Biological pathway inference with answer set programming

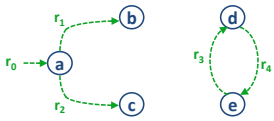
Oliver Ray
Dept. of Computer Science
Univ. of Bristol, UK

ISSSB 2011, Shonan, Japan
15th November 2011

Summary of 3 Investigations

- IIBM 2010
  "Logic-based Steady-State Analysis and Revision of Metabolic Networks with Inhibition"
  - trivial networks: substrates, products
  - focus on reasoning about cycles (ignore inhibition here)
- ANB 2010
  "Analysing Pathways Using ASP-Based Approaches"
  - reaction networks: substrates, products, modifiers
  - focus on ranking hypotheses (via numerical parameters)
- ILP 2009
  "Automatic Revision of Metabolic Networks through Logical Analysis of Experimental Data"
  - real networks: substrates, products, inhibitors, (iso) enzyme (complexes)
  - focus on revising models (via abduction and induction on real data)

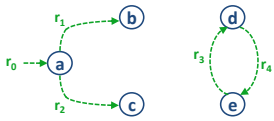
Trivial networks



Reactions: Substrates & Products

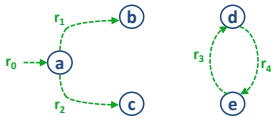
a metabolite is present iff it is product of active reaction

a reaction is active iff all substrates are present



Answer Set Programming

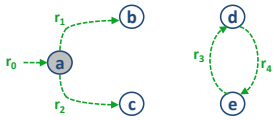
a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
active(R) :- present(M):substrate(M,R).



logical description (black) and ASP formulation (blue)

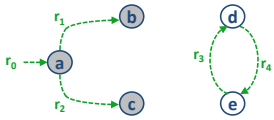
Sanity Check

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
active(R) :- present(M):substrate(M,R).



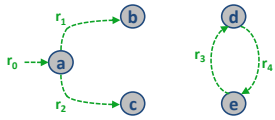
So Far, So Good!

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
active(R) :- present(M):substrate(M,R).



What on Earth?

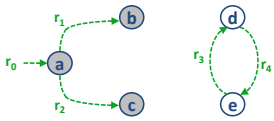
a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
active(R) :- present(M):substrate(M,R).



n.b. this unfounded solution of the logical rules (black) is eliminated by ASP (blue)

Reachability = Stability (Free in ASP)

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
active(R) :- present(M):substrate(M,R).

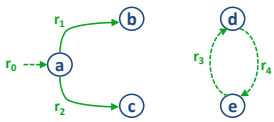


the available metabolites can be synthesised (from the initial metabolites)!

Some Contributions of IIBM'2010

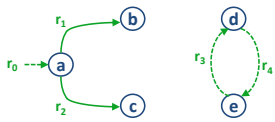
- shows that the ASP notion of stability can be exploited to enforce the reachability of steady states in metabolic networks
- shows that the ASP formalism is well-suited to representing and reasoning about metabolic networks

Reaction networks



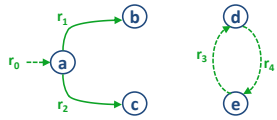
Competition

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all inputs are present and no competitor is active
active(R) :- present(M):input(M,R) , not active(R'):compete(R,R').



Reactants & Modifiers

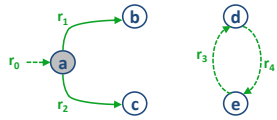
a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all inputs are present and no competitor is active
active(R) :- present(M):input(M,R) , not active(R'):compete(R,R').
an input is a reactant or a modifier
input(M,R) :- 1{reactant(M,R) , modifier(M,R)}.
two reactions compete iff there is an input of both that is a reactant of one
compete(R1,R2) :- 2{input(M,R1;R2)} , 1{reactant(M,R1;R2)} , R1≠R2.



n.b. unlike modifiers (dotted), reactants (solid) can only be used in one reaction

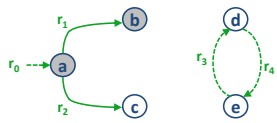
Sanity Check

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present and no competitor is active
active(R) :- present(M):input(M,R) , not active(R'):compete(R,R').
an input is a reactant or a modifier
input(M,R) :- 1{reactant(M,R) , modifier(M,R)}.
two reactions compete iff there is an input of both that is a reactant of one
compete(R1,R2) :- 2{input(M,R1;R2)} , 1{reactant(M,R1;R2)} , R1≠R2.



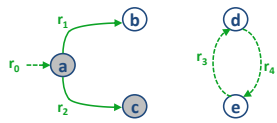
Solution 1

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present and no competitor is active
active(R) :- present(M):input(M,R) , not active(R'):compete(R,R').
an input is a reactant or a modifier
input(M,R) :- 1{reactant(M,R) , modifier(M,R)}.
two reactions compete iff there is an input of both that is a reactant of one
compete(R1,R2) :- 2{input(M,R1;R2)} , 1{reactant(M,R1;R2)} , R1≠R2.



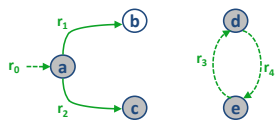
Solution 2

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present and no competitor is active
active(R) :- present(M):input(M,R) , not active(R'):compete(R,R').
an input is a reactant or a modifier
input(M,R) :- 1{reactant(M,R) , modifier(M,R)}.
two reactions compete iff there is an input of both that is a reactant of one
compete(R1,R2) :- 2{input(M,R1;R2)} , 1{reactant(M,R1;R2)} , R1≠R2.



Doh!

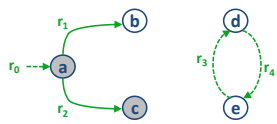
a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present and no competitor is active
active(R) :- present(M):input(M,R) , not active(R'):compete(R,R').
an input is a reactant or a modifier
input(M,R) :- 1{reactant(M,R) , modifier(M,R)}.
two reactions compete iff there is an input of both that is a reactant of one
compete(R1,R2) :- 2{input(M,R1;R2)} , 1{reactant(M,R1;R2)} , R1≠R2.



n.b. this unfounded solution of the logical rules (black) is eliminated by ASP (blue)

Reachability = Stability (Free in ASP)

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present and no competitor is active
active(R) :- present(M):input(M,R) , not active(R'):compete(R,R').
an input is a reactant or a modifier
input(M,R) :- 1{reactant(M,R) , modifier(M,R)}.
two reactions compete iff there is an input of both that is a reactant of one
compete(R1,R2) :- 2{input(M,R1;R2)} , 1{reactant(M,R1;R2)} , R1≠R2.



the available metabolites can be synthesised (from the initial metabolites)!

Weight Constraints

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is viable iff all substrates are present and no competitor is active
viable(R) :- present(M):input(M,R) , not active(R'):compete(R,R').
an input is a reactant or a modifier
input(M,R) :- 1{reactant(M,R) , modifier(M,R)}.
two reactions compete iff there is an input of both that is a reactant of one
compete(R1,R2) :- 2{input(M,R1;R2)} , 1{reactant(M,R1;R2)} , R1≠R2.
a reaction is active iff it is viable and we don't force it off…
active(R) :- viable(R) , not force_off(R).
…or (it is not viable but) we force it on
active(R) :- force_on(R).
we can force any viable reactions off
{force_off(R)} :- viable(R)
we can force any reactions on (that we didn't force off)
{force_on(R)} :- not force_off(R)
but we want to minimize the cost of doing so
#minimize{force_off(R) = c1 , force_on(R) = c2}.

A MaxSat Approach (Tiwiari et al. ANB'07)

- a reaction is active iff inputs present and all its competitors are of

$$r_i \leftrightarrow \bigwedge_{s \in R(r_i) \cup M(r_i)} \left(\bigvee_{r_j \in P^{-1}(s)} r_j \wedge \bigwedge_{r_j \in R^{-1}(s), j \neq i} \neg r_j \right)$$

- in fact these constraints are actually broken into parts with different weights and the aim is to compute maximal weight solutions
- in fact dummy reactions are added to produce initial species and/or species with no producing reactions
- we can do all that in ASP as well (details in paper)!

Target & Forbidden Species

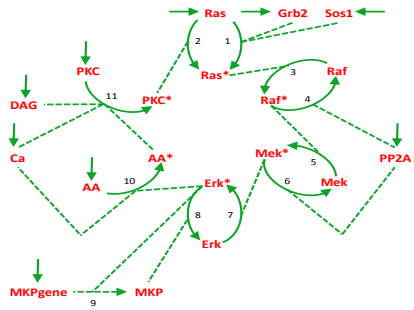
target species should be present
:- target(S), not present(S).

$$\bigvee_{r_j \in P^{-1}(s)} r_j$$

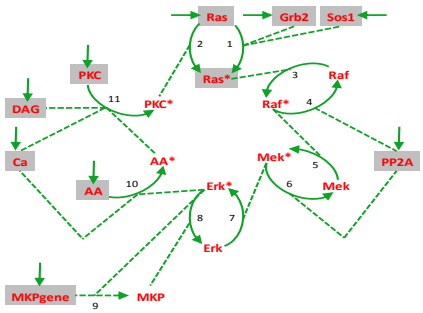
forbidden species should be absent
:- forbidden(S), present(S).

$$\bigwedge_{r_j \in P^{-1}(s)} \neg r_j$$

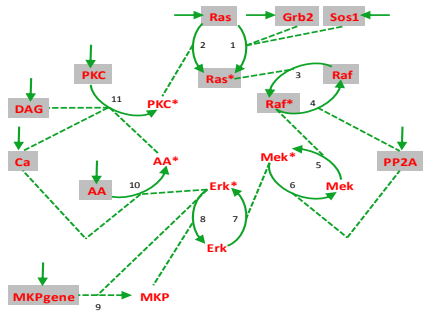
Example: MAPK



Optimal Solution

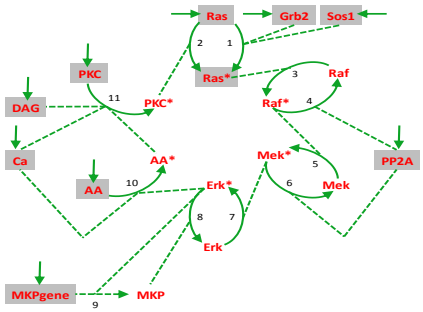


Doh!



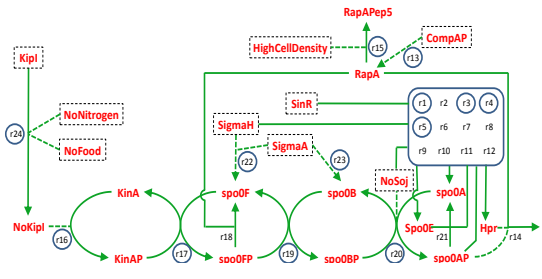
n.b. this unfounded minimal-cost MaxSat solution is eliminated by ASP

Reachability = Stability (Free in ASP)



the available metabolites can be synthesised (from the initial metabolites)!

Example: Sporulation Initiation



n.b. here again, unfounded minimal-cost MaxSat solutions are eliminated by ASP

Some Contributions of ANB'2010

- Identify a limitation of Tiwari's approach with respect to its handling of cyclic networks

  Show the biological importance of this limitation using Tiwari's MAPk and sporulation studies

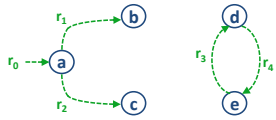
- Develop a theoretical correction for Tiwari's approach based on total order relations

  Establish a formal characterisation of these approaches using stable and supported models

- Develop a practical implementation of the new approach by means of Answer Set Programming

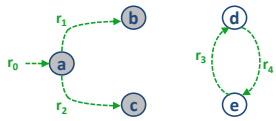
Back to the Beginning

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
active(R) :- present(M):substrate(M,R).



What about Learning of substrates?

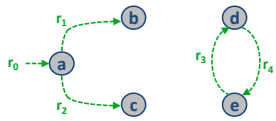
a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
~~active(R) :- present(M):substrate(M,R).~~
active(R) :- not absent_substrate(R).
absent_substrate(R) :- substrate(M,R) , not present(M).



n.b. the fact substrate is on longer a domain predicate means it cannot be used within a conditional literal so we are forced to try this classical reformulation

Doh!

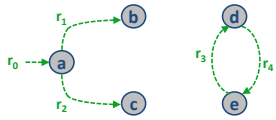
a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
~~active(R) :- present(M):substrate(M,R).~~
active(R) :- not absent_substrate(R).
absent_substrate(R) :- substrate(M,R) , not present(M).



n.b. the fact substrate is on longer a domain predicate means it cannot be used within a conditional literal so we are forced to try this classical reformulation
BUT the double loops through negation now add unwanted stable models!

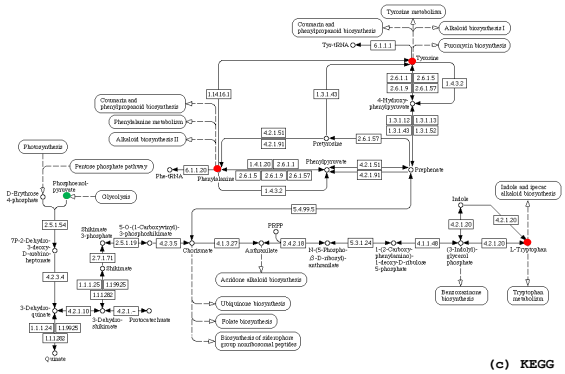
Explicitly specify well-foundedness

a metabolite is present iff it is product of active reaction
present(M) :- active(R) , product(M,R).
a reaction is active iff all substrates are present
~~**active(R) :- present(M):substrate(M,R).**~~
active(R) :- not absent_substrate(R).
absent_substrate(R) :- substrate(M,R) , not present(M).

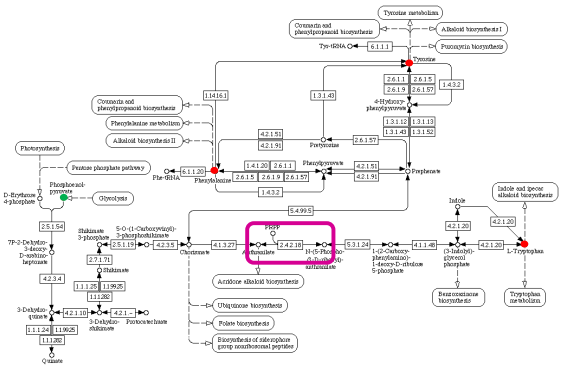


precedes(R1,R2) :- active(R1), active(R2).
precedes(R1,R3) :-precedes(R1,R2), precedes(R2,R3).
:- precedes(R,R).
justified(M,R2) :-precedes(R1,R2), product(M,R1).
:- substrate(M,R), active(R), not justified(M,R).
the available metabolites can be synthesised (from the initial metabolites)!

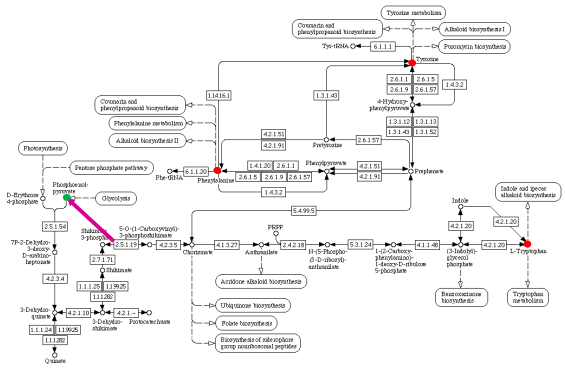
Aromatic Amino Acid Synthesis



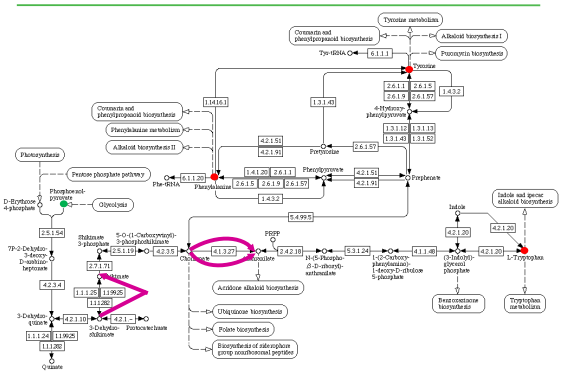
Ambiguities



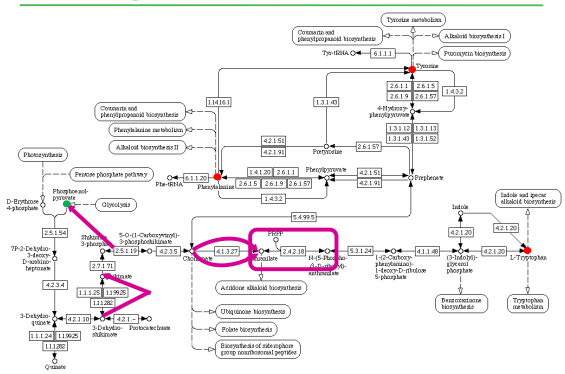
Omissions



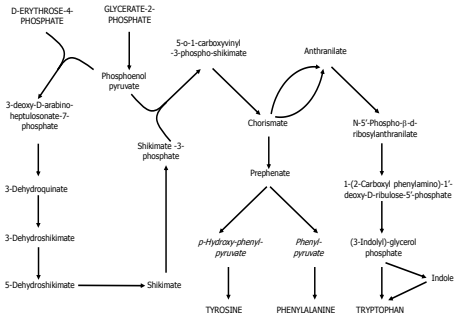
Elaborations



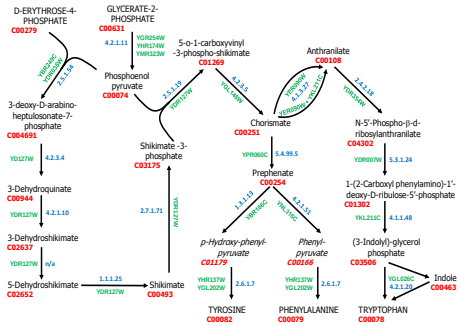
Ambiguities, Omissions, Elaborations



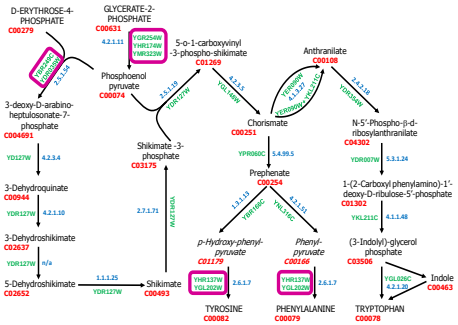
Saccharomyces cerevisiae



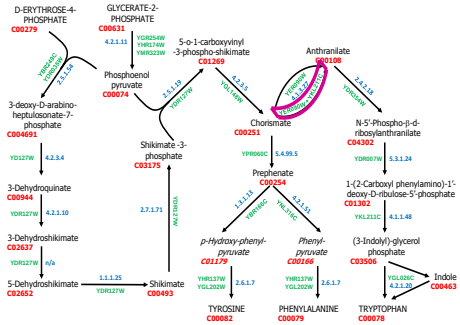
Gene-Enzyme Mappings



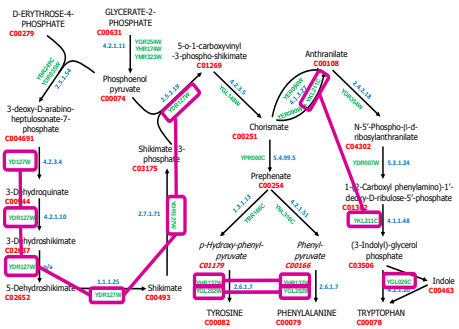
Iso-Enzymes



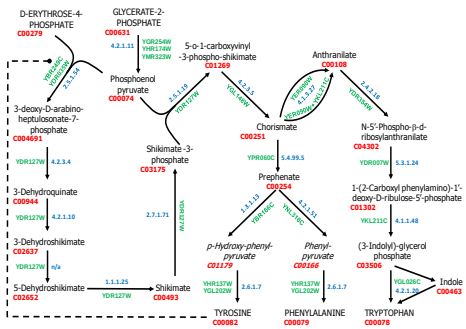
Enzyme-Complexes



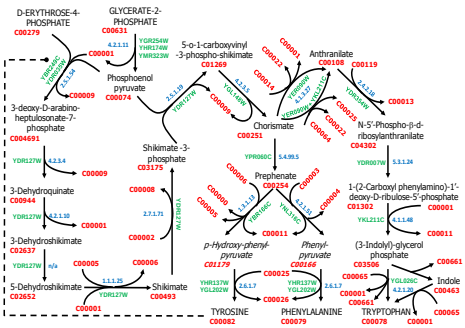
Multi-Functional Enzymes



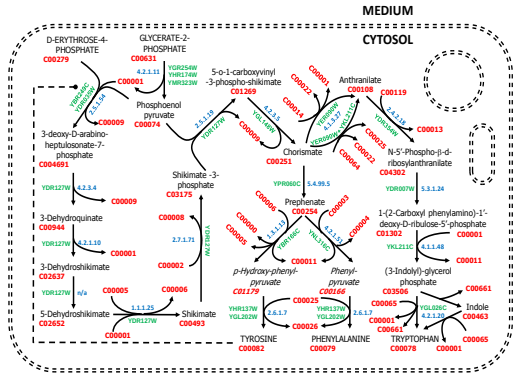
Enzyme-Inhibitions



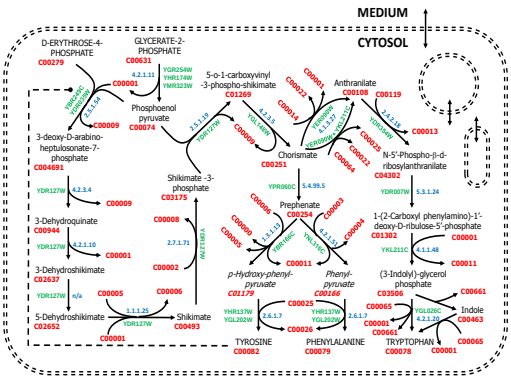
Full Chemical Reactions



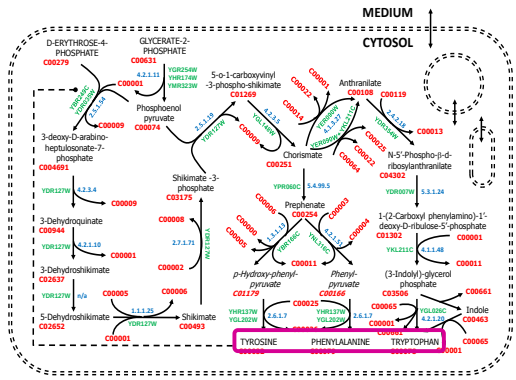
Cellular Compartments



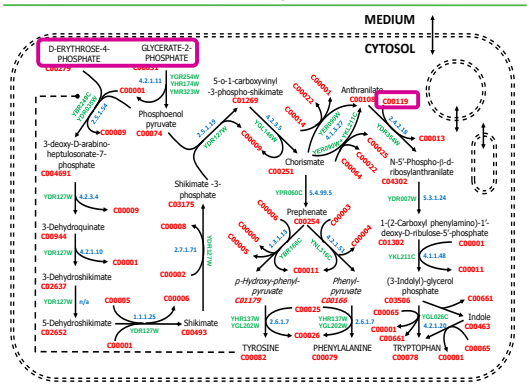
Membrane Reactions



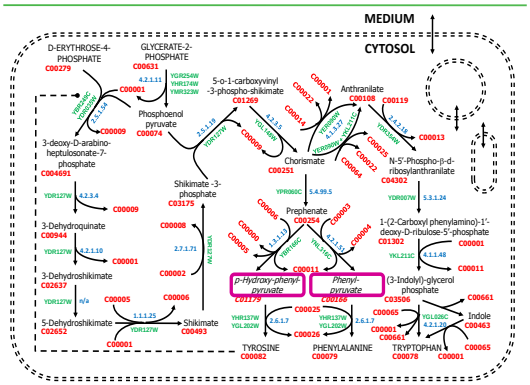
Essential Molecules



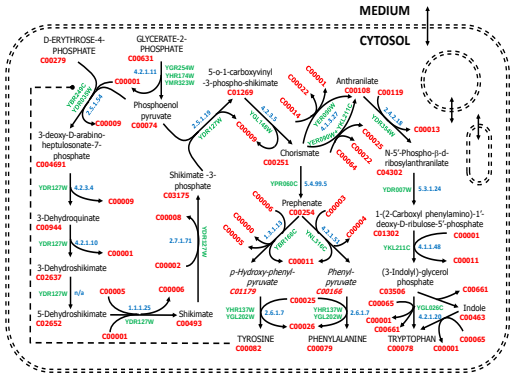
Start Compounds



Rate Information



Saccharomyces cerevisiae



Logical Representation

Reactions: Syntax

```
reaction(
    31, "2.5.1.19", 1,                          % rid , label, rate
  [ reactant( cytosol, 1, "C00074" ),           % substrates
    reactant( cytosol, 1, "C03175" )  ],
  [ reactant( cytosol, 1, "C00009" ),           % products
    reactant( cytosol, 1, "C01269" )  ]
).
```

Reactions: Syntax & Semantics

```
reaction(
    31, "2.5.1.19", 1,                          % rid , label, rate
  [ reactant( cytosol, 1, "C00074" ),           % substrates
    reactant( cytosol, 1, "C03175" )  ],
  [ reactant( cytosol, 1, "C00009" ),           % products
    reactant( cytosol, 1, "C01269" )  ]
).
```

```
in_compartment(Experiment, "C00009", cytosol, Day) :-
                                                    , Day),
                                                    , Day),
in_compartment(Experiment, "C01269", cytosol, Day) :-
  Day >= 1,
  in_compartment(Experiment, "C00074", cytosol, Day),
  in_compartment(Experiment, "C03175", cytosol, Day),
  catalyst(31, Complex),
  not inhibited(Experiment, Complex, Day),
  not deleted(Experiment, Complex),
  not exclude(31).
```

% CERTAIN / RETRACTABLE / ASSERTABLE

Enzymes: Syntax

```
enzyme(
    10,                                         % eid
   "4.1.3.27",                                  % label
   [ "YKL211C","YER090W" ]                      % orfs
   [ 34 ]                                       % rids
).
```

Enzymes: Syntax & Semantics

```
enzyme(
    10,                                         % eid
   "4.1.3.27",                                  % label
   [ "YKL211C","YER090W" ]                      % orfs
   [ 34 ]                                       % rids
).
```

```
catalyst(34, 10).
component("YKL211C", 10).
component("YER090W", 10).
```

```
catalyst(Reaction, unknown) :-
   not enzyme_info(Reaction).
enzyme_info(Reaction) :-
   catalyst(Reaction, Complex),
   not (Complex = unknown).
```

Inhibitions: Syntax

```
inhibitor(
    17,                    % complex
   "C00082",               % metabolite
    cytosol                % compartment
).
```

Inhibitions: Syntax & Semantics

```
inhibitor(
    17,                    % complex
   "C00082",               % metab
    cytosol                % compart
).
```

```
inhibited(Experiment, Complex, Day) :-
   inhibitor(Complex, Metabolite, Compart),
   in_compartment(Experiment, Metabolite, Compart, Day),
   additional_nutrient(Experiment, Metabolite, Compart').
```

Nutrients: Syntax

```
start_compound("C00631", medium).        % metab, compart

essential_compound("C00078", cytosol).   % metab, compart
```

Nutrients: Syntax & Semantics

```
start_compound("C00631", medium).        % metab, compart
```

```
in_compartment(Experiment, Metabolite, Compart, Day) :-
   start_compound(Metabolite, Compart).
```

```
essential_compound("C00078", cytosol).   % metab, compart
```

```
deficient(Experiment, Metabolite, Day) :-
   essential_compound(Metabolite, Compart),
   not in_compartment(Experiment, Metabolite, Compart, Day).
arrested(Experiment, Day) :-
    deficient(Experiment, Metabolite, Day).
predicted_growth(Experiment, Day) :-
   not arrested(Experiment, Day).
```

Experiments: Syntax

```
knockout(1, "YKL211C").                  % exp, orf

additional_nutrient(1, "C00108", medium).  % exp, metab

-observed_growth(1, 1).                  % exp, day
```

Experiments: Syntax & Semantics

```
knockout(1, "YKL211C").                  % exp, orf
```

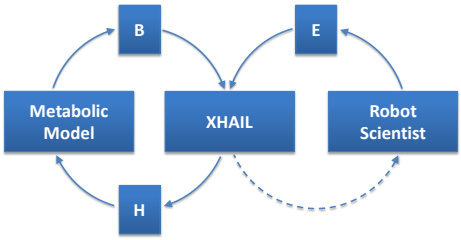
```
deleted(Experiment, Complex) :-
   component(ORF, Complex),
   knockout(Experiment, ORF).
```

```
additional_nutrient(1, "C00108", medium).  % exp, metab
```

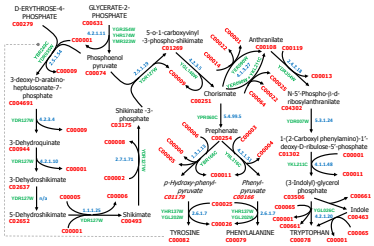
```
in_compartment(Experiment, Metabolite, Compartment, Day) :-
   additional_nutrient(Experiment, Metabolite, Compartment).
```

```
-observed_growth(1, 1).                  % exp, day
```

```
:-  observed_growth(Exp,Day), not predicted_growth(Exp,Day).
:- -observed_growth(Exp,Day), predicted_growth(Exp,Day).
```

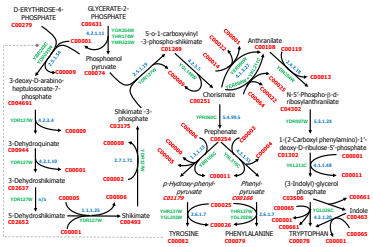


Validation 1: Relearn Inhibition



Validation 1: Relearn Inhibition

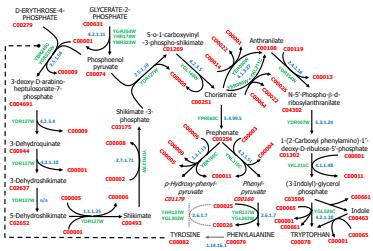
modeh(0,10,min,inhibitor("#enzymeID","#metabolite",cytosol)).



H1 = [inhibitor(14, "C00082", cytosol)]

H2 = [inhibitor(14, "C00078", cytosol)]

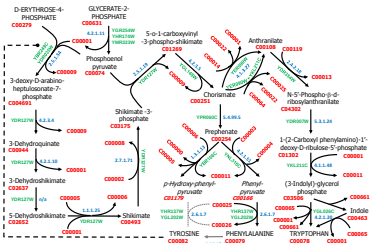
Validation 2: Add & Remove Reactions



Validation 2: Add & Remove Reactions

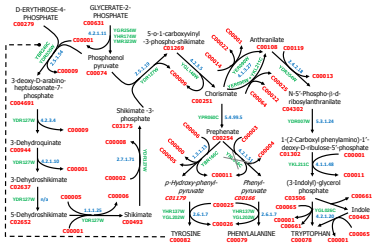
modeh(0,25,min,include("#assertable_reaction")).

modeh(0,25,min,exclude("#retractable_reaction")).



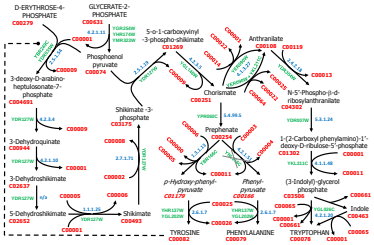
H1 = [exclude(0), include(45)]

Validation 3: Relearn Enzyme Complexes



Validation 3: Relearn Enzyme Complexes

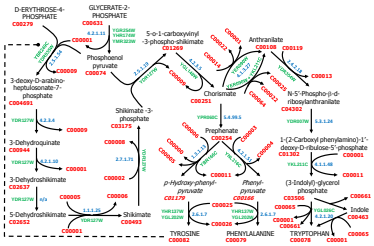
newcomplex("C1"). enzymeID("C1").
modeh(0,2,min,catalyst("#reactionID","#newcomplex")).
modeh(0,2,min,component("#orf","#newcomplex")).



H1 = [catalyst(19, "C1"), component("YNL316C", "C1")]
H2 = [catalyst(42, "C1"), component("YNL316C", "C1")]

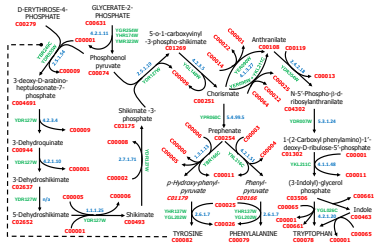
Preliminary Results

1. Indole Opacity X



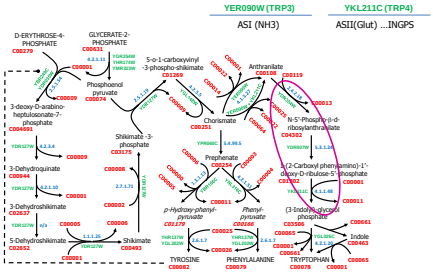
H = [predicted_growth(Experiment,Day) :-
additional_nutrient(Experiment,"C00463").]

2. Indole Contamination ?



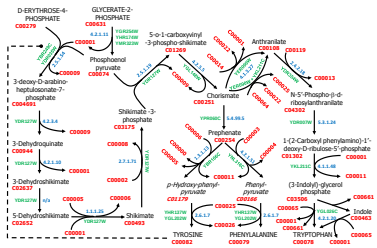
H = [additional_nutrient(Experiment,"C00078",medium) :-
additional_nutrient(Experiment,"C00463",medium).]

3. YER090W Complex in Trypotphan Pathway?

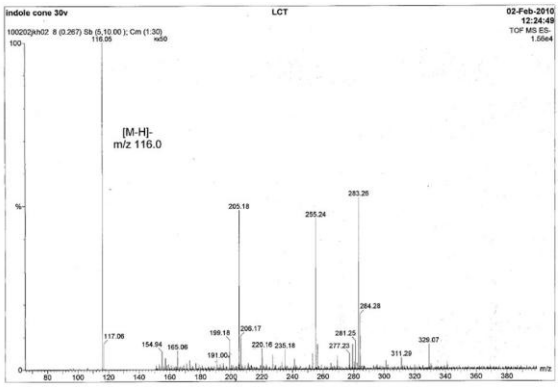


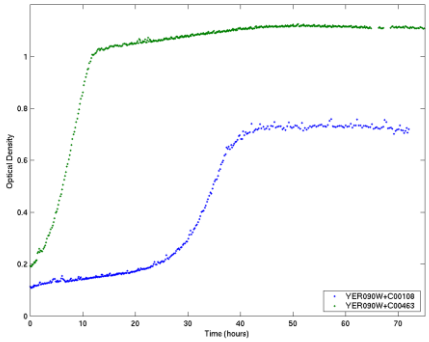
H1 = [orf_complex("YER090W", 9)] H2 = [orf_complex("YER090W", 8)]
H3 = [orf_complex("YER090W", 7)]

4. Slow Anthranilate Import ?



H = [inhibited(Experiment,25,Day) :- Day < 2.]





Some Contributions of ILP'09

1. State-of-the-art Biological Model
   - genes, enzymes, metabolites
   - iso-enzymes, enzyme complexes,
   - enzyme inhibitions, multi-functional enzymes
   - multiple substrates, products, cycles
   - compartments, (rudimentary) rates
2. State-of-the-art nonmonotonic Inference System
   - deduction, abduction, induction
3. Biologically verified hypotheses from Robot Scientist Data
4. Method for adding/removing reactions and generally revising the model to ensure the existence of a steady-states consistent with experimental data