Programmation par ensemble-réponse (ASP) et application à la reconstruction et la correction de réseaux biologiques

Anne Siegel

Ecole Thematique CNRS - Porquerolles. 2013

Location

University Rennes
& CNRS & Inria

↓

Computer science dpt

↓

Bioinformatics &
Systems biology

DyLiSS

UMR IRISA

Personal presentation

Background

- Thesis. Discrete dynamical systems. Ergodic theory and fractals.
- Arriving in Rennes.... Applications of dynamical systems in theoretical computer sciences (numeration systems, discrete geometry).
- ... Now. Applications of dynamical systems in integrative and systems biology.

Main questions

- **Which invariant properties of dynamical systems lead to a better understanding of biological systems?**
- How can we take such information into account?

Motivation

Integration of data and knowledge (usual approach)

- Characteristics. Uncertainty and multi-scale observations.
- Methods. Discriminative approaches selecting the most accurate model.
- Weakness. Very dependent on the quality of data.

Goal : reasoning over a complete family of feasible models instead of selecting one model

- Exhibit properties of the system to reduce the space of feasible models.
  $\rightarrow$ **Dynamical systems**
- Develop methods to exhaustively parse or sample the space.
  $\rightarrow$ **Knowledge reasoning**

Extract robust information about the subsystems which deserve a specific dynamical study?

Several examples of applications

Introducing *dynamical background* in *knowledge reasoning* ?

Model correction and curation

- Dynamics: qualitative model of steady-state shifts
- Potential regulators of a cancer phenotype

Network inference

- Reconstruct metabolic network.
- Optimize the number of interactants.
- Identify regulatory modules in a network

Learning the boolean dynamical of a model

- Model steady-state of boolean networks.
- Derive controllers.

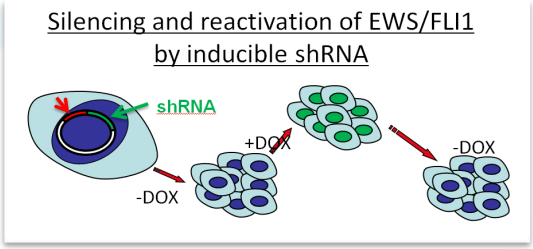
Learn hidden correlations from quantitative data

- Use ergodic invariants.
- Predict sensitive reactions

First case-study example

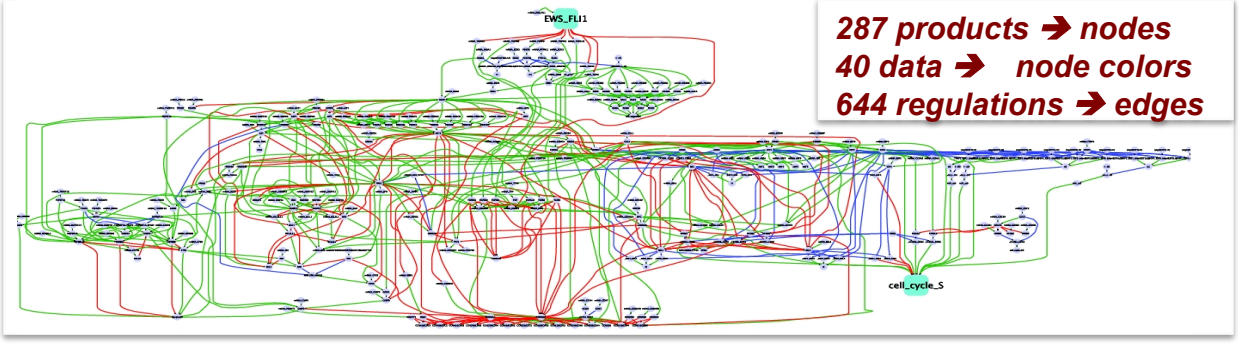
Ewing Sarcoma

- Chimeric protein
- Institut Curie. Inactivation of the protein expression.



Data *[Institut curie. Barillot & Delattre]*

- Litterature-based regulatory network
- Time-series genes expression after the protein inactivation

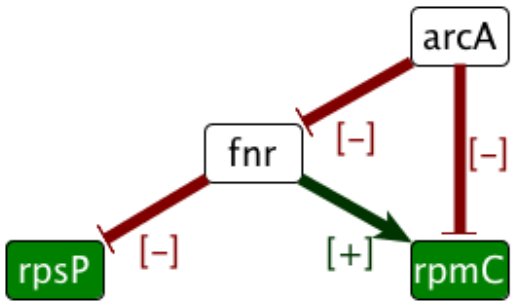


→ **What can be surely predicted from this information?**

Problem reformulation?

Knowledge representation

- Regulations. Signed oriented graph.
- Edge colors. Regulatory effects.
- Node colors. Expression data.



→ **What would we like to say about *fnr* ?**

Problem reformulation?

Dynamical systems *[Radulescu, Roy. Soc. Interface'06, Siegel, Biosystems'06]*

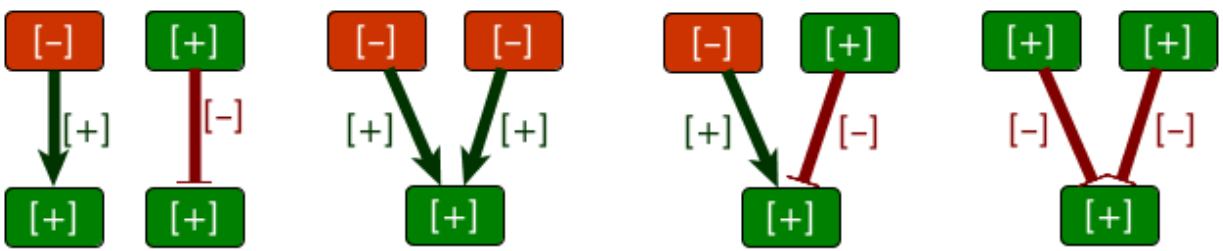
Theorem. *Assume that*

- *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:

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

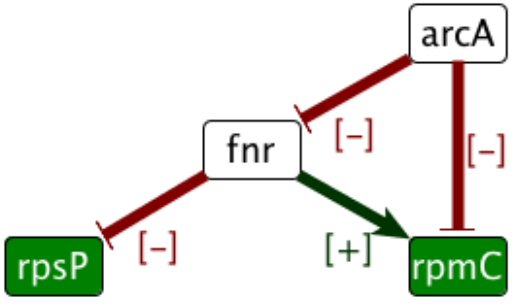
Take-home message **The expression of each target gene MUST BE explained by the consistent regulation of a source**

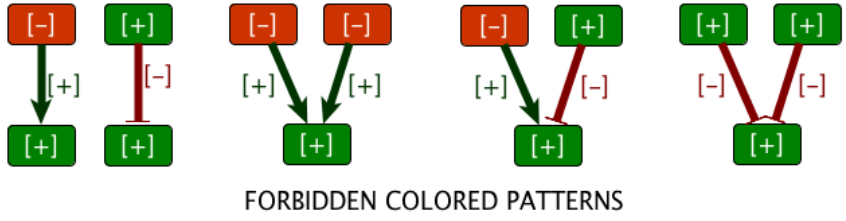


FORBIDDEN COLORED PATTERNS

Validation/Correction of (possibly inferred) networks

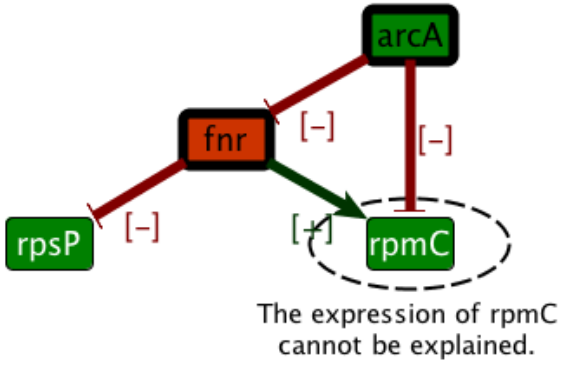
Constraint over graph-coloring Explain the expression of each target gene by the consistent regulation of a source.





Application *[Guziolowski-BMCGenomics'09, Gebser-KR'10]*

- Prediction. *rpsP* and *fnr* have fixed colors according to allowed patterns.
- Diagnosis. An extra forbidden pattern appears on *rpmC*.
- Correction. Also possible.



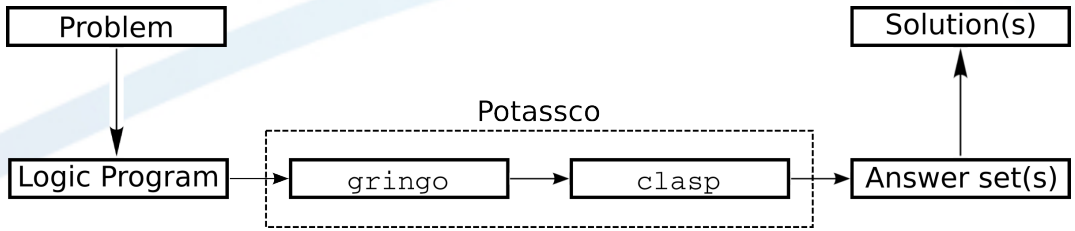
How can you solve this question efficiently ???

```
vertex(rpsP). vertex(fnr).
vertex (arcA). vertex(rpmC).
edge(fnr,rpsP). observedE(fnr,rpsP,-).
edge(fnr,rpmC). observedE(fnr,rpmC,+).
edge(arcA,fnr).  observedE(arcA,fnr,-).
edge(arcA,rpmC). observedE(arcA,rpmC,-).
observedV(rpsP,-).  observedV(rpmC,-).
labelV(I ,+) ; labelV (I ,-) :- vertex(I).
labelV(I ,S) :- observedV(I,S).
labelE(J,I,+) ; labelE (J,I,-) :- 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).
```

Natural question: what is this?

Answer Set Programming: *what?* instead of *how?*

- Declarative logical problem solving paradigm
- Knowledge representation and reasoning problems
- Combinatorial search problems in NP $\rightarrow$ *constraint satisfaction, diagnosis, repairing, planning...*



Potassco: **Potsdam** Answer Set Solving Collection

`http://potassco.sourceforge.net`

Rich modeling language

- Encoding problems as queries on propositional logical programs.
- *Gringo* grounder

Highly efficient inference engines

- Boolean constraint solving technology
- *Clasp* solver
- Competing with the power of SAT algorithms.

Programmation par ensembles-réponses: les bases

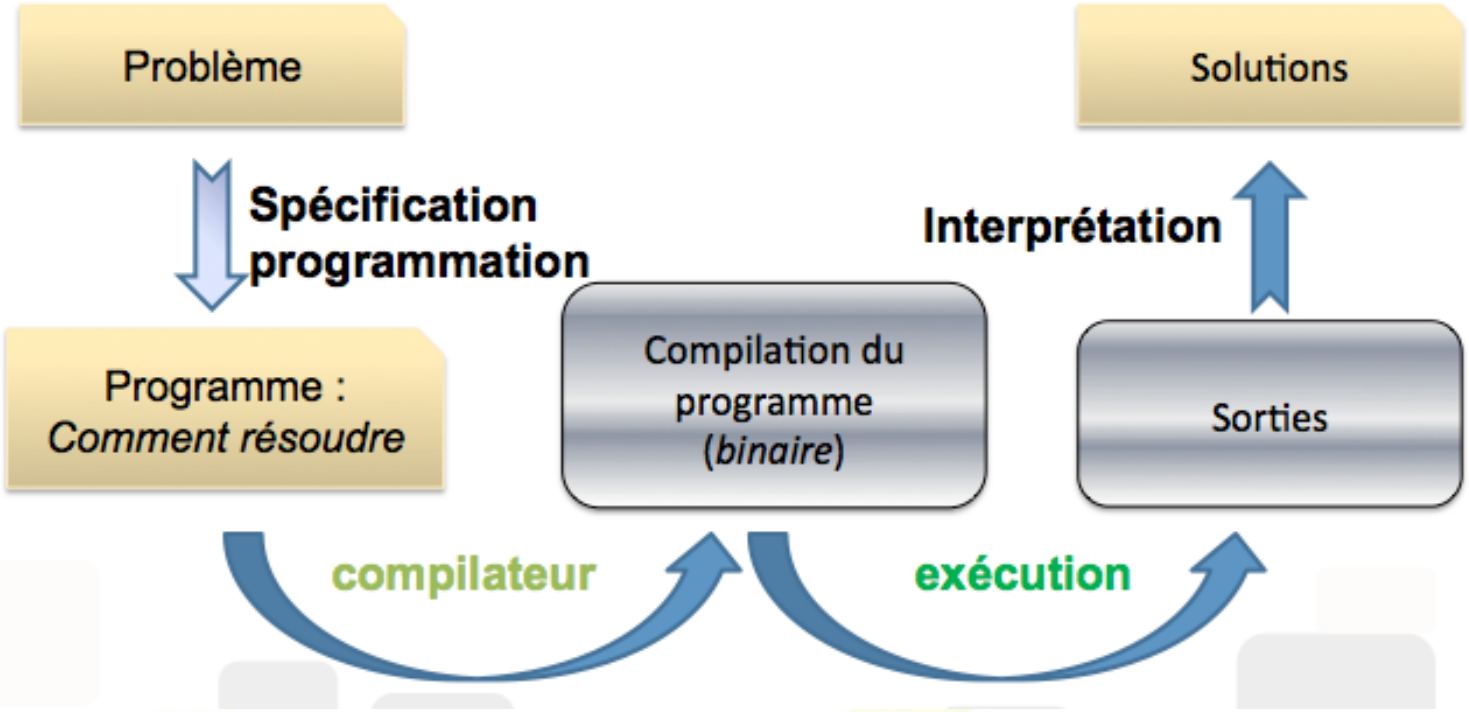
Principe généraux

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.

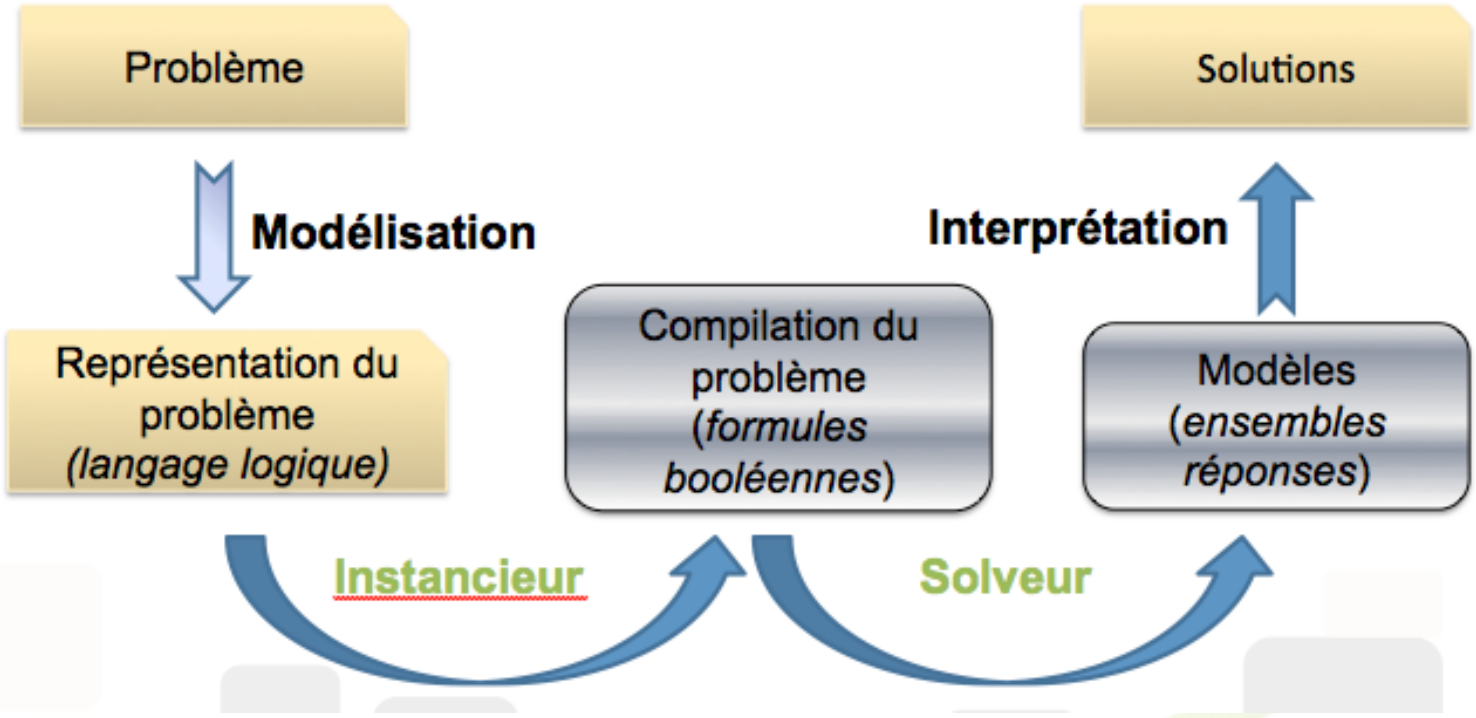
Principe Des solveurs associés effectuent la recherche d'une, de plusieurs, ou de l'ensemble des solutions.

Programmation usuelle



- Compilateur: version dans un langage de bas-niveau de l'algorithme.

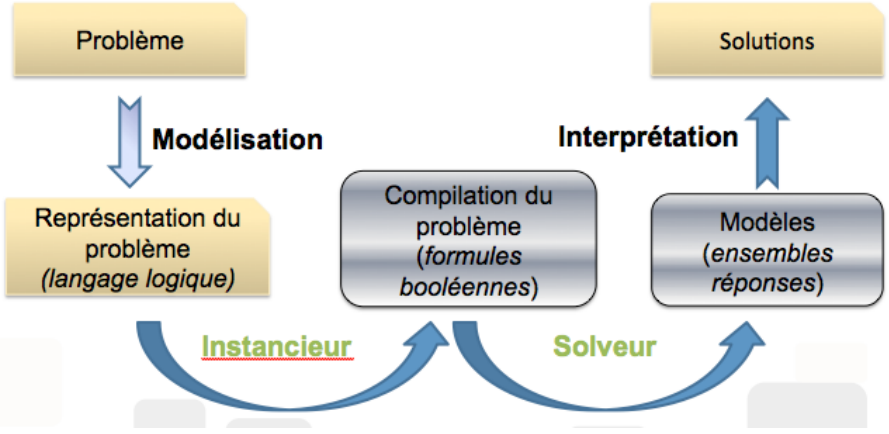
ASP



- L'instancieur prend la représentation du problème et le transforme en des formules booléennes de bas niveau.
  → remplacement des variables par des constantes.
  → Remplace le binaire.

- Le solveur remplace l'executeur.
  → résoud les contraintes

”Quoi” plutôt que ”comment”



- La spécification (**modélisation**) est fondamentale puisque c’est là qu’on décrit le programme.
- Le solveur résoud le problème plutôt que de faire un algorithme qui dit comment résoudre le problème.

On décrit des relations, pas des procédures !

Pourquoi maintenant ?

Il existe déjà des approches déclaratives (Prolog en particulier)...

Les spécificités d'ASP

- Négation (monde clos). Un prédicat est faux tant qu'aucune indication ne permet de dire qu'il est vrai.

- Expressivité. Logique propositionnelle: on ne peut travailler sur un ensemble fini d'atome.

  $\rightarrow$ Raisonnement sur l'infini "délicat".

- **Résolution**. Techniques novatrices avancées basées sur l'énumération (SAT, techniques de gestions de BDD déductives) alors que Prolog réalise une réécriture de programme.

  $\rightarrow$ L'ordre des clauses n'a pas d'impact sur la solution.

  $\rightarrow$ Totalement décidable: pas de "boucle infinie" dans la résolution.

- Optimisation avec différents critères.
- Raisonnement. Différents modes possibles (intersection, union, énumération).

Short description

Disjunctive rules

$$\underbrace{k\,\{\,a_1\,;\,\ldots\,;\,a_n\,\}\,l}_{\text{head}}\ \text{:-}\ \underbrace{a_{n+1},\ldots,a_r,\,not\ a_{r+1},\ldots,not\ a_s}_{\text{body}}$$

- Atoms. $a_1 \ldots a_n$ can be considered as facts.
- Deduction

Whenever all facts of the body are satisfied, between k and l facts in the head shall be true.

- Facts. ” $a.$ ” is always true.
- Integrity constraint. ” $:-a.$ ” is always ”false”.

Answer Set a at roughly glance (see the following for details)

- Set of atoms satisfying all logical rules
- Minimality and stability properties
- Every atom of an answer set appears in the head of at least one rule.

Who killed Mr Boddy ?

Program

```
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.

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

#hide.
#show murderer/1.  #show place_of_crime/1.  #show weapon_of_crime/1.
```

Answer Sets? *Exercise: do it yourself!*

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
#hide.
#show murderer/1.  #show place_of_crime/1.  #show weapon_of_crime/1.
```

Answer Sets? *Exercise: do it yourself!*

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
#hide.
#show murderer/1.  #show place_of_crime/1.  #show weapon_of_crime/1.
```

Answer Sets? *Exercise: do it yourself!*

```
weapon_of_crime(candlestick) murderer(colonel_Mustard) place_of_crime(hall)
weapon_of_crime(candlestick) murderer(ms_Scarlet) place_of_crime(dining_room)
   weapon_of_crime(candlestick) murderer(ms_Scarlet) place_of_crime(hall)
```

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

Syntaxe: Termes

Constante : entier, mot de {a-z, A-Z,0-9, _ }* débutant par une minuscule.

Variable : mot de {a-z,A-Z,0-9,_ }* débutant par une majuscule.

Fonction

- constante(terme,... terme)
- opérateurs infixes (a+b)
- notations de liste. `L=[a,b,c]=[a|U], U =[b,c], [c]=[c|[ ]]='.'(c,[])`

Litteral

- Négation `not q`
- Calculs

  `X<Y  U!=V  X+Y<5`
- Condition

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

  → sous-ensemble de taille 1 à 2 de l'ensemble des r(X), où X est dans le domaine défini par q.

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

  → *Exercise: do it yourself !*

Syntaxe: Termes

Constante : entier, mot de {a-z, A-Z,0-9, _ }* débutant par une minuscule.

Variable : mot de {a-z,A-Z,0-9,_ }* débutant par une majuscule.

Fonction

- constante(terme,... terme)
- opérateurs infixes (a+b)
- notations de liste. `L=[a,b,c]=[a|U], U =[b,c], [c]=[c|[ ]]='.'(c,[])`

Litteral

- Négation `not q`
- Calculs

  `X<Y  U!=V  X+Y<5`
- Condition

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

  → sous-ensemble de taille 1 à 2 de l'ensemble des r(X), où X est dans le domaine défini par q.

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

  → *Exercise: do it yourself !* 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).`
→ "p(X) et q(X)" est faux.

`:- not {r(U+1):q(U)}M, max(M).`
→ *Exercise: do it yourself !*

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).`
→ "p(X) et q(X)" est faux.

`:- not {r(U+1):q(U)}M, max(M).`
→ *Exercise: do it yourself !*Quelle que soit la valeur de M dans son domaine défini par max, il y a au plus M valeurs de U dans son domaine défini par q telles que r(U+1) est vérifié.

Syntaxe: Clauses

Optimisation

```
#minimize [ t(X):s(X)=poids(X) ].
```

→ chercher à minimiser au sens de leur poids global l'ensemble d'éléments pondérés constitué des `t(X)`, chacun de poids X, où X varie dans le domaine de variation défini par s.

```
#maximize [ p=1, q(X,Y):X>0:Y<0=2].
```

→ *Exercise: do it yourself !*

Mutli-optimisation

```
#maximize [ p=1@2, q(X,Y):X>0:Y<0=2@2].
#minimize [ t(X):s(X)=X-1@1].
```

→ On cherche d'abord à maximiser (priorité @2) avant de minimiser (priorité @1 $<$ @2).

Sorties

```
#hide.
#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)`.

Syntaxe: Clauses

Optimisation

```
#minimize [ t(X):s(X)=poids(X) ].
```

→ chercher à minimiser au sens de leur poids global l'ensemble d'éléments pondérés constitué des `t(X)`, chacun de poids X, où X varie dans le domaine de variation défini par s.

```
#maximize [ p=1, q(X,Y):X>0:Y<0=2].
```

→ *Exercise: do it yourself !* maximiser au sens de leur poids global la présence de l'ensemble d'éléments pondérés constitué de `p` de poids 1 et de tous les `q(X,Y)` de poids 2 où X et Y varient dans leur domaines de variation définis par ailleurs dans les clauses définissant q mais avec la restriction supplémentaire que X soit strictement positif et Y strictement négatif.

Mutli-optimisation

```
#maximize [ p=1@2, q(X,Y):X>0:Y<0=2@2].
#minimize [ t(X):s(X)=X-1@1].
```

→ On cherche d'abord à maximiser (priorité @2) avant de minimiser (priorité @1 $<$ @2).

Sorties

```
#hide.
#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)`.

Utiliser potasco

Sources

- Installation : http://potassco.sourceforge.net/
- Version binaire linux/windows/mac : http://www.cs.uni-potsdam.de/ sthiele/bioasp/downloads/bin/

Appel

- `gringo cluedo.lp -t` Pour voir le modèle instancié généré (suppression des variables)
- `gringo input.lp model.lp -c chain_size=28 | clasp -stats` 1 solution
- `clingo cluedo.lp -n 10` 10 solutions
- `gringo cluedo.lp | clasp -n 0` Toutes les solutions

Utilisation du grounder

Le programme suivant ne fait pas appel à l'instancieur : pas de variables

```
% 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
#hide.  #show murderer/1.  #show place_of_crime/1.  #show
weapon_of_crime/1.
```

Ce programme n'est pas "canonique".

Meilleur programme

Utilisation de 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 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.
```

Exercice

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 ?

→ *Exercise: do it yourself !*

Caractéristiques d'une solution `sol(Taxi, Client)` doit être tel qu'il y a 1 taxi par client et 1 client par taxi

→ *Exercise: do it yourself !*

Optimisation Minimiser le coût d'affectation ?

→ *Exercise: do it yourself !*

Exercice

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 ?

→ *Exercise: do it yourself !* `agent(T):- cout(T,_,_).  client(T):- cout(_,T,_).`

Caractéristiques d'une solution `sol(Taxi, Client)` doit être tel qu'il y a 1 taxi par client et 1 client par taxi

→ *Exercise: do it yourself !*

Optimisation Minimiser le coût d'affectation ?

→ *Exercise: do it yourself !*

Exercice

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 ?

→ *Exercise: do it yourself !* `agent(T):- cout(T,_,_).  client(T):- cout(_,T,_).`

Caractéristiques d'une solution `sol(Taxi, Client)` doit être tel qu'il y a 1 taxi par client et 1 client par taxi

→ *Exercise: do it yourself !*

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

Optimisation Minimiser le coût d'affectation ?

→ *Exercise: do it yourself !*

Exercice

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 ?

→ *Exercise: do it yourself !* `agent(T):- cout(T,_,_).  client(T):- cout(_,T,_).`

Caractéristiques d'une solution `sol(Taxi, Client)` doit être tel qu'il y a 1 taxi par client et 1 client par taxi

→ *Exercise: do it yourself !*

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

Optimisation Minimiser le coût d'affectation ?

→ *Exercise: do it yourself !*

```
#minimize[sol(Taxi, Client):cout(Taxi, Client, Temps)=Temps].
```

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

Classification

Données

```
ville(brest;rennes;nantes;paris).
```

Optimisation Quelle est la plus grande ville de France ?

```
plusgrandeville(V):- ville(V), U<=V:ville(U).
```

→ la condition `U<V` utilise la longueur des mots et l'ordre lexicographique !!

Deuxième question Quelle est la plus petite ville de France ?

→ *Exercise: do it yourself !*

Nouvelles données

```
population(brest,20000).  population(rennes,300000).
population(paris,10000000).  population(nantes,500000).
```

Dernière question Quelle est la plus grande ville de France en terme de population ?

→ *Exercise: do it yourself !*

Classification

Données

```
ville(brest;rennes;nantes;paris).
```

Optimisation Quelle est la plus grande ville de France ?

```
plusgrandeville(V):- ville(V), U<=V:ville(U).
```

→ la condition `U<V` utilise la longueur des mots et l'ordre lexicographique !!

Deuxième question Quelle est la plus petite ville de France ?

→ *Exercise: do it yourself !*

```
pluspetiteville(V):- ville(V), not ville(U):ville(U):U<V.
```

Nouvelles données

```
population(brest,20000).  population(rennes,300000).
population(paris,10000000).  population(nantes,500000).
```

Dernière question Quelle est la plus grande ville de France en terme de population ?

→ *Exercise: do it yourself !*

Classification

Données

```
ville(brest;rennes;nantes;paris).
```

Optimisation Quelle est la plus grande ville de France ?

```
plusgrandeville(V):- ville(V), U<=V:ville(U).
```

→ la condition `U<V` utilise la longueur des mots et l'ordre lexicographique !!

Deuxième question Quelle est la plus petite ville de France ?

→ *Exercise: do it yourself !*

```
pluspetiteville(V):- ville(V), not ville(U):ville(U):U<V.
```

Nouvelles données

```
population(brest,20000).  population(rennes,300000).
population(paris,10000000).  population(nantes,500000).
```

Dernière question Quelle est la plus grande ville de France en terme de population ?

→ *Exercise: do it yourself !*

```
plusgrandepopulation(V):- population(V,W),
                                    not
population(X,Y):population(X,Y):W<Y.
```

ASP est utile dans quel cas ?

Résolution des problèmes combinatoires: problèmes NP-complets et NP-difficiles (graphes, raisonnement, ...)

- Cadre unifié où sont intégrés des aspects bases de données, bases de connaissances, résolution de contraintes et programmation logique.
- Déclarativité Exprimer facilement des propriétés mathématiques sur un espace de recherche : plus il y a de propriétés, mieux ça marchera !
- Optimisation
- Différents modes de raisonnement: une solution, toutes, les meilleures, leur intersection ou leur union.
- Facilité pour mettre au point et tester différents modèles et différentes heuristiques.

Limites

Limite

- On n'échappe pas à la complexité mais on tire parti des particularités d'une instance de problème à résoudre.
- Difficile de tracer un programme ASP pour savoir où se cache la partie complexe.
  $\rightarrow$ Procéder par étapes : découpage modulaire en clauses indépendantes.
  $\rightarrow$ Prototype de profileur de l'exécution d'un programme.

A ne pas utiliser...

- Pour remplacer des algorithmes classiques maîtrisés (tri, calcul traitement de séquences)
  $\rightarrow$ Le langage de script Lua est inclus dans gringo pour les pré et post-traitements;
- travailler dans des espaces peu structurés et peu contraints
- travailler dans des espaces difficiles à discrétiser.
- résoudre un problème qui nécessiterait de nombreuses étapes successives qui communiquent
  $\rightarrow$ un programme ASP ne permet pas d'écrire des procédures.

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)

$$\underbrace{A}_{tete} :- \underbrace{B_1, B_2, \ldots B_m}_{corps}$$

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 $\neg B_1 \vee \neg B_2 \vee \neg \vee \neg B_m \vee A$

$\rightarrow$ Les programmes positifs sont des clauses disjonctives avec exactement un atome positif. (clauses définies).

Ensemble clos d'un programme positif
Un ensemble d'atomes X est *clos pour un programme positif* si pour l'ensemble de ses règles r

$$\text{corps}(r) \subset X \implies \text{tête}(r) \subset X$$

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

$$\text{corps}(r) \subset X \implies \text{tête}(r) \subset X$$

Programme

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

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}, $\emptyset$.

Sous-ensembles clos

$\rightarrow$ *Exercise: do it yourself !*

Ensemble-réponse?

Exemple 1

$$\text{corps}(r) \subset X \implies \text{tête}(r) \subset X$$

Programme

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

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}, $\emptyset$.

Sous-ensembles clos

$\rightarrow$ *Exercise: do it yourself !*

{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}, $\emptyset$.

Ensemble-réponse?

Exemple 1

$$\text{corps}(r) \subset X \implies \text{tête}(r) \subset X$$

Programme

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

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}, $\emptyset$.

Sous-ensembles clos

$\rightarrow$ *Exercise: do it yourself !*

{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}, $\emptyset$.

Ensemble-réponse? $\emptyset$

$\rightarrow$ 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

$$\text{corps}(r) \subset X \implies \text{tête}(r) \subset X$$

Programme

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

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}, $\emptyset$.

Sous-ensembles clos

Ensemble-réponse?

Exemple 2

$$\text{corps}(r) \subset X \implies \text{tête}(r) \subset X$$

Programme

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

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

{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.

Ensemble-réponse? {**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 NOT?

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

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

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

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

Calcul

- 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.`

- Réduit par rapport à { `q` } ? *Exercise: do it yourself !*

- Réduit par rapport à { `a` } ? *Exercise: do it yourself !*

Programmes avec NOT?

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

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

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

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

Calcul

- 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.`

- Réduit par rapport à { `q` } ? *Exercise: do it yourself !*

```
b:- p, q.
p.
```

- Réduit par rapport à { `a` } ? *Exercise: do it yourself !*

```
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 minimal pour l'inclusion.

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: ?
- {q}. Réduit: ?
- ∅. Réduit: ?

Pour chaque réduit, calculer l'ensemble clos minimal

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

Garder les clos minimaux
Ensembles-réponses ?

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: Ensemble clos minimal?
- ∅. Réduit: Ensemble clos minimal?

Garder les clos minimaux
Ensembles-réponses ?

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 minimaux
Ensembles-réponses ?

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 minimaux
Ensembles-réponses ? {q}

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}. →
- {p}. →
- {q}. →
- ∅. Réduit: ?

Pour chaque réduit, calculer l'ensemble clos minimal

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

Garder les clos minimaux
Ensembles-réponses ?

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

Garder les clos minimaux
Ensembles-réponses ? {p}, {q}

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}. → ∅
- {p}. → p.
- {q}. → 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}. →
- ∅. →

Pour chaque réduit, calculer l'ensemble clos minimal

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

Garder les clos minimaux
Ensembles-réponses ?

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?

Garder les clos minimaux
Ensembles-réponses ? Aucun !!!

Exemple 3

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

Enumérer les sous-ensembles X d'atomes.
$\{p\}$, $\emptyset$.
Pour chacun, calculer le réduit.

- $\{p,q\}$. $\rightarrow \emptyset$
- $\emptyset$. $\rightarrow$ `p.`

Pour chaque réduit, calculer l'ensemble clos minimal

- $\{p,q\}$. Réduit: programme vide . Ensemble clos minimal? rien
- $\emptyset$. Réduit: `p.` Ensemble clos minimal? $\{p\}$

Garder les clos minimaux
Ensembles-réponses ? Aucun !!!

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.

Catalogue plus avancé...

Avec différents éléments du langage, on a des règles de remplacement en introduisant éventuellement des atomes supplémentaires

Variables On instancie les variables en réduisant au mieux le nombre d'instances produites (*gringo*).

Contraintes d'intégrité On remplace "`:- p.`" par "`atomesup:- p, not atomesup.` "

Contraintes ensemblistes On remplace "`{a,b}:- p.`" par

```
atomesup:- p.
a:- atomesup, not atomesup_a.  atomesup_a :- atomesup, not a
b:- atomesup, not atomesup_b.  atomesup_b :- atomesup, not b
```

Contraintes de cardinalité D'autres règles de remplacement.

Le piège...

Explosion lors de l'instanciation

- ASP peut gérer des programmes avec quelques millions d'atomes.
- MAIS : ce nombre est vite atteint !
- Limiter au maximum le nombre de variables différentes dans une clause !

Compromis grounding/solving

- Gringo (instancieur) dépense de la mémoire. Précalcule (tabule) tout ce qui peut être instancié à l'avance de manière déterministe.
- Clasp (solveur) dépense du CPU, il effectue les choix et élague dynamiquement l'espace de recherche.
- Il faut faire en sorte que le solveur apprenne de nouvelles contraintes en cours de résolution.

Le choix du solveur

Dépend du problème considéré !

clasp: Calcul des ensemble-réponses minimaux en taille.

unclasp: Autre heuristique pour calculer des ensembles-réponses minimaux en taille $\rightarrow$ petits sous-ensembles d'un espace de grande taille.

claspD(isjunctive) utilise des metaprogrammes additionnels pour calculer des ensembles-réponses minimaux pour l'inclusion ensembliste.

hclasp utilise des prédicats supplémentaires pour guider la rechercher et réaliser des "mémorisations de solutions" pour calculer des ensembles minimaux pour l'inclusion ensembliste.

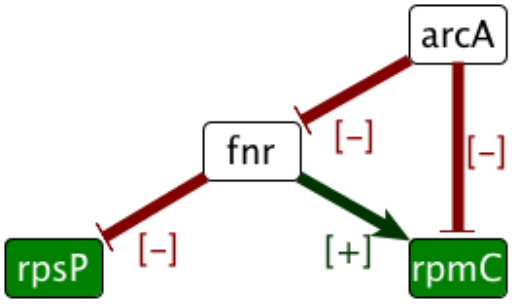
clingo(gringo+clasp) Utilisation "tout en un".

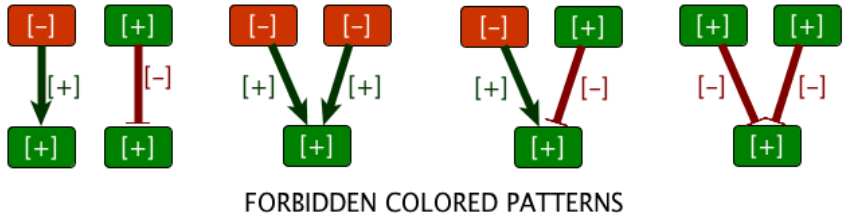
iclingo Solveur incrémental : (base + paramètre incrémentant d'un pas de résolution à l'autre)

clingcon Solveur de contraintes sur entiers

Jouons un peu...

Constraint over graph-coloring Explain the expression of each target gene by the consistent regulation of a 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.

Chaque sommet a exactement une prédiction `labelV`, qui est compatible avec les observations si elles existent.

Chaque sommet peut se voir prédire exactement un signe

Les observations permettent de spécifier des signes

une prédiction ne peut pas être différente d'une observation de sommet

Suite

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

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

Chaque sommet a exactement une prédiction `labelV`, qui est compatible avec les observations si elles existent.

```
1{labelV(I,S):signe(S)}1 :- vertex(I).
labelV(I,S) :- observedV(I,S).
```

Chaque sommet peut se voir prédire exactement un signe

```
1{receive(I,S):signe(S)}1 :- vertex(I).
```

Les observations permettent de spécifier des signes

```
receive(I,up) :- observedE(J,I,S), labelV(J,S).
receive(I,down) :- observedE(J,I,S), labelV(J,T), S!=T.
```

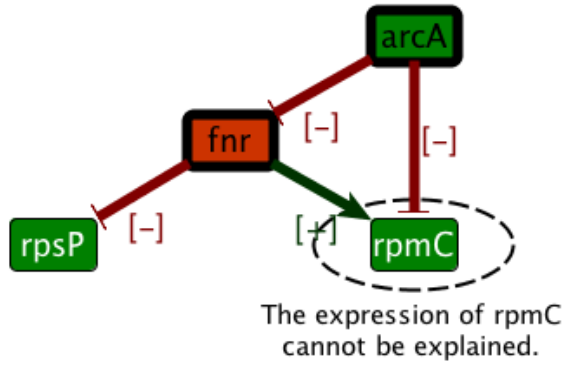
une prédiction ne peut pas être différente d'une observation de sommet

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

Extensions

Application *[Guziolowski-BMCGenomics'09, Gebser-KR'10]*

- Prediction. *rpsP* and *fnr* have fixed colors according to allowed patterns.
- Diagnosis. An extra forbidden pattern appears on *rpmC*. *[Guziolowski-BMCGenomics'09]*
- Repair. Also possible with intersection mode. *[Gebser-KR'10]*
- Sign inference. *[Veber-BMC bioinfo'2008]*



Availability

- Encodings http://www.cs.uni-potsdam.de/bioasp/sign_consistency.html
- Python http://pypi.python.org/pypi/ingranalyze
- Online http://mobyle.genouest.org/cgi-bin/Mobyle/portal.py#forms::bioasp-check_inflgraph

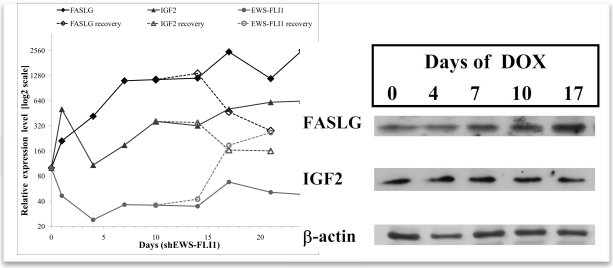
Qu'est ce qui fait que ca marche ?

- l'expressivité du langage
- le mode de raisonnement par intersection

Cancer application

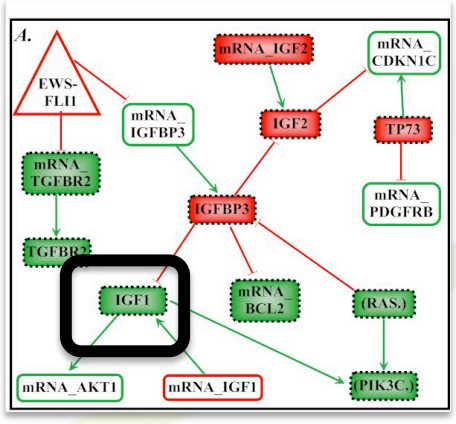
Explain and predict

- Effect of multi-scale competitions.
- Validation of predictions.



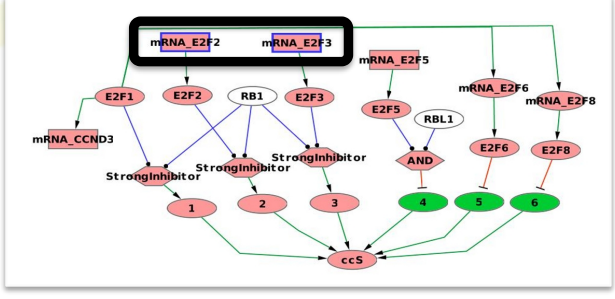
Key pathways *[Baumuratova-BMC syst. bio'10]*

- Missing regulations over IGF1



Design? *[Guziolowski TCBB'11]*

- Two new possible targets for EWI-FLI1
- si-RNA confirmation (unpublished)



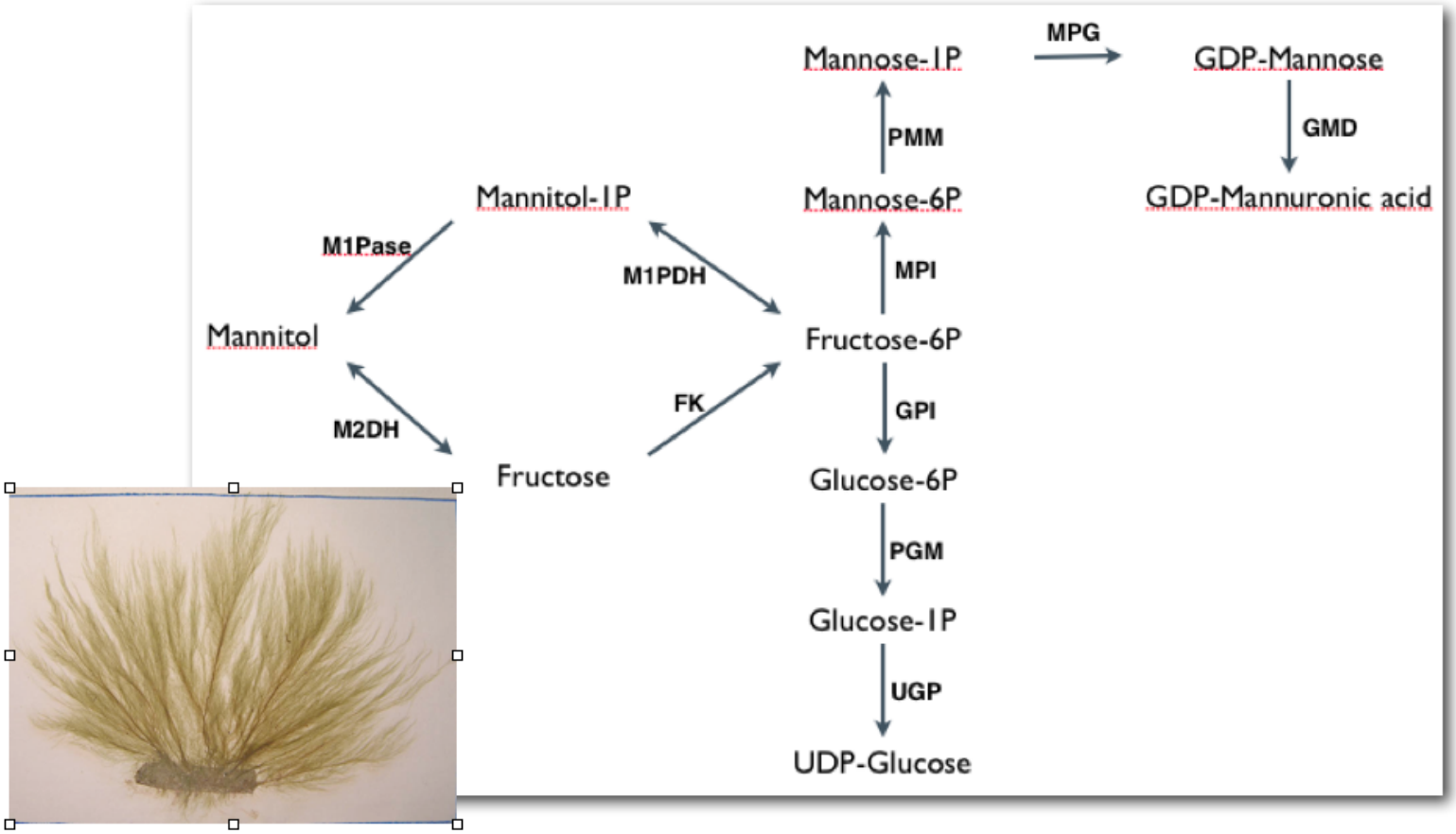
Scalability ?

Can this approach of reasoning over data and (dynamical) rules be useful in larger scale applications ?

- topological study of metabolic networks.
- identification of regulatory modules.
- pruning network after their inference.

First question: case-study

How to reconstruct a metabolic network for non-model species ?



Brown algae: manitol cycle

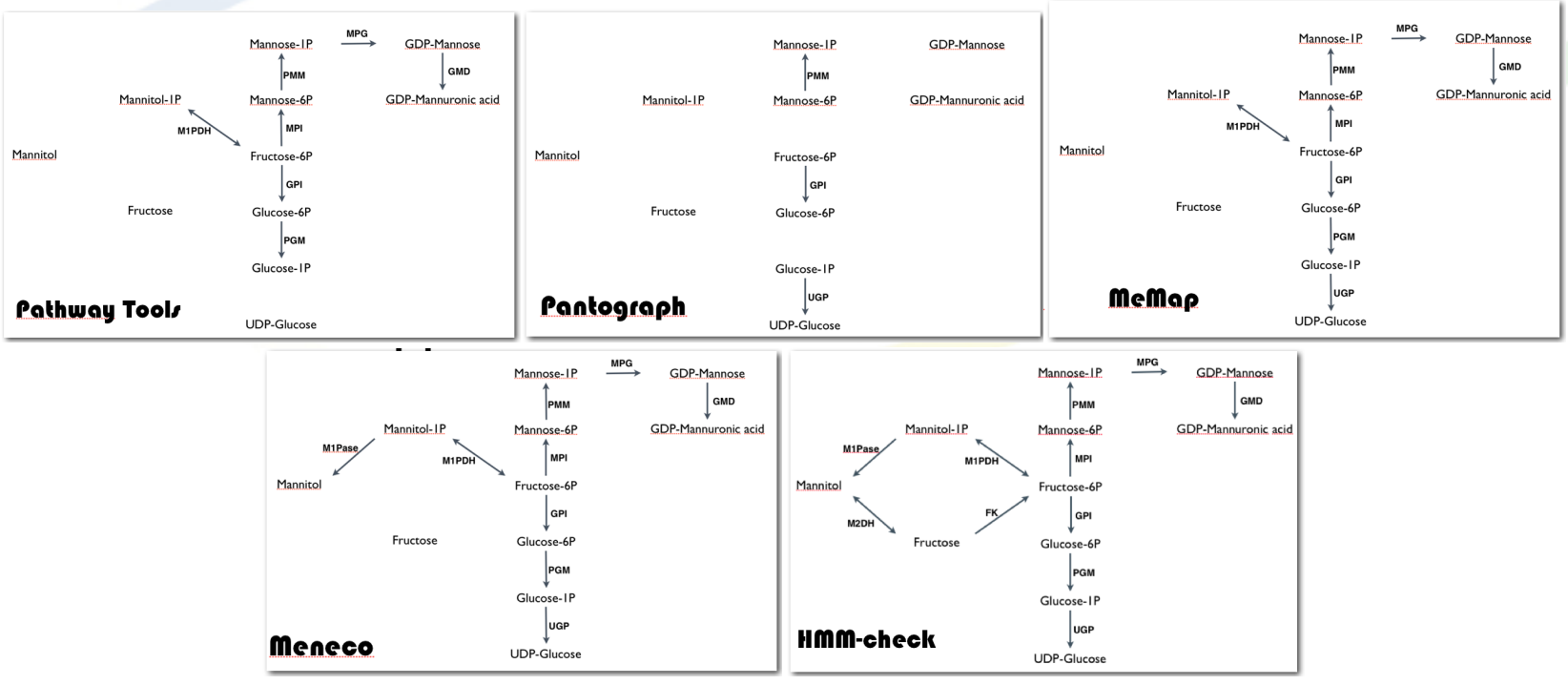
How to reconstruct a network ?

Use many data sources

- Gene sequences
- Protein sequences
- Gene annotations
- Reaction databanks
- Metabolic evidences

Many steps for the integration...

- Mapping gene annotation to reactions
- Use sequence alignments
- **Network completion**
- Genome post-validation
- Functional validation



The key combinatorial point: network completion

Metabolic functionality ?

- $\simeq$ 1800 reactions, $\simeq$ 2000 metabolites.
- Among 51 targets, only half are producible before completion.
-

Combinatorial problem: find all the sets of reactions with a minimal size that restore the connectivity between the seed and the targets.

The key point: network completion

ASP solves the problem *[Collet&Prigent&Thiele, LPNMR'2013]*

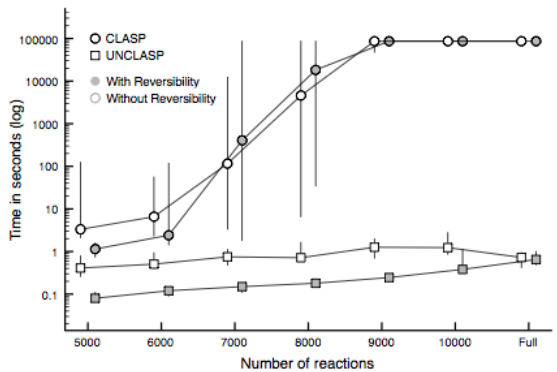
- Knowledge representation: reversible reaction.
- Compute the minimum of interactions in network completion. Heuristics (*unclasp*).
- Enumerate the intersection of all solution.
- Study the combinatorics.

$\rightarrow$ about 10 reactions are necessary to restore the connectivity of the network but have no enzyme candidates on the genome...*[Prigent, work in progress]*

Focus on characteristic pathways/actors of the metabolic response...

Listing 1.2. Ground logic program instance with reversiblity.

```
seed(H2). seed(O). target(H2O).

{ reaction(r) }. reversible(r).
reactant(H2,r). reactant(O,r).
product(H2O,r).

scope(H2) :- seed(H2).
scope(O) :- seed(O).

scope(H2O) :- product(H2O,r), reaction(r),
              scope(H2), scope(O), reactant(H2,r), reactant(O,r).

scope(H2) :- reactant(H2,r), reaction(r), reversible(r),
              scope(H2O), product(H2O,r).

scope(O) :- reactant(O,r), reaction(r), reversible(r),
              scope(H2O), product(H2O,r).

:- target(H2O), not scope(H2O).

#minimize[ reaction(r) ].
```

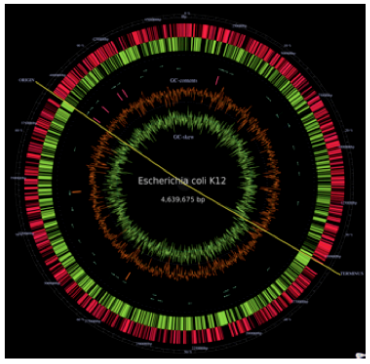


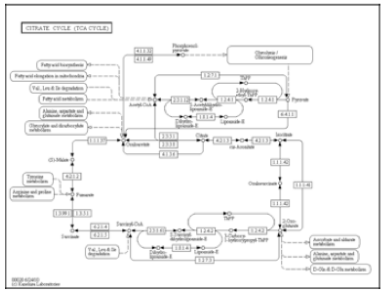
Availability

- Encodings http://www.cs.uni-potsdam.de/bioasp/expansion.html
- Python http://pypi.python.org/pypi/meneco
- Online http://mobyle.genouest.org/cgi-bin/Mobyle/portal.py#forms::bioasp-network_expansion

Further development in integrative biology

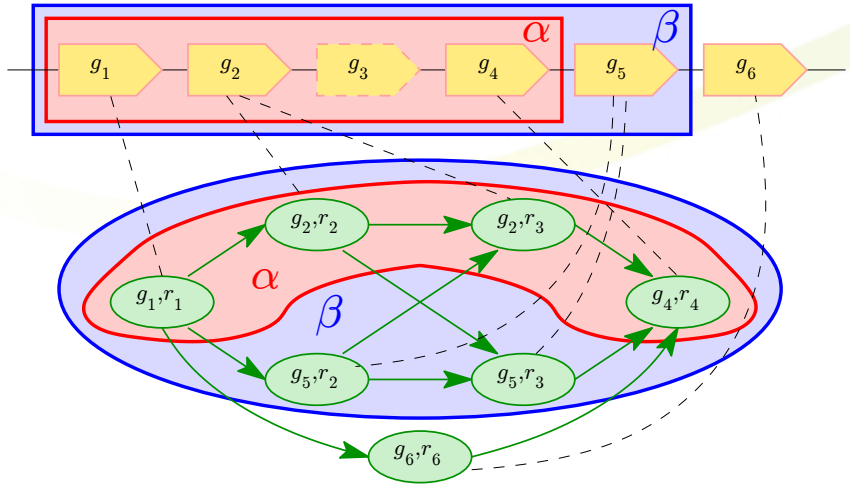
Combine genomic and metabolic information ?





Dynamics What could be a functional module ?

Portions of genomes with a high density of gene coding for enzymes that are *close* in a metabolic pathway



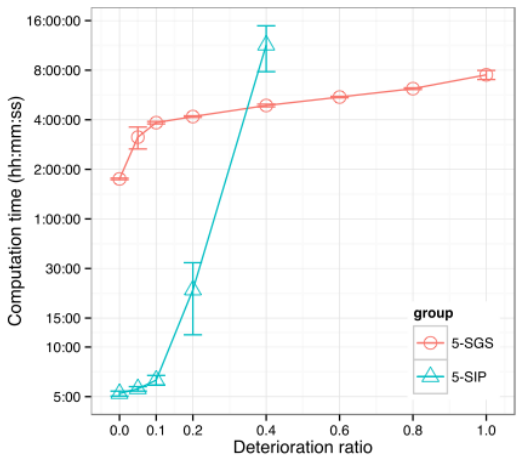
Enumeration

ASP-based approach *[Bourdon, Thiele et al , LPNMR'13]*

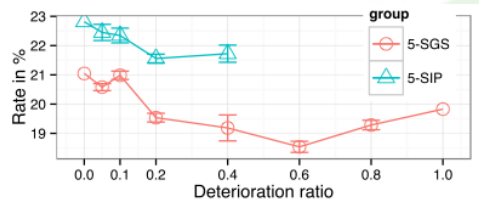
- Step 0: Model shortest genome segments
- Step 1: computing the minimum length of a segment that can catalyze the desired metabolic pathway.
- Step 2: enumerating all solution from the minimum length to a fixed maximum length.

Listing 1.1. sgs.lp: ASP encoding of shortest genome segments.

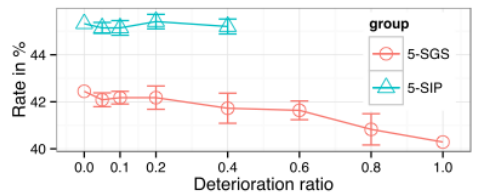
```
sgene(G) :- start(R),cat(G,R).
egene(G) :- end(R),cat(G,R).

pse(F,L) :- gene(F;L), F<L,
            (L-F)+1 <= max, (L-F)+1 >= min,
            sgene(S), F > S-max, L < S+max,
            S >=F, S <= L,
            egene(E), F > E-max, L < E+max,
            E >=F, E <= L.

1{ se(F,L) : pse(F,L) }1.

on_segment(G) :- se(F,L), gene(G), G>=F, G<=L.

aedge(X,Y) :- edge(X,Y), cat(G1,X), cat(G2,Y),
              on_segment(G1;G2).

from_start(X) :- start(X), on_segment(G), cat(G,X).
from_start(Y) :- from_start(X), aedge(X,Y).
:- not from_start(X), end(X).

to_end(Y) :- end(Y), on_segment(G), cat(G,Y).
to_end(X) :- to_end(Y), aedge(X,Y), cat(G,X).

aunit(G) :- on_segment(G), cat(G,X), from_start(X),to_end(X).
:- se(F,L), not aunit(F).
:- se(F,L), not aunit(L).

length((L-F)+1) :- F<=L, se(F,L).

:- length(X), X < min.

#minimize [ length(L) = L ].
```



(a) Computation time



(b) Operonic relevance



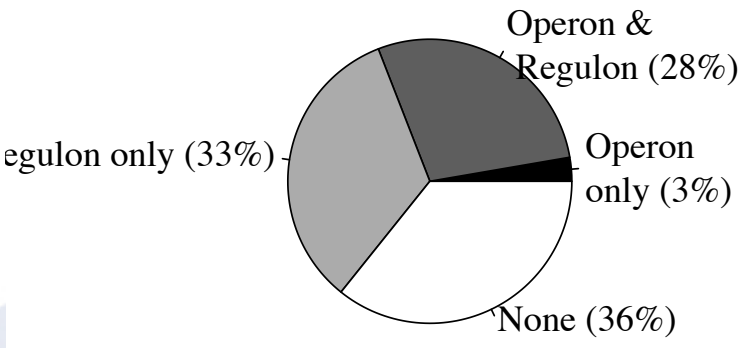
(c) Operonic justification

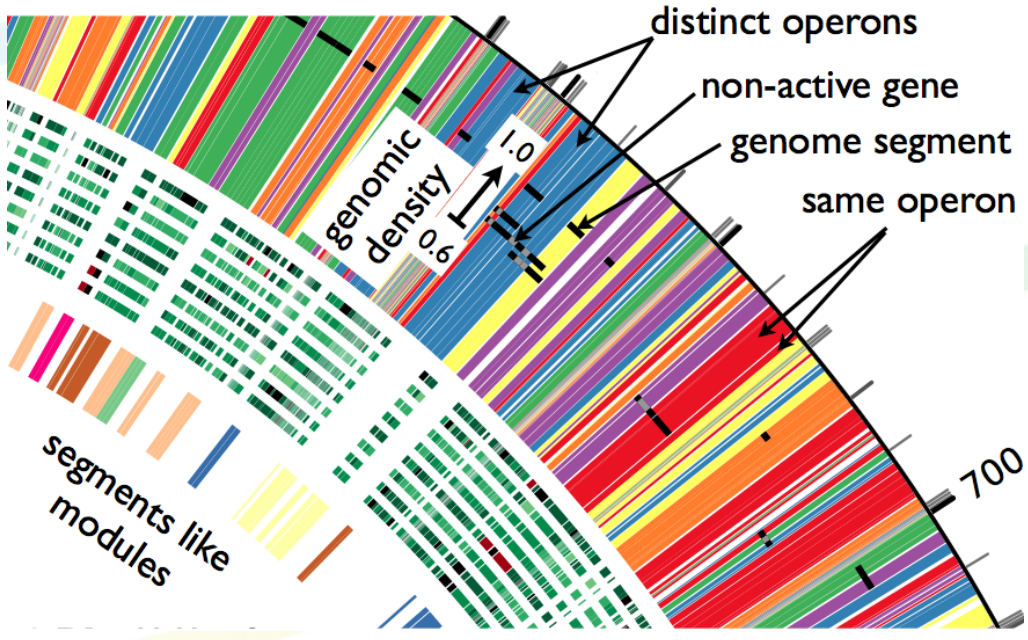
Good computational performances: the complete set of modules can be exhaustively computed

Validation

ASP-based approach *[Bourdon, work in progress]*

- Validation with lists of operons and regulons.
- 90% have a GO enrichment with a good p-value.





Color: union of modules sharing co-expressed genes

Prospects

- Prediction of functional modules for exotic bateria $\rightarrow$ validation of functional modules.
- Elucidate regulatory relationships between functional modules?

Relationship with network inference?

Regulatory network inference in few words Identify the main actors and functions involved in the response of a system.

State-of-the-art methods

- Data-mining. Statistics. Machine Learning...
- Metaheuristics. Search for a local optimal (genetic algorithms...).
- Optimization. Look for best-score solution (ILP).

Weakness: most approaches are discriminative:
their output is a "single" most-probable solution.

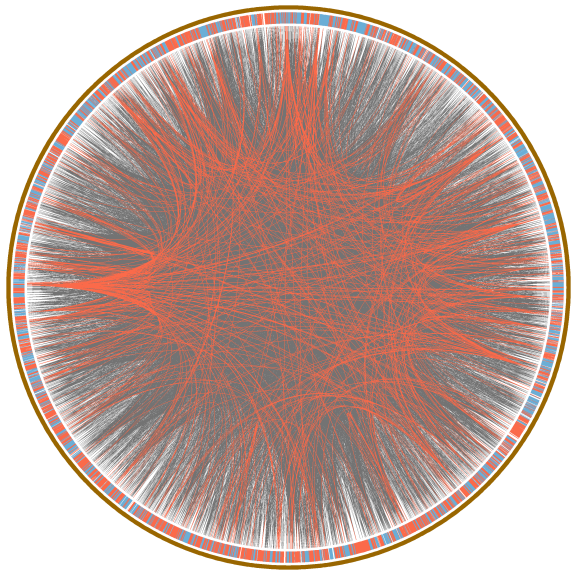
Limitations Uncertainties appear at different stages of the identification process

- **Confidence** in the resulting predictions?
- Relevance of a **unique** solution?

Case-study: E. coli regulatory network

- Data Regulatory evidences between genes (TF motif, Prodoric, Regulon DB).
- Combine genes when they belong to the same operon.
  $\rightarrow$ direct regulations between operons.

- Analysis
- Very few regulations have biological evidence: false-positive.



How can we use reasoning of expression data to reduce false-positive?

Idea: **explain** mutual information with **few** indirect regulations.

Reasoning over the graph

- Data 2: expression data $\rightarrow$ Mutual information.
  $\rightarrow$ pairs of genes sharing mutual information.
- Dynamical rule **Each pair of gene with mutual information should be explained by an indirect common regulator.**
- Implementation Select interactions appearing in **shortest explanatory paths** for pairs of mutual information.

  $\rightarrow$ Good prediction score + relatively short path

The trick: strong need for a discretized version of the problem

Difficult combinatorial problem: ASP

Generate all minimal explanations for each pair of gene sharing mutual information

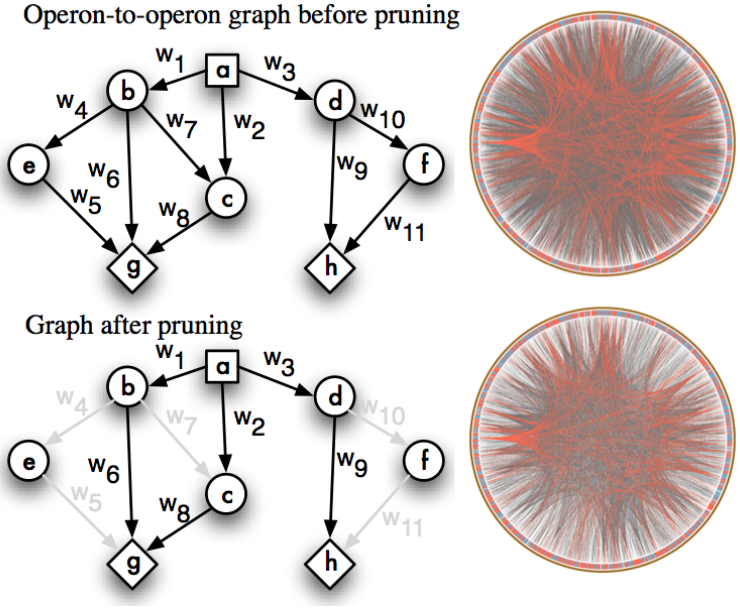
- Optimization.
- Union reasoning mode.
- Specific heuristics: *unclasp* solver.

For each explanatory node, find the nodes in explanatory pathways

- Grounding.

Results *[Aravena, work in progress]*

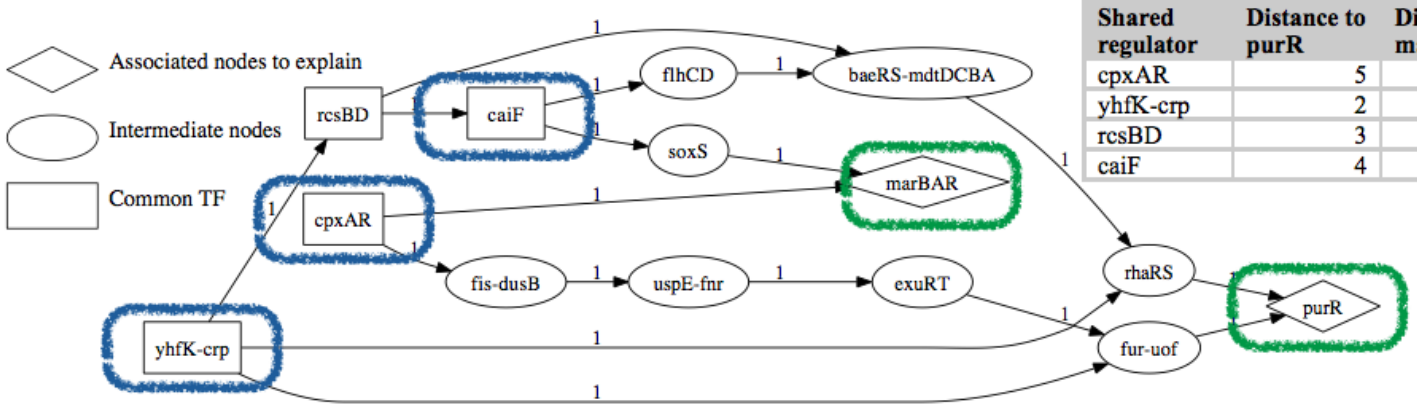
- ASP can enumerate all the solutions.
- The pruning removes false-positive interactions from the inferred network while keeping true-positive



Deduce regulators....

- modeling regulatory relationship with discrete rules
- using efficient solvers

→ **elucidating causalities within correlations and propose regulators with higher confidence**.

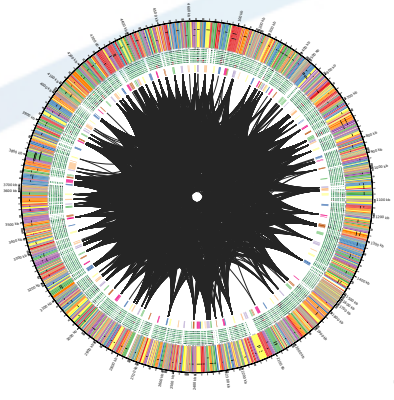


Shared regulator	Distance to purR	Distance to marBAR
cpxAR	5	1
yhfK-crp	2	4
rcsBD	3	3
caiF	4	2

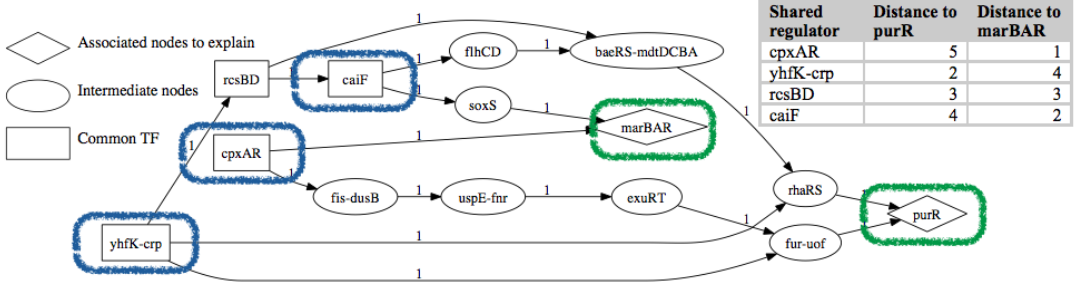
To do next

Combine everything....

+



+

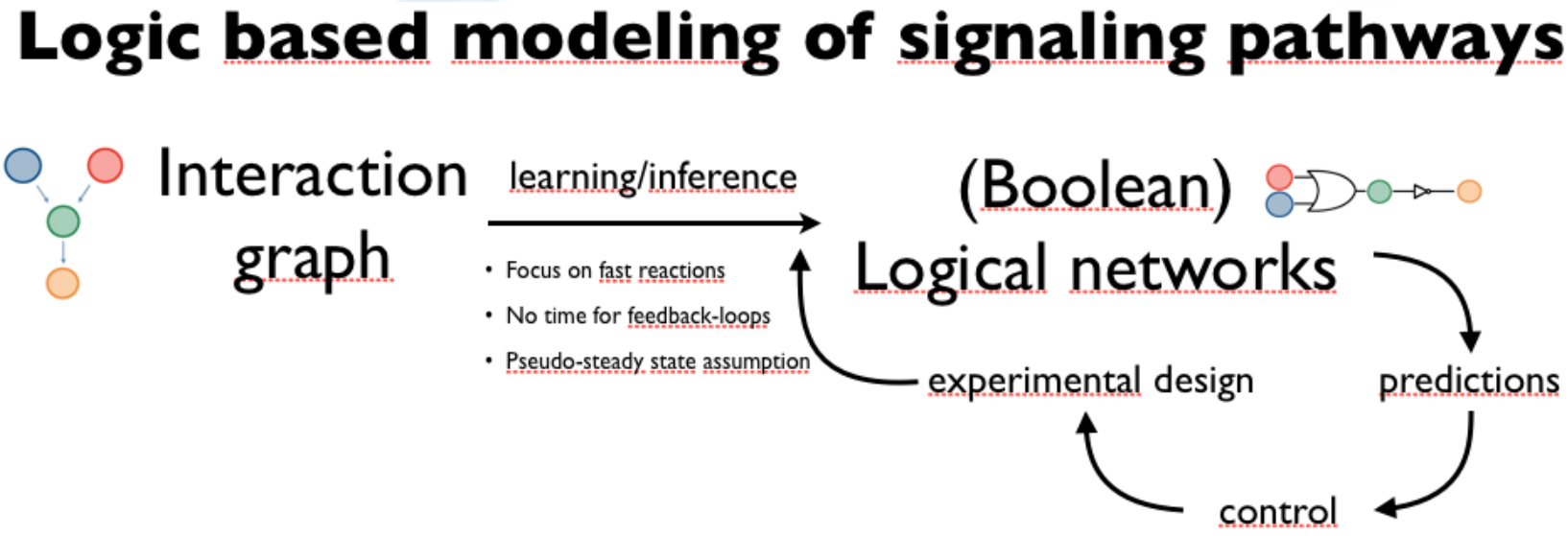


Shared regulator	Distance to purR	Distance to marBAR
cpxAR	5	1
yhfK-crp	2	4
rcsBD	3	3
caiF	4	2

... to decipher key regulators of exotic organisms ?

Towards dynamics...

How do actors act together ?

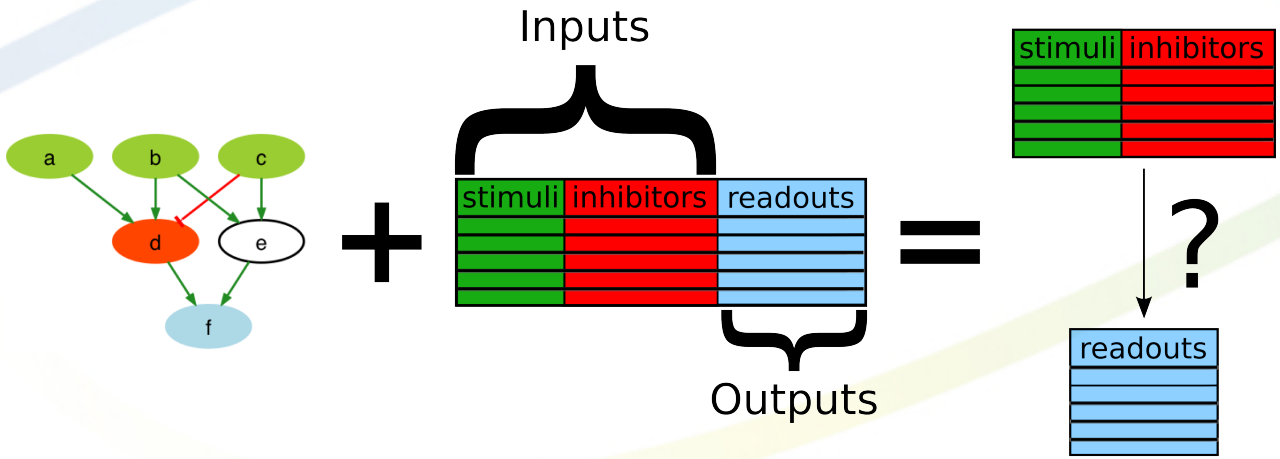


Learning logical static rules

Data

- Signed and directed causal interactions among proteins
- Phosphorylation activity in time *t* after stimulation

Goal Predictive models of immediate-early protein signaling pathways

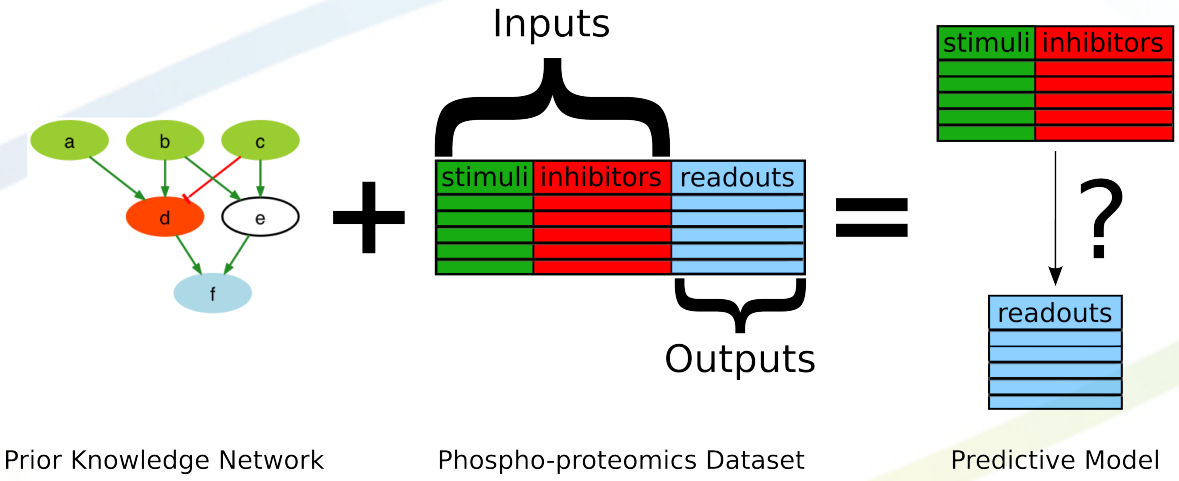


Underlying assumptions

- Focus on fast reactions
- No time for feedback mechanisms
- Pseudo-steady state assumption

Predictive Signaling Network Challenge *[Prill'11]*.

12 groups with different formalisms (ODEs, machine learning, boolean logic)

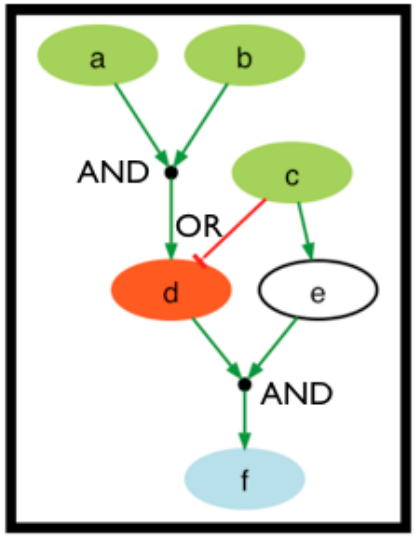


Score **Trade-off between fitness and model size**.

- Biological Property: consistency with experimental data
- Parsimony Principle: minimal/simplest explanation

Discrete approach

a	b	c	d	f
1	1	1	1	0,7
1	1	1	0	0,2



a,b,c are stimulated and **d** is not inhibited

$$((d \wedge e) \equiv f) \wedge (c \equiv e) \wedge a \wedge b \wedge c \wedge (((a \wedge b) \vee \neg c) \equiv d)$$

a,b,c are stimulated and **d** is is inhibited

$$((d \wedge e) \equiv f) \wedge (c \equiv e) \wedge a \wedge b \wedge c \wedge \neg d$$

a	b	c	d	f	f_M
1	1	1	1	0,7	1
1	1	1	0	0,2	0

$$\mathcal{B}(v) = \begin{cases} (a \wedge b) \vee \neg c & \text{if } v = d \\ c & \text{if } v = e \\ (d \wedge e) & \text{if } v = f \end{cases}$$

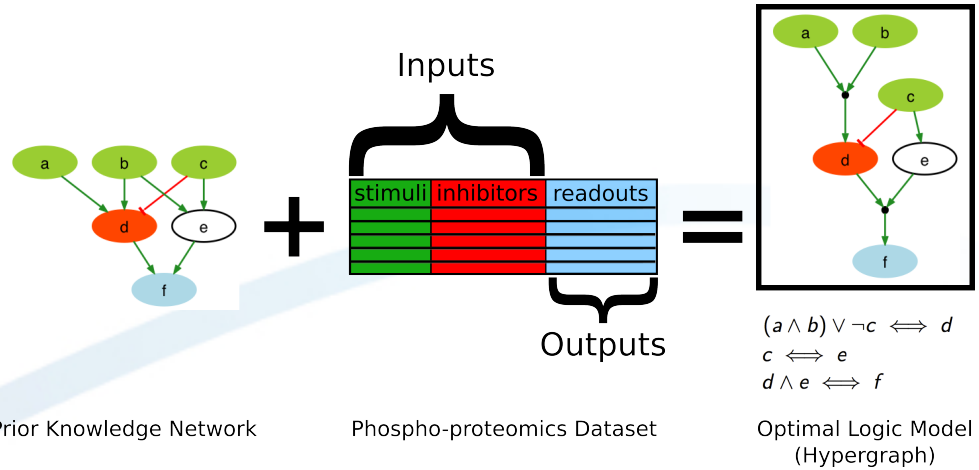
- Residual: $(0.2 - 0)^2 + (0.7 - 1)^2 = 0.04 + 0.09 = 0.13$
- Size: $3 + 1 + 2 = 6$

1	1	0	0	?	0
1	1	0	1	?	0
...	...	...	...	?	...
0	0	0	0	?	0

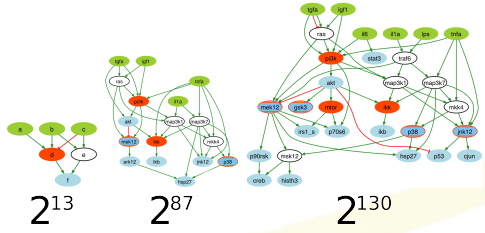
Complete input-output behavior:
Global Truth Table (GTT)

Discrete approach

Optimization Learning **Logic** Models or **hypergraphs**?



Search space. Hypergraphs compatible with the graph: exponential growth

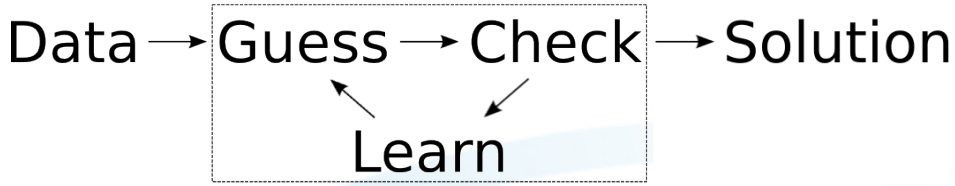


State-of-the-Art CellNOpt (genetic algorithm) *[Saez-Rodriguez'09]*

Questions

- Scalability ?
- Exhaustive enumeration of the space of solutions ?
- Include inherent variability: sub-optimals ?

ASP. Guess & Check methodology



- Data: PKN and phospho-proteomics dataset (facts)

```
node(tnfa). node(p38). edge(tnfa,p38,1). exp(1,tnfa,1). obs(1,p38,0).
```

- Guess: Generate candidates models (non-deterministic)

```
{clause(A,N)} :- hyperedge(A,N).
```

- Check: Eliminate invalid models (integrity constraints)

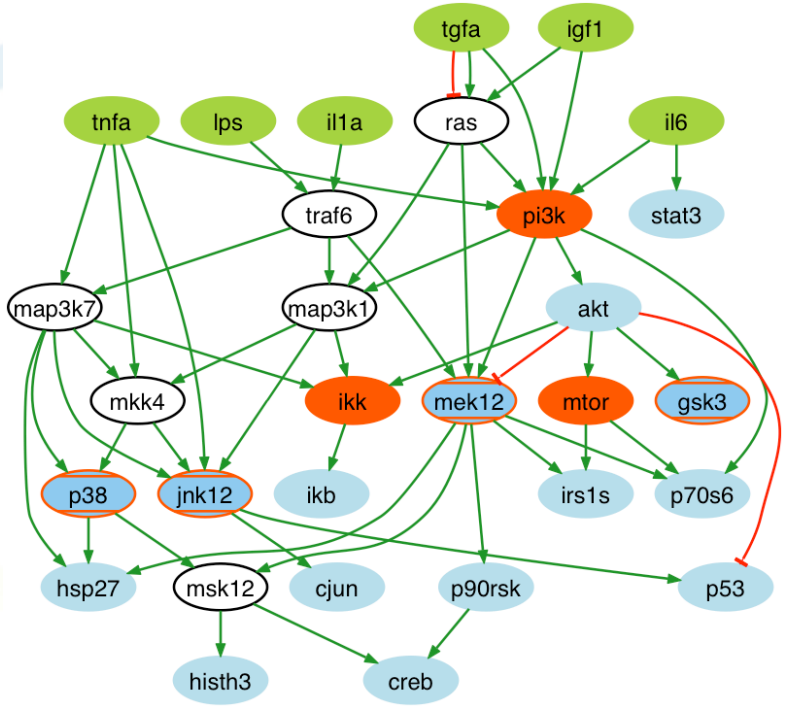
```
:- clause(A,N), clause(B,M), A!=B, redundant(A,B).
```

- Learn: Loop between ”guess” and ”check”
- Optimize: Minimize cost function (weighted sum of atoms)

```
#minimize[mismatch(E,R,W) = W, clause(A,N) : param(P) = N*P].
```

Real case example

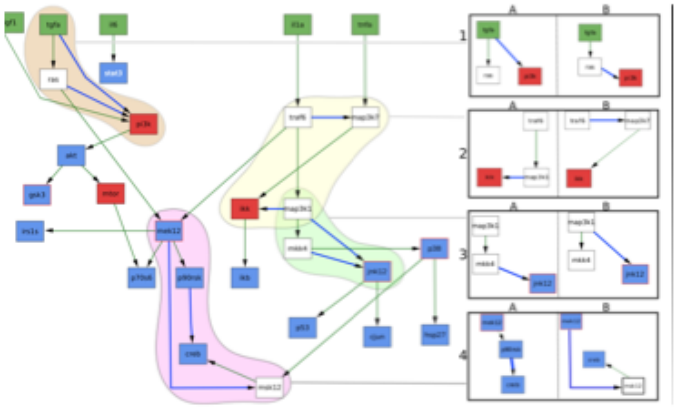
- Case-study: model of pro-growth and inflammatory pathways in human liver cells
- Dataset: Phosphorylation activity after perturbation of 15 proteins under 64 combinations of stimuli and inhibitions



What can we say about boolean networks which are compatible with these data?

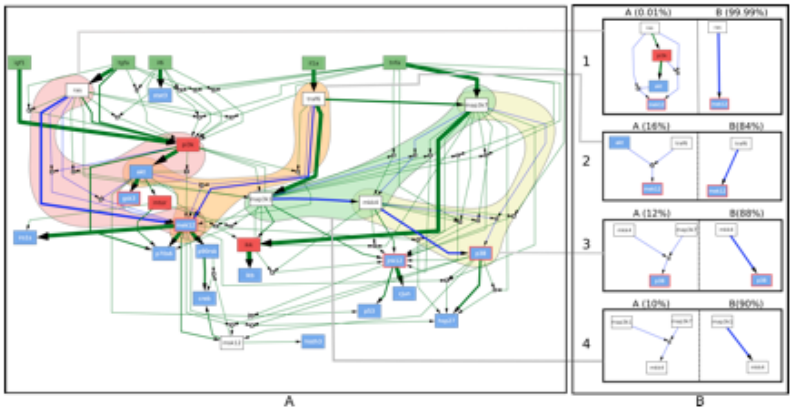
Complete enumeration of sub-optimal models

Minimal MSE



- 16 optimal models (~1 sec.)
- 1 input-output behavior

Tolerance of 10%



- 11,700 suboptimal models (~4 sec.)
- 91 input-output behaviors

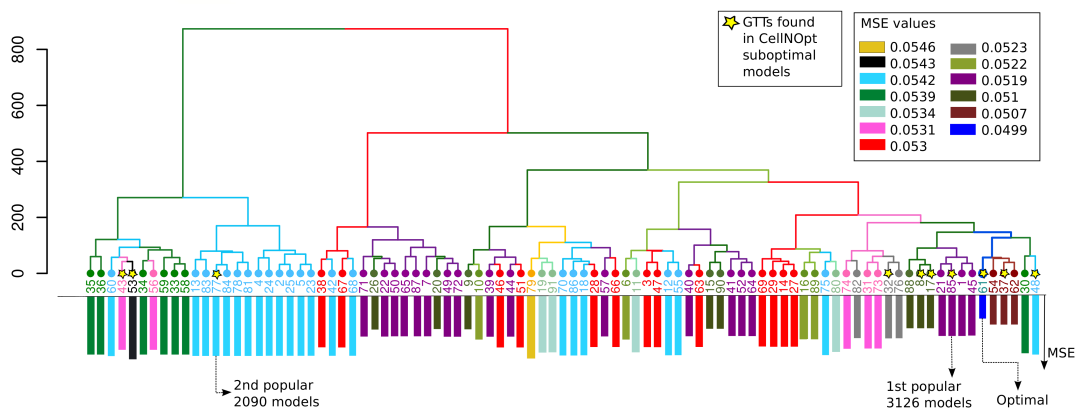
- ASP: Good time performances *[Videla et al, CMSB'12]*
- **Thousands of compatible models**
- Studying the combinatorial structure becomes possible *[Videla-Guziolowski + EBI, submitted]*
- Less variability among truth tables (input/output behaviors).

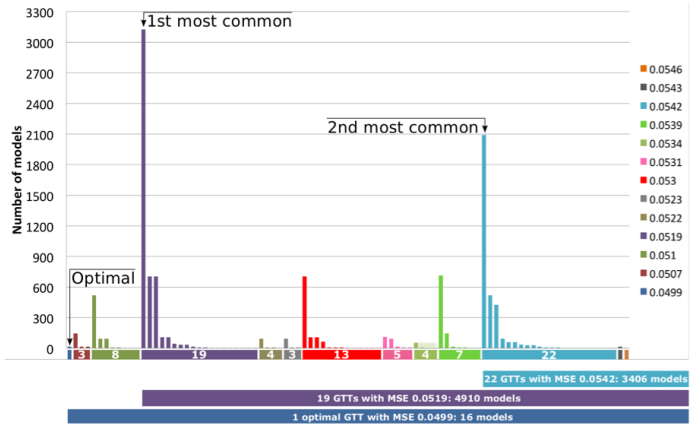
What can be really deduced from the model ?

Focus on truth tables allows reducing the variability

Main observations

- 30% of input conditions give the same output among the 91 GTTs
  → Strong/robust predictions independent of the choice of the model
- Some GTTs appear clearly more relevant than others (2 gather 50% of models)
  → Most interesting feature to reduce variability !
- 1st most common GTT is very close to the optimal GTT





Towards experimental design ?

How addressing the question of experiment design ?

- Several criteria for ”simplest” experimental condition.
- Several criteria for ”most informative” experiment
- ASP appears as a powerful tool to address such questions.

Preliminary results

- A few new experiments can discriminate most relevant truth tables.
- There are two *simplest* experimental conditions (inputs) maximizing differences between GTTs at the same time

	Stimuli			Inhibitors		Readouts			
	TFGa	IGF1	IL1a	mTORi	MEK12i	p70s6	CREB	p53	GTT
Experiment 1	1	0	1	1	1	0	1	1	Optimal
						1	0	1	1° most common
Experiment 2	0	1	1	1	1	0	0	0	2° most common

Availability

- Python http://pypi.python.org/pypi/caspo
- Online http://mobyle.genouest.org/cgi-bin/Mobyle/portal.py#forms::caspo

What is the trick: Modeling dynamics !!

Modeling approach

Modeling steady-states of boolean networks?

- ASP: order one logics.
- **Iterative computation of logical rules until it stops**.
- Closely related to fixed-point semantics in logics !!

2-valued classical logics:	3-valued Kleene's logics:
{0,1}	{*,0,1}
0 < 1	* < 0, * < 1

Impact of semantics

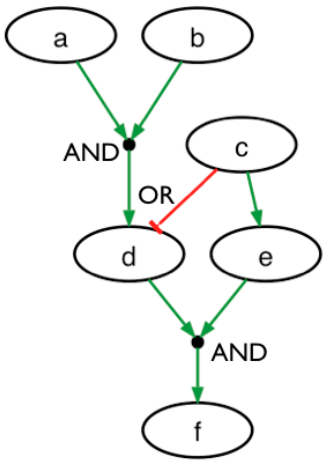
- LSS2: everything remains false unless there is a cause to make it true (may not exists due to negative loops).
- LSS3: everything remains undefined unless there is a cause to make it either true or false (always exists).

$LSS_{\{2,3\}}$ starting from the "false" or "undefined" state and certain clamped nodes. **=** Least fixpoint starting from the "false" or "undefined" assignment and certain clamped variables.

Application: control of signaling network

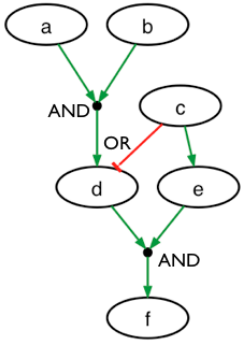
Minimal intervention sets?

- Based on Minimal Cut Sets for metabolic networks (Klamt et al'2006, Samaga et al '2006).
- Hard combinatorial problem.



- Given:
  - Constraints
  - Goals
- Find:
  - Required interventions (knock-in/out) over nodes in order to satisfy goals?

Modelling with LLS_3

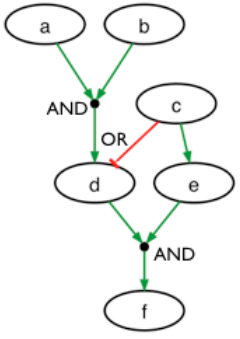


- Constraints: {}
- Goals: {d=-1}
- Interventions:
  - {a=-1, c=1}
  - {b=-1, c=1}

i: {a=-1, c=1}

LSS$_3$ →

t	a	b	c	d	e	f
t_0	-1	*	1	*	*	*
t_1	-1	*	1	-1	1	*
t_2	-1	*	1	-1	1	-1



- Constraints: {a=-1,b=-1}
- Goals: {f=1}
- Interventions?
  - {a=1,b=1,c=1}
  - {a=1,b=1,e=1}
  - {c=1, d=1}
  - {c=0, e=1}
  - {d=1, e=1}

Iterative computation of the LSS$_3$ starting from the undefined state and certain clamped nodes: {a=1, b=1, e=1}

LSS$_3$ →

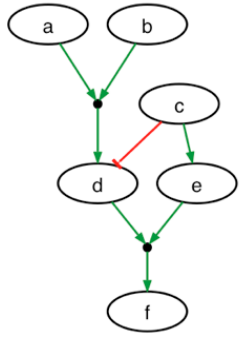
t	a	b	c	d	e	f
t_0	1	1	*	*	1	*
t_1	1	1	*	1	1	*
t_2	1	1	*	1	1	1

LLS_3 captures "by nature" the early-response in logic models of signaling transduction *[Videla et al, ICLP'2013]*

Modelling with ASP

instance

```
variable(a). variable(b). variable(c).
variable(d). variable(e). variable(f).
candidate(a). candidate(b).  candidate(c).
candidate(d). candidate(e).

formula(d,0). formula(e,1). formula(f,2).
dnf(0,0). dnf(0,1). dnf(1,2). dnf(2,3).
clause(0,a,1).  clause(0,b,1).  clause(1,c,-1).
clause(2,c,1).  clause(3,d,1). clause(3,e,1).

scenario(1).
constrained(1,a,-1). constrained(1,b,-1).
goal(1,f,1).
```



- Constraints: {a=-1,b=-1}
- Goals: {f=1}

guess

```
goal(T,S)        :- goal(_,T,S).
goal(T)          :- goal(_,T,_).
constrained(Z,E) :- constrained(Z,E,_).
constrained(E)   :- constrained(_,E).
```

auxiliary domain predicates

```
{ intervention(V,-1;1) : candidate(V) }.
```

Guess an intervention set among all candidates (optimization: some interventions can be omitted in advance)

```
:- intervention(V,1), intervention(V,-1).
intervention(V) :- intervention(V,S).
```

Eliminate contradictory interventions and project intervened variables

fix point

```
eval(Z,V,S) :- scenario(Z), intervention(V,S).
eval(Z,E,S) :- constrained(Z,E,S),
               not intervention(E).
free(Z,V,D) :- formula(V,D), scenario(Z),
               not constrained(Z,V),
               not intervention(V).
```

Set truth value for interventions and side constraints. Neither intervened nor constrained, are "free"

```
eval_clause(Z,C,-1) :- clause(C,V,S),
                       eval(Z,V,-S).
```

A clause evaluates to false if at least one of its literals evaluates to false

```
eval(Z,V, 1) :- free(Z,V,D), dnf(D,C),
                eval(Z,W,T) : clause(C,W,T).
```

A (free) variable evaluates to true if at least, in one of its DNF clauses all literals evaluates to true

```
eval(Z,V,-1) :- free(Z,V,D),
                eval_clause(Z,C,-1) : dnf(D,C).
```

A (free) variable evaluates to false if all its DNF clauses evaluates to false

check&minimize

```
:- goal(Z,T,S), not eval(Z,T,S).
```

A goal variable does not evaluates as required

```
#const max=0.
:- max>0, max + 1 #count{ intervention(_) }.
```

Optional maximal number of interventions

```
#minimize { intervention(_) }.
```

Minimize over interventions

Application...

smart encoding + *hclasp* allows computing all MIS in a reasonable time...*[Videla et al, ICLP'2013]*

	EGFR ($3^{13} \times 2^{19}$)			EGFR* ($3^{52} \times 2^{15}$)			TCR ($3^{35} \times 2^{28}$)			TBH6b ($3^{85} \times 2^{24}$)		
k	#MISs	claspD	hclasp	#MISs	claspD	hclasp	#MISs	claspD	hclasp	#MISs	claspD	hclasp
2	15	0,11	0,00	0	0,93	0,08	0	0,05	0,00	0	0,81	0,00
3	21	0,13	0,00	8	1,15	0,10	7	0,05	0,00	6	0,92	0,00
4	21	0,14	0,00	38	11,16	0,15	26	0,36	0,03	6	18,00	0,00
5	21	0,17	0,00	74	41,57	0,21	1196	2,04	0,08	6	17,40	0,00
...	...	...	...	...	...	...	...	....	...	...	...	...
10	21	0,19	0,00	83	84,60	0,23	9704	669,64	0,94	1248	667,85	**0,23**
inf	21	0,13	**0,00**	83	91,83	**0,19**	13016	512,52	**1,19**	-	-	-

Summary We can use programming logics to reason efficiently over large-scale biological data by including qualitative dynamical rules.

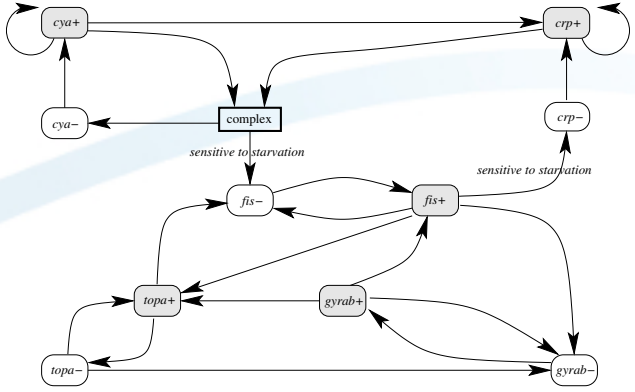
What about quantitative data ?

$\rightarrow$ **new invariants can be used...**

Small illustration

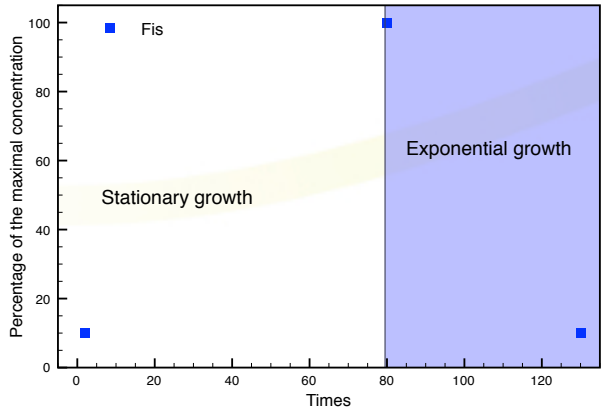
Imagine that you are given

A qualitative model of a system dynamics



E. Coli, [*Ropers et al, 2006*]

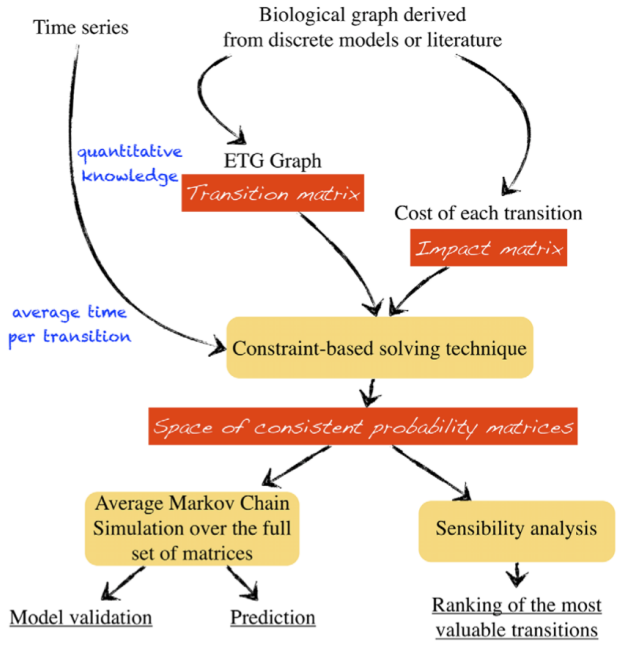
Some quantitative time-series information

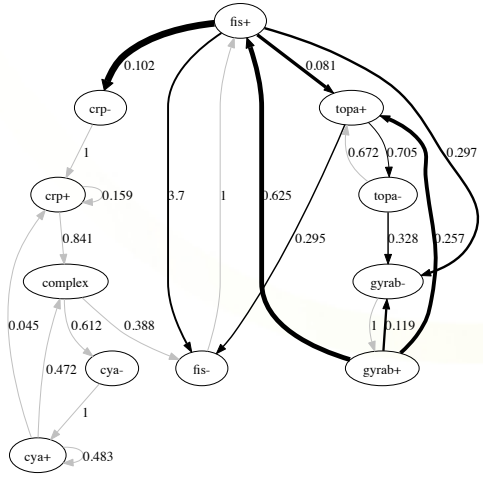


3 time-series measures of *Fis* protein concentration.

Accumulative laws over Markov Chains

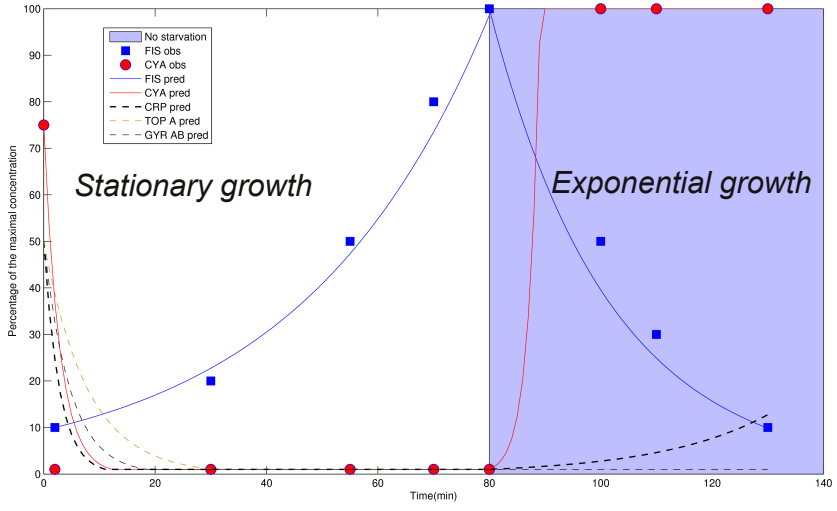
- Model Multiplicative accumulation law over a Makov chain which describes transitions at the transcriptional scale.
- Dynamics The asymptotic behavior follows a log-normal law with explicit parameters.
- Application *[Bourdon et al, PLOS CB'2011]*
  - Space of compatible Markov chains (local search methods)
  - Interaction sensitivity.





Event Transition Graph

Sensitivity of transition with respect to observations



Validation

Ergodic invariants allow predicting the main features of other proteins in the network !

Conclusion: a huge mess

(Too many) problems

- Curation of regulatory networks in cancer.
- Pruning regulatory network inferred from genome analysis and mutual information
- Identifying functional modules
- Learning signaling networks
- Controlling signaling networks
- Predicting time-series protein quantities

What is the common point?

- Never discriminate between models.
- Introduce qualitative rules deduced from dynamical studies.
- Reduce uncertainty (raised by fluctuation in data) by making the problems discrete.
- Use very efficient programming logics (ASP) to integrate data and knowledge.
- cross your fingers (good compromise between efficiency and flexibility in the problem statement).

Take home message

- **We now have efficient enough methods to address enumeration problems in integrative and systems biology.**

Credits

Modeling and Dynamics

- IRISA/Inria (Rennes). G. Collet. M. Le Borgne. V. Picard. S. Prigent. S. Thiele. S. Videla.
- LINA (Nantes). D. Eveillard, J. Bourdon.
- Ircyyn (Nantes). C. Guziolowski.
- Univ. Luxembourg. T. Baumuratova.
- Univ. Montpellier. O. Radulescu.

ASP solvers and grounders

- Potsdam university. T. Schaub. M. Gesber. M. Ostrowski.
- Rennes. J. Nicolas.

Biological applications

- Institut Curie (Ewing sarcoma). G. Stoll. D. Surdez. O. Delattre. A. Zinoviev.
- EBI (signaling network). J. Saez-Rodriguez. F. Eduarti.
- University of Chile (exotic bacteria). A. Maass. P. Bordon. A. Aravena.
- Station Biologique Roscoff (algae). T. Tonon. C. Boyen.

Support ANR blanc Biotempo, Inria-Chile/CIRIC, ANR Inv. avenir Idealg.

Aknowledgments

Thanks to the many people who contributed to the construction of this lecture

- **Jacques Nicolas**.
- Santiago Videla
- Andres Aravena.
- Philippe Bordron.
- Guillaume Collet.
- Sylvain Prigent.
- Sven Thiele.

More about ASP

- Higher level introduction course. http://www.irisa.fr/dyliss/jacques.nicolas
- binaries: http://www.cs.uni-potsdam.de/ sthiele/bioasp/downloads/bin/
- Tutorial. http://sourceforge.net/projects/potassco/files/potassco_guide/20101004/
- Advanced courses. http://potassco.sourceforge.net/teaching.html
- mailing list, google+ group
- Free book. Answer Set Solving in Practice Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub Synthesis Lectures on Artificial Intelligence and Machine Learning, Dec. 2012, Vol. 6, No. 3 , Pages 1-238