COMPILING MATHEMATICAL FUNCTIONS IN BIOCHEMICAL REACTION NETWORKS

Internship INRIA Lifeware - Julien Bienvenu

Soutenance de stage juillet 2022

Context – Practical view point

M. Patrick Amar & Sys2Diag

- Synthetic vesicles
- Multiple usages (cheap medical testing)

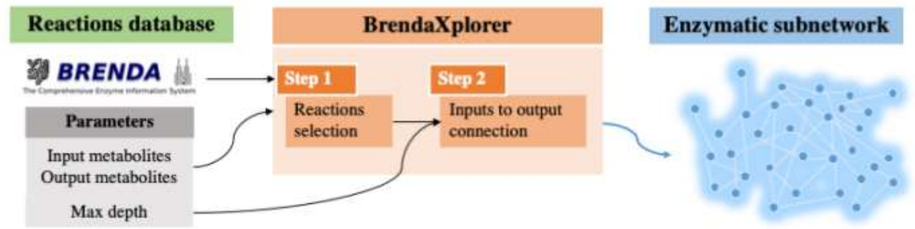


Fig. 2 Database pipeline implemented in BrendaXplorer software to extract concrete chemical reactions, currently used with SiliCell Maker software for implementing logical circuits.

But : Need to figure out real-life CRNs

- Brenda database & BrendaXplorer

Context – Theoretical viewpoint

- Lifeware & BioCham
- Able through polynomialization to perform the following pipeline :

Context – Theoretical viewpoint

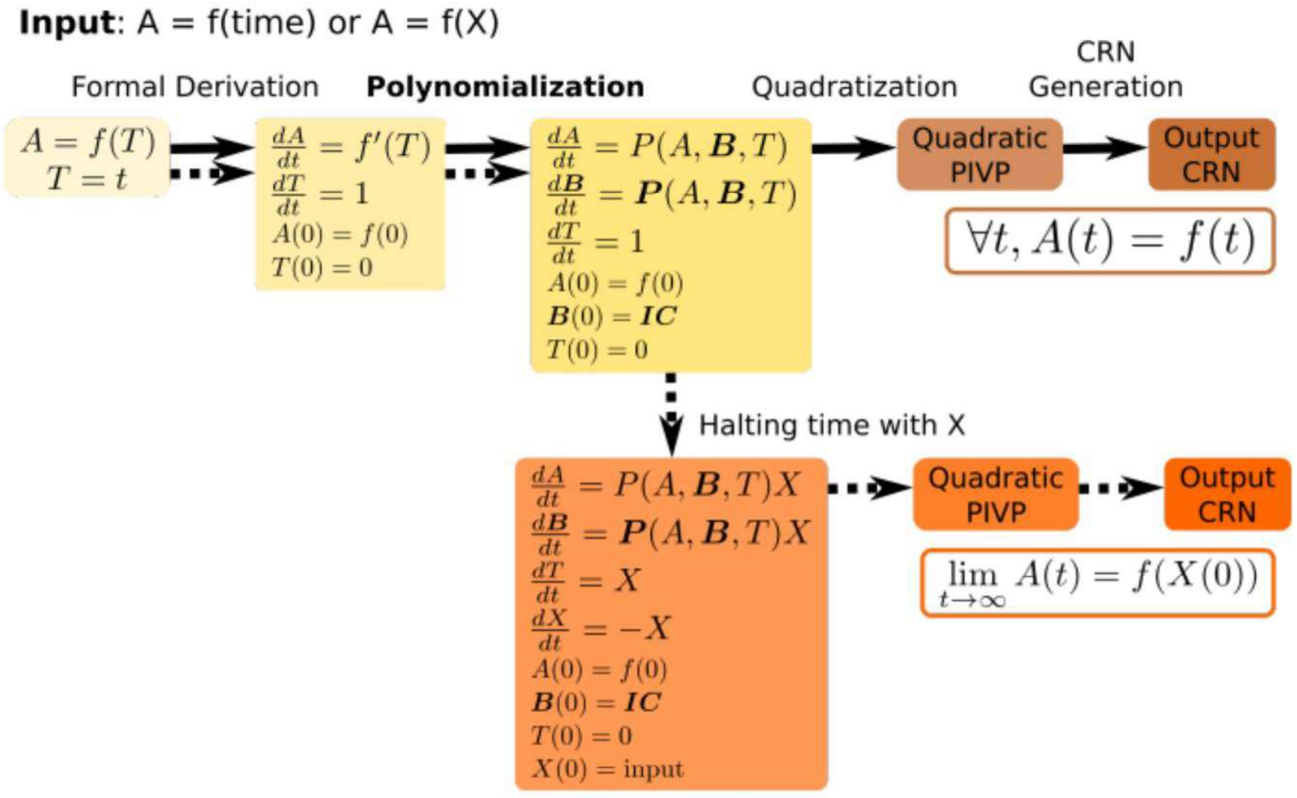


Fig. 1. Symbolic compilation pipeline implemented in BIOCHAM v4 to transform any elementary mathematical function f into an abstract CRN [13,17,26], either a function of time (plain arrows) or an input/output function (dashed arrows): P and **P** are polynomials, **B** denotes the set of abstract auxiliary species introduced by polynomialization given with initial conditions **IC**.

Internship's objective(s)

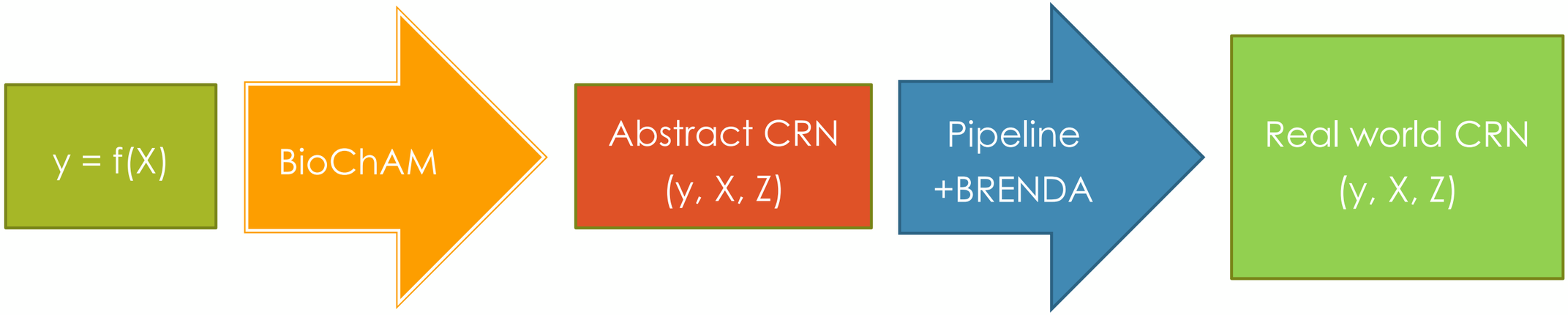
Compile general elementary mathematical functions in concrete CRN, by constraining our compilation pipeline to use the ODE terms of a limited set of well-characterized real enzymatic reactions

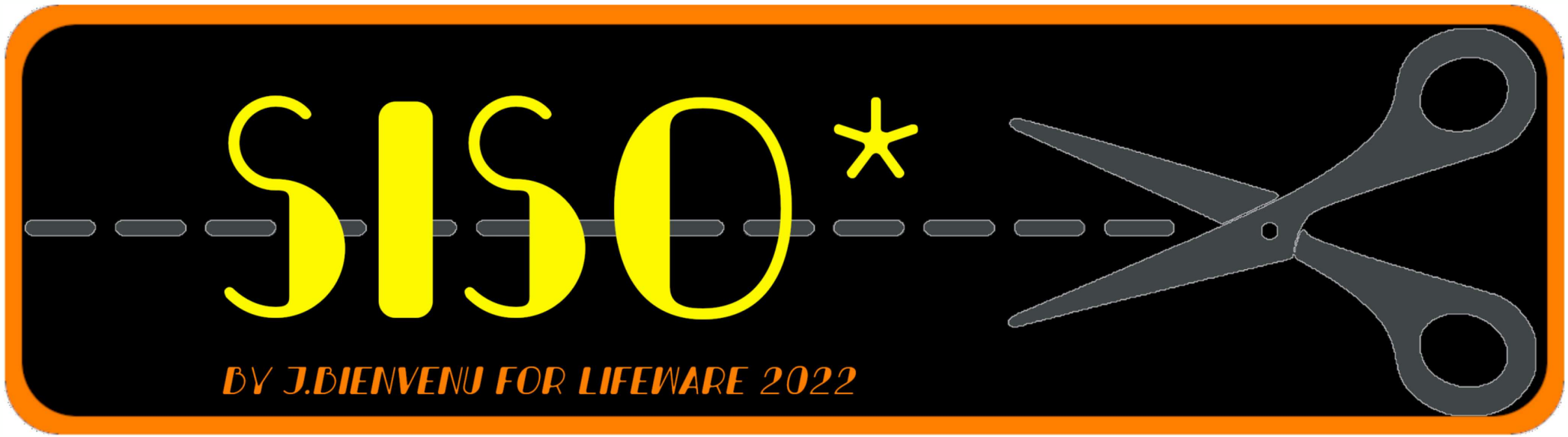
Static mapping from the abstract CRN to a concrete CRN (using Brenda as a backend module to Biocham)

Bonus objective (if there's time):

Symbolic computation theory of our CRN compilation pipeline in order to constrain it to the limited resources of a reaction catalogue in the early ODE transformation phases

Internship's objective(s)




SISO*
BY J.BIENVENU FOR LIFEWARE 2022

My approach – Dividing in three

Biocham:
Figuring out how to use and exploit the abstract CRNs that it returns

Brenda(Xplorer):
Figuring out how to use it and exploit the database of reactions it returns

Processing :
Where the magic happens

Pre-Processing Bricks

Biocham:
Figuring out how to use and exploit the abstract CRNs that it returns

Brenda(Xplorer):
Figuring out how to use it and exploit the database of reactions it returns

Processing :
Where the magic happens

Pre-Processing Bricks

Biocham:
Figuring out how to use and exploit the abstract CRNs that it returns

Brenda(Xplorer):
Figuring out how to use it and exploit the database of reactions it returns

Processing :
Where the magic happens

Pre-Processing Bricks

What do we have in Brenda ?

EC code	Enzyme Name	Organism	Reactants	Products	Reversibility	KM	Kcat	pH	…

SOAP API

- Brenda has a smart API allowing us to get an ID number for every different molecule
- Ex : for ID = 1 :

['OH-', 'H2o', 'h2O', 'H2O[side 2]', 'H3O+', '1alpha-hydroxy-vitamin D3', 'HO-', '1alpha-hydroxyvitamin D3', 'H2O']

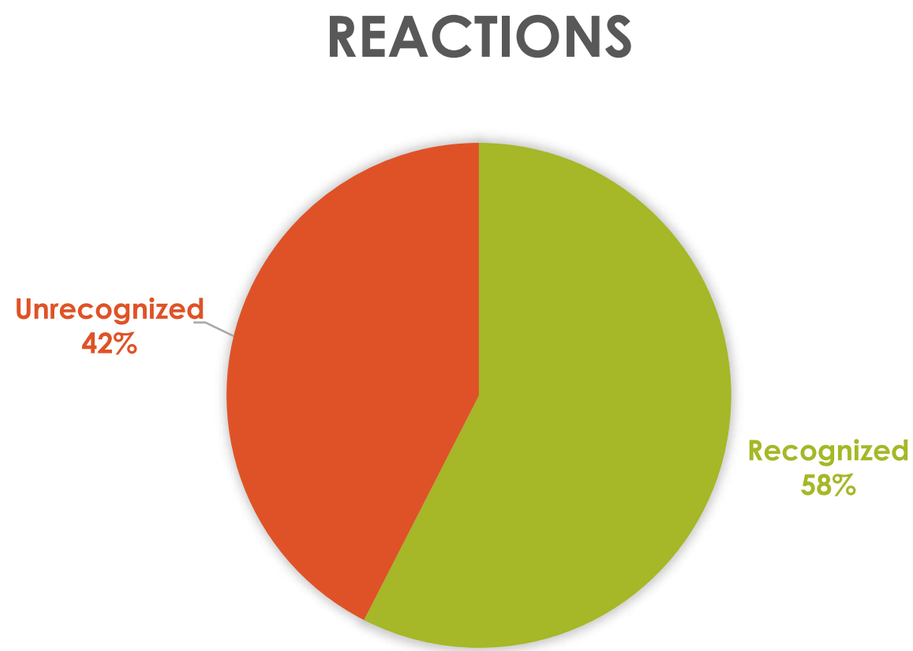
- Limited to 10 requests/second in theory (in reality ~100/mn)
- >100 000 requests leads to a long time of computing
- Some metabolites' name are still not recognized

Some statistics

304 848 reactions
116 215 different metabolite name
6475 different enzyme names
6502 different ECs

Among the 116 215, there are :
35 755 names not recognized by the API
53 162 molecules considered as different by the API

Among the 304 848 reactions of the data-base, there are 129 482 of them with at least one unrecognized metabolite.



Weird ECs

('1.14.14.143', '(+)-menthofuran synthase')
('1.14.13.104', '(+)-menthofuran synthase')
('1.14.19.7', '(S)-2-hydroxypropylphosphonic acid epoxidase')
('1.11.1.23', '(S)-2-hydroxypropylphosphonic acid epoxidase')
('1.14.21.2', '(S)-cheilanthifoline synthase')
('1.14.19.65', '(S)-cheilanthifoline synthase')
('1.14.14.99', '(S)-limonene 3-monooxygenase')
('1.14.13.47', '(S)-limonene 3-monooxygenase')
('1.14.21.1', '(S)-stylopine synthase')
('1.14.19.64', '(S)-stylopine synthase')
('1.14.14.133', '1,8-cineole 2-endo-monooxygenase')
('1.14.13.156', '1,8-cineole 2-endo-monooxygenase')
('1.14.13.53', "4'-methoxyisoflavone 2'-hydroxylase")
('1.14.14.89', "4'-methoxyisoflavone 2'-hydroxylase")
('5.3.2.8', '4-oxalomesaconate tautomerase')
('5.3.3.16', '4-oxalomesaconate tautomerase')
('5.6.2.1', 'DNA topoisomerase')
('5.99.1.2', 'DNA topoisomerase')
('5.99.1.3', 'DNA topoisomerase (ATP-hydrolysing)')
('5.6.2.2', 'DNA topoisomerase (ATP-hydrolysing)')
('1.14.13.71', "N-methylcoclaurine 3'-monooxygenase")
('1.14.14.102', "N-methylcoclaurine 3'-monooxygenase")
('3.1.11.7', "adenosine-5'-diphospho-5'-[DNA] diphosphatase")
('3.6.1.71', "adenosine-5'-diphospho-5'-[DNA] diphosphatase")
('1.14.13.158', 'amorpha-4,11-diene 12-monooxygenase')
('1.14.14.114', 'amorpha-4,11-diene 12-monooxygenase')
('1.14.13.120', 'costunolide synthase')
('1.14.14.150', 'costunolide synthase')
('1.14.14.120', 'dammarenediol 12-hydroxylase')
('1.14.13.183', 'dammarenediol 12-hydroxylase')
('4.1.3.2', 'malate synthase')
('2.3.3.9', 'malate synthase')
('1.10.3.12', 'menaquinol oxidase (H+-transporting)')
('7.1.1.5', 'menaquinol oxidase (H+-transporting)')
('1.14.13.202', 'methyl farnesoate epoxidase')
('1.14.14.127', 'methyl farnesoate epoxidase')
('1.14.19.70', 'mycocyclosin synthase')
('1.14.21.9', 'mycocyclosin synthase')
('4.3.1.5', 'phenylalanine ammonia-lyase')
('4.3.1.24', 'phenylalanine ammonia-lyase')
('1.14.13.185', 'pikromycin synthase')
('1.14.15.33', 'pikromycin synthase')
('1.14.13.121', 'premnaspirodiene oxygenase')
('1.14.14.151', 'premnaspirodiene oxygenase')
('1.14.14.121', 'protopanaxadiol 6-hydroxylase')
('1.14.13.184', 'protopanaxadiol 6-hydroxylase')
('1.14.21.4', 'salutaridine synthase')
('1.14.19.67', 'salutaridine synthase')
('1.14.19.62', 'secologanin synthase')
('1.3.3.9', 'secologanin synthase')
('1.14.13.70', 'sterol 14alpha-demethylase')
('1.14.14.154', 'sterol 14alpha-demethylase')
('1.14.13.75', 'vinorine hydroxylase')
('1.14.14.104', 'vinorine hydroxylase')

Biocham:
Figuring out how to use and exploit the abstract CRNs that it returns

Brenda(Xplorer):
Figuring out how to use it and exploit the database of reactions it returns

Processing :
Where the magic happens

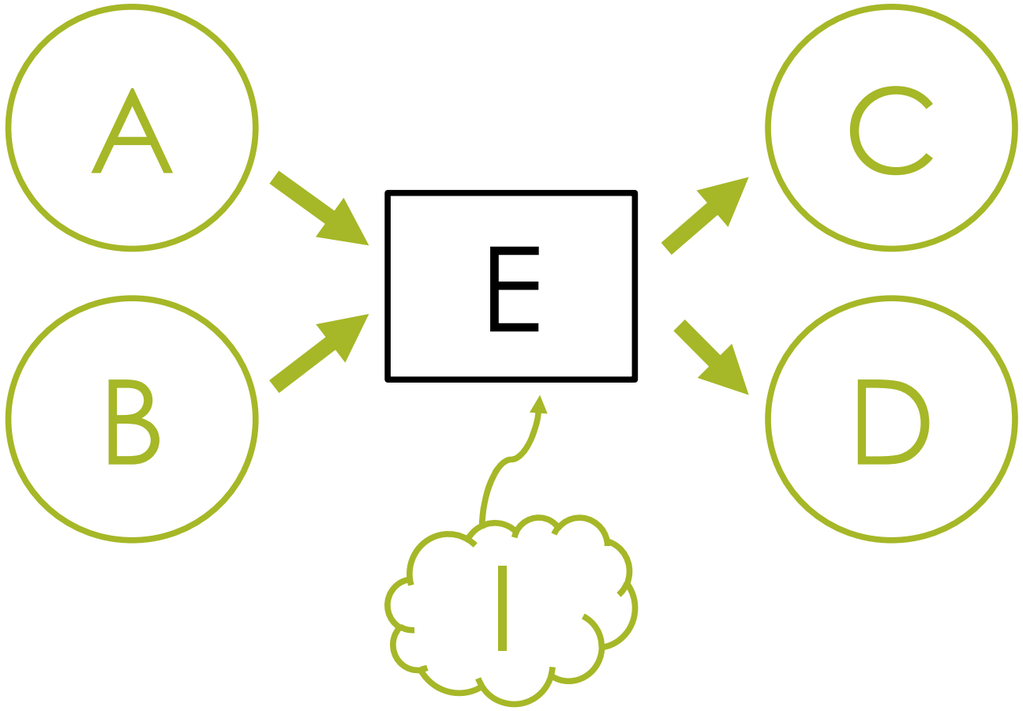
Pre-Processing Bricks

My approach – General ideas

- Various objects and classes :
  - Metabolite
  - Reaction
  - Kinetics
  - Environment…

- Loose concept of « matching »
- Abstract CRNs as graphs, with a pre-traversal algorithm
- Recursive programming, with the previous heuristic to find matching reactions

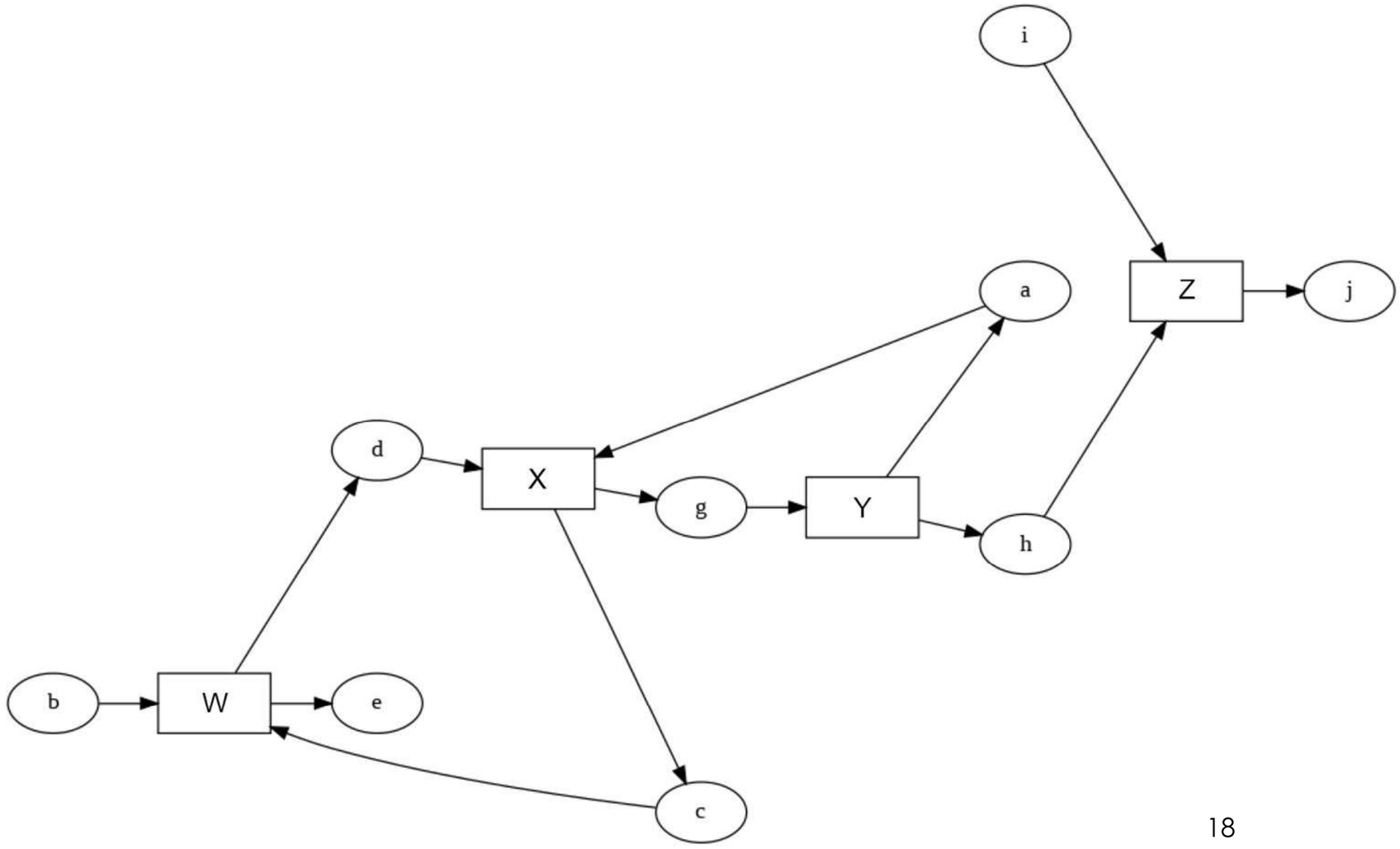
Some description of aCRNs



	E
A	in
B	in
C	out
D	out
I	inhib

Matrix of an aCRN

	W	X	Y	Z
A		in	out	
B	in			
C	in	out		
D	out	in		
E	out			
G		out	in	
H			out	in
I				in
J				out

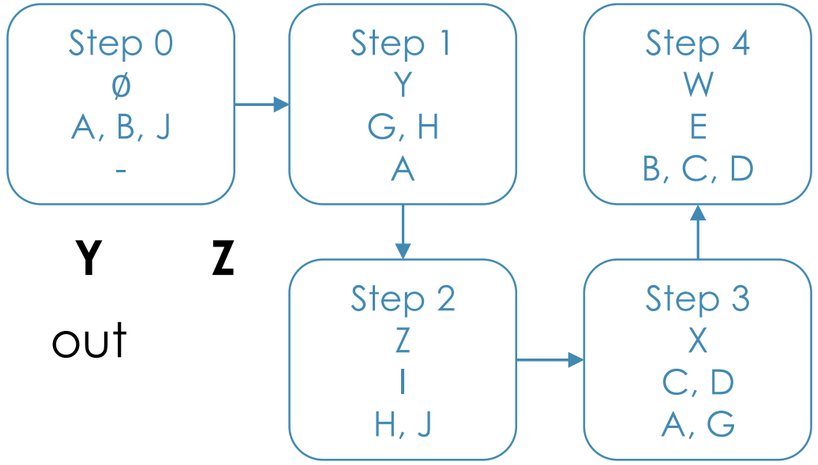


Pre-traversal of aCRN

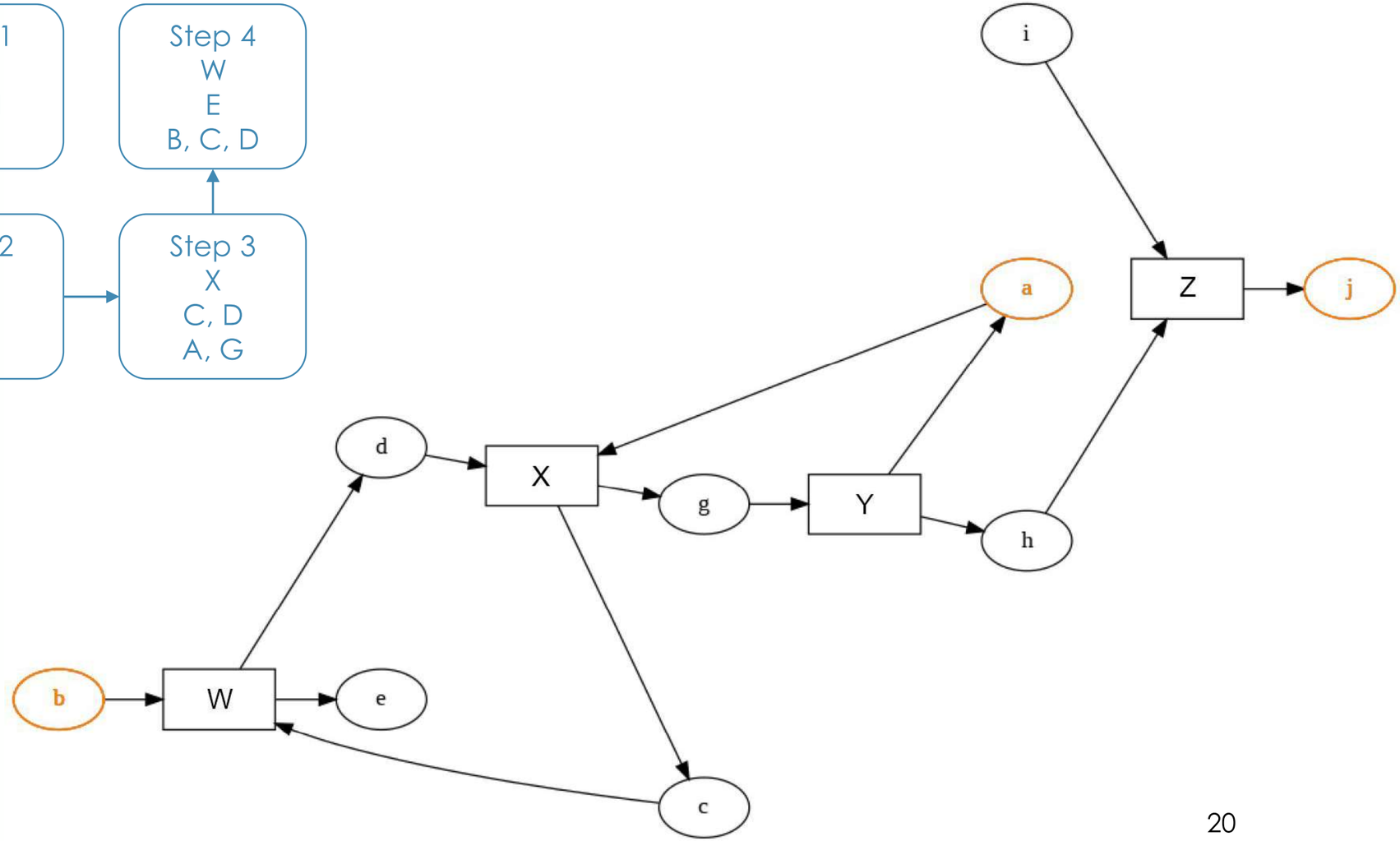
- Step 0 : Consider and color as green the predefined metabolites
- Step 1 : Select the enzyme with the most green metabolites and the least red
- Step 2 : Note the enzyme and the new filled metabolites
- Step 3 : Green-light these metabolites for all of their enzymes
- Step 4 : Go back to step 1…

Real-time demo on whiteboard !

Pre-traversal an aCRN



	W	X	Y	Z
A		in	out	
B	in			
C	in	out		
D	out	in		
E	out			
G		out	in	
H			out	in
I				in
J				out



Notations

- Abstract metabolites/enzymes will be denoted with capital letters (sometimes refered to as lettered-metabolites or lettered-enzymes orally)
- A step of the pre-traversal is described as follow :
  - An enzyme E to fill
  - A series of metabolites (Ai) to fill
  - A series of already filled metabolites (Xj) which are involved in E

Final pipeline

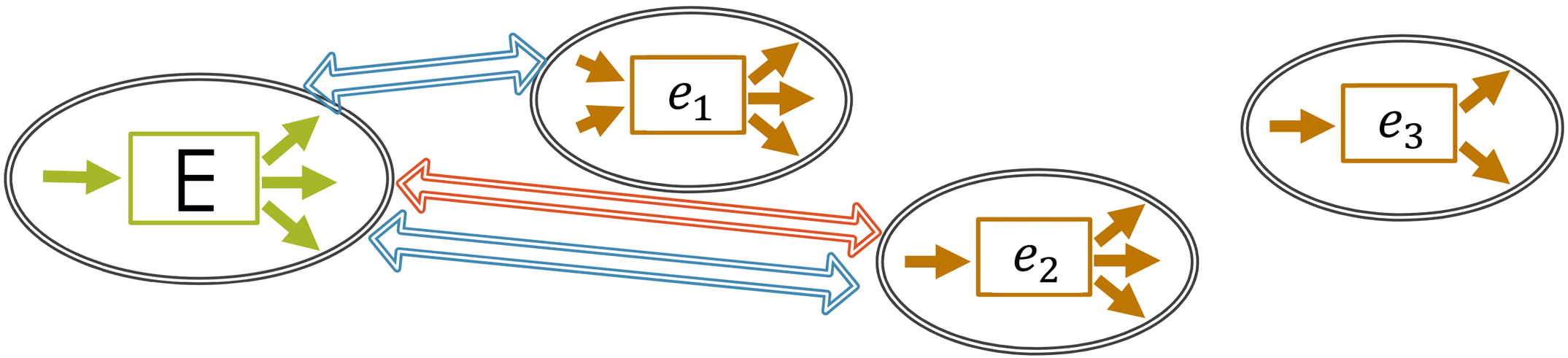
- Pre-traversal optimises the pipeline : aims to show incompatibilities ASAP
- Pre-selecting enzyme candidates for every abstract one in advance, matching with it's abstract kinetic
- Recursive programming

When do abstract and concrete reactions match ?

« hard-match » exactly same number of ins and outs

« soft-match » abstract has less ins and less outs than concrete
→ Grey-listing

- « kinetics-match » : takes kinetics into account as well
- Consequences on the graphs considered !



Recursive algorithm

For the n-th step proceed as follows :

- If $n \geq$number of enzymes, (D_E, D_M, G_M) is a solution
- Else, let E, A, X be a description of the n-th step and K set of candidates for E
  - For $c \in K$, if $c \in B_E$ or $x \notin c.in \cup c.out$, c is discarded
  - For every remaining c of K :
    - Let $e_{in} = c.in\ (x.in \cup G_M)$ and $e_{out} = c.out\ (x.out \cup G_M)$.
    - If $|e_{in}| < |A.in|$ or $|e_{out}| < |A.out|$, c is discarded
    - Else : $D_E[E] \leftarrow c$ and are considered every $|A.in|$-permutation ai of e_{in} and $|A.out|$-permutation ao of e_{out}
    - For every couple (ai, ao) : $D_M[A.in] \leftarrow ai$, $D_M[A.out] \leftarrow ao$, and the step $n+1$ is initiated with :
      - $B_M \leftarrow B_M \cup ai \cup ao$
      - $B_E \leftarrow B_E \cup c$
      - $G_M \leftarrow G_M \cup (e_{in} \setminus ai) \cup (e_{out} \setminus ao)$.

Case Study – D-glucose Λ Acetone → Resorufin

- Same aCRN as this one, but with letters
- First goal : find the same solution as this one
- Secondary goal : acquire new solutions
- Resazurin → Resorufin still not found
- Added end step
- 2-propanone == acetone
- Some complete results !

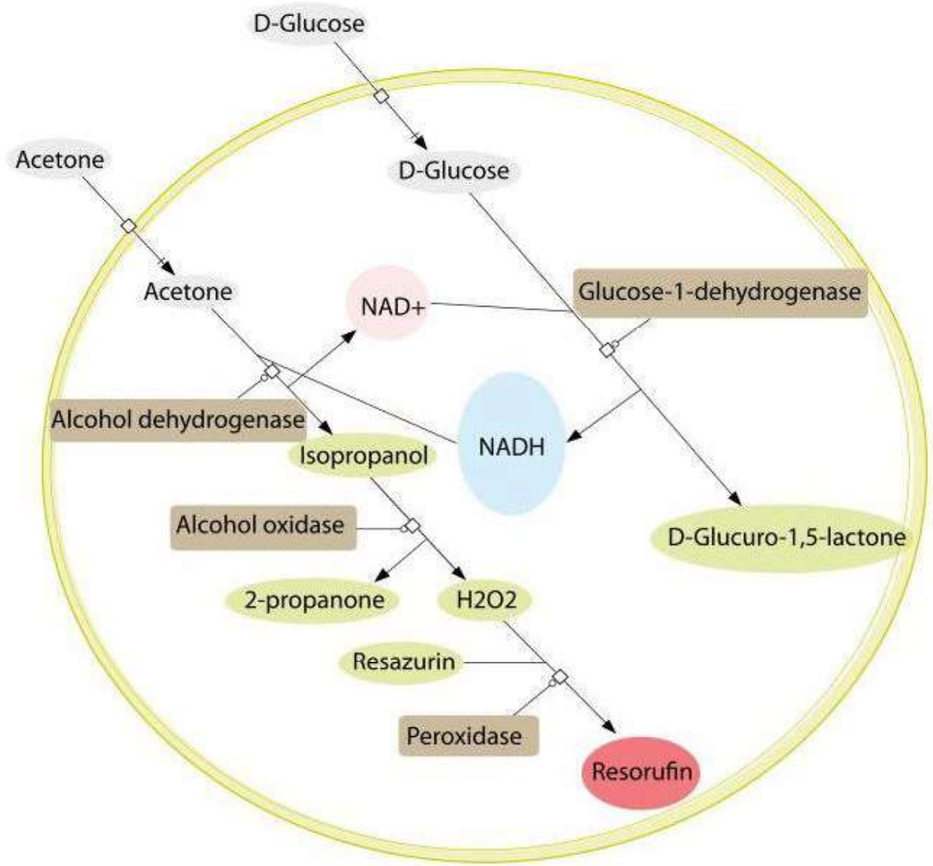


Figure 2.

Many new results

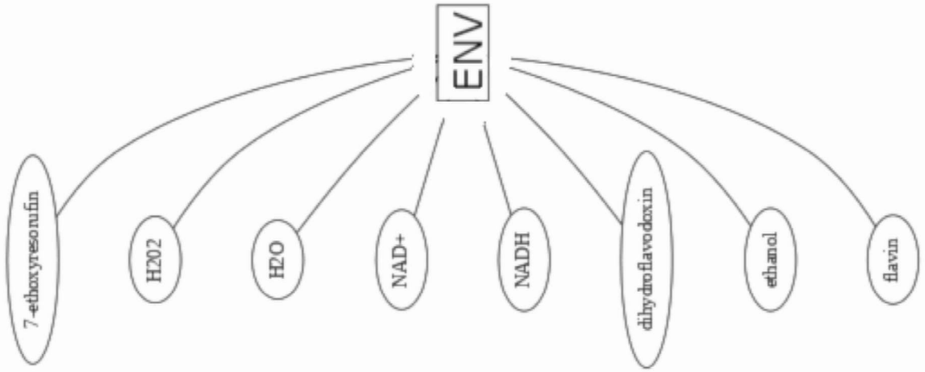
- ~600 with distinct enzymes for 9h computation time on a regular computer
- Not kinetic dependent
- Quite convincing
- But really time-costly to get every solution)(doesn't seem usable with a more SEPI-like search)

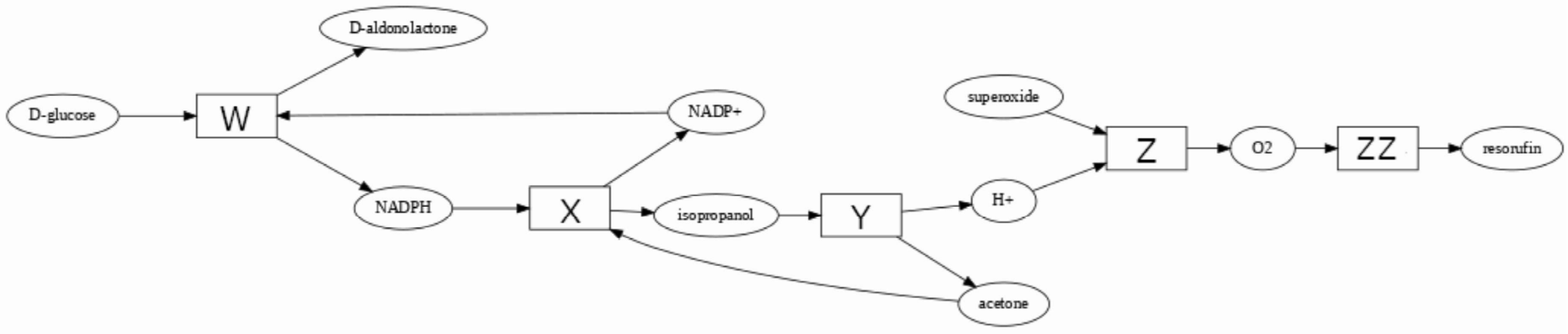
Some of the solutions

- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'aldose 1-dehydrogenase (NAD+)'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'D-xylose 1-dehydrogenase'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'inositol 2-dehydrogenase'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'glucose/galactose 1-dehydrogenase'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'glucose-6-phosphate 3-dehydrogenase'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'aldehyde dehydrogenase (NAD+)'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'L-glutamate gamma-semialdehyde dehydrogenase'}
- {'Y': 'alcohol oxidase', 'X': 'carbonyl reductase (NADPH)', 'W': 'alcohol dehydrogenase (NADP+)'}
- {'Y': 'alcohol oxidase', 'X': 'carbonyl reductase (NADPH)', 'W': 'glucose/galactose 1-dehydrogenase'}
- {'Y': 'alcohol oxidase', 'X': 'carbonyl reductase (NADPH)', 'W': 'alcohol dehydrogenase [NAD(P)+]'}
- {'Y': 'alcohol oxidase', 'X': 'carbonyl reductase (NADPH)', 'W': '(R)-aminopropanol dehydrogenase'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase (NADP+)', 'W': 'glucose/galactose 1-dehydrogenase'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase (NADP+)', 'W': 'alcohol dehydrogenase [NAD(P)+]'}
- {'Y': 'alcohol oxidase', 'X': 'alcohol dehydrogenase (NADP+)', 'W': '(R)-aminopropanol dehydrogenase'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'D-xylose 1-dehydrogenase'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'inositol 2-dehydrogenase'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'glucose/galactose 1-dehydrogenase'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'aldehyde dehydrogenase (NAD+)'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'alcohol dehydrogenase', 'W': 'L-glutamate gamma-semialdehyde dehydrogenase'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'carbonyl reductase (NADPH)', 'W': 'alcohol dehydrogenase (NADP+)'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'carbonyl reductase (NADPH)', 'W': 'glucose/galactose 1-dehydrogenase'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'carbonyl reductase (NADPH)', 'W': 'alcohol dehydrogenase [NAD(P)+]'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'carbonyl reductase (NADPH)', 'W': '(R)-aminopropanol dehydrogenase'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'alcohol dehydrogenase (NADP+)', 'W': 'glucose/galactose 1-dehydrogenase'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'alcohol dehydrogenase (NADP+)', 'W': 'alcohol dehydrogenase [NAD(P)+]'}
- {'Y': 'secondary-alcohol oxidase', 'X': 'alcohol dehydrogenase (NADP+)', 'W': '(R)-aminopropanol dehydrogenase'}
- …

Case Study – a couple of solutions

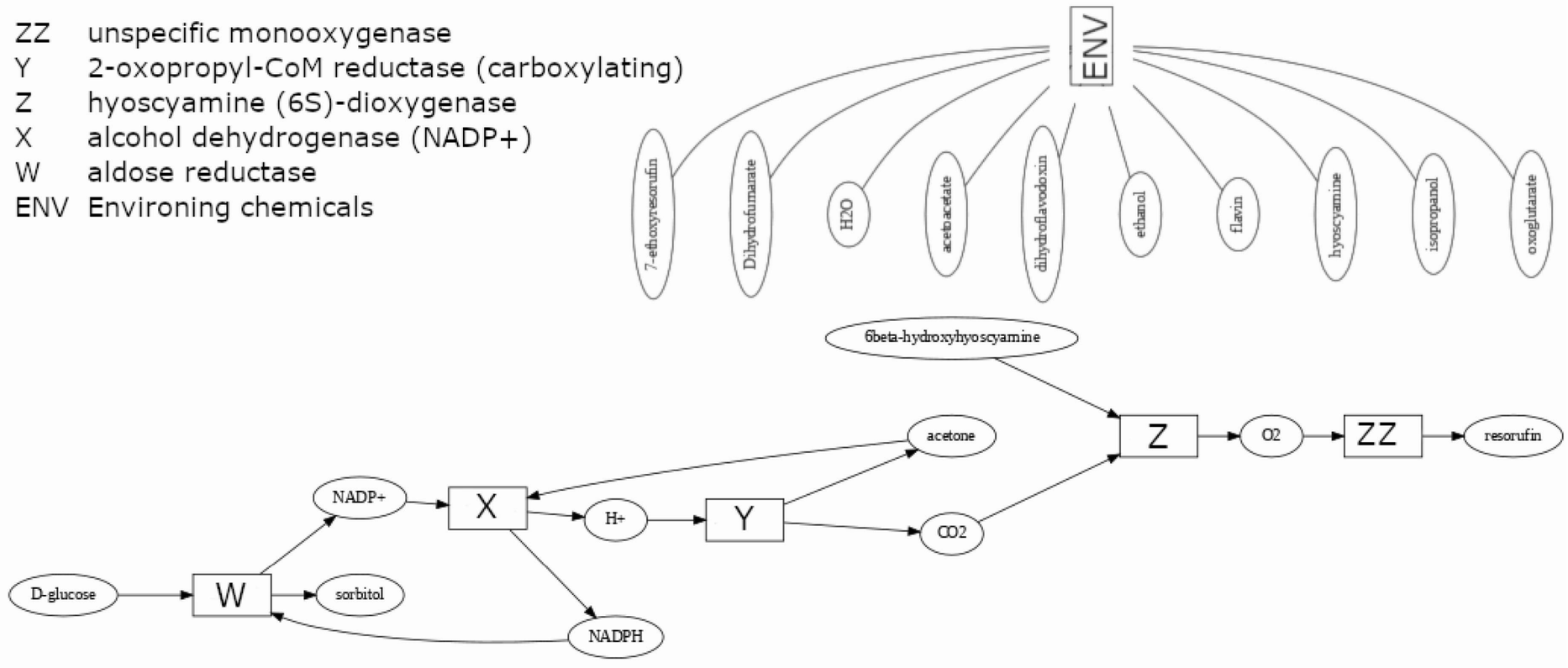
ZZ unspecific monooxygenase
Y methanol dehydrogenase
Z superoxide dismutase
X alcohol dehydrogenase (NADP+)
W D-xylose 1-dehydrogenase (NADP+, D-xylono-1,5-lactone-forming)
ENV Environing chemicals





Case Study – a couple of solutions

ZZ unspecific monooxygenase
Y 2-oxopropyl-CoM reductase (carboxylating)
Z hyoscyamine (6S)-dioxygenase
X alcohol dehydrogenase (NADP+)
W aldose reductase
ENV Environing chemicals



Kinetics – Chain contraction

$$a_0 \xrightarrow{MA(k_0)} a_1 \xrightarrow{MA(k_1)} \dots \xrightarrow{MA(k_{n-1})} a_n \xrightarrow{MA(k_n)} b$$

With $D : u \mapsto \frac{du}{dt}$ and $Q_{(k_j)_{0 \le j \le n}} = \frac{X}{\prod_{j=0}^{n} k_j} \prod_{j=1}^{n} (X + k_j)$

b and a_0 can be linked by $a_0 = Q_{(k_j)_{0 \le j \le n}} \circ \mathrm{D(b)}$

What can be deduced from it on a kinetics standpoint ?

$$a_0 \overset{?}{\to} b$$

Kinetics – Chain contraction proof

$$a_0 \xrightarrow{MA(k_0)} a_1 \xrightarrow{MA(k_1)} \dots \xrightarrow{MA(k_{n-1})} a_n \xrightarrow{MA(k_n)} b$$

- $\frac{db}{dt} = k_n a_n \qquad \frac{da_0}{dt} = -k_0 a_0 \qquad \frac{da_j}{dt} = k_{j-1} a_{j-1} - k_j a_j$
- $n = 0$: $\frac{db}{dt} = k_0 a_0$ ie $Q_{k_0} = \frac{X}{k_0}$
- $n \mapsto n+1$:

  $\frac{da_1}{dt} = k_0 a_0 - k_1 a_1$ ie $\frac{1}{\mathrm{k_0}}\left(\frac{\mathrm{d}}{\mathrm{dt}} + \mathrm{k_1}\right)(a_1) = a_0$

  so $Q_{(k_j)_{0 \le j \le n+1}} = \frac{1}{\mathrm{k_0}}(X + \mathrm{k_1}) Q_{(k_j)_{1 \le j \le n+1}}$ ■

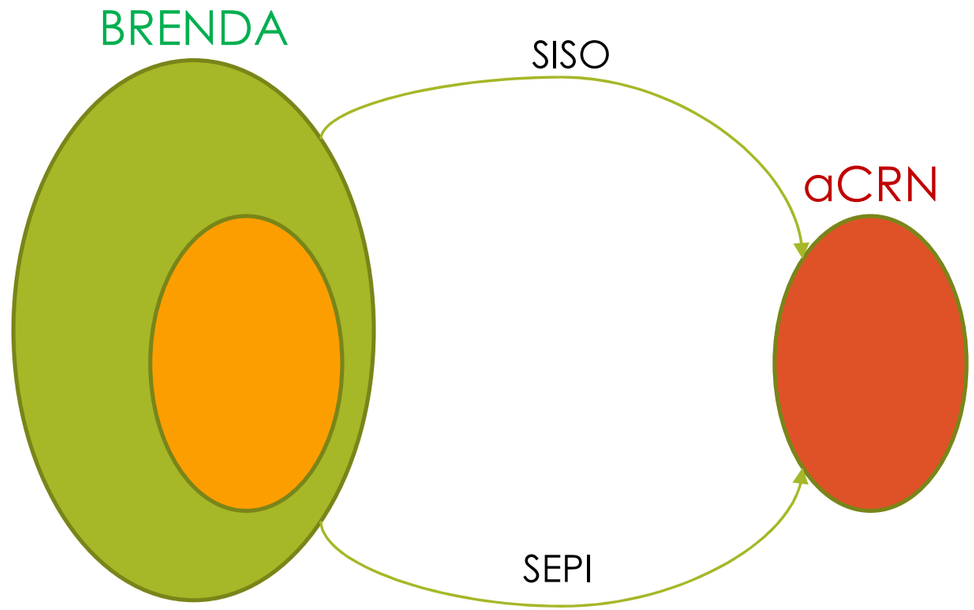
Kinetics - Independence

Hypothesis for the following reductions :

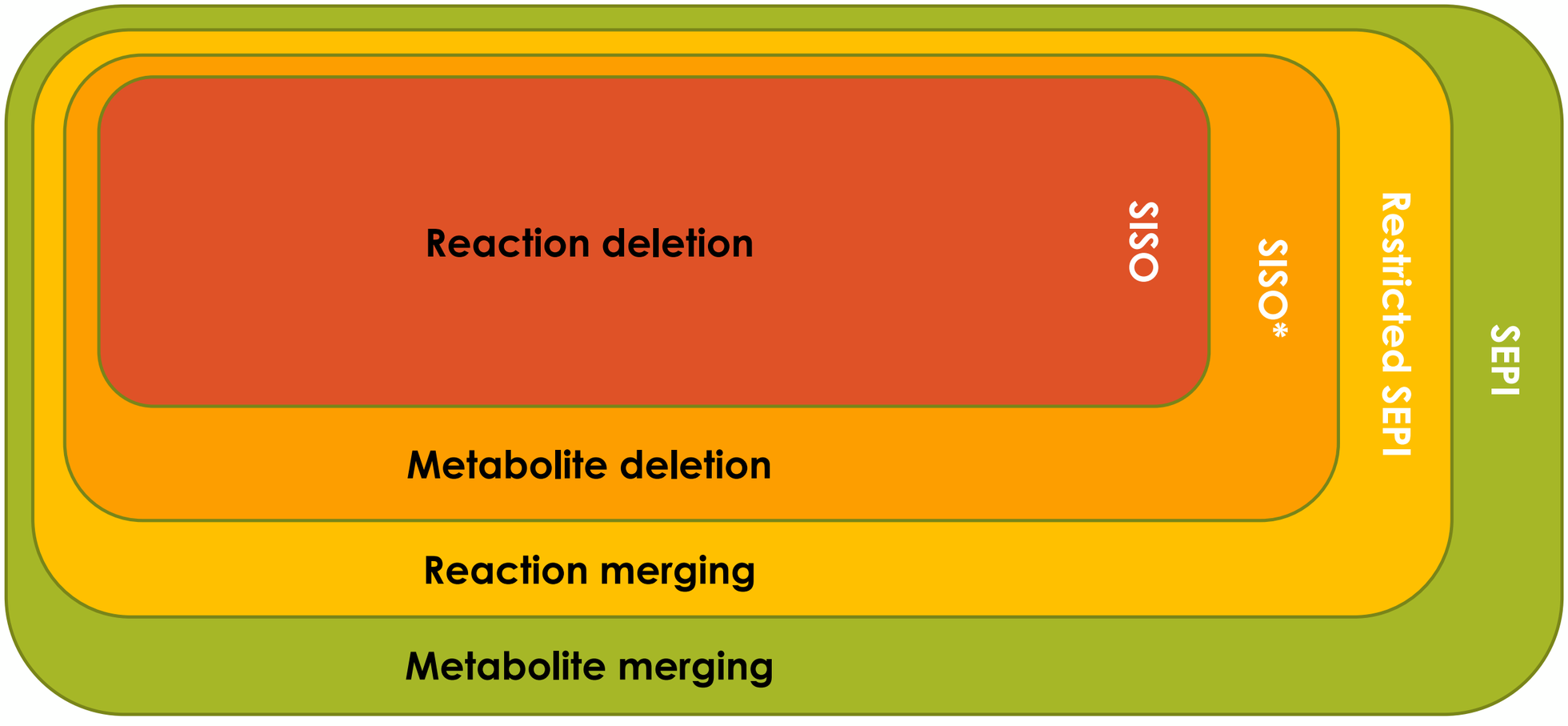
The aCRN considered are « morally » rate-independent.

SEPI, SISO…

- *SEPI :* subgraph epimorphism
- *SISO :* subgraph isomorphism
- We will describe them by the operations they permit



SEPI, SISO… – Nuances



SEPI, SISO – Implementations

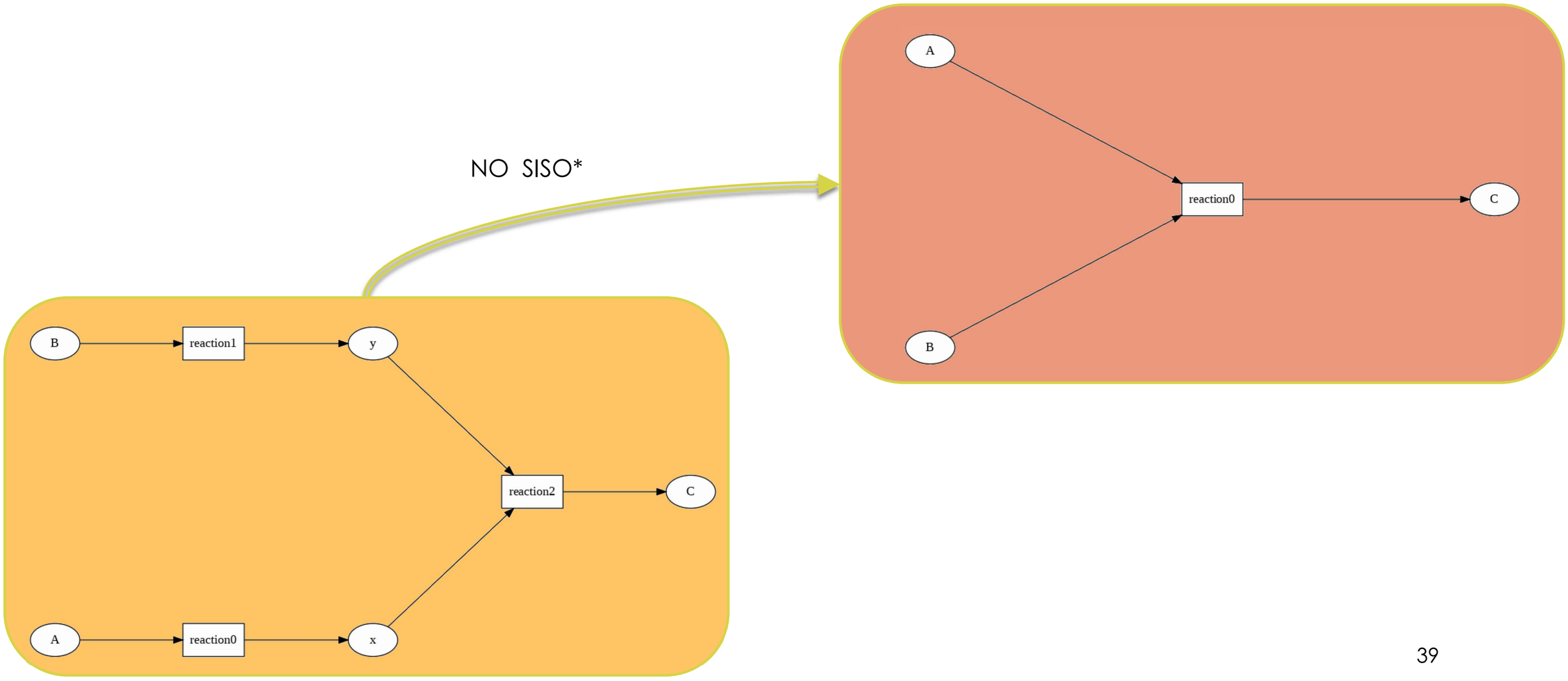
BIOCHAM

- SEPI
- Restricted SEPI
- But only up to ~100 reactions

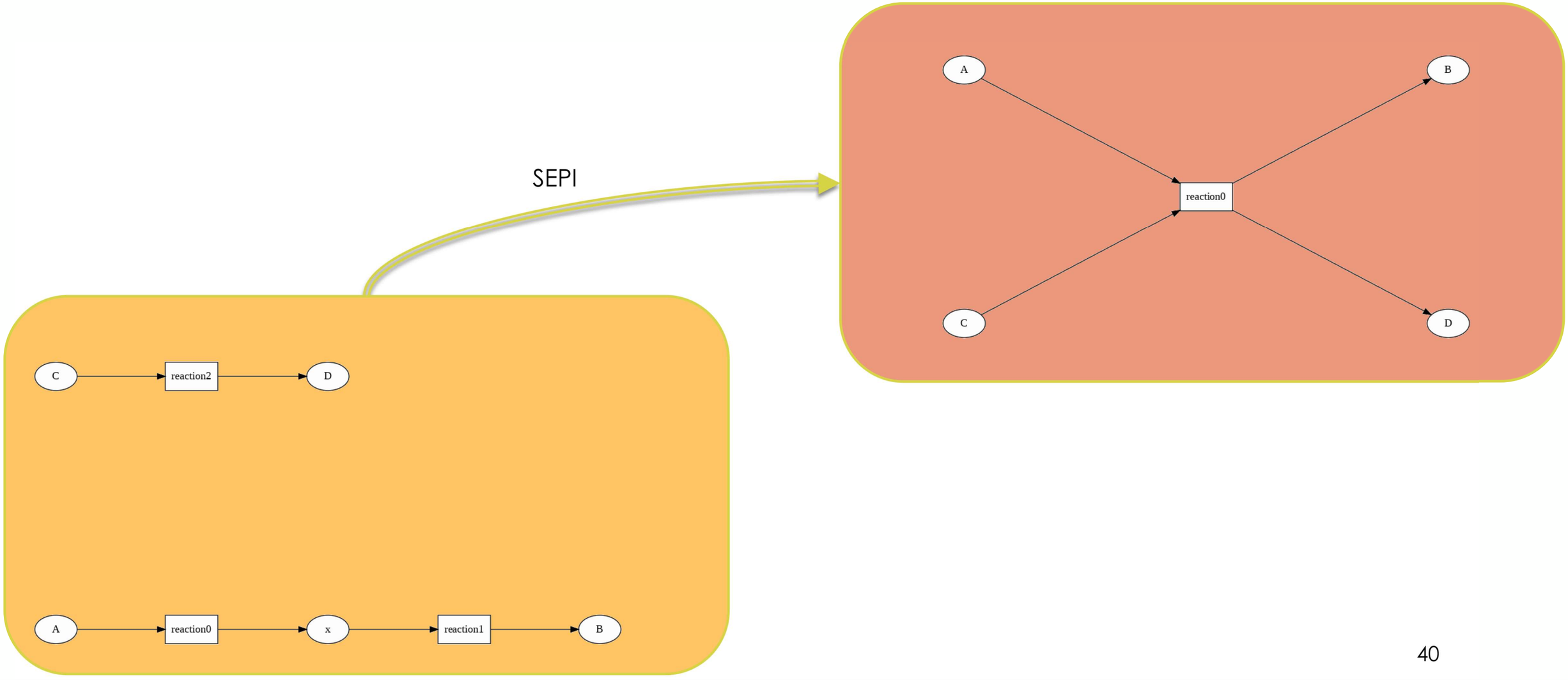
My Pipeline

- SISO
- SISO*
- Can handle BRENDA

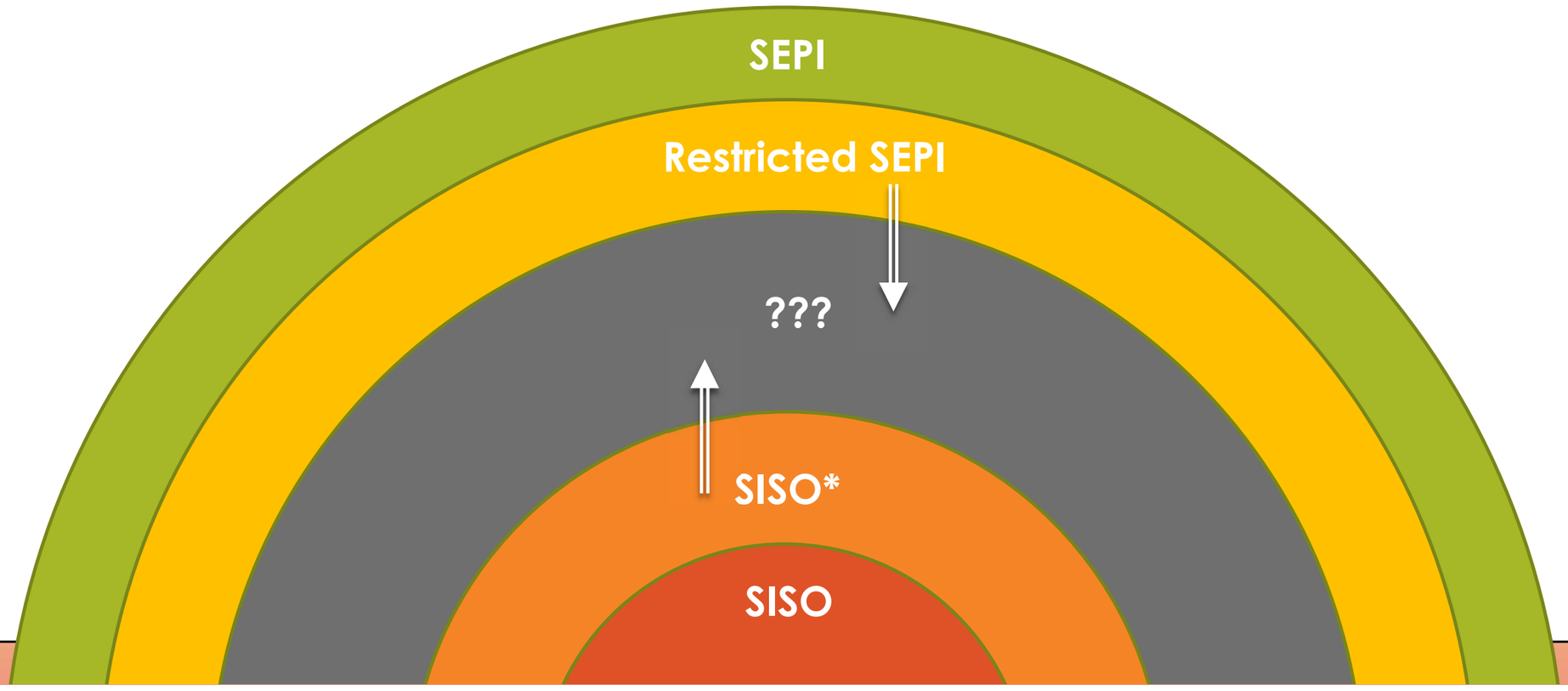
SISO* – Too restrictive



Restricted SEPI – Too lenient



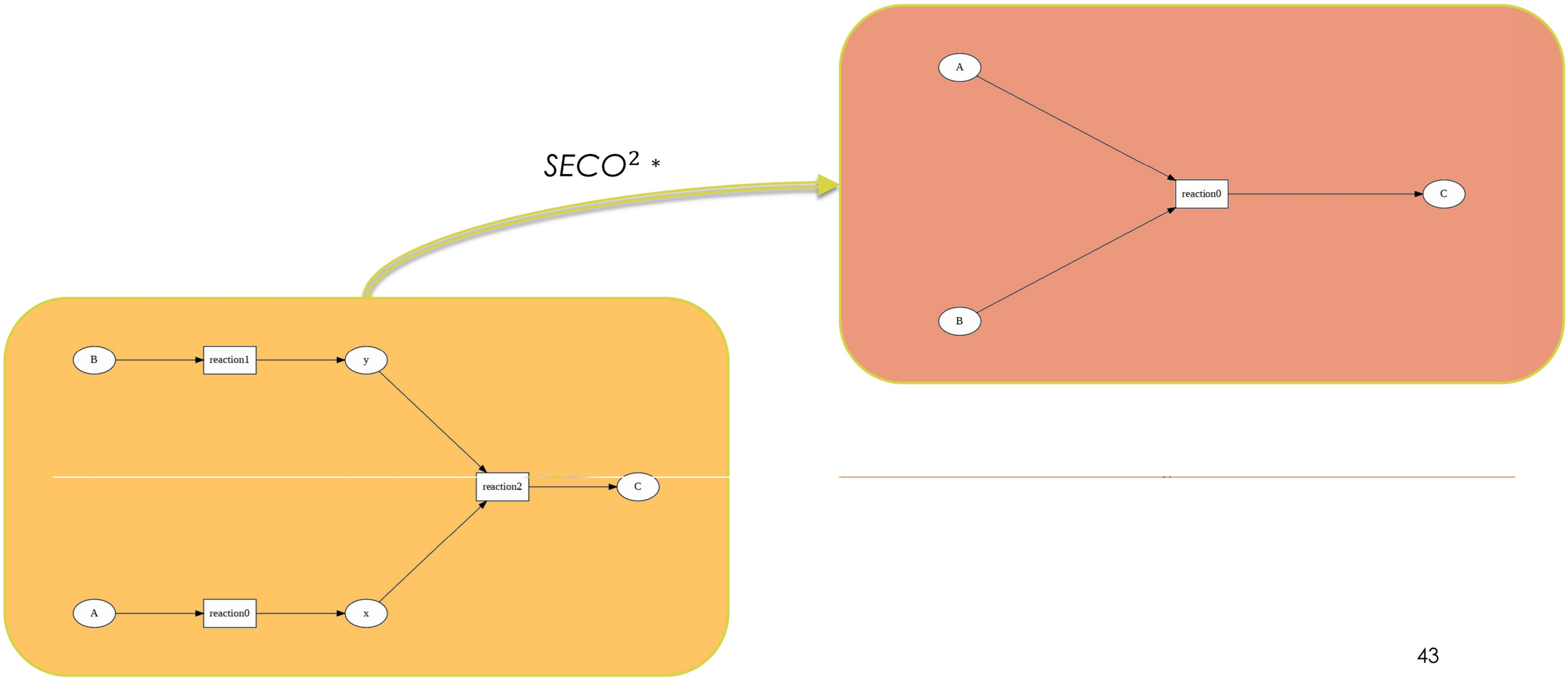
SEPI, SISO – Somewhere in between



SEPI, SISO – Solution ?

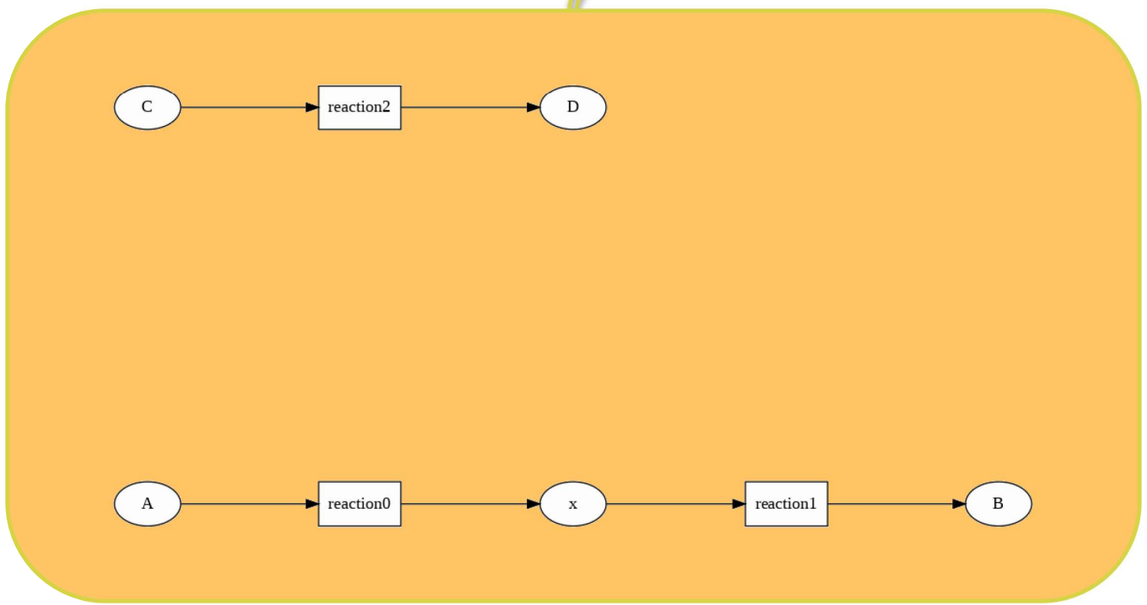
- The concretization morphism is somewhere between SISO* and restricted SEPI
- No definitive solution as of now but some ideas have emerged the past week, including :
- *SECO*^{2} (« subgraph epimorphism connexity-conserving »), merges reactions only if the resulting connexity is weaker than weaker than the original.

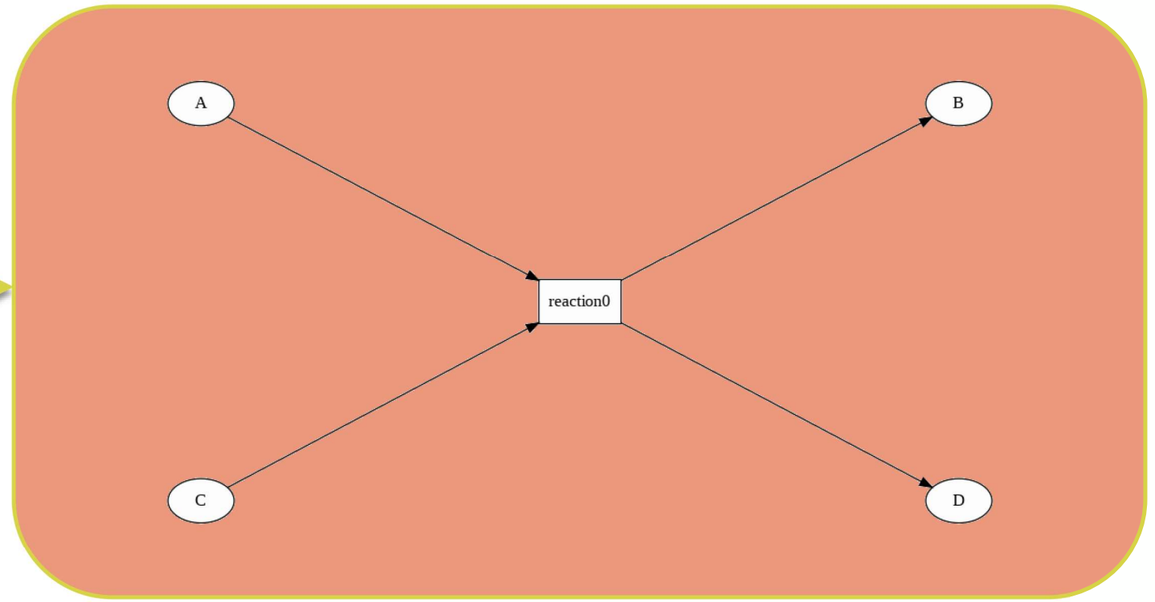
$SECO^2$– Too restrictive ?



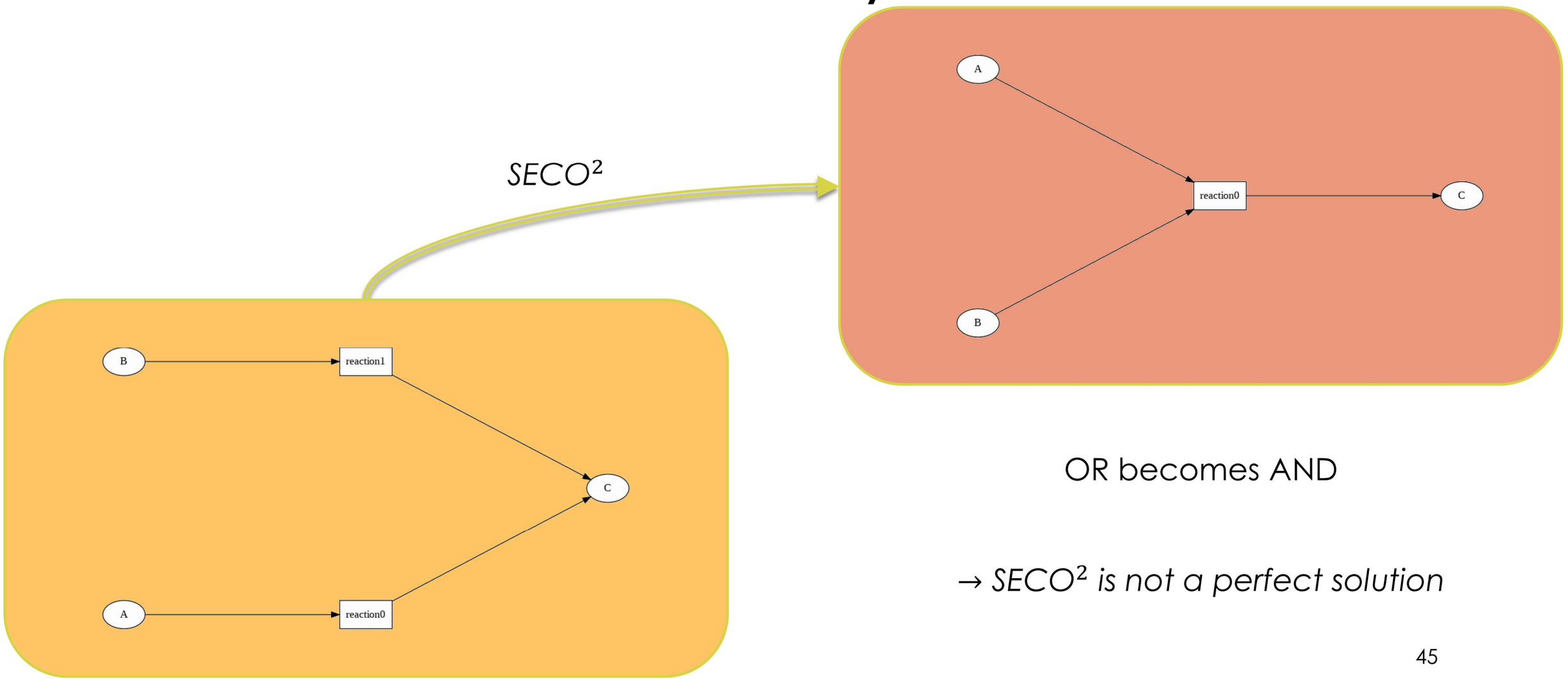
SECO^2– Too lenient ?

No *SECO*^{2}





SECO² – Solves many issues but…



OR becomes AND

→ *SECO² is not a perfect solution*

What about now ?

- Kinetics – Adding them to *SISO* ⋆, how contractions affects them…
- Inhibtors not yet incorporated
- Between *SECO*^{2} and *SISO* ⋆, find the morally correct description of what we are looking for
  - *SECO*^{2} → more restrictions to the fusions ?
  - *SISO* ⋆ → automate insertions ?