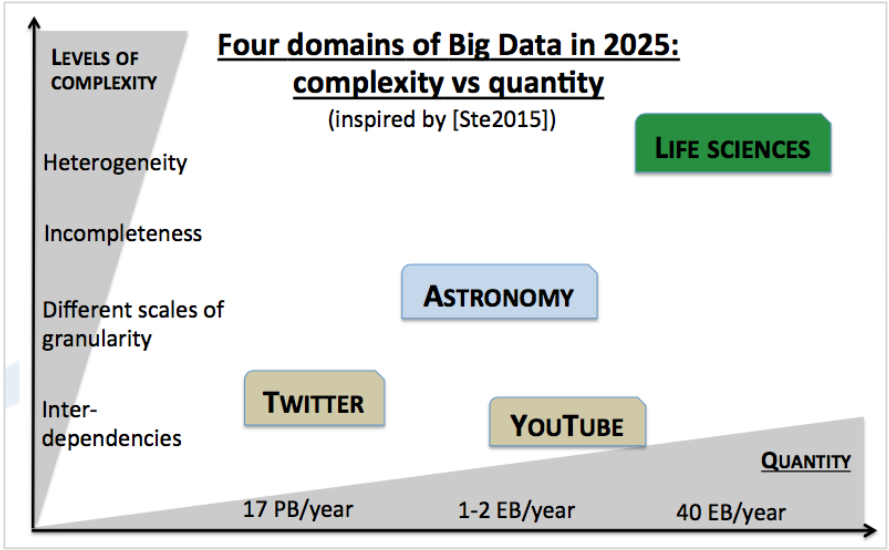


Confront genomes with their observed consequences (phenotypes)

LIFE-SCIENCES DATA NIGHTMARE



Data deluge

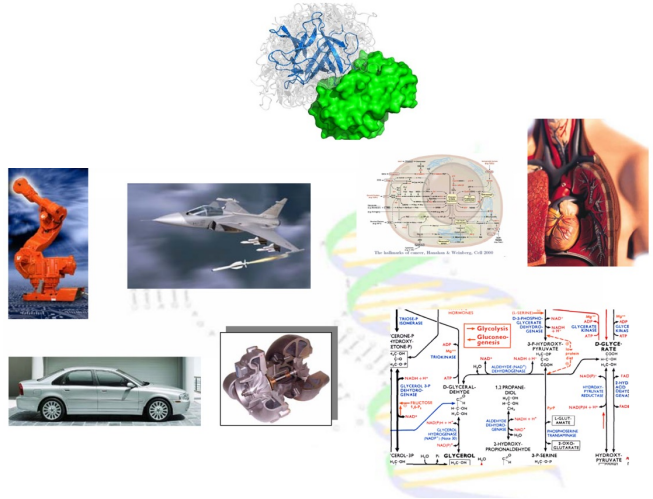


Worst data that may exist (size and quality) (stephens, 2015)

Data-based induction : characterize diseases (imaging), diagnosis.

MASSIVE GENOMES ANALYSES

Strong deception: life-science complexity (species, diseases) cannot be explained by the genome diversity



Limits of the reductionist approach

« Biology requires to examine the structure and dynamics of a cellular function rather than the characteristics of isolated parts of a cell » *(Kitano, 2002)*

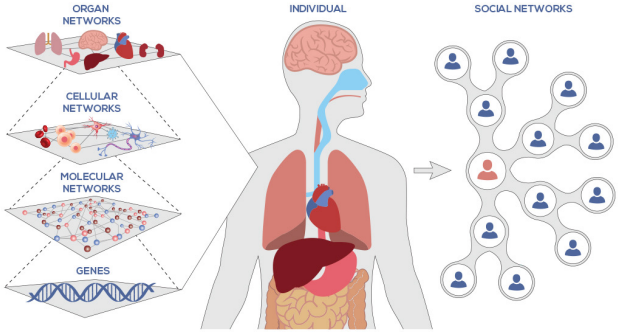
IN OTHER WORDS: BIOLOGY IS A COMPLEX SYSTEM

… data-based induction discovery is far from possible

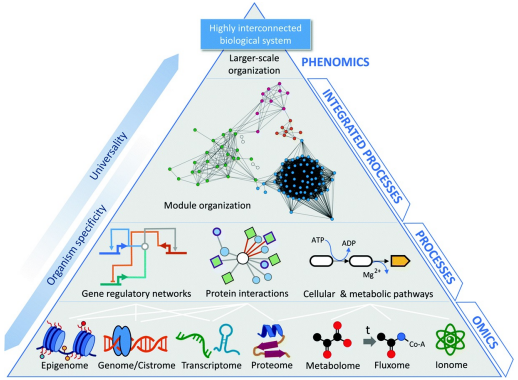
2. (Identification of) dynamical systems

1. Short historical reminder: from observations to discoveries in life sciences
2. **(Identification of) dynamical systems**
3. Focus on the metabolic scale.
4. Reasoning over discrete dynamical systems
5. Knowledge representation : introduction to ASP
6. More insights on ASP
7. Back to biology towards more tricky problems
8. Still new issues

SYSTEMS BIOLOGY



https://isbscience.org/about/what-is-systems-biology/

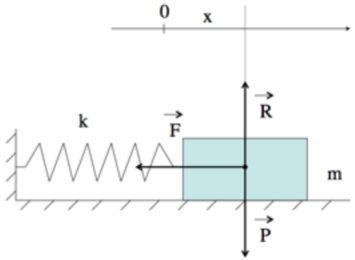


https://link.springer.com/chapter/10.1007/978-3-030-18601-2_8

Systems biology. Biological observations results from the superposition and the chaining c multi-layer mechanisms.

- Biology is a complex (dynamical) system
- Produce predictive statements that can be experimentally validated (deduction) ?

MODELLING A COMPLEX SYSTEM



Système masselotte-ressort.
La masselotte de masse m est supposée glisser sans frottement. $\vec{R}$ est la réaction du support, égale au poids $\vec{P}$. La masselotte n'est donc soumise qu'à la seule force de rappel $\vec{F}$, proportionnelle à la distance x à sa position à l'équilibre (où $\vec{F}$ est nulle).

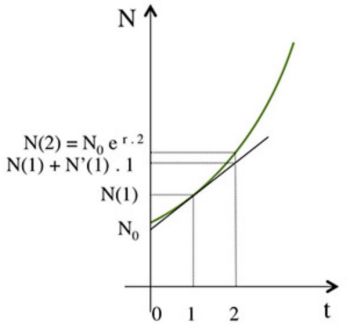
Describe mechanisms

$$m\,x''(t) = -\,k\,x(t)$$

$$x(t) = A\cos(\omega t + \phi)$$

$$\frac{d\mathbf{M}(t)}{dt} = \mathbf{S}\,\mathbf{v}(\mathbf{M}(t))$$

Write differential equations



Si, à partir du temps $t_1 = 1$, la croissance de N se poursuivait à une vitesse constante égale à la dérivée en ce point, c'est-à-dire avec la valeur $N'(t_1) = N'(1)$. N croîtrait linéairement pour atteindre, au temps $t_2 = 2$, la valeur $N(t_1) + N'(t_1)$. $(t_2 - t_1) = N(1) + N'(1) \,.\, 1 \approx 1061{,}36$. Mais la valeur de la dérivée N'(t) varie à chaque instant, puisqu'elle est égale à r . N(t). De fait, la valeur $N(t_2) = N(2)$ est égale à $N_0\, e^{r.2} \approx 1061{,}83$.

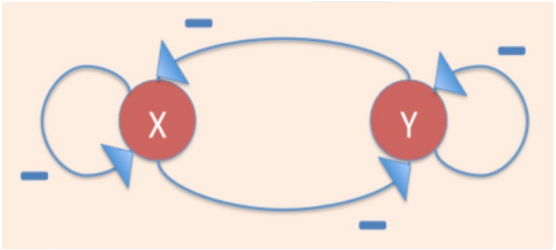
Solve differential equations or simulate them

Source : https://interstices.info/systemes-dynamiques-et-equations-differentielles/

DYNAMICAL SYSTEMS

Historical motivation Model the evolution of the set of components in a system according to time.

$$F : \underbrace{\mathbb{T}}_{(t\text{ime}} \times \underbrace{\mathbb{S}}_{\text{state})} \rightarrow \mathbb{S} \quad (t, \mathbf{z}) \mapsto \underbrace{F(t, \mathbf{z})}_{\text{new state at time } t}$$



$$\frac{dX}{dt} = \frac{k}{K + Y^n} - aX$$
$$\frac{dY}{dt} = \frac{l}{L + X^n} - bY$$

Parameterized numerical system

$$f(X) \leftarrow 1 - Y$$
$$f(Y) \leftarrow 1 - X$$

Boolean model with asynchronous update scheme

Identification/calibration of a dynamical system Find the best function F which parcimounously explains and describes the observed responses of a system.

MODEL IDENTIFICATION/CALIBRATION SINCE THE 18TH CENTURY

What has always allowed a model identification

- **A priori knowledge about the (conservation/behavior) laws governing the system**
  - Predetermined shape for the function F
- **Limited number of components**
  - Reduction of the search space
- **Wide panel of sensors and perturbations**
  - Discriminate parameters

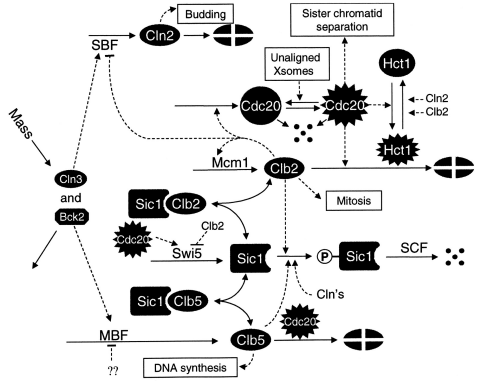
Where is the complexity ?

- The search space grows exponentially with the number of measured compounds

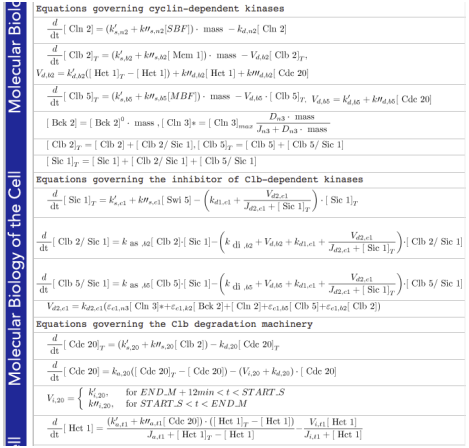
The more compounds we measure, the less calibrated a system can be.

EXAMPLE

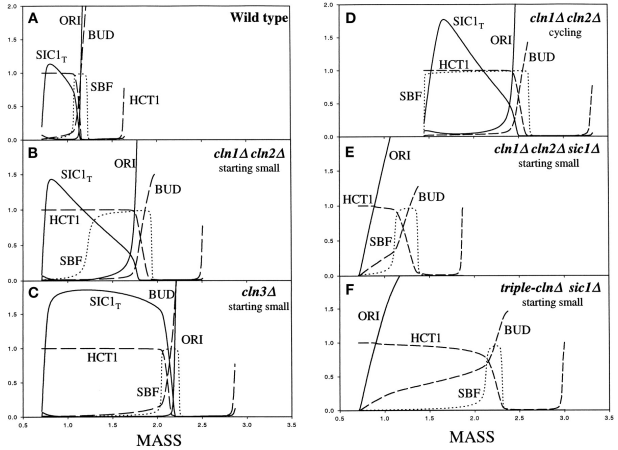
Yeast cell cycle (Tyson 1990)



Graph model of interactions, transformations and controls



Fitting of a numerical model (ODE)

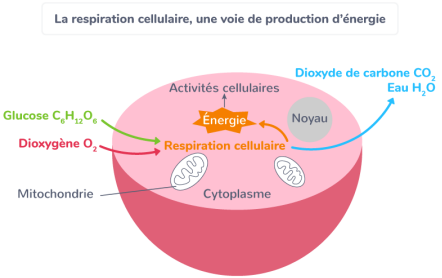


Simulation of the effect of mutations

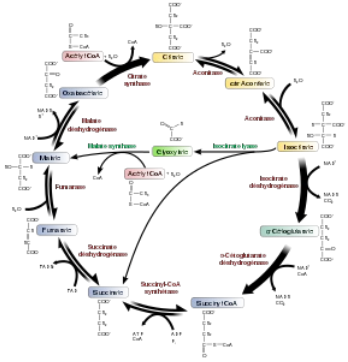
Limitation: to add a molecule, we need to find many new parameters

FROM BIOLOGICAL FUNTIONS TO METABOLIC MAPS

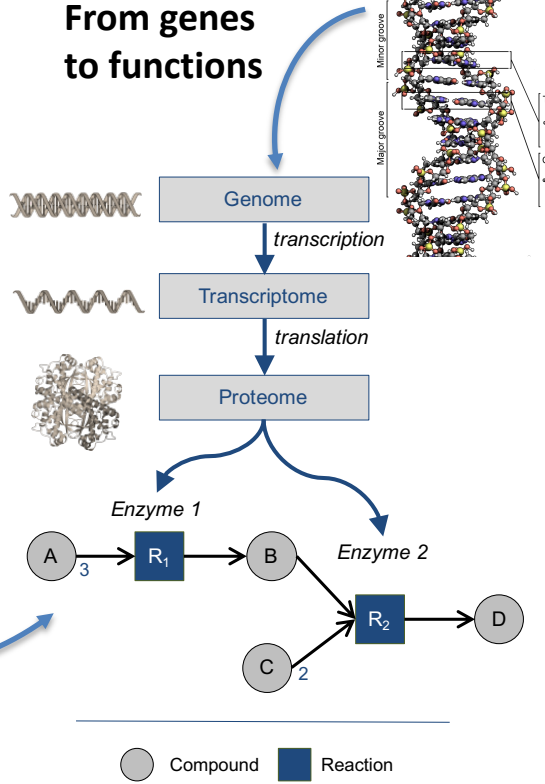
Metabolic function

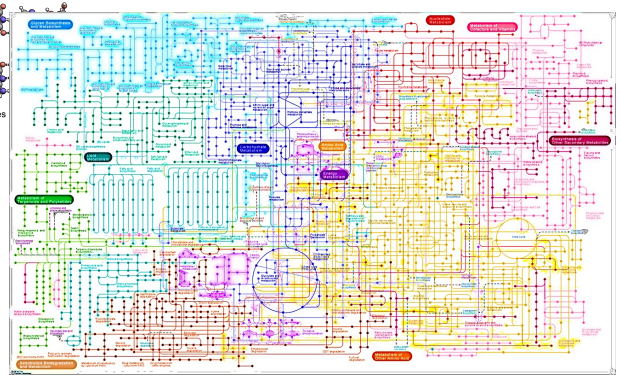


Molecules involved in the function



From genes to functions





Metabolic map = bipartite directed graph

$$\frac{d\mathbf{M}(t)}{dt} = \mathbf{S}\,\mathbf{v}(\mathbf{M}(t))$$

Dynamical model
1500 reactions
2 parameters/reaction to fit

Biomolecular dynamical systems cannot be identified !!!

DIFFERENCES BETWEEN SCIENCES

Physics

- **Knowledge**
  Fundamental laws
- **Sensors**
  Numerous
- **Perturbations**
  Controlled reactors

DIFFERENCES BETWEEN SCIENCES

Physics

- **Knowledge**
  Fundamental laws
- **Sensors**
  Numerous
- **Perturbations**
  Controlled reactors

Biology

- **Knowledge**
  Empirical laws
- **Sensors**
  Qualitative
- **Perturbations**
  Few possible perturbations

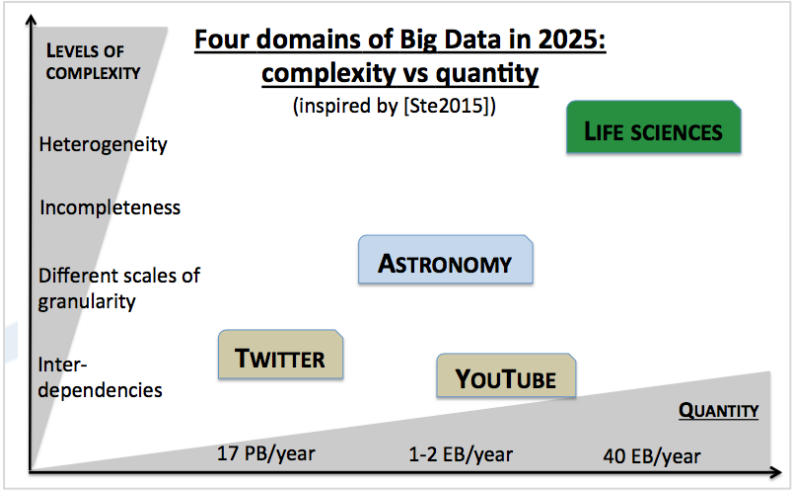
DIFFERENCES BETWEEN SCIENCES

Physics

- **Knowledge**
  Fundamental laws
- **Sensors**
  Numerous
- **Perturbations**
  Controlled reactors

Biology

- **Knowledge**
  Empirical laws
- **Sensors**
  Qualitative
- **Perturbations**
  Few possible perturbations



Molecular biology cannot be studied with usual complex system approaches

STRATEGY: DYNAMICAL SYSTEMS & LOGICAL CONSTRAINTS PROGRAMMING

Describe a system by a family of abstract models
- **Avoid quantitative predictions**
- Reason over a family of models instead of selecting a single one

Logical knowledge representation
- Search space **description**

Discrete dynamical systems
- **Links** between multi-scale observations

Combinatorial optimisation problems
- **Replace laws by combinatorial constraints**

Forget about deduction or induction…
and come back to abduction to mimic life science discovery reasoning

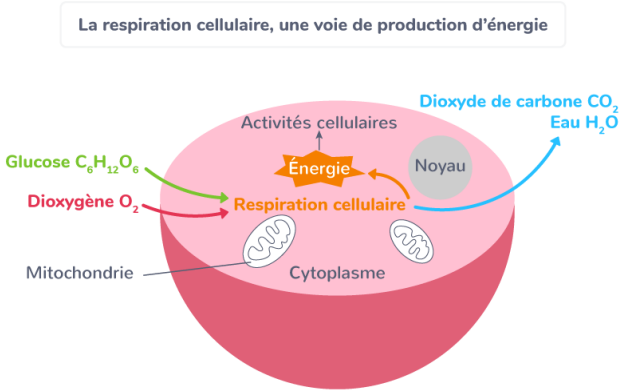
3. Focus on the metabolic scale

1. Short historical reminder: from observations to discoveries in life sciences
2. (Identification of) dynamical systems
3. **Focus on the metabolic scale.**
4. Reasoning over discrete dynamical systems
5. Knowledge representation : introduction to ASP
6. More insights on ASP
7. Back to biology towards more tricky problems
8. Still new issues

FOCUS SUR L'ECHELLE DU METABOLISME

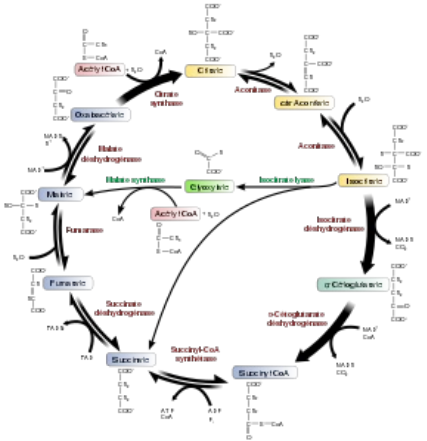
Voies métaboliques: ensemble de **réactions chimiques** permettant de **convertir un composé chimique en un autre à travers des transformations successives**, catalysées par des enzymes.

Fonction métabolique



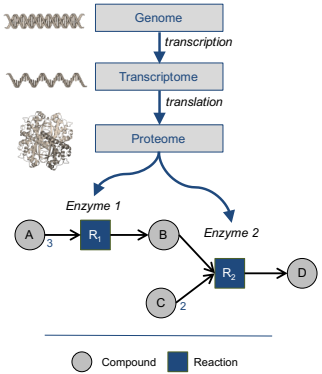
https://www.kartable.fr

Acteurs de cette fonction



https://fr.wikipedia.org/wiki/Cycle_du_glyoxylate

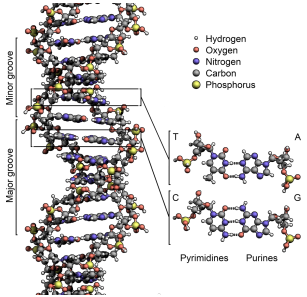
Du gène à la fonction



@C. Frioux

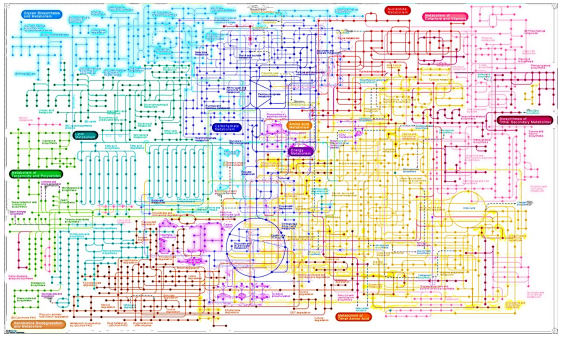
Les fonctions métaboliques sont contrôlées par des gènes

EXPLOITATION DU GENOME A L'ECHELLE DU METABOLISME



https://fr.wikipedia.org

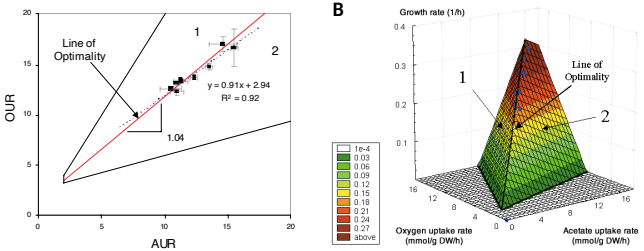
Transformation du **génome annoté** en carte métabolique



Source : KEGG

Analyse des flux: produire un maximum de composés sans provoquer d'embouteillages ?

- → Systèmes dynamiques non identifiables
- → **Principe thermodynamique & entropique du métabolisme**
- → Recherche opérationnelle, raisonnement



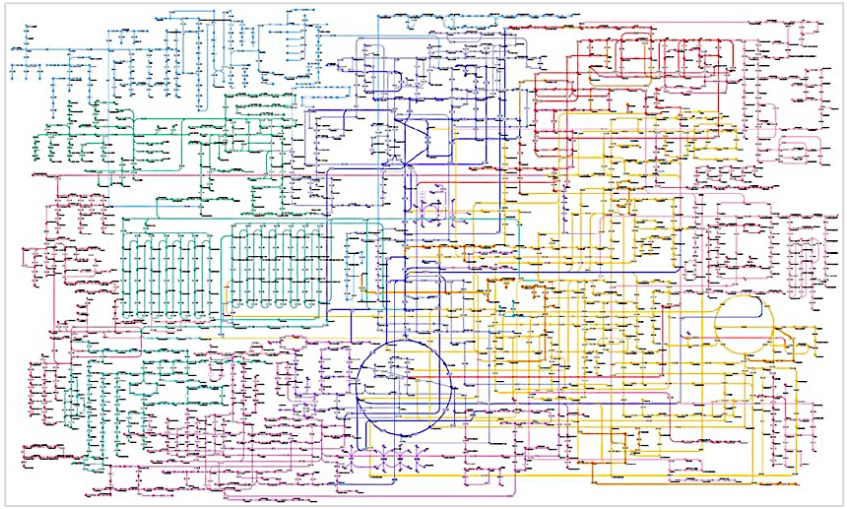
Source : Edwards, Ibarra Palsson, nature biotech. 2001

Génie génétique (CRISPR) pour optimiser la production de molécules

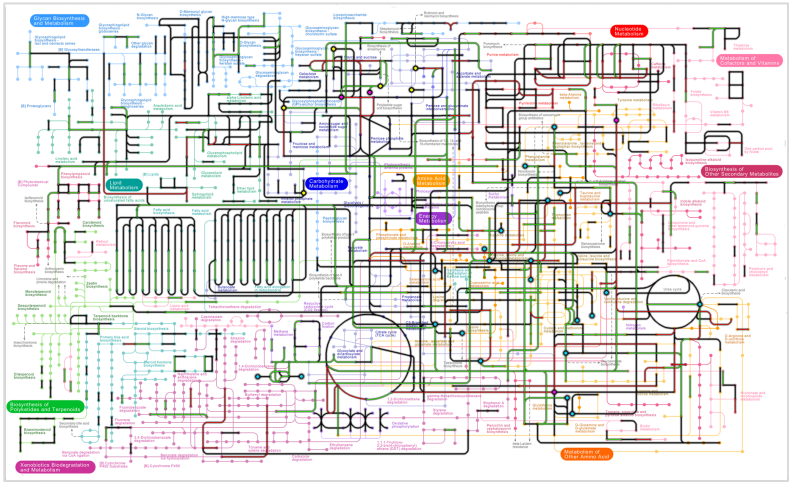
- insuline
- vitamines
- cosmétiques
- Acides aminés
- ethanol

Le schéma est valable pour les bactéries, levures et microalgues (unicellulaires) qui ont des cartes de bonne qualité

DANS LA VRAIE VIE



Expected metabolic map



Real metabolic map

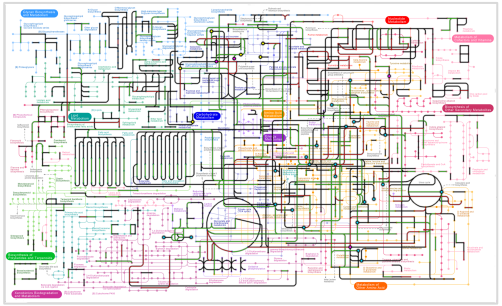
Holes → impossible predictions

EUKARYOTES MULTICELLULAIRES : EXEMPLE DES MACRO-ALGUES

Les cartes sont incomplètes

- on ne connaît que très partiellement les fonctions des protéines
- Il manque des voies de synthèse complètes dans les bases de connaissances

> MGnify
> @MGnifyDB
>
> Using their fast ESMFold's language model this provides a structure for almost every cluster representative in the protein database, of which just under half of the proteins currently have an unknown functional annotation. #protein #darkmatter

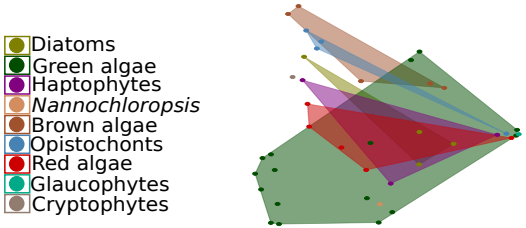


(Tapia et al, 2016)

La carte reconstruite à partir du génome d'E. siliculosus ne pouvait produire in silico aucun des composés mesurés expérimentalement

(PIA IDEALG, Prigent et al, 2015)

Les cartes dépendent fortement des annotations et des technologies de séquençage

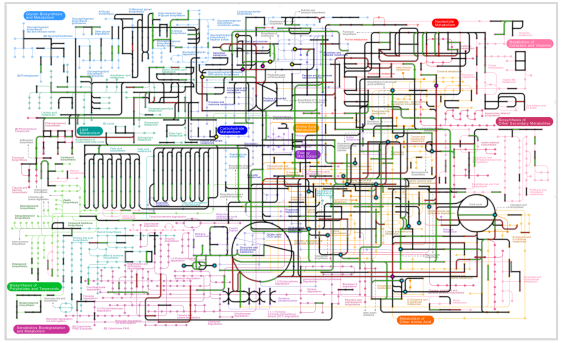


Les réseaux métaboliques reconstruits pour une famille de 36 macro-algues n'ont aucune cohérence phylogénétique

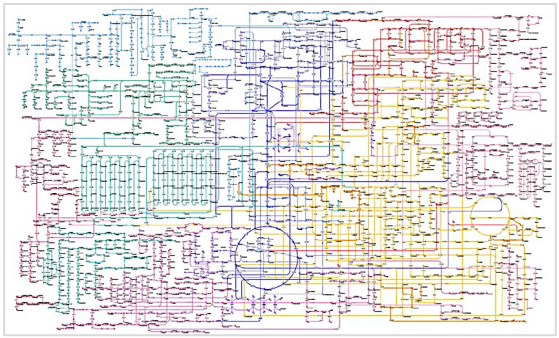
(Belcour et al, genome research, 2023)

Chaque lacune dans une carte est un potentiel de découverte scientifique

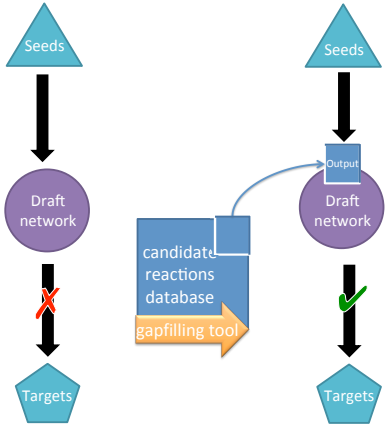
GAP-FILLING A METABOLIC NETWORK : A COMPUTATIONAL PROBLEM



Draft network (holes)



Functional network



Gap-filling problem(s)

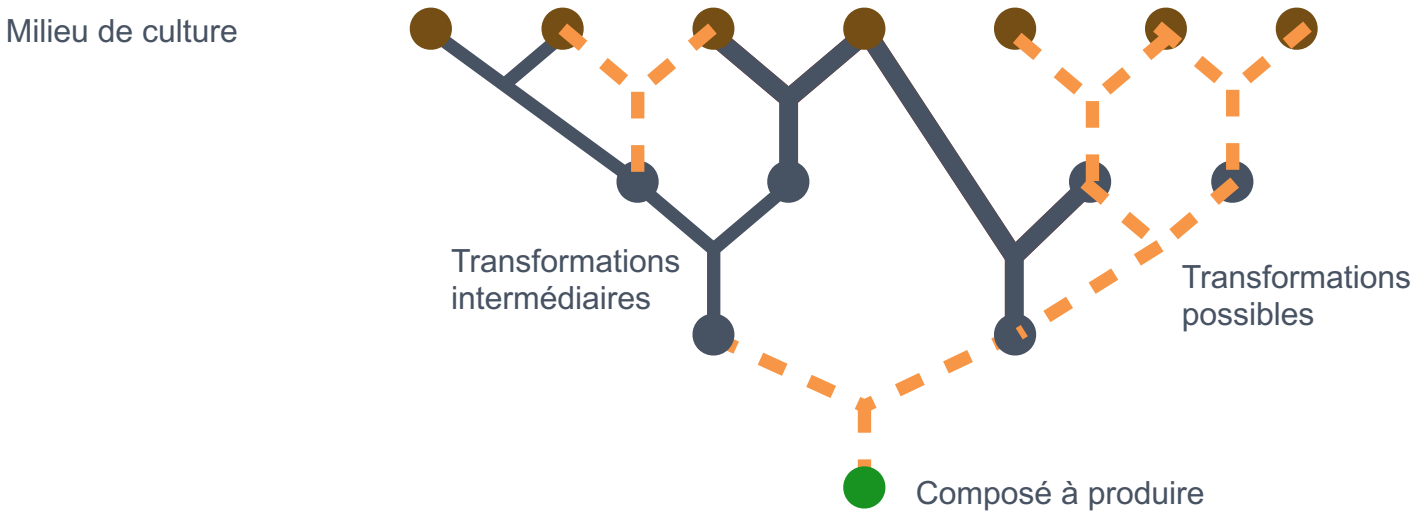
Input:

- network with non-accessible target components
- Knowledge database of possible reactions

Output

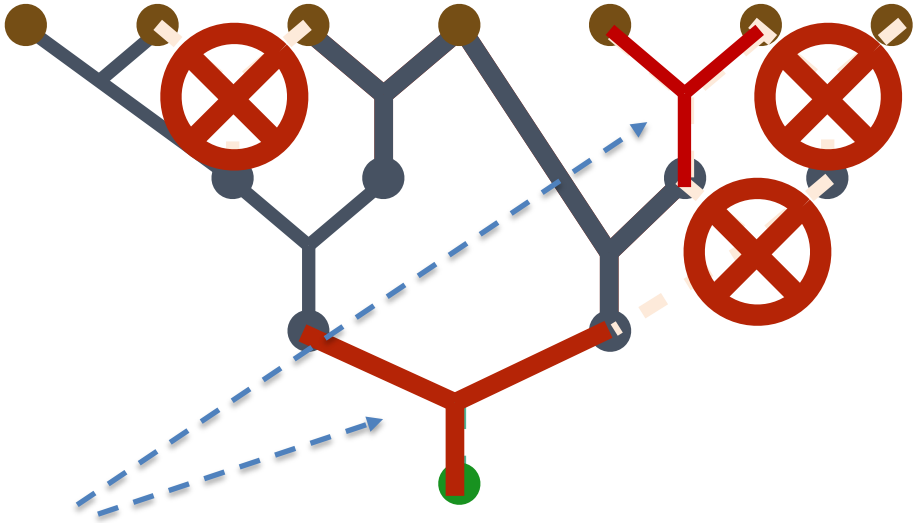
- (minimal) set of reactions restoring target accessibility

GAP-FILLING A METABOLIC NETWORK : NUTSHELL



Which reactions (orange reactions) must be added to produce the green target ?

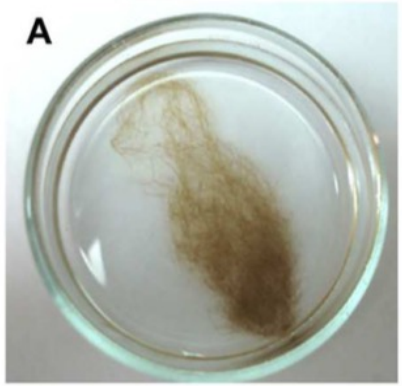
GAP-FILLING A METABOLIC NETWORK : NUTSHELL



Two reactions are sufficient

Warning: less easy if you have to take into account biological features: cycles, co-factors, accumulations…

GAP-FILLING A METABOLIC NETWORK : NUTSHELL



E. Siliculosus [Plant Journal'15]

- Metabolic network: 1785 reactions, 1981 compounds
- Flux (flow) analysis : no production of the 54 experimentally characterized molecules
- Flow-based gap-filling: add 500 reactions ???
- Unrealistic and intractable

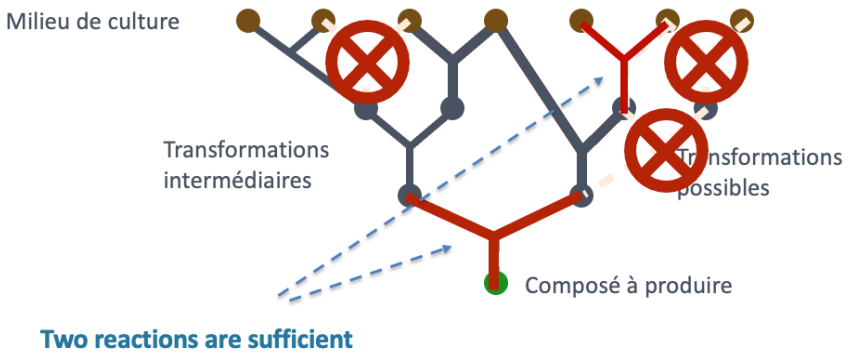
Use genomic data to deduce/predict cellular behaviors → Dead-end ?

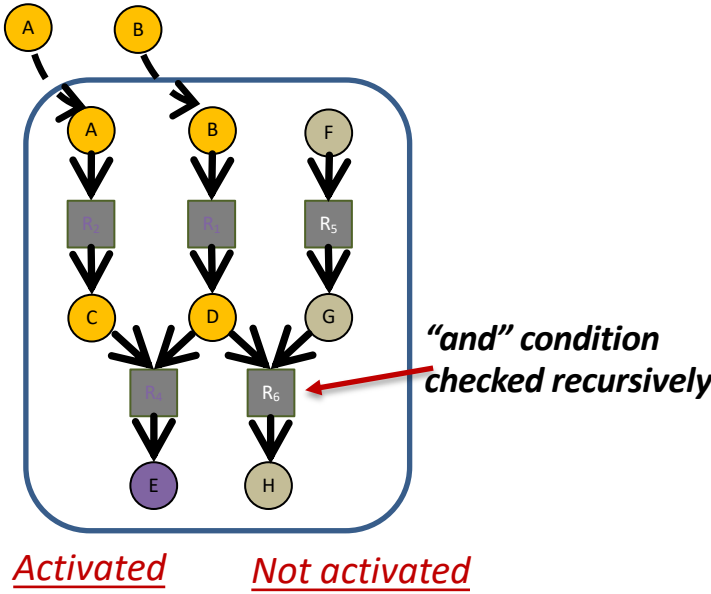
- **Bottleneck 1** : not enough data to infer paramaterized dynamical models
  → Quasi steady-state approximation – MILP problems
- **Bottleneck 2**: imprecise information about the structure of the dynamical models
  → Solve optimisation problems – Too many models (untractable)

4. Reasoning over discrete dynamical systems

1. Short historical reminder: from observations to discoveries in life sciences
2. (Identification of) dynamical systems
3. Focus on the metabolic scale.
4. **Reasoning over discrete dynamical systems**
5. Knowledge representation : introduction to ASP
6. More insights on ASP
7. Back to biology towards more tricky problems
8. Still new issues

WHAT ABOUT BEING LESS PRECISE





Graph-based semantics *[Handorf, Ebenhoeh, Sagot, Schaub/Thiele]*

A reaction can be activated if all its substrates are

- either in the growth medium
- or the product of an activated reaction

Computation of graph-based activated reactions

- **Easily modeled in logical programming**
- Not that easy with linear constraints (Thuillier, Roadef) or with algorithmic procedures (sagot team).

UNDERLYING DYNAMICAL SYSTEMS

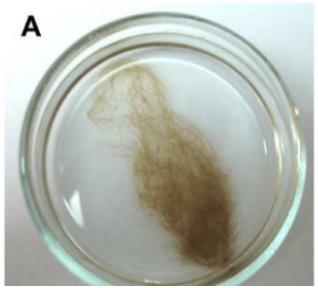
The graph-based approximation corresponds to a minimal fixed-point computation in a Boolean framework (Frioux, S., 2020)

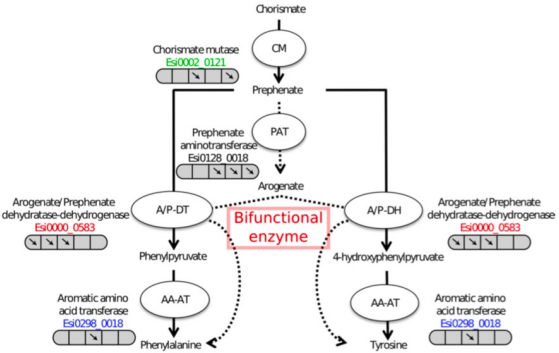
	Continuous framework (ODEs)	Discrete framework (Boolean)
Dynamics	$\frac{d\mathbf{M}(t)}{dt} = \mathbf{S}\,\mathbf{v}(\mathbf{M}(t))$	$\phi(\overline{m}) = \bigvee_{r \in R \mid m \in \text{product}(r)} \bigwedge_{s \in \text{substrate}(r)} \overline{s}.$
Steady-state	$\frac{d\mathbf{M}(t)}{dt} = 0$	$\phi(\overline{m}) = \overline{m} \quad \forall m \in M$ $\phi(\overline{m}) = 1 \quad \forall m \in \text{medium}$
Side-constraints	*Biomass optimisation* *Flux upper/lower values*	***Minimal number of activated nodes***
Resolution framework	*Linear (MILP) solving*	*Logical programming with appropriate negation model (ASP)*

Gap-filling problem → solve a combinatorial optimisation problem

IN REAL LIFE

S. Prigent's PhD
Now @Inra





E. Siliculosus [Plant Journal'15]

- Flow-based gap-filling:. untractable
- Discrete-based gap-filling solved with ASP :
  → 3500 solutions, 50 reactions to check.

Biological discovery (knowledge confrontation)

One of the holes was badly predicted… since one of the enzymes had acquired a specific function.

Towards new computer science issues: *C. Crispus* [G. Markov]

→ SMT-based approaches [TPLP'18]

A single reaction was missing (false gene-assignement due to genome-assembly)

WHAT DID WE DO ?

Describe a system by a family of abstract models
- **Avoid quantitative predictions**
- Reason over a family of models instead of selecting a single one

Logical knowledge representation
- Search space description

Discrete dynamical systems
- Links between multi-scale observations

Combinatorial optimisation problems
- Replace laws by combinatorial constraints

Discrete framework (Boolean)

Dynamics	$\phi(\overline{m}) = \bigvee_{r \in R \mid m \in \text{product}(r)} \bigwedge_{s \in \text{substrate}(r)} \overline{s}.$
Steady-state	$\phi(\overline{m}) = \overline{m} \quad \forall m \in M$ $\phi(\overline{m}) = 1 \quad \forall m \in \text{medium}$
Side-constraints	***Minimal number of activated nodes***
Resolution framework	*Logical programming with appropriate negation model (ASP)*

Efficiently solve optimisation combinatorial problems

```
{reaction(r)}.
scope(M):- seed(M).
scope(M):- product(M,R), reaction(R), scope(M') : reactant(M',R).
:- target(T), not scope(T).
#minimize{ reaction(r) }.
```

IRISA UMR

ANSWER SET PROGRAMMING

$$\underbrace{K\ \{\ atom_1;\ \ldots;\ atom_n\ \}\ L}_{\text{head}}\ \underbrace{:-}_{\text{"smiley"}}\ \underbrace{atom_{n+1}\ ;\ldots;\ atom_r;\ not\ atom_{r+1};\ \ldots;\ not\ atom_s\ .}_{\text{body}}$$

High-level model language

- Propositional logics
- **Model for negation: everything is false a priori**
- **Reasoning modes**: enumeration, union, intersection of solutions
- **Optimisation**

Highly performant solving technics

- SAT-based and deductive-DB technics
- Decidable: no infinite loop

Optimisation rule

#maximize{W,atom(X): condition(X),W}.

Linear constrains atoms

&sum{a1*x1;...;al*xl} <= k

WHY IS IT USEFUL ?

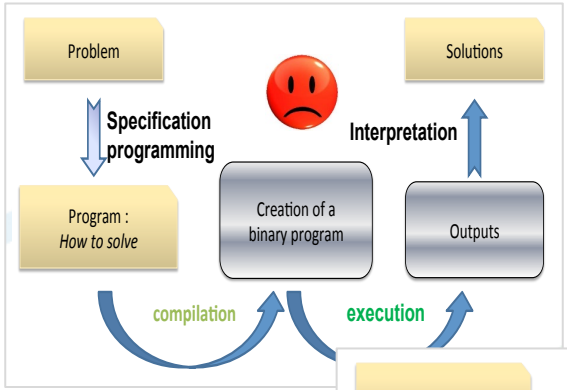
- Relevant semantics of negation
- Optimisation : parcimony hypothesis
- Enumeration mode : complete solution space
- Union reasoning mode : families of candidates
- Projection reasoning mode : what can be surely deduced from knowledge

Integrative and systems biology is a very relevant field to challenge ASP technologies

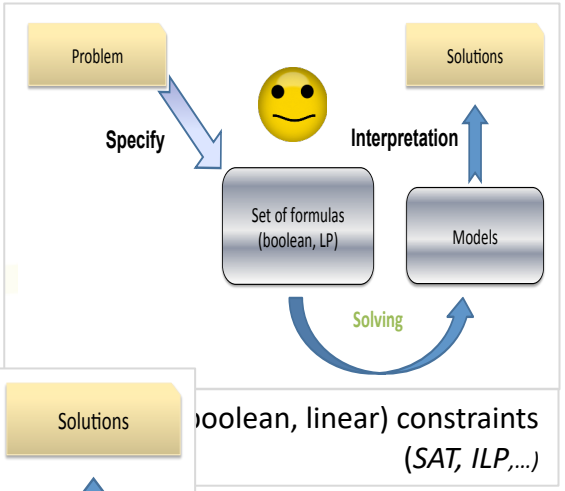
5. Knowledge representation : introduction to answer set programming

1. Short historical reminder: from observations to discoveries in life sciences
2. (Identification of) dynamical systems
3. Focus on the metabolic scale.
4. Reasoning over discrete dynamical systems
5. **Knowledge representation : introduction to ASP**
6. More insights on ASP
7. Back to biology towards more tricky problems
8. Still new issues

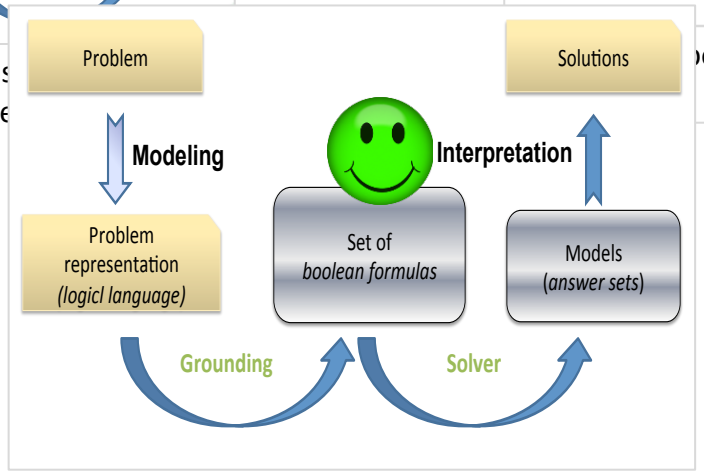
Solving combinatorial problems



Write a program which s
problem should be solve



oolean, linear) constraints
(*SAT, ILP,…*)

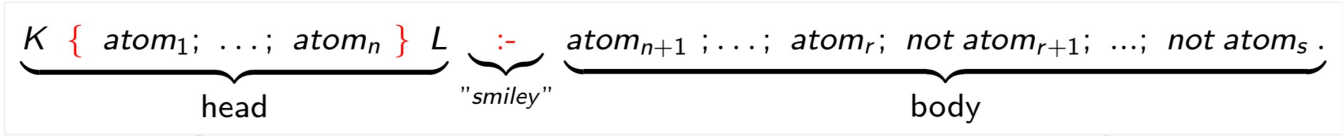


Answer set programming.
Describe what you want to solve

Declarative programing

- Problem = axioms & rules
- No need of algorithm

ASP logical rules : declarative programming



If **all terms** on the **right side** are true,
then **at least K and at most L** terms are true on the **left side**.

If **nothing** on the **left side**,
then **always false.**

```
:- K{atom1, .. atomN}L.
```

If **nothing** on the **right side**,
then **always true.**

```
K{atom1, .. atomN}L.
```

Optimisation rule

```
#maximize{W,atom(X): condition(X),W}.
```

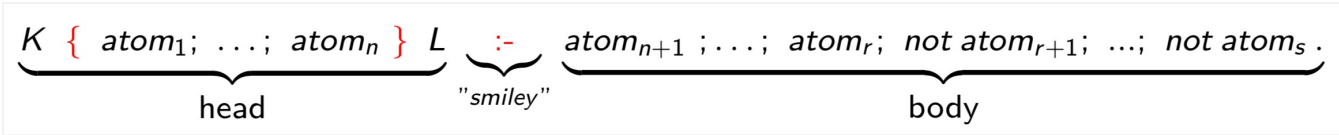
High-level model language

- Propositional logics
- Model for negation

Highly performant solving technics

- SAT-based and deductive-DB technics
- Decidable: no infinite loop

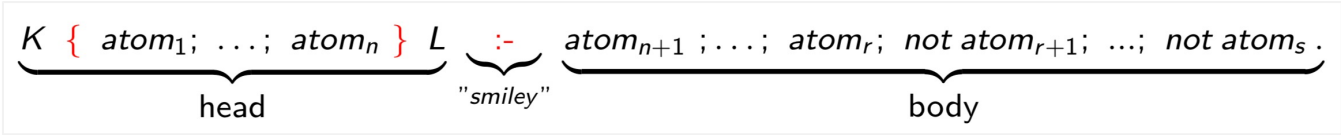
ASP logical rules




```
1{murderer(ms_Scarlet); murderer(colonel_Mustard)}1.
1{weapon_of_crime(revolver); weapon_of_crime(candlestick)}1.
1{place_of_crime(kitchen); place_of_crime(hall);
                                    place_of_crime(dining_room)}1.

crim_record(ms_Scarlet,4). crim_record(colonel_Mustard,7).

weapon_of_crime(candlestick).
:- place_of_crime(kitchen).
place_of_crime(hall) :- murderer(colonel_Mustard), not
                                          weapon_of_crime(revolver).

sol(X,Y,Z) :- murderer(X),weapon_of_crime(Y),place_of_crime(Z).



#show sol/3.
```

How many solutions ?

ASP logical rules




```
1{murderer(ms_Scarlet); murderer(colonel_Mustard)}1.
1{weapon_of_crime(revolver); weapon_of_crime(candlestick)}1.
1{place_of_crime(kitchen); place_of_crime(hall);
                                    place_of_crime(dining_room)}1.

crim_record(ms_Scarlet,4). crim_record(colonel_Mustard,7).
```

Declare what you know

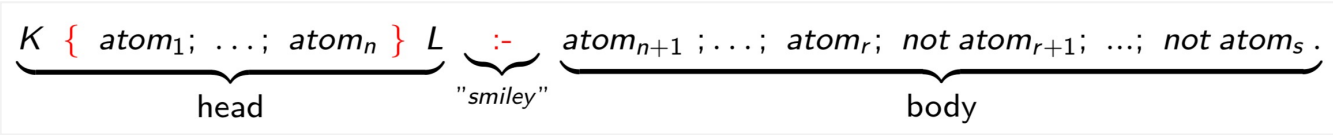
```
weapon_of_crime(candlestick).
:- place_of_crime(kitchen).
place_of_crime(hall) :- murderer(colonel_Mustard), not

     weapon_of_crime(revolver).

sol(X,Y,Z) :- murderer(X),weapon_of_crime(Y),place_of_crime(Z).
```

What you have deduced up to now

```
#show sol/3.
```

What you look for

ASP logical rules




```
1{murderer(ms_Scarlet); murderer(colonel_Mustard)}1.
1{weapon_of_crime(revolver); weapon_of_crime(candlestick)}1.
1{place_of_crime(kitchen); place_of_crime(hall);
                                    place_of_crime(dining_room)}1.

crim_record(ms_Scarlet,4). crim_record(colonel_Mustard,7).

weapon_of_crime(candlestick).
:- place_of_crime(kitchen).
place_of_crime(hall) :- murderer(colonel_Mustard), not

     weapon_of_crime(revolver).

sol(X,Y,Z) :- murderer(X),weapon_of_crime(Y),place_of_crime(Z).

#minimize{W , sol(X,Y,Z) : sol(X,Y,Z) , crim_record(X,W) , murderer(

#show sol/3.
```

Introduce optimization rules

Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).


1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).

receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.

:- labelV (I,S), not receive(I,S).
```

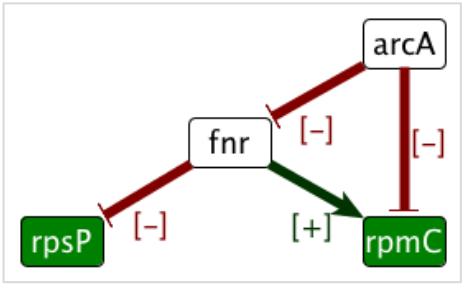
Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).


1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).

receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.

:- labelV (I,S), not receive(I,S).
```

Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).
```

(1) Declare nodes, edges and signs

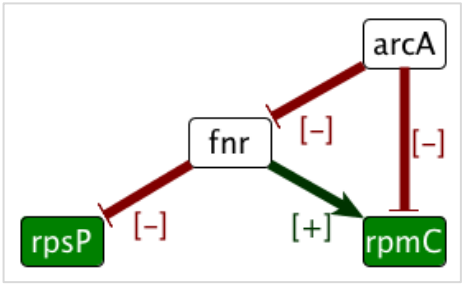



```
1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).

receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.

:- labelV (I,S), not receive(I,S).
```

Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).
```

(1) Declare nodes, edges and signs

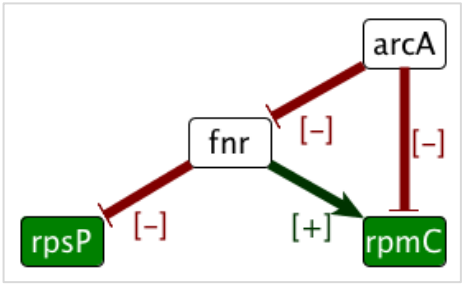



```
1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).

receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.

:- labelV (I,S), not receive(I,S).
```

Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).
```

(1) Declare nodes, edges and signs



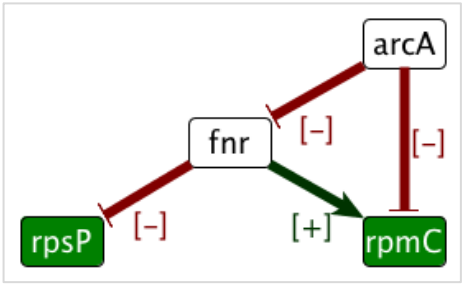

```
1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).

receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.

:- labelV (I,S), not receive(I,S).
```

(2) Associate a predicted sign to each node and edge

Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).
```

(1) Declare nodes, edges and signs




```
1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).
```

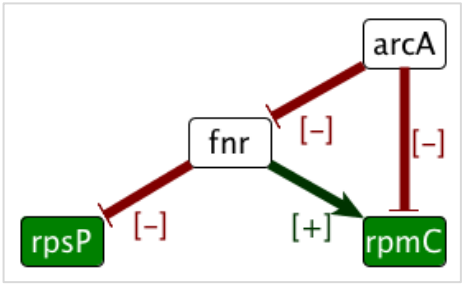
(2) Associate a predicted sign to each node and edge

```
receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.

:- labelV (I,S), not receive(I,S).
```

Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).
```

(1) Declare nodes, edges and signs




```
1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).
```

(2) Associate a predicted sign to each node and edge

```
receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.
```

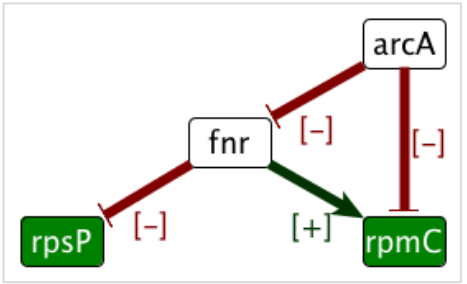
(3) Dynamical propagation of the signal

```
:- labelV (I,S), not receive(I,S).
```

Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).
```

(1) Declare nodes, edges and signs




```
1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).
```

(2) Associate a predicted sign to each node and edge

```
receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.
```

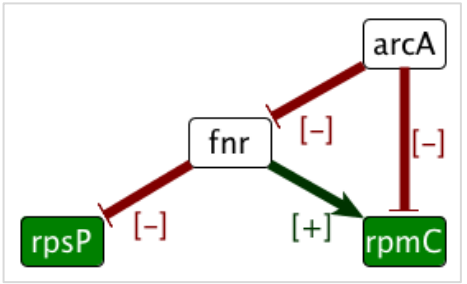
(3) Dynamical propagation of the signal

```
:- labelV (I,S), not receive(I,S).
```

Example : which rule is encoded in the following program ?

```
vertex(rpsP). observedV(rpsP,-).
vertex(rpmC). observedV(rpmC,-).
vertex(fnr).
vertex(arcA).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr). observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).
```

(1) Declare nodes, edges and signs




```
1{labelV(I,+) ; labelV (I,-)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
1{labelE(J,I,+) ; labelE (J,I,-)}1 :- edge(J,I).
labelE(J,I,S) :- observedE(J,I,S).
```

(2) Associate a predicted sign to each node and edge

```
receive(I,+) :- labelE(J,I,S), labelV(J,S).
receive(I,-) :- labelE(J,I,S), labelV(J,T), S!=T.
```

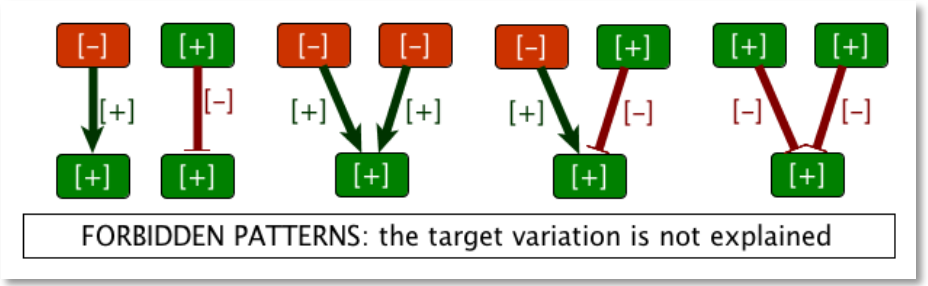
(3) Dynamical propagation of the signal

```
:- labelV (I,S), not receive(I,S).
```

(4) Ensure that propagation is consistent with prediction

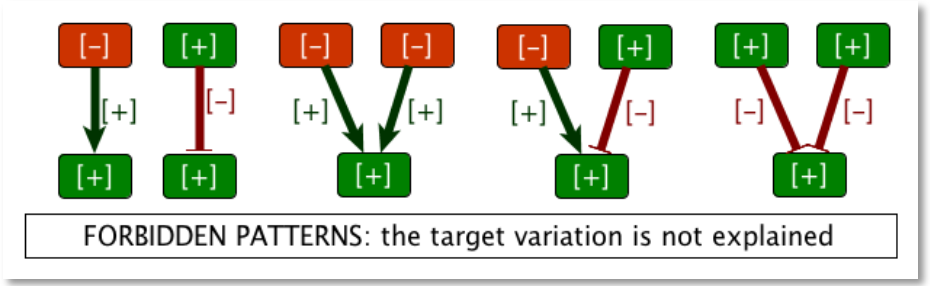
Implicit dynamical rule ?

Interpretation. For each gene which does not self-induces itself, the gene expression variation must be explained by a consistent regulation from at least one predecessor.



Implicit dynamical rule ?

Interpretation. For each gene which does not self-induces itself, the gene expression variation must be explained by a consistent regulation from at least one predecessor.



Theorem. *Assume that*

- *Specy i autoregulates itself negatively:* $\frac{\partial \mathbf{F}_{X(i)}}{\partial X(i)} < -C, \quad C > 0.$
- *There is no direct influence from* $\mathbf{P}$ *on* $X(i)$
- *When i is absent, the system produces it* $\mathbf{F}_{X(i)}(\{\mathbf{X},\ X(i) = 0\}) > 0$
- *For every predecessor $k \to i$, the sign of the action* $\frac{\partial \mathbf{F}_{X(i)}}{\partial X(k)}$ *is constant during the experimentation*

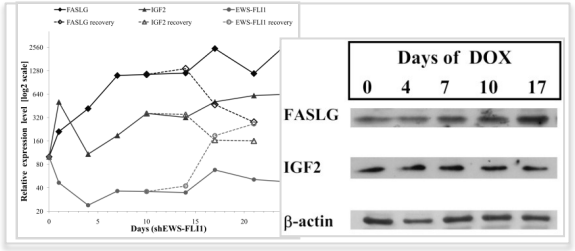
Then the variations of the species between two steady states (for different parameters) satisfy the following relationship:

$$sign(\Delta X(i)) \simeq \sum_{k \neq i, k \to i} sign\left(\frac{\partial \mathbf{F}_{X(i)}}{\partial X(k)}\right) \times sign(\Delta X(k)).$$

Examples of application

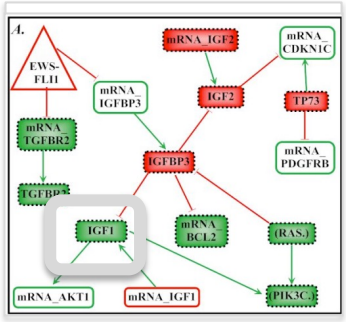
Explain and predict

- Competing effects
- Significant predictions



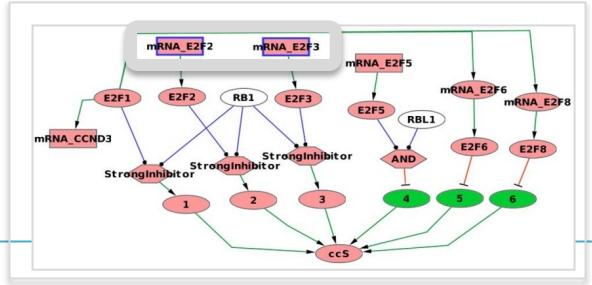
Key processes

- Incomplete IGF1- pathway



Better understanding of regulation

- Prediction of two new potential targets
- Confirmed by si-RNA



Applications

OPEN ACCESS Freely available online

PLOS | COMPUTATIONAL BIOLOGY

Detecting and Removing Inconsistencies between Experimental Data and Signaling Network Topologies Using Integer Linear Programming on Interaction Graphs

Ioannis N. Melas[1], Regina Samaga[2], Leonidas G. Alexopoulos[1], Steffen Klamt[2]*

1 National Technical University of Athens, Athens, Greece, 2 Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany

Thiele *et al. BMC Bioinformatics* (2015) 16:345
DOI 10.1186/s12859-015-0733-7

BMC Bioinformatics

METHODOLOGY ARTICLE Open Access

Extended notions of sign consistency to relate experimental data to signaling and regulatory network topologies

SCIENTIFIC REPORTS

OPEN

Logic programming reveals alteration of key transcription factors in multiple myeloma

Received: 29 November 2016
Accepted: 25 July 2017
Published online: 23 August 2017

Bertrand Miannay[1,2], Stéphane Minvielle[2,3], Olivier Roux[1], Pierre Drouin[1], Hervé Avet-Loiseau[4], Catherine Guérin-Charbonnel[2,5], Wilfried Gouraud[2,5], Michel Attal[6], Thierry Facon[7], Nikhil C Munshi[8,9], Philippe Moreau[2,3], Loïc Campion[2,5], Florence Magrangeas[2,3] & Carito Guziolowski[1]

Klamt's group & Guziolowski's group

→ Discriminate patient responses from a large-scale interaction graph

6. More insights on ASP

1. Short historical reminder: from observations to discoveries in life sciences
2. (Identification of) dynamical systems
3. Focus on the metabolic scale.
4. Reasoning over discrete dynamical systems
5. Knowledge representation : introduction to ASP
6. **More insights on ASP**
7. Back to biology towards more tricky problems
8. Still new issues

Description rapide

Règles disjonctives

$$\underbrace{K\ \{\ atom_1;\ \ldots;\ atom_n\ \}\ L}_{\text{head}}\ \underbrace{:-}_{\text{"smiley"}}\ \underbrace{atom_{n+1}\ ;\ldots;\ atom_r;\ not\ atom_{r+1};\ \ldots;\ not\ atom_s\ .}_{\text{body}}$$

- $fact_1, \ldots, fact_n$ sont des *atomes*
- Règle de déduction

  Si tous les faits apparaissant dans le corps sont vrais,
  alors la tête contient entre K et L faits qui sont vérifiés.

- Contrainte d'intégrité. S'il n'y a pas de tête: le corps est *toujours faux*

  $:- fact_0.$

- Faits vrais. S'il n'y a pas de corps, la tête est *toujours vraie*

  $fact_0.$

Les *Ensembles-réponses* sont tous les modèles valides

- Ce sont des ensembles d'atomes qui vérifient toutes les règles.
- Tout atome d'un modèle valide apparait dans la tête d'au moins une règle.

Who killed Mr Boddy ?

Program

```
% Specify that there is only one murder
1{murderer(ms_Scarlet);murderer(colonel_Mustard)}1.
1{weapon_of_crime(revolver);weapon_of_crime(candlestick)}1.
1{place_of_crime(kitchen);place_of_crime(hall);place_of_crime(dining-room)}1.

% Declare what you can deduce from your cards
:- place_of_crime(kitchen).
place_of_crime(hall) :- murderer(colonel_Mustard);
                                                not weapon_of_crime(revolver).
weapon_of_crime(candlestick).

% Enumerate the solutions
#show murderer/1.
#show place_of_crime/1.
#show weapon_of_crime/1.
```

Answer Sets? **Exercice**

```
Solving...
Answer:  1
weapon_of_crime(candlestick) murderer(colonel_Mustard) place_of_crime(hall)
Answer:  2
weapon_of_crime(candlestick) murderer(ms_Scarlet) place_of_crime(dining_room)
Answer:  3
weapon_of_crime(candlestick) murderer(ms_Scarlet) place_of_crime(hall)
SATISFIABLE
```

ASP: Principe généraux à retenir

Approche déclarative pour la résolution de problèmes combinatoires de complexité NP

"*What* instead of *How* ?"

Approche déclarative

Modélisation du problème à résoudre sous formes d'axiomes et de contraintes exprimées dans un langage logique.

Pourquoi ce nom ?

Les modèles logiques solutions de l'ensemble de formules, les ensembles réponses, sont les résultats du programme.

Principe

Des solveurs associés effectuent la recherche d'une, de plusieurs, ou de l'ensemble des solutions.

Un peu plus précisément

Langage de modélisation à haut-niveau

expressivité: ASP $\simeq$ Prolog

- Logique propositionnelle

  $\rightarrow$ *On ne peut travailler que sur un ensemble fini d'atomes.*

- Négation : formalisme élégant (monde clos).

  $\rightarrow$ *Un prédicat est faux tant qu'aucune indication ne permet de dire qu'il est vrai.*

Capacité de résolution de problèmes très élevée

ASP $\simeq$ SAT, ILP

- Décidable techniques de résolution SAT

  $\rightarrow$ *Pas de réécriture de programmes*

  $\rightarrow$ L'ordre des clauses n'a pas d'impact

  $\rightarrow$ Pas de boucle infinie dans la résolution

- Optimisation, raisonnement flexible (préférences), solveurs hybrides.

Une spécificité très utile en biologie moléculaire et bioinformatique

Différents modes de raisonnements

Déduire différents types d'information biologique

- Enumération des solutions à un problème (`clingo cluedo.lp -n 0`)

  → Variabilité et taille de l'espace des modèles possibles

- Intersection (cautious reasoning) des atomes présents dans toutes les solutions (`clingo cluedo.lp --enum-mode=cautious`)

  → Déductions robustes des données et connaissances

- Union (brave reasoning) des atomes présents dans toutes les solutions (`clingo cluedo.lp --enum-mode=brave`)

  → Espace des déductions possibles

Programmation par ensembles-réponses: d'un point de vue concret

Syntaxe: Termes

CONSTANTE : entier, mot de {a-z, A-Z, 0-9, _ }*

→ débute par une MINUSCULE. `porquerolles` `jean-Paul_Comet`

VARIABLE : mot de {a-z, A-Z, 0-9, _ }*

→ débute par une MAJUSCULE. `Lieu` `Organisateur`

FONCTION

- de la forme *constante(terme, ..., terme)*
- inclut les opérateurs : `a+b`
- inclut la notation de liste. `L=[a,b,c]=[a|U], U =[b,c]`

Question: `Cout(Repas(Participants,Jour),nuits(participants,jour))*5` est-il valide ?

Réponse: `cout(repas(Participants,Jour),nuits(Participants,Jour))*5`

LITTERAL

- Négation `not q`
- Calculs `X<Y` `U!=V` `X+Y<5`

Syntaxe: Termes

Condition

TailleMin { *atome*1 : *domaine_validite*1 ; *atome*2 : *domaine_validite*2 } *TailleMax*

- On construit l'ensemble des atomes1 et atomes 2 qui vérifient toutes les conditions de validité.
- On isole tous les sous-ensembles de taille qui contiennent entre *TailleMin* et *TailleMax* atomes valides.

```
25{chambreSimple(X):particip(X) ; chambreDouble(Y,Z):particip(Y),particip(Z)}30
```

→ Sous-ensembles de 25 à 30 chambres simples ou doubles qui hébergent des participants.

```
5 { chambreSimple(X) : participant(X) ; dureeSejour(X)=DureeMax } 8
```

→ Sous-ensembles de 5 à 8 chambres simples qui hébergent des participants dont la durée du séjour est égale à la constante DureeMax.

Exemple

Variables

```
% Declare what you are playing with

place(kitchen; hall; diningroom).
weapon(revolver; candlestick).
character(colonel_Mustard;ms_Scarlet).


% Specify that there is only one murder
1{murderer(X):character(X)}1.
```

→ Le grounder (`gringo cluedo.lp -t`) va instancier cette clause en `1{murderer(ms_Scarlet) ; murderer(colonel_Mustard)}1.`

```
1{weapon_of_crime(X):weapon(X)}1.
```

→ grounding: `1{weapon_of_crime(revolver) ; weapon_of_crime(candlestick)}1.`

```
1{place_of_crime(X):place(X)}1.
```

→ grounding:

```
1{place_of_crime(kitchen);place_of_crime(hall);place_of_crime(dining-room)}1.
```

Syntaxe: Termes

```
1{bouchanourrirs(X):perso(X)}2
```

$\rightarrow$ sous-ensemble de taille 1 à 2 de l'ensemble des atomes *bouchanourrirs*(X) qui sont vrais, où X est une variable dans le domaine défini par le prédicat *perso*.

Sans indication supplémentaire sur les prédicats `bouchanourrirs` et `perso`, ce terme est strictement équivalent à

```
1{r(X):q(X)}2
```

```
2{r(U):q(U)=U ; s(V):t(V)=2}
```

$\rightarrow$ Sous-ensemble de taille au moins 2 de l'ensemble composé

- des $r(U)$ vrais pour lesquels U est tel que $q(U) = U$,
- ET de l'ensemble des $s(V)$ vrais pour lesquels V est tel que $t(V) = 2$.

Syntaxe: Clauses

Faits `p(2,a).`

→ p(2,a) est vrai.

Règle `q(X) :- p(X,Y) ; q(Y) .`

→ Si p(X,Y) est vrai et q(Y) est vrai, alors q(X) est vrai aussi.

Contrainte `:- q(2) .`

→ q(2) est faux.

`:- p(X) ; q(X) .`

→ Le prédicat "p(X) et q(X)" est faux.

Optimisation

```
#minimize{Score , ensembleDeRecherche(X,...,Z) : ensembleDeRecherche(X,...,Z) ,
contrainte1(X,...,Z) , contrainte2(X,...,Z) ,
contrainteVariablesScore(X,..,Z,Score) ,
```

Sorties

```
#show p(X). #show p/1.  #show s(X):r(X).
```

Le `/1` désigne l'arité de `p`. `p/1` est équivalent à `p(X)`

Exemple optimisation

```
place(kitchen; hall; diningroom).
weapon(revolver; candlestick).
character(colonel_Mustard;ms_Scarlet).
1{murderer(X):character(X)}1.
1{weapon_of_crime(X):weapon(X)}1.
1{place_of_crime(X):place(X)}1.
:- place_of_crime(kitchen).
place_of_crime(hall) :- murderer(colonel_Mustard) ;
                                not weapon_of_crime(revolver).
weapon_of_crime(candlestick).

sol(X,Y,Z) :- murderer(X) ; weapon_of_crime(Y) ; place_of_crime(Z).

criminal_record(colonel_Mustard,20).  criminal_record(ms_Scarlet,18).

number_previous_crime(hall,8).  number_previous_crime(dining_room,6).
number_previous_crime(kitchen,2).

 #show sol/3.  #show murderer/1.  #show place_of_crime/1.  #show weapon_of_crime/1.
```

```
Answer:  1
weapon_of_crime(candlestick) murderer(ms_Scarlet) place_of_crime(dining_room) sol(ms_Scarlet,candlestick,dining_room)
Answer:  2
weapon_of_crime(candlestick) place_of_crime(hall) murderer(ms_Scarlet) sol(ms_Scarlet,candlestick,hall)
Answer:  3
weapon_of_crime(candlestick) place_of_crime(hall) murderer(colonel_Mustard) sol(colonel_Mustard,candlestick,hall)
```

Question: écrire une règle qui permet de caractériser les solutions de poids minimal?

Synthaxe

```
#minimize{W , sol(X,Y,Z) : sol(X,Y,Z) , criminal_record(X,W) ,
                           character(X) , place(Z) , weapon(Y)}.
```

```
(clingo cluedo.lp --opt-mode=optN)

Answer:  1
weapon_of_crime(candlestick) place_of_crime(hall)
murderer(ms_Scarlet)
sol(ms_Scarlet,candlestick,hall)
Optimization:  3
Answer:  2
weapon_of_crime(candlestick) murderer(ms_Scarlet)
sol(ms_Scarlet,candlestick,dining_room)
place_of_crime(dining_room)
Optimization:  3
OPTIMUM FOUND
```

Exercice 1

Données : nombre de minutes que doit mettre un taxi pour joindre un client

```
cout(taxi1,client1,10).  cout(taxi2,client1,8).
cout(taxi3,client1,12).  cout(taxi1,client2,11).
cout(taxi2,client2,15).  cout(taxi3,client2,13).
cout(taxi1,client3 ,7).  cout(taxi2,client3 ,7).
cout(taxi3,client3,10).
```

Dérivation d'un graphe : comment identifier les taxis et les clients ?

```
taxi(T):- cout(T,_,_).  client(T):- cout(_,T,_).
```

Caractéristiques d'une solution

Exercice: introduire des clauses pour définir le prédicat `sol(Taxi, Client)` qui affecte un taxi par client et un client par taxi

Exercice 1

Données : nombre de minutes que doit mettre un taxi pour joindre un client

```
cout(taxi1,client1,10).  cout(taxi2,client1,8).
cout(taxi3,client1,12).  cout(taxi1,client2,11).
cout(taxi2,client2,15).  cout(taxi3,client2,13).
cout(taxi1,client3 ,7).  cout(taxi2,client3 ,7).
cout(taxi3,client3,10).
```

Dérivation d'un graphe : comment identifier les taxis et les clients ?

```
taxi(T):- cout(T,_,_).  client(T):- cout(_,T,_).
```

Caractéristiques d'une solution

Exercice: introduire des clauses pour définir le prédicat `sol(Taxi, Client)` qui affecte un taxi par client et un client par taxi

```
1{sol(X,Y):taxi(X)}1:- client(Y).
1{sol(X,Y):client(Y)}1:- taxi(X).
```

Optimisation

Exercice: minimiser le coût d'affectation ?

```
#minimize { Temps, sol(Taxi, Client):  sol(Taxi, Client), cout(Taxi, Client, Temps) } .
```

.... généralisable à de nombreux problèmes de graphes.

Modélisation de la négation

Ensemble-réponse = solutions données par le programme

- Un ensemble réponse contient des atomes se trouvant dans au moins une tête de règle du programme.
- Tout élément d'un ensemble réponse est supporté par une règle.
- Tout ce qui n'est pas prouvé par un fait est considéré comme faux !
- Logique non monotone: rajouter des faits à une théorie peut faire réduire l'ensemble de conclusions.

Exemple 1

Programme

```
a :- b;p;q.
b :- q.
```

Quel est l'ensemble-réponse du programme ?

L'ensemble-réponse du programme est l'ensemble vide $\emptyset$

$\rightarrow$ Le programme ne contient que des implications, et rien n'est "vrai".

Tout ce qui n'est pas prouvé par un fait est considéré comme faux!

Exemple 2

Programme

```
a :- b;p;q.
b :- q.
q.
```

Quel est l'ensemble-réponse?

L'ensemble réponse du programme est {**b,q**}

→ On n'a pas gardé p puisqu'on n'a aucune preuve de p.

Les atomes d'un ensemble-réponse d'un programme positif apparaissent dans la tête d'au moins une règle

Exemple 3

```
p:- p.
q:- not p.
```

Quel est l'ensemble-réponse?

Ensembles-réponses ? {q}

(Par défaut, p est considéré comme faux !)

Exemple 4

```
p:- not q.  q:- not p.
```

```
p:- not q.
q:- not p.
```

Quel est l'ensemble-réponse?

Ensembles-réponses ? {p}, {q}

(Si p est faux, on obtient une preuve de q, et inversement, donc deux solutions)

Exemple 5

```
p:- not p.
```

Quel est l'ensemble-réponse?

Ensembles-réponses ? Aucun: il n'y a pas d'ensemble stable !

A retenir
Un ensemble réponse contient des atomes se trouvant dans une tête de règle du programme.
Tout élément d'un ensemble réponse est supporté par une règle.

Logique non monotone Rajouter des faits à une théorie peut faire réduire l'ensemble de conclusions.

Exemple 6

```
a :- b ; not q.
b:- p ; q.
p.
```

Quel est l'ensemble-réponse?

Exemple 6

```
a :- b ; not q.
b:- p ; q.
p.
```

Quel est l'ensemble-réponse?

Ensembles-réponses ? C'est p. (puisqu'on n'a pas de preuve de b pour déduire a...)

Calcul effectif

- **On réduit le programme par rapport à** { q } (=on enlève les règles dans lesquelles { q }) apparait avec une négation

  ```
  b:- p; q.
  p.
  ```

- **Exercice: calculer le réduit du programme par rapport à** { a } , { b } , { p }

  ```
  a :- b.
  b:- p; q.
  p.
  ```

Exemple 7 : contrainte d'intégrité vs règle de déduction

RAPPEL DE MATH "$A \implies B$" est équivalent à vérifier que la formule B ou $notA$ est vraie
Ce qui est équivalent à verifier que la formule $notB$ et A est fausse.

Et en ASP ? Comparer les deux programmes

```
b :- a.
a.
```

```
:- not b; a.
a.
```

Exemple 7 : contrainte d'intégrité vs règle de déduction

RAPPEL DE MATH "$A \implies B$" est équivalent à vérifier que la formule B ou $notA$ est vraie
Ce qui est équivalent à verifier que la formule $notB$ et A est fausse.

Et en ASP ? Comparer les deux programmes

```
b :- a.
a.
```

Solution : `a. b.`

Règle de déduction. Puisque a est vrai, alors b est vrai.

```
:- not b; a.
a.
```

Solution : `insatisfiable.`

Contrainte d'intégrité : on ne peut pas déduire b puisque qu'on n'a aucune preuve de b (qui n'apparait dans aucune tête de règle). Mais si vrai est faux alors "not b; a" est vérifié et cela n'est pas autorisé.

Conclusion : modélisation de la négation

Ensemble-réponse = solutions données par le programme

- Un ensemble réponse contient des atomes se trouvant dans au moins une tête de règle du programme.
- Tout élément d'un ensemble réponse est supporté par une règle.
- Tout ce qui n'est pas prouvé par un fait est considéré comme faux !
- Logique non monotone: rajouter des faits à une théorie peut faire réduire l'ensemble de conclusions.

Exercice

Question : que veut dire ?

```
:- {proprieteLambda(U+1):conditionQuelconque(U)} M ; domaineValidite(M).
```

- **Il est faux d'avoir** :
  si M qui vérifie le prédicat `domaineValidite`,
  il existe un ensemble contenant au plus M atomes vrais de la forme `proprieteLambda(U+1)`,
  où U vérifie le prédicat `conditionQuelconque`.

- **Autre formulation**:
  Pour tout M qui vérifie `domaineValidite`,
  le prédicat `proprieteLambda(U+1)` est vrai au moins $M + 1$ fois
  lorsqu'on parcours les U qui vérifient `conditionQuelconque`.

- **Encore une autre formulation**:
  Pour tout M qui vérifie `domaineValidite`,
  il y a au moins $M + 1$ valeurs de U qui vérifie `conditionQuelconque` et qui sont telles que `proprieteLambda(U+1)` est vraie.

–¿ Introduction de quantificateurs...

Nouvelle question : que veut dire ?

```
:- {r(U+1):q(U)}M ; p(M).
```

Généralisation

Question : proposer des reformulations de la règle suivante

```
:- not {proprieteLambda(U+1):conditionQuelconque(U)}M ; domaineValidite(M).
```

- **Il est faux d'avoir** :
  si M vérifie le prédicat `domaineValidite`,
  il n'existe pas d'ensemble contenant au plus M atomes vrais de la forme `proprieteLambda(U+1)`,
  où U vérifie le prédicat `conditionQuelconque`,

Généralisation

Question : proposer des reformulations de la règle suivante

```
:- not {proprieteLambda(U+1):conditionQuelconque(U)}M ; domaineValidite(M).
```

- **Il est faux d'avoir** :
  si M vérifie le prédicat `domaineValidite`,
  il n'existe pas d'ensemble contenant au plus M atomes vrais de la forme `proprieteLambda(U+1)`,
  où U vérifie le prédicat `conditionQuelconque`,

- **Autre formulation**:
  si M vérifie `domaineValidite`,
  le prédicat `proprieteLambda(U+1)` est vrai au plus M fois
  lorsqu'on parcours les U qui vérifient `conditionQuelconque`.

- **Encore une autre formulation**:
  Pour tout M qui vérifie `domaineValidite`,
  il y a au plus M valeurs de U qui vérifie `conditionQuelconque` et qui sont telles que `proprieteLambda(U+1)` est vraie.

Introduction de nouveaux quantificateurs et du concept de "au moins"...

Exercice 2

Proposer un programme ASP pour modéliser le système suivant

Le métabolisme du lactose chez Escherichia coli nécessite la présence de 3 enzymes: la β-galactosidase LacZ, la périmease LacY et la transacetylase LacA. La perméase LacY transporte le lactose de l'extérieur de la cellule à l'intérieur du milieu cellulaire. LacZ hydrolyse le lactose interne en glucose et en allolactose. Les protéines LacY et LacZ sont sous contrôle direct d'un inhibiteur appelé LacI. Inversement, un complexe formé d'AMP cyclique et de la protéine CRP (cAMP − CRP) agit sur l'activité des ARN-polymérases et permet d'activer fortement la transcription de LacY et LacZ. Le glucose inhibe la production d'AMP cyclique. Enfin, l'allolactose a une action inhibitrice sur LacI via la formation d'un complexe.

Exercice 2

1. Ecrire un programme pour l'opéron-lactose qui prédit les enzymes, métabolites, autres molécules présentes dans le système en fonction de la présence ou non de LacI, LacA, LacZ et du lactose externe.
2. Quelle est la différence entre les deux codages suivants pour décrire le lien entre LacI et LacY
   - `:- enzyme(lacI), enzyme(lacY).` (contrainte d'intégrité)
   - `enzyme(lacY) :- not enzyme(lacI).` (règle de déduction)

(Début de solution de l') Exercice

```
%enzyme(lacI).
enzyme(lacZ).
%enzyme(lacY).
%enzyme(lacA).
metabolite(lactoseExterne).
metabolite(lactoseInterne) :- enzyme(lacY) ; metabolite(lactoseExterne).
metabolite(allolactose) :- enzyme(lacZ) ; metabolite(lactoseInterne).
metabolite(glucose) :- enzyme(lacZ) ; metabolite(lactoseInterne).
```

```
%:- enzyme(lacI), enzyme(lacY).
```

(Intégrité: si lacI et lacY sont présents, le système est incompatible)

```
enzyme(lacY) :- not enzyme(lacI).
```

(règle: si lacI est absent, alors lacY est présent)

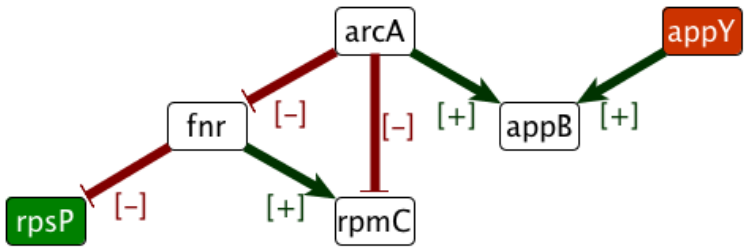
```
:- enzyme(lacI), enzyme(lacZ).
enzyme(lacZ):- coumpound(ampcyc) ; proteine(crp).
enzyme(lacY):- coumpound(ampcyc) ; proteine(crp).
:-coumpound(ampcyc) ; metabolite(glucose).
:-metabolite(allolactose) ; enzyme(lacI).
#show enzyme/1.
#show metabolite/1.
#show coumpound/1.
```

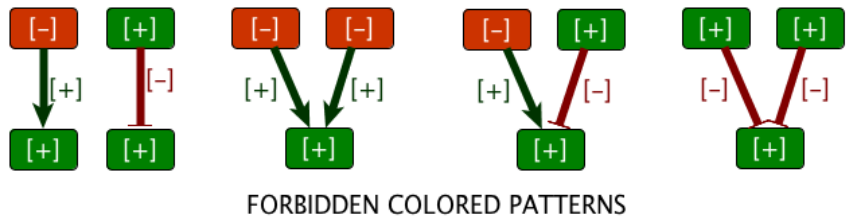
Le codage des règles est très sensible à ce que nous voulons exprimer et peut changer tous les résultats !

Application réelle

Contrainte sur la coloration de graphes

Expliquer l'expression de chaque gène cible par la régulation consistante d'au moins une source.





Exercice

Ecrire un programme qui vérifie que le modèle est inconsistant ou, s'il est consistant, qui donne les prédictions existantes

Suite

Déclarer les signes, sommets, observations des sommets, aretes avec leurs observations.

Les observations sur les arêtes permettent de déclarer des arêtes.

Le terme `predictedV(I,S)` spécifie que chaque sommet a exactement un signe prédit par la méthode.

Les prédictions sur les sommets sont compatibles avec les observations si elles existent.

Un sommet reçoit une influence positive si le signe d'un de ses régulateurs est égal au signe de l'interaction correspondante

Un sommet reçoit une influence négative d'un de ses régulateurs est opposé au signe de l'interaction correspondante

Pour tout sommet *I* qui a au moins un régulateur dans le graphe, si *I* est prédit comment étant de signe S, alors *I* doit recevoir une influence de signe *S* d'un de ses régulateurs

(1) On caractérise les sommets sans prédécesseurs

(2) Pour modéliser le "quelque soit", on exprime le fait que la contraposée est fausse

`Afficher les prédictions sur les sommets`

Suite

Déclarer les signes, sommets, observations des sommets, aretes avec leurs observations.

```
signe(down;up).
vertex(rpsP;fnr;arcA;rpmC;appY;appB).
observedV(rpsP,up).
observedV(appY,down).
observedE(fnr,rpsP,down).  observedE(fnr,rpmC,up).
observedE(arcA,fnr,down).  observedE(arcA,rpmC,down).
observedE(arcA,appB,up).  observedE(appY,appB,up).
```

Les observations sur les arêtes permettent de déclarer des arêtes.

```
edge(I,J) :- observedE(I,J,S); signe(S).
```

Le terme `predictedV(I,S)` spécifie que chaque sommet a exactement un signe prédit par la méthode.

```
1predictedV(I,S):signe(S)1 :- vertex(I).
```

Les prédictions sur les sommets sont compatibles avec les observations si elles existent.

```
predictedV(I,S) :- observedV(I,S).
```

Un sommet reçoit une influence positive si le signe d'un de ses régulateurs est égal au signe de l'interaction correspondante

```
influenced(I,up) :- observedE(J,I,S); predictedV(J,S); signe(S).
```

Un sommet reçoit une influence négative d'un de ses régulateurs est opposé au signe de l'interaction correspondante

```
influenced(I,down) :- observedE(J,I,T); predictedV(J,S) ; T!=S.
```

Pour tout sommet *I* qui a au moins un régulateur dans le graphe, si *I* est prédit comment étant de signe S, alors *I* doit recevoir une influence de signe *S* d'un de ses régulateurs

(1) On caractérise les sommets sans prédécesseurs

```
sansPredecesseur(I):- vertex(I); 0edge(J,I):vertex(J)0.
```

(2) Pour modéliser le "quelque soit", on exprime le fait que la contraposée est fausse : il est faux d'avoir au moins un couple (I, S) tel que I a un régulateur, I est prédit comme ayant le signe S et I ne reçoit pas une influence de signe S.

```
:- vertex(I); signe(S); predictedV(I,S); not influenced(I,S); not sansPredecesseur(I).
```

Afficher les prédictions sur les sommets

```
#show predictedV/2.
```

Application

Enumeration

clingo consistance.lp -n 0

```
Answer:  1
predictedV(appY,down) predictedV(rpsP,up) predictedV(appB,up)
predictedV(rpmC,down) predictedV(arcA,up) predictedV(fnr,down)
Answer:  2
predictedV(appY,down) predictedV(rpsP,up) predictedV(appB,down)
predictedV(rpmC,down) predictedV(arcA,up) predictedV(fnr,down)
SATISFIABLE
```

Intersection des solutions : ce qui est certainement déduit

clingo consistance.lp −−enum-mode=cautious

```
Cautious consequences:
predictedV(appY,down) predictedV(rpsP,up) predictedV(rpmC,down)
predictedV(arcA,up) predictedV(fnr,down)
SATISFIABLE
```

Union des solutions: grouper tout ce qui peut être déduit

./clinfo consistance.lp −−enum-mode=brave

```
Brave consequences:
predictedV(appY,down) predictedV(rpsP,up) predictedV(appB,up)
predictedV(appB,down) predictedV(rpmC,down) predictedV(arcA,up)
predictedV(fnr,down)
SATISFIABLE
```

Extensions

Programme pour trouver une coloration qui contient le plus de signe "-" possibles ?

```
poids(up,0).
poids(down,10).
#maximize {T, predictedV(I,S): poids(S,T), predictedV(I,S), vertex(I), signe(S)}.
```

Concrètement. On recherche tous les modèles optimaux avec

clingo programme.lp −−opt-mode=optN

```
Solving...
Answer: 1
predictedV(appY,down) predictedV(rpsP,up) predictedV(fnr,down) predictedV(arcA,up) predictedV(rpmC,down)
predictedV(appB,down)
Optimization: -20
Answer: 2
predictedV(appY,down) predictedV(rpsP,up) predictedV(fnr,down) predictedV(arcA,up) predictedV(rpmC,down)
predictedV(appB,up)
Optimization: -30
Answer: 1
predictedV(appY,down) predictedV(rpsP,up) predictedV(fnr,down) predictedV(arcA,up) predictedV(rpmC,down)
predictedV(appB,up)
Optimization: -30
OPTIMUM FOUND

Models : 3
Optimum : yes
Optimization : -30
```

Lecture de la sortie

- Le solveur cherche l'optimum (Answer:1 ... Answer: 2) jusqu'à à trouver la bonne valeur optimale (-30).
- Ensuite, le solveur recommence sa recherche et énumère toutes les solutions dont le score prend cette valeur optimale (Answer:1 ... OPTIMUM FOUND)

ANSWER SET PROGRAMMING

$$\underbrace{K\ \{\ atom_1;\ \ldots;\ atom_n\ \}\ L}_{\text{head}}\ \underbrace{:-}_{\text{"smiley"}}\ \underbrace{atom_{n+1}\ ;\ldots;\ atom_r;\ not\ atom_{r+1};\ \ldots;\ not\ atom_s\ .}_{\text{body}}$$

High-level model language

- Propositional logics
- **Model for negation: everything is false a priori**
- **Reasoning modes**: enumeration, union, intersection of solutions
- **Optimisation**

Highly performant solving technics

- SAT-based and deductive-DB technics
- Decidable: no infinite loop

Optimisation rule
#maximize{W,atom(X): condition(X),W}.

Linear constrains atoms
&sum{a1*x1;...;al*xl} <= k

Problem statement & modelling

IRISA UMR

Universität Potsdam

Solving heuristics & problem reformulation

ASP programs never scale…
… until they do scale thanks to specific heuristics

- Repair large-scale interaction graph with **branch and bound** [KR'10]
- Scale metabolic network gap-filling problem with **unsatisfiable core** [LPNMR'13]
- Design experiments with **incremental solving** [Frontiers'15]
- Implement and benchmark **constraints propagators** [TPLP'18]

7. Back to biology & more tricky problems

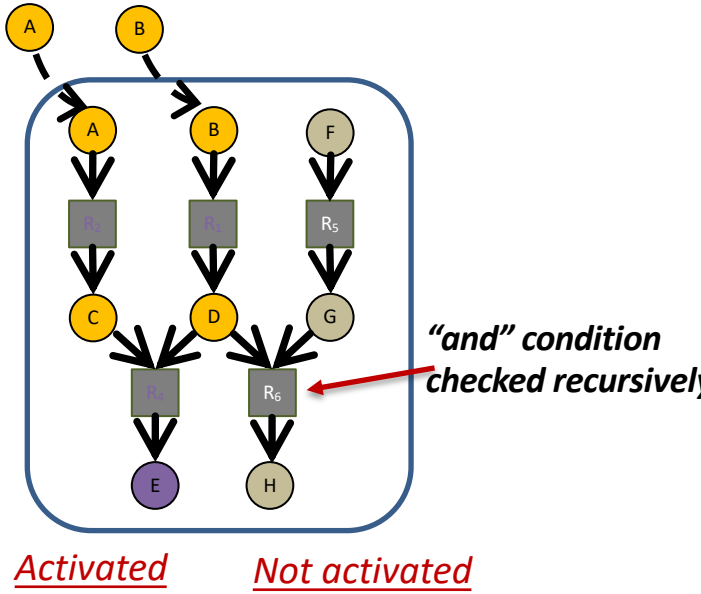
1. Short historical reminder: from observations to discoveries in life sciences
2. (Identification of) dynamical systems
3. Focus on the metabolic scale.
4. Reasoning over discrete dynamical systems
5. Knowledge representation : introduction to ASP
6. More insights on ASP
7. **Back to biology towards more tricky problems**
8. Still new issues

COMING BACK TO METABOLISM

Graph-based semantics *[Handorf, Ebenhoeh, Sagot, Schaub/Thiele]*

A reaction can be activated if all its substrates are

- either in the growth medium
- or the product of an activated reaction



From dynamics to reasoning:
caracterize metabolite production

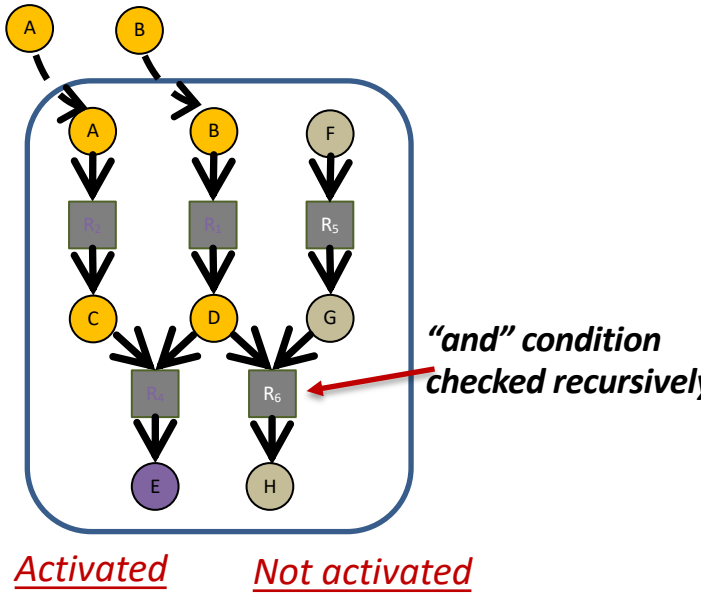
Solve the gap-filling problem

COMING BACK TO METABOLISM

Graph-based semantics *[Handorf, Ebenhoeh, Sagot, Schaub/Thiele]*

A reaction can be activated if all its substrates are

- either in the growth medium
- or the product of an activated reaction



From dynamics to reasoning: caracterize metabolite production

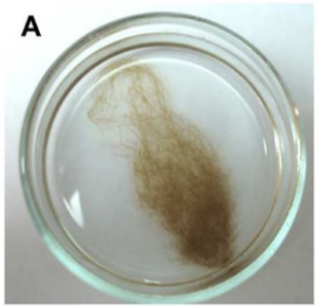
```
scope(M):- seed(M).
scope(M):- product(M,R), reaction(R), scope(M') : reactant(M',R).
```

Solve the gap-filling problem

```
{reaction(r)}.
scope(M):- seed(M).
scope(M):- product(M,R), reaction(R), scope(M') : reactant(M',R).
:- target(T), not scope(T).
#minimize{ reaction(r) }.
```

APPLICATION….

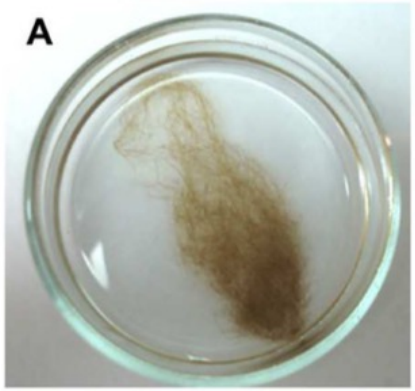
S. Prigent's PhD
Now @Inra

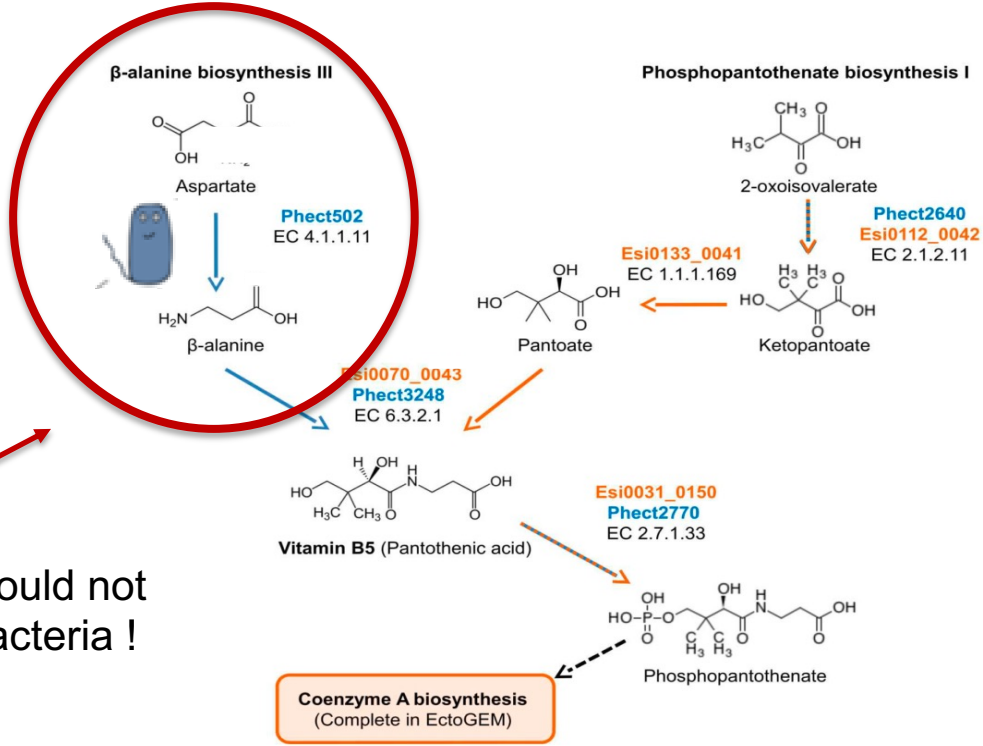


E. Siliculosus [Plant Journal'15]

- Flow-based gap-filling:. untractable
- Discrete-based gap-filling solved with ASP (union mode) :
  → 3500 solutions, 50 reactions to check (union reasoning mode).

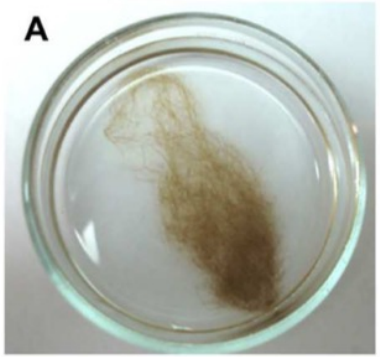
THE METABOLIC MAP HAD NEW SECRETS



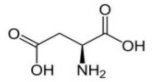


One of the necessary filled reactions could not be operated by the algae… but by a bacteria !

Biological discovery = inconsistency between predictions and knowledge

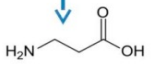


β-alanine biosynthesis III

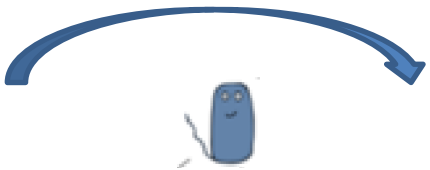


Aspartate

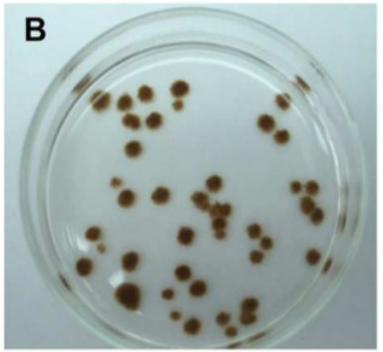
Phect502
EC 4.1.1.11



β-alanine

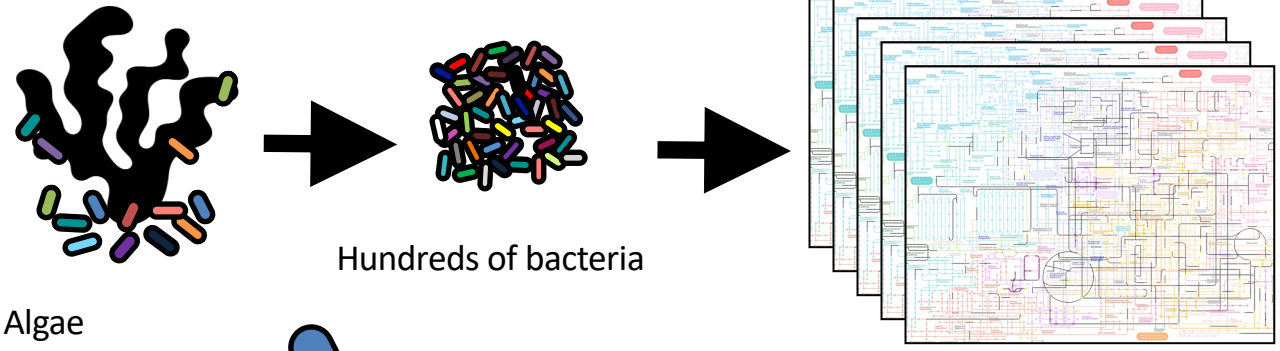


?



NEW COMBINATORIAL PROBLEM

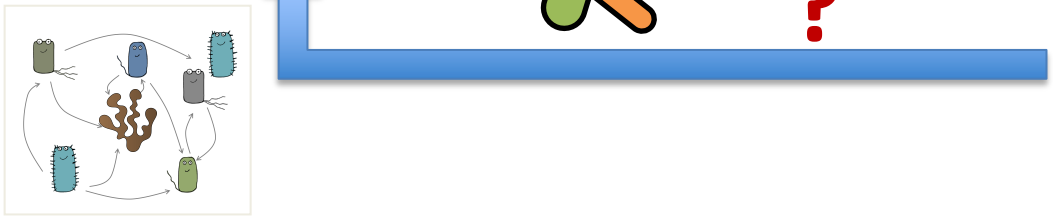
C. Frioux's phD.
Now @Inria

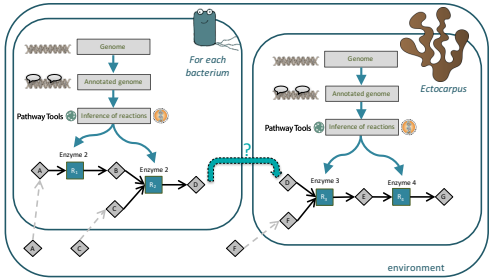


Algae

Hundreds of bacteria

Hundreds of metabolic maps with possible exchanges



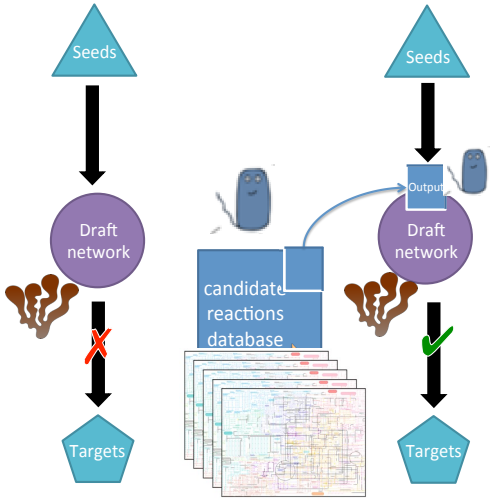


Smallest number of bacteria useful for the algae ?

NEW GAP-FILLING PROBLEM

Synthetic community problem

- Pick–up reactions in the database made of bacterial GSMs.
- For few bacteria: Steiner graph approach [Sagot team, 2017]



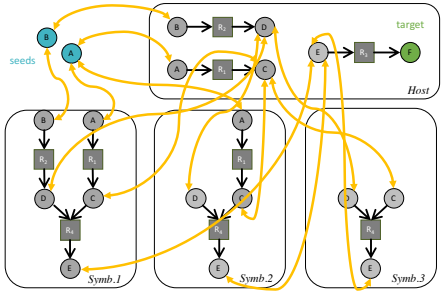
Gapfilling database: Bacterial GSMs

Gapfilling complexity

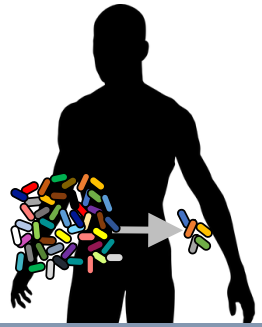
- depends on the number of hyperarcs

Size of the search space

- depends on the number of symbionts



499,177 combinations of <6 exchanges



$1.62.10^{81}$ combinations of <10 exchanges

Highly combinatorial optimization problem

PROBLEM RESOLUTION

C. Frioux's phD.
Now @Inria

Heuristics for the community selection problems,
→ playing with projection and union reasoning modes

Who problem

- Get rid of boundaries and select all minimal symbiot families
- **ASP-based gapfilling** with a meta-bacterial GSM as database

$$\text{mxdbagCnity}(S, T, G_1..G_N) = \underset{\{G_{i_1}..G_{i_L}\} \subset \{G_1..G_N\}}{\arg\min} \begin{pmatrix} \text{size}\left(T \backslash \text{mxdbagScope}(G_{i_1}..G_{i_L}, S)\right), \\ \text{size}\{G_{i_1}..G_{i_L}\}. \end{pmatrix}$$

How problem

- Sort the selected families according to the number of exchanges
- **ASP-based gapfilling** with union of compartimentalized GSMs as database.

$$\text{cptCnity}(S, T, G_1..G_N) = \underset{\substack{\{G_{i_1}..G_{i_L}\} \\ \subset \{G_1..G_N\}}}{\arg\min} \begin{pmatrix} \text{size}\left(T \backslash \text{mxdbagScope}(G_{i_1}..G_{i_L}, S)\right), \\ \text{size}\{G_{i_1}..G_{i_L}\}, \\ \text{size}\{\mathcal{E} \subset \text{exchg}(G_{i_1}..G_{i_L}) | \\ \quad T \cap \text{cptScope}(G_{i_1}..G_{i_L}, \mathcal{E}, S) \\ \quad = T \cap \text{mxdbagScope}(G_{i_1}..G_{i_L}, S)\}. \end{pmatrix}$$

Curation

- Model prior knowledge / co-culture / cultivable constraints
- Ask your favorite biologist to select the final community

VALIDATION AND BENCHMARKING ON THE HUMAN MICROBIOTE

C. Frioux's phD.
Now @Inria

Benchmark

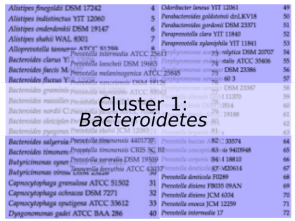
- Human model + 773 AGORA bacteria. 46 molecules to produce

Applying the miscoto tool [ECCB/bioinformatics 2018]

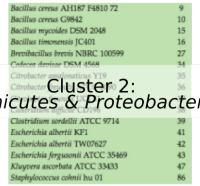
- Enumeration : 381 minimal communities of 3 bacteria,
- Union: 89 bacteria in total

Analysis of solutions (power graphs) Clustering of bacteria

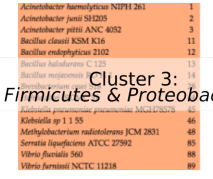
Each of the 381 communities is composed of
1 *Bacteoidetes* (/58) + 1 *Firmicute* or *Proteobacteria* (/15) + 1 *Firmicute* or *Proteobacteria* (/16)



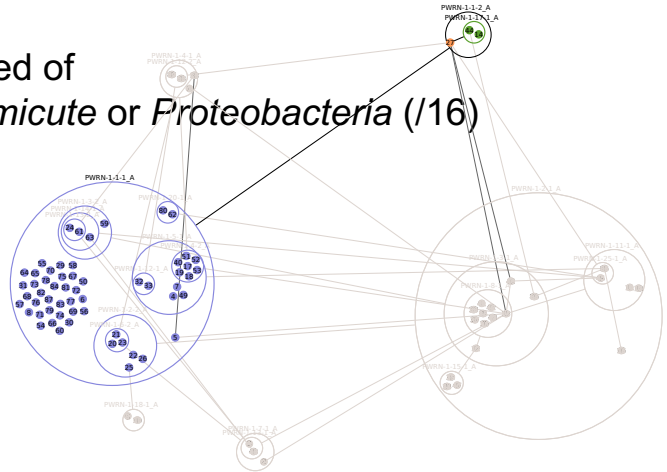
Cluster 1:
Bacteroidetes



Cluster 2:
Firmicutes & Proteobacteria



Cluster 3:
Firmicutes & Proteobacteria



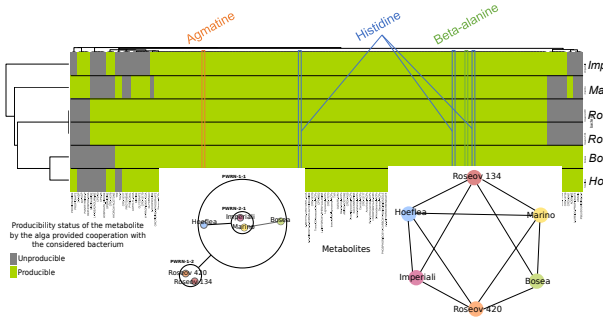
Back to biological discovery: Screening of individual metabolic functions
The functionality of individual targets explains the communities

IRISA

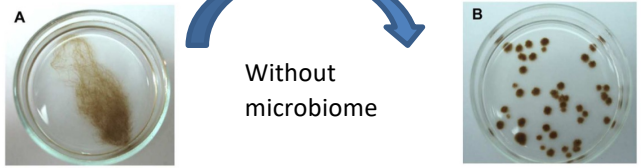
APPLICATION TO ALGAL METABOLISM

Preliminary community selection for E. Siliculosus

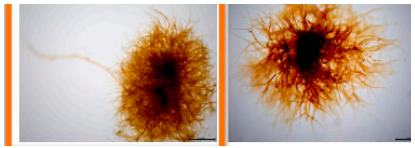
- 10 cultivable bacteria
- Prediction: 6 equivalent communities of 3 bacteria



Experimentations



+ 3 selected bacteria among 10 cultivable bacteria



With S. Dittami, Bertille Burgunter-Delamare [Frontiers, 2020]

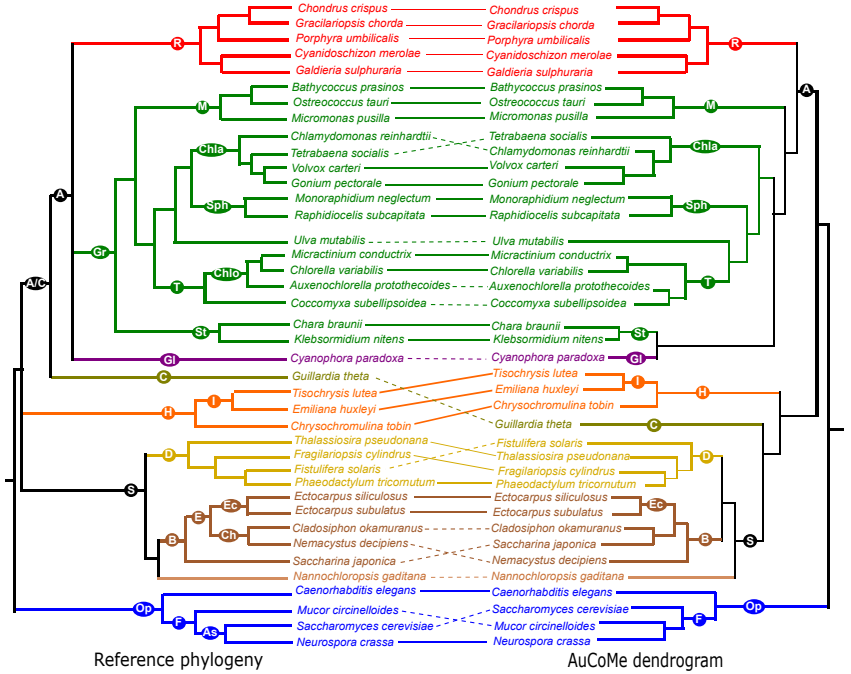
The algae grew again… but with strange behaviors … (to be continued)

APPLICATION A L'ETUDE D'UNE GRANDE FAMILLE D'ALGUES

Projet Phaeoexplorer

- **61 échantillons d'algues** séquencés
- Données bactériennes retrouvées malgré les lavages antibiotiques
  - **285 espèces bactériennes**

Reconstruction des réseaux métaboliques
De toutes les algues et de toutes les bactéries
(A. Belcour, P. Hamon-Giraud, C. Lucas)

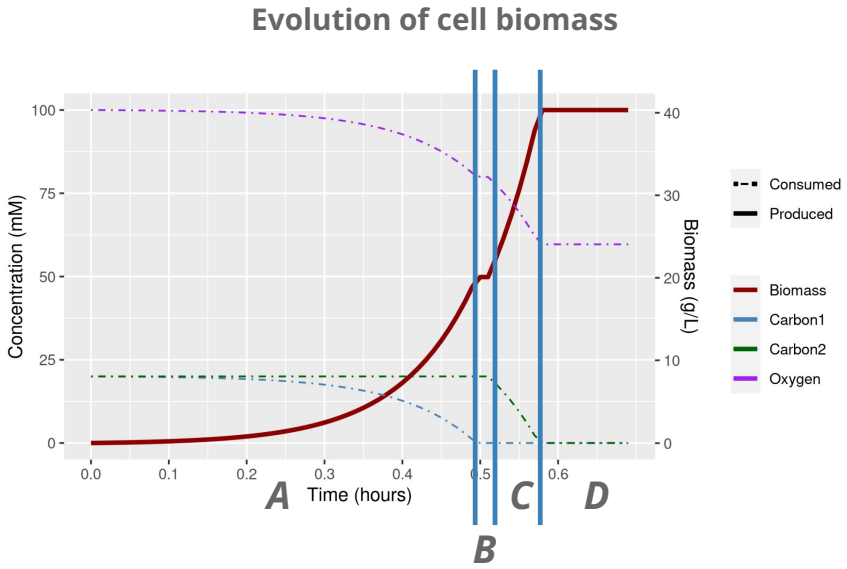


Genome Research 2023

8. Still new issues : discrete-continuous systems

1. Short historical reminder: from observations to discoveries in life sciences
2. (Identification of) dynamical systems
3. Focus on the metabolic scale.
4. Reasoning over discrete dynamical systems
5. Knowledge representation : introduction to ASP
6. More insights on ASP
7. Back to biology towards more tricky problems
8. **Still new issues**

COMPRENDRE DES CINETIQUES COMPLEXES DU METABOLISME ?



Diauxic shift

→ Successive growth phases on different mediums

→ Control by regulations

Divided in 4 phases

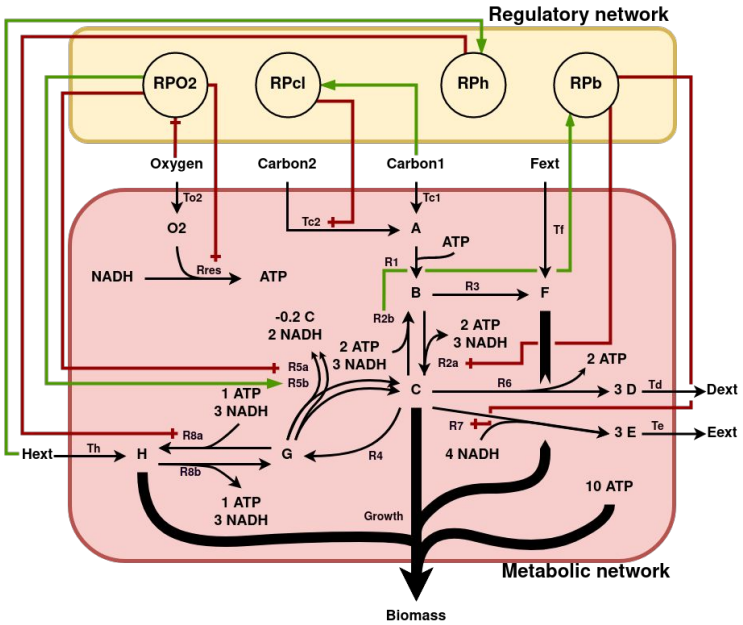
Characterize by different qualitative behaviours (e.g. growth medium)

A *→ Growth on* ***Carbon1 only***
B *→* ***No growth*** *due to regulations*
C *→ Growth on* ***Carbon2 only***
D *→* ***No growth****, no growth medium*

@Kérian Thuillier, collaboration L. Paulevé, A. Bockmayr, L. Cottret, C. Baroukh

MODELE SOUS-JACENT : MODELES MULTI-NIVEAUX

Gold standard instance *(Covert et al, 2001)*



Toy model based on *E.coli*

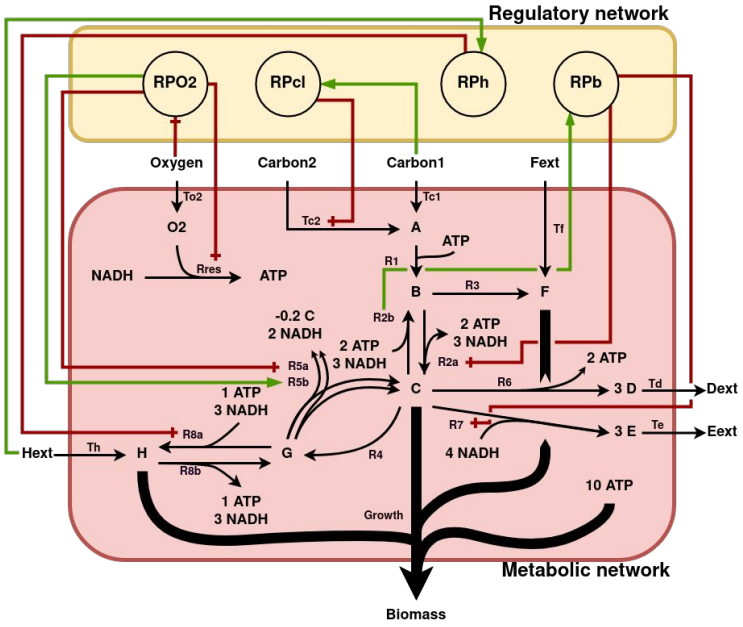
20 reactions, 4 regulatory proteins, 11 regulations

Model complex behaviours

Diauxic shift, aerobic/anaerobic growth, etc.

[1] M. W. Covert et al., **Journal of theoretical biology**, 2001

MODELE SOUS-JACENT : MODELES MULTI-NIVEAUX

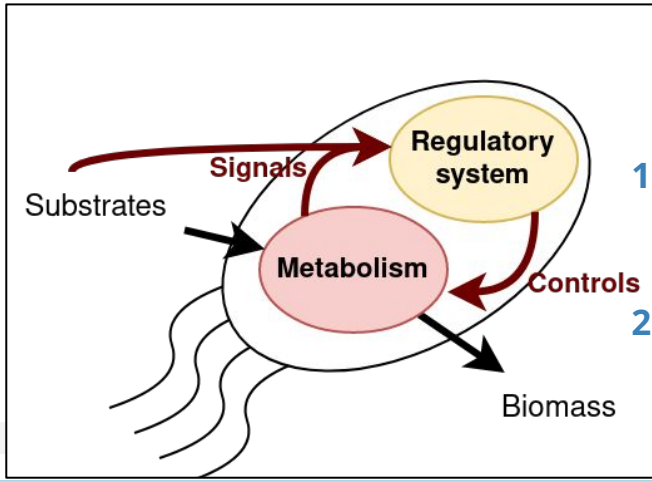


Toy model based on *E.coli*

20 reactions, 4 regulatory proteins, 11 regulations

Model complex behaviours

Diauxic shift, aerobic/anaerobic growth, etc.



Model as two interconnected systems

1. **Metabolic system**
   Chemicals reactions converting substrates to energy and biomass
2. **Regulatory system**
   Rules constraining the metabolism to adapt itself to its environment

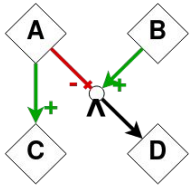
[1] M. W. Covert et al., **Journal of theoretical biology**, 2001

@Kérian Thuillier, collaboration L. Paulevé, A. Bockmayr, L. Cottret, C. Baroukh

Two models with different dynamics

Regulatory

Discrete dynamics[1]



Model

$$f_A(x_A, x_B, x_C, x_D) = x_A$$
$$f_B(x_A, x_B, x_C, x_D) = x_B$$
$$f_C(x_A, x_B, x_C, x_D) = x_A$$
$$f_D(x_A, x_B, x_C, x_D) = \neg x_A \wedge x_B$$

Logical rules
(Boolean network)

Inputs				Outputs			
x_A	x_B	x_C	x_D	x_A	x_B	x_C	x_D
0	0	x	x	0	0	0	0
0	1	x	x	0	1	0	1
1	0	x	x	1	0	1	0
1	1	x	x	1	1	1	0

Truth table (simulation)

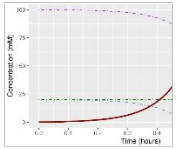
Interactions

Metabolic

Steady-states approximation[2]

$$\begin{aligned} \text{maximise} \quad & x_{\text{Growth}} \\ \text{such that:} \quad & S \cdot x = 0 \\ & l_r \cdot f_r \leq x_r \leq u_r \cdot f_r \qquad \forall r \in \mathcal{R} \end{aligned}$$

Dynamic flux balance analysis (dFBA)



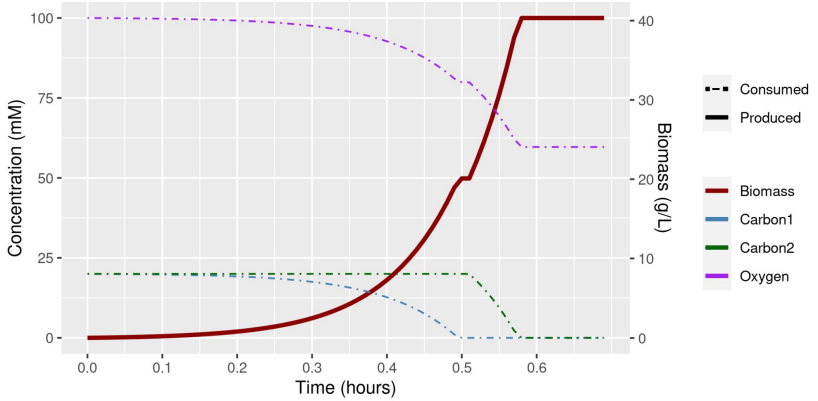
Metabolic traces

[1] S. Videla et al., **Bioinformatics**, 2016
[2] M. W. Covert et al., **Journal of theoretical biology**, 2001

@Kérian Thuillier, collaboration L. Paulevé, A. Bockmayr, L. Cottret, C. Baroukh

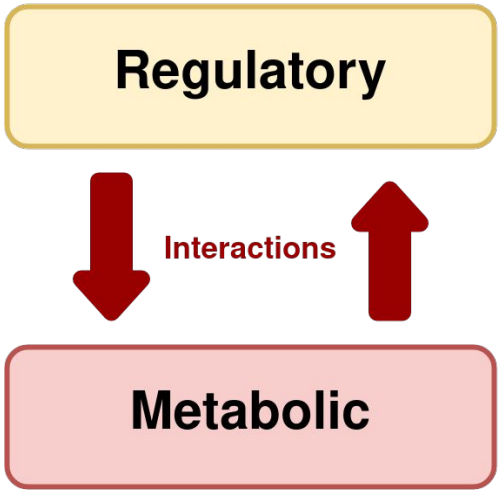
IRISA UMR

Evolution of cell biomass



Several simulations approaches

*Based on **regulatory Flux Balance Analysis**[1] (dynamic + regulations)*



rFBA timestep:

1. Update the **regulatory system**

   1 synchronous update of the Boolean network

$$
\begin{aligned}
f_A(x_A, x_B, x_C, x_D) &= x_A \\
f_B(x_A, x_B, x_C, x_D) &= x_B \\
f_C(x_A, x_B, x_C, x_D) &= x_A \\
f_D(x_A, x_B, x_C, x_D) &= \neg x_A \wedge x_B
\end{aligned}
$$

2. Update the **metabolic system**

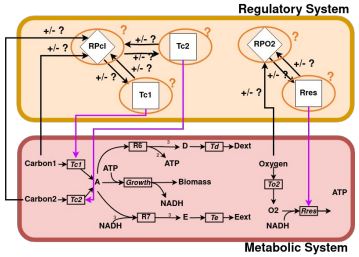
   Solve FBA — LP problem

$$
\begin{aligned}
\text{maximise} \quad & x_{\text{Growth}} \\
\text{such that:} \quad & S \cdot x = 0 \\
& l_r \cdot f_r \leq x_r \leq u_r \cdot f_r \qquad \forall r \in \mathcal{R}
\end{aligned}
$$

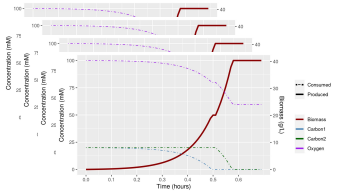
3. Update the cell environment

UN PROBLEME D'INFERENCE : RETROUVER LES REGLES DE REGULATION A PARTIR DES DONNEES ?

Inferring regulatory rules from time series observations



Prior Knowledge Network

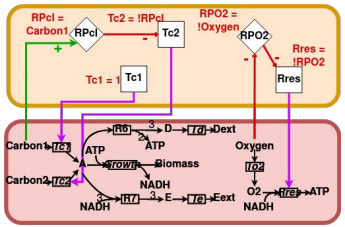


Time series
(kinetics, fluxomics, transcriptomics)

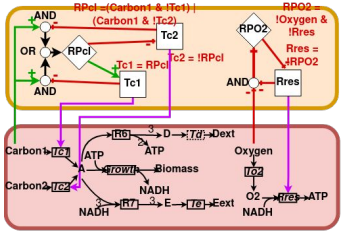
Input:

➔ **Prior Knowledge Network (PKN)**
Set of admissible interactions between components of the regulatory network

➔ **Time series data**
Kinetics, fluxomics and/or transcriptomics



Solution n°1

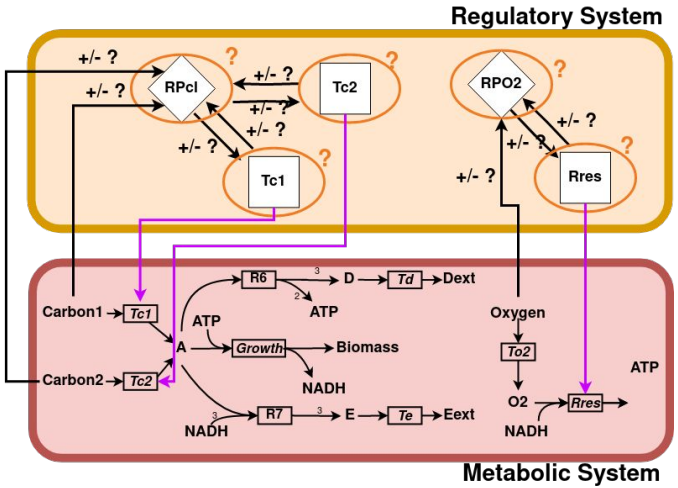


Solution n°2

Output:

➔ **Set of consistent regulatory networks**
Respecting the admissible interactions
Allowing to reproduce the input time series

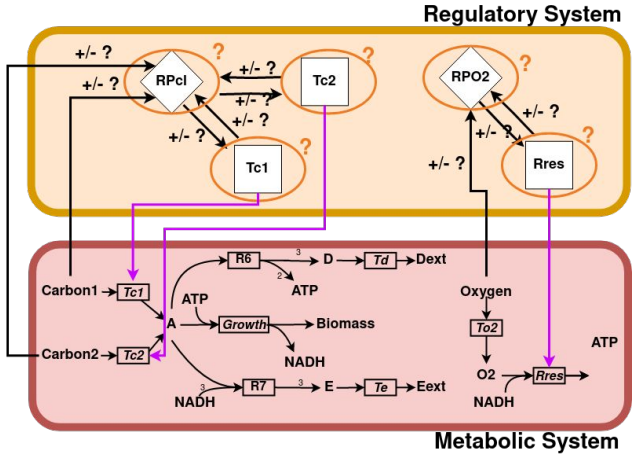
ESPACE DE RECHERCHE : CONTRAINT PAR UN GRAPHE D'INTERACTION A PRIORI



Prior Knowledge Network

Only specific interactions (*activations*, *inhibitions*) between proteins and enzymes can be used in Boolean regulatory rules

ESPACE DE RECHERCHE : CONTRAINT PAR UN GRAPHE D'INTERACTION A PRIORI



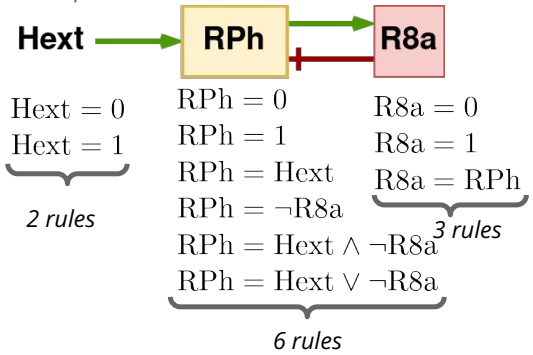
Prior Knowledge Network

Only specific interactions (*activations*, *inhibitions*) between proteins and enzymes can be used in Boolean regulatory rules

Prior Knowledge Network:

Set of authorised interactions: activation and inhibition effects

Example:



Regulatory rule of RPh can only depend on:
*- **activation of Hext***
*- **inhibition of R8a***

***6 potential Boolean regulatory rules** for these 2 interactions*

Size of the search space

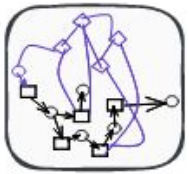
$O(2^{2^n})$ ***in the number*** n ***of interactions***

ETAT DE L'ART : NE TIENT PAS COMPTE DU METABOLISME !!!

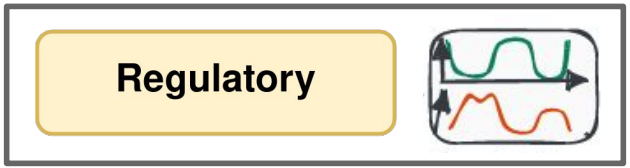
Several inferring approaches[1,2]

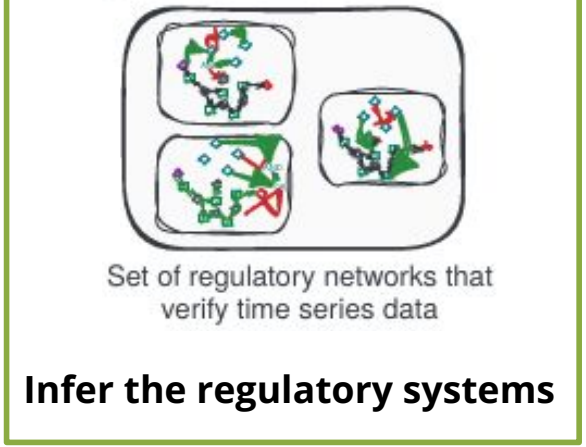
→ Based on **constraint programming**
→ **Discrete modelling** of the regulatory system dynamics
→ **Observations of the regulatory system + PKN**

PKN

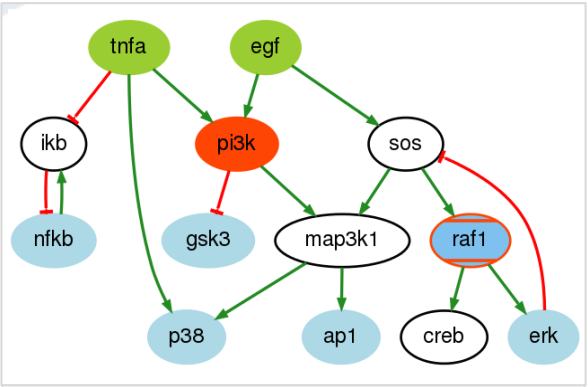


Observations

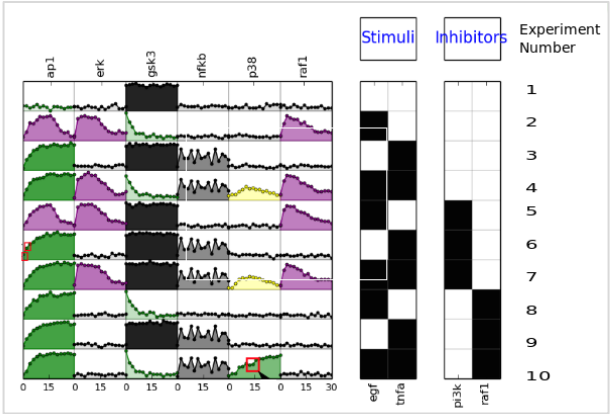




Si on oublie le métabolisme: décrire la réponse combinatoire d'un modèle



Domain knowledge



Phosphoproteomic data

Goal. Explain individual data

- Propagation of regulations among a discrete system
- Common hidden mutation

Sample	mutation	RNA-seq
Patient	-	AP1 =0
Patient 1	SOS	AP1 ++
Patient 2	pi3K	P38 = 0

Genomics (inter-individual data)

Reformulating the inference problem

Mathematical problem: Optimize $\Theta_{rss}((V,\phi)) + 0.1\Theta_{size}((V,\phi))$

1st objective ($\in \mathbb{R}$). difference between observations and predictions

$$\Theta_{rss}((V,\phi), \underbrace{(P_1,\ldots,P_n)}_{n \text{ experiments}}) = \sum_{i=1}^{n} \sum_{v \in dom(P_i)} \left(\underbrace{P_i(v)}_{\text{observations} \in [0,1]} - \underbrace{\pi_i(v)}_{\text{predictions} \in \{0,1\}} \right)^2$$

2nd objective ($\in \mathbb{N}$). model complexity

$$\Theta_{size}((V,\phi)) = \sum_{v \in dom(\phi)} |\phi(v)|$$

- Local minima
- No global enumeration
- No data noise

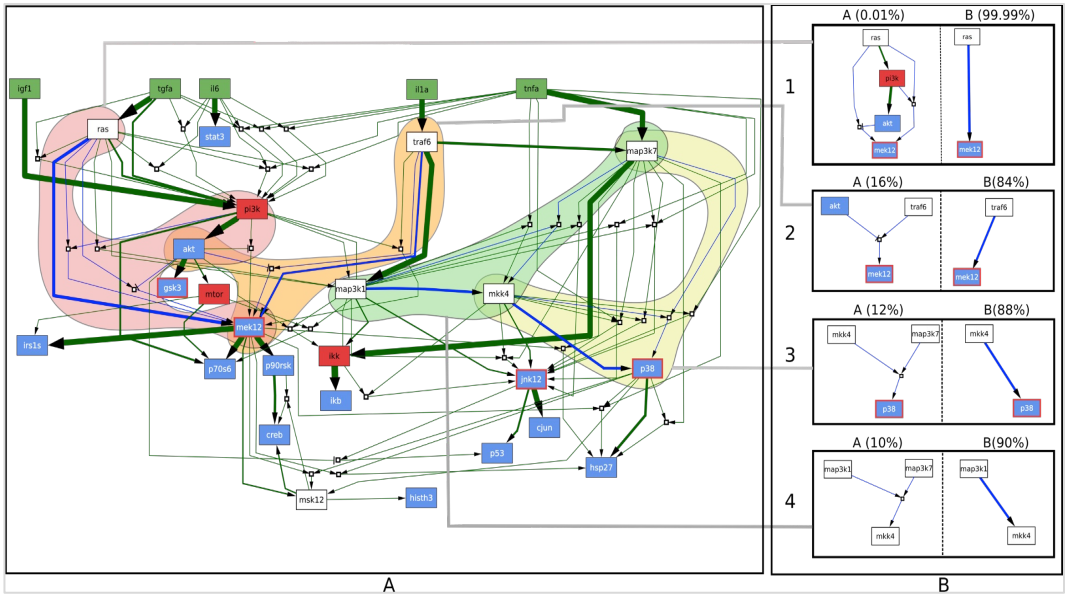
Impact of data discretization
on the solutions of the optimization problem

Computing all sub-optimal solutions
to a combinatorial **multi-optimization problem → ASP encoding**

$$\left\{ (V,\phi) \in \underbrace{\mathbb{M}_{(V,E,\sigma)}}_{\text{Structure}} \mid \underbrace{Score_{rss}((V,\phi), Exp\_Data)}_{\text{Fitting}} \leq \underbrace{(1 + 10\%) \min Score_{rss}}_{\text{Noise tolerance}} \text{ and } \underbrace{Score_{size}((V,\phi))}_{\text{Parsimony}} \leq \underbrace{\min Score_{rss} + 2}_{\text{Size tolerance}} \right\}$$

Videla, et al, Journal of Theoretical Computer Science, 2015

Network identification from two time points



Within the search space of 2^{87} models,
exactly 5306 boolean models explain equivalently
well the 2009 Dream challenge data.

Videla, et al, Journal of Theoretical Computer Science, 2015
Guziolowski et al, *Bioinformatics*, 2013

Boolean network inference from time-series data

Main limitation:
ASP is not a model-checker: finite trajectories only.

➢ **How to take dynamical traces into account ?**

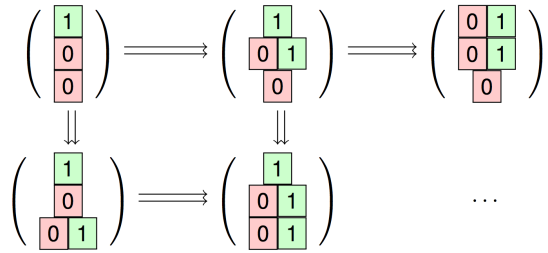
$$\begin{pmatrix} u_{1..i-1} \\ \boxed{a} \\ u_{i+1..n} \end{pmatrix} \rightrightarrows \begin{pmatrix} u_{1..i-1} \\ 0\ 1 \\ u_{i+1..n} \end{pmatrix} \quad \text{if } \exists x \in u : f_i(x) \neq a$$

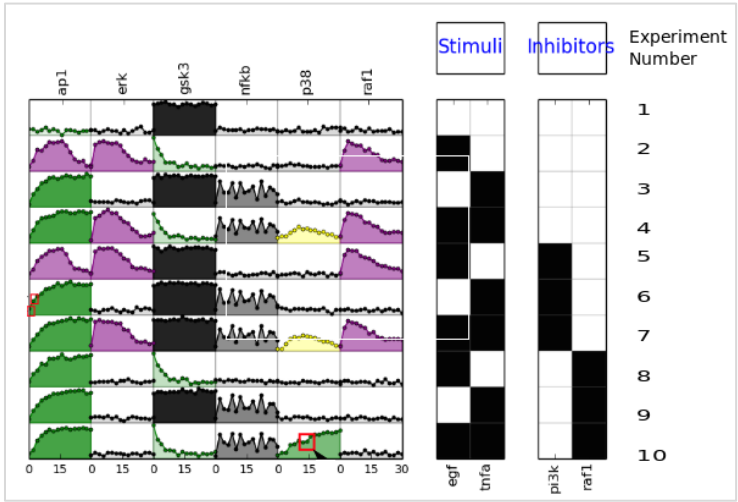
Example

$f_1(x) = \neg x_2 \vee x_3$

$f_2(x) = x_1$

$f_3(x) = \neg x_2 \vee x_3$



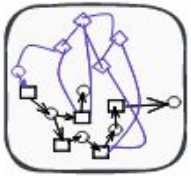


➢ **Abstraction of logical states with meta-states**

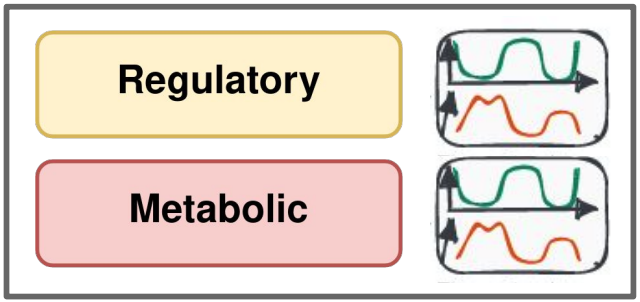
➢ A posteriori-filtering with model-checkers

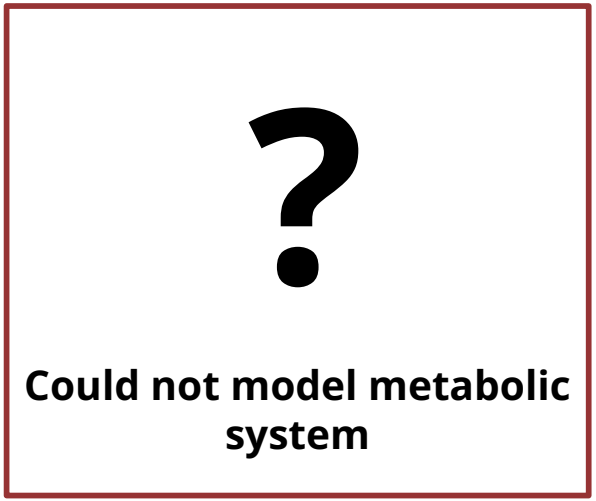
RETOUR AU PROBLEME DU METABOLISME

PKN



Observations

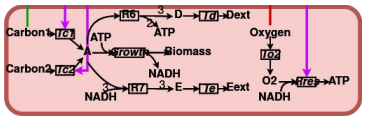




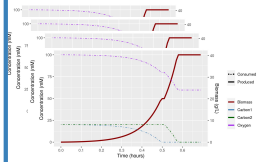
RAPPEL

Inferring problem — *in a nutshell*

Inputs:



Metabolic network $\mathcal{N}$



Pair of successive observations of:

1. *Metabolic fluxes* $\mathcal{V}$
2. *Substrate* $\mathcal{W}$
3. *Regulatory state* $\mathcal{X}$

Set of time series $\{T_i\}_i$

(kinetics, fluxomics, transcriptomics)

Set of authorised interactions: activation and inhibition effects

$\mathrm{Hext} = 0$
$\mathrm{Hext} = 1$

$\mathrm{RPh} = 0$
$\mathrm{RPh} = 1$
$\mathrm{RPh} = \mathrm{Hext}$
$\mathrm{RPh} = \neg \mathrm{R8a}$
$\mathrm{RPh} = \mathrm{Hext} \wedge \neg \mathrm{R8a}$
$\mathrm{RPh} = \mathrm{Hext} \vee \neg \mathrm{R8a}$

$\mathrm{R8a} = 0$
$\mathrm{R8a} = 1$
$\mathrm{R8a} = \mathrm{RPh}$

36 compatible regulatory networks

$O(2^{2^n})$ ***in the number*** n ***of interactions***

Prior Knowledge Network (PKN)

Define a search space $\mathcal{F}$

Outputs:

All the Boolean networks $f \in \mathcal{F}$ such that there is a rFBA simulations matching the input time series $\{T_i\}_i$

Inferring problem — *formal definition*

Input: *metabolic network* $\mathcal{N}$*, search space* $\mathcal{F}$*, set of time series* $\{T_i\}_i$

Output: *all Boolean networks* $f \in \mathcal{F}$ *such that:*

Boolean constraints

$$\bigwedge_{T_i} \bigwedge_{(s,s') \in T_i} \Big(f(x) = x' \Big)$$

$$\wedge\, \exists \hat{v} \in \mathbb{R}^{\text{Reactions}}, \left(S \cdot \hat{v} = 0 \wedge \bigwedge_{r \in \text{Reactions}} l_r \cdot x'_r \leq \hat{v}_r \leq u_r \cdot x'_r \wedge \hat{v}_{\text{growth}} \geq v'_{\text{growth}} - \epsilon \right)$$

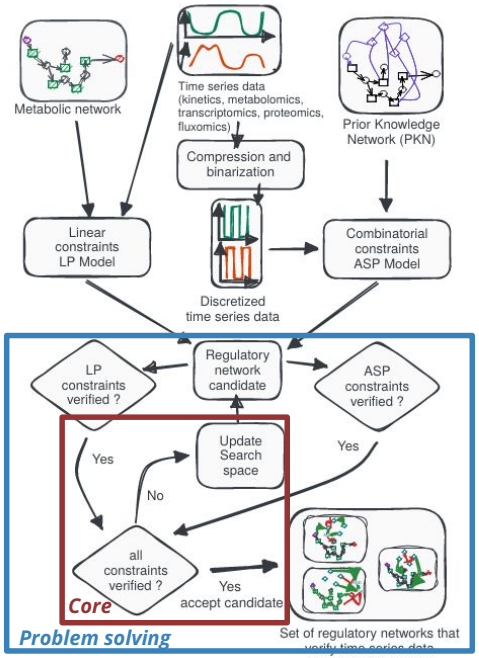
$$\wedge\, \forall \hat{v} \in \mathbb{R}^{\text{Reactions}}, \left(S \cdot \hat{v} = 0 \wedge \bigwedge_{r \in \text{Reactions}} l_r \cdot x'_r \leq \hat{v}_r \leq u_r \cdot x'_r \right) \implies \hat{v}_{\text{growth}} \leq v'_{\text{growth}} + \epsilon \Big)$$

Quantified linear constraints

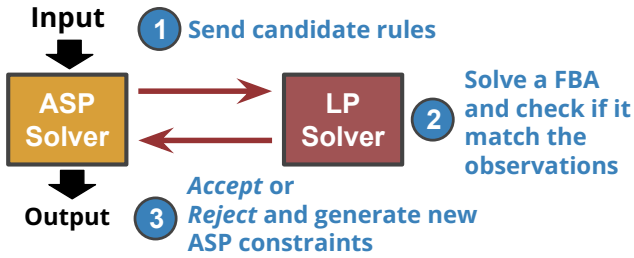
Hybrid problem: combinatorial + quantified linear constraints

Resolution framework

MERRIN – published at ECCB22



Based on the *conflict driven clause learning* algorithm[1] and satisfiability modulo theory[2] methods

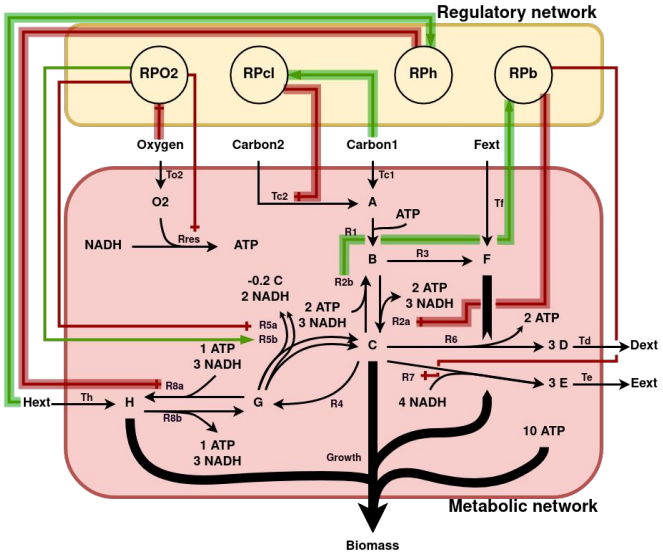


New constraints (*nogoods*) are generated for each conflict

- Generalisation of conflicts
- Reduce the number of call to the LP solver

[1] **CDCL:** J. P. Marques-Silva and K. A. Sakallah, **IEEE Transactions on Computers**, 1999
[2] **SMT:** C. Barrett and C. Tinelli, **Handbook of model checking**, 2018

Validation and robustness testing



Learn more parsimonious model than ground truth

- Reproduce exactly the input time series
- Unrecovered regulations can be explained

Validation on a benchmark of 240 instances (*in silico*)

- 4 data types
- 6 level of degradations (0% to 50%)

Perfectly reproduce the time series with:

- *kinetics* and *transcriptomics* data
- < *20%* of degradation

Bunch of conclusions

TAKE HOME MESSAGE

Life sciences are becoming a data-science….

But they cannot be adressed with usual complex system technics.

→ **Curated and reliable dynamical models are difficult to obtain (**Large-scale data, lacks of knowledge and information)

Our strategy: explore and integrate data with discrete abstractions

- Favor combinatorial optimisation problems to gain in robustness
- **Use boolean abstractions** of the dynamics (*minimal* fixed-point)

ASP technologies allow for a scalable reasoning on data and networks

(Win-win collaboration with your BFF ASP-tech developers)

New discoveries in life science may result rather from inconsistencies between knowledge and data (abduction) than from data-science technics.

ANSWER SET PROGRAMMING

$$\underbrace{K\ \{\ atom_1;\ \ldots;\ atom_n\ \}\ L}_{\text{head}}\ \underbrace{:-}_{\text{"smiley"}}\ \underbrace{atom_{n+1}\ ;\ldots;\ atom_r;\ not\ atom_{r+1};\ \ldots;\ not\ atom_s\ .}_{\text{body}}$$

High-level model language

- Propositional logics
- **Model for negation: everything is false a priori**
- **Reasoning modes**: enumeration, union, intersection of solutions
- **Optimisation**

Highly performant solving technics

- SAT-based and deductive-DB technics
- Decidable: no infinite loop

Optimisation rule

```
#maximize{W,atom(X): condition(X),W}.
```

Linear constrains atoms

```
&sum{a1*x1;...;al*xl}  <=  k
```

Problem statement & modelling

IRISA UMR

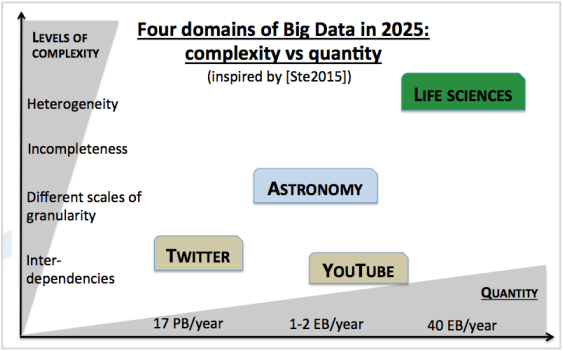
Universität Potsdam

Solving heuristics & problem reformulation

ASP programs never scale…
… until they do scale thanks to specific heuristics

- Repair large-scale interaction graph with **branch and bound** [KR'10]
- Scale metabolic network gap-filling problem with **unsatisfiable core** [LPNMR'13]
- Design experiments with **incremental solving** [Frontiers'15]
- Implement and benchmark **constraints propagators** [TPLP'18]

REASONING IN LIFE SCIENCES



Life sciences and systems biology are a wonderful playing field

- For new dynamical modeling frameworks (competitions…)
- For new knowledge representation and reasoning paradigms

Addressing the data nightmare in life sciences ?

- Machine learning/deep learning approaches : inductive reasoning, correlations
- **Biology is also/mainly interested in causalities, knowledge discovery, and hypothesis formulation, leading to inconsistencies… and new theories.**

→ **Produce hypotheses rather than replacing biologists by in silico models**

Hypothesis…

- **Life sciences = priviledged field to experiment new theories of abductive reasoning applied to large-scale data ?**
- Do no forget dynamical systems behind…

PLACE DES ERREURS DANS LES PREDICTIONS

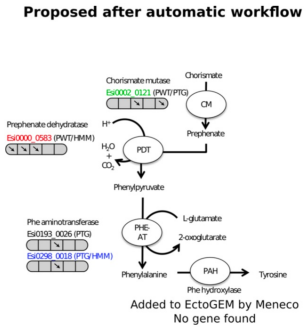
Méthode(s) in silico pour boucher les trous dans les cartes

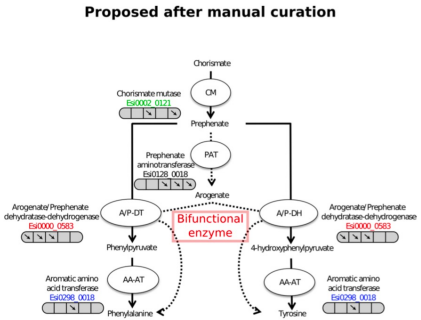
(Logiciels: meneco, miscoto, AuReMe, AuCoMe…)

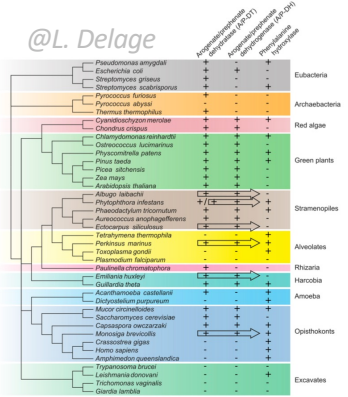
Synthèse de la tyrosine chez E. siliculosus :

gène absent mais compréhension du du rôle bifonctionnel d'une enzyme

(Prigent et al, 2014)



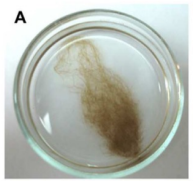




Arrows: bifunctional enzymes

Voie de synthèse de la coenzyme A : nécessite de la beta-alanine et du pantoate

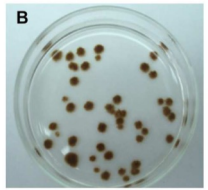
- L'algue sait produire le pantoate… mais n'a pas le gène prédit in silico pour produire la beta-alanine
- Ce gène est présent chez une bactérie symbiotique rescapée de la phase d'axénisation → Holobionte !



Sans microbiote

Condition axénique

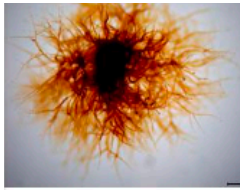
(Tapia et al, 2016)



Ajout de 3 bactéries prédites in silico

(Frioux et al, 2018)

(Burgunter-Delamare, 2020)



Identification de candidats (inexacts) pour faire progresser la connaissance

Remerciements

Team Dyliss@IRISA

- C. Frioux (now @Inria Bordeaux)
- S. Prigent (now @Inrae)
- C. Guziolowski (now@Univ Nantes)
- **J. Got**
- **S. Blanquart**
- **A. Belcour**
- **K. Thuillier**
- **P. Hamon Giraud**
- **C. Lucas**

Algae@SBR Roscoff

- **S. Dittami**
- H. Kliijean
- B. Burgunter
- T. Tonon
- **G. Markov**

Merrin team

Loic Paulevé

- Alexander Bockmayr
- Ludovic Cottret
- Caroline Baroukh

ASP tech@Potsdam university

- **T. Schaub's** team
- S. Thiele

www.irisa.fr @irisa_lab

Institut de Recherche en Informatique et Systèmes Aléatoires

Programmation par ensembles-réponses: un peu de sémantique

Ensemble-réponse dans le cas de programmes positifs

Programme positif

Un programme positif est constitué d'un ensemble de *règles* (clauses définies)

$$\underbrace{A}_{tete} \text{ :- } \underbrace{B_1; B_2; \ldots B_m}_{corps}.$$

où A (la *tête* de la règle) et les B (le *corps*) sont des *atomes* (variables booléennes)

A :- $B_1; B_2; \ldots B_m.$ correspond à la formule $\neg B_1 \vee \neg B_2 \vee \neg \vee \neg B_m \vee A$

Les programmes positifs sont des clauses disjonctives avec exactement un atome positif.

Ensemble clos d'un programme positif

Un ensemble d'atomes X est *clos pour un programme positif P* si

$$\textbf{pour toute règle } r \textbf{ du programme } P, \quad \textbf{corps}(r) \subset X \implies \textbf{tête}(r) \subset X$$

C'est un *modèle du programme* vu comme une formule.

Ensemble-réponse

L'ensemble réponse $Cn(P)$ (*answer set*) d'un programme positif P est le plus petit ensemble d'atomes qui soit clos pour P. Il existe et est unique.

Exemple 1

Vérifier que pour toute règle r, on a bien $\text{corps}(r) \subset X \implies \text{tête}(r) \subset X$

Programme

```
a :- b;p;q.
b :- q.
```

Liste des sous-ensembles d'atomes

{a,b,p,q}, {a,b,p}, {a,b,q}, {a,p,q}, {b,p,q}, {a,b}, {a,p}, {a,q}, {b,p}, {b,q}, {p,q}, {a}, {b}, {p}, {q}, ∅.

Sous-ensembles clos **Question: quels sont-ils ?**

{a,b,p,q}, {a,b,p}, {a,b,q}, NONCLOS{a,p,q}, NONCLOS{b,p,q},
{a,b}, {a,p}, NONCLOS{a,q}, NONCLOS{b,p},
{b,q}, NONCLOS{p,q}, {a}, {b}, {p}, NONCLOS{q}, ∅.

Quel est l'ensemble-réponse du programme ?

L'ensemble-réponse du programme est l'ensemble vide ∅

→ Le programme ne contient que des implications, et rien n'est "vrai".

Tout ce qui n'est pas prouvé par un fait est considéré comme faux!

Exemple 2

Vérifier que pour toute règle r, on a bien $\text{corps}(r) \subset X \implies \text{tête}(r) \subset X$

Programme

```
a :- b;p;q.
b :- q.
q.
```

Liste des sous-ensembles d'atomes

{a,b,p,q}, {a,b,p}, {a,b,q}, {a,p,q}, {b,p,q}, {a,b}, {a,p}, {a,q}, {b,p}, {b,q}, {p,q}, {a}, {b}, {p}, {q}, ∅.

Liste des sous-ensembles clos ?

{a,b,p,q}, NONCLOS{a,b,p}, {a,b,q}, NONCLOS{a,p,q}, NONCLOS{b,p,q}, NONCLOS{a,b}, NONCLOS{a,p}, NONCLOS{a,q}, NONCLOS{b,p}, {b,q}, NONCLOS{p,q}, NONCLOS{a}, NONCLOS{b}, NONCLOS{p}, NONCLOS{q}, NONCLOS ∅.

→ Comme ∅ est contenu dans un corps, q doit être contenu dans tous les sous-ensembles clos.

Quel est l'ensemble-réponse?

L'ensemble réponse du programme est {**b,q**}

→ On n'a pas gardé p puisqu'on n'a aucune preuve de p.

Les atomes d'un ensemble-réponse d'un programme positif apparaissent dans la tête d'au moins une règle

Programmes avec une négation ?

Programme "quelconque"

Un *programme* est constitué de manière plus générale d'un ensemble de règles

$$\texttt{r:} \quad \underbrace{A}_{tete} \texttt{ :- } \underbrace{B_1; B_2; \ldots B_m}_{corps^+(r)} \underbrace{not\ B_{m+1};\ not\ B_{m+2}; \ldots not\ B_{m+n}}_{corps^-(r)}.$$

Réduit par rapport à un ensemble

Le réduit d'un programme par rapport à un ensemble d'atomes X est l'ensemble des règles de la forme

$\texttt{r:} \quad \texttt{tête(r) :- corps}^+\texttt{(r).}$ pour tout règle r telle que $corps^-(r) \cap X = \emptyset$.

Identification

- On choisit un sous-ensemble d'atomes X
- On ne garde que les règles r : A :- $B_1; B_2; \ldots B_m$ pour lesquelles aucun des $B_{m+1}, B_{m+2}, \ldots B_{m+n}$ n'appartiennent à X.

Exemple `a :- b; not q.` `b:- p; q.` `p.`

- **Exercice: calculer le réduit du programme par rapport à** { `q` }

  ```
  b:- p; q.
  p.
  ```

- **Exercice: calculer le réduit du programme par rapport à** { `a` }

  ```
  a :- b.
  b:- p; q.
  p.
  ```

"Not": ensemble-réponse

Ensemble-réponse

Un *ensemble réponse* (*modèle stable*) d'un programme P est un sous-ensemble X d'atomes tel que

- X est clos pour le programme obtenu comme réduit de P par rapport à X.
- X est **stable**: X **est égal à l'ensemble-réponse de son programme réduit**.
- X est minimal pour l'inclusion parmi tous les ensembles clos et stables.

Remarque

- Il faut d'abord réduire le programme par rapport à X et ensuite vérifier que ce qu'on obtient est clos vis à vis de X.
- Il peut y en avoir 0, 1 ou plusieurs.

Contenu d'un ensemble-réponse

Un modèle contient l'atome d'une tête de règle

- si la règle est un fait (règle réduite à une tête) après en avoir enlevé les littéraux négatifs qui ne sont pas dans le modèle
- ou si tous les littéraux positifs du corps sont dans le modèle et aucun des littéraux négatifs.

Exemple 1

```
p:- p.  q:- not p.
```

Enumérer les sous-ensembles X d'atomes.
{p,q}, {p}, {q}, ∅.

Pour chacun, calculer le réduit.

- {p,q}. Réduit: ? p:-p.
- {p}. Réduit: ? p:-p.
- {q}. Réduit: ? p:-p. q.
- ∅. Réduit: ? p:-p. q.

Pour chaque réduit, calculer l'ensemble clos minimal

- {p,q}. Réduit: p:-p. . Ensemble clos minimal? ∅
- {p}. Réduit: p:-p. Ensemble clos minimal? ∅
- {q}. Réduit: p:-p. q. Ensemble clos minimal? {q}
- ∅. Réduit: p:-p. q. Ensemble clos minimal? {q}

Garder les clos stables minimaux
Ensembles-réponses ? {q}
(c'est le seul ensemble qui est égal à l'ensemble-réponse de son réduit.)

Exemple 2

```
p:- not q.   q:- not p.
```

Enumérer les sous-ensembles X d'atomes.
{p,q}, {p}, {q}, ∅.

Pour chacun, calculer le réduit.

- {p,q}. Réduit: ?→ ∅
- {p}. Réduit: ? → p.
- {q}. Réduit: ? → q.
- ∅. Réduit: ? → p. q.

Pour chaque réduit, calculer l'ensemble clos minimal

- {p,q}. Réduit: programme vide . Ensemble clos minimal? rien
- {p}. Réduit: p. Ensemble clos minimal? {p}
- {q}. Réduit: q. Ensemble clos minimal? {q}
- ∅. Réduit: p. q. Ensemble clos minimal? {p,q}

Garder les clos minimaux

Ensembles-réponses ? {p}, {q}

Exemple 3

```
p:- not p.
```

Enumérer les sous-ensembles X d'atomes.

{p}, ∅.

Pour chacun, calculer le réduit.

- {p,q}. → ∅
- ∅. → p.

Pour chaque réduit, calculer l'ensemble clos minimal

- {p,q}. Réduit: programme vide . Ensemble clos minimal? rien
- ∅. Réduit: p. Ensemble clos minimal? {p}

Garder les clos minimaux

Ensembles-réponses ? Aucun: il n'y a pas d'ensemble stable !

A retenir

Un ensemble réponse contient des atomes se trouvant dans une tête de règle du programme.
Tout élément d'un ensemble réponse est supporté par une règle.

Logique non monotone Rajouter des faits à une théorie peut faire réduire l'ensemble de conclusions.