RBA

S. Fischer - Biosys - MAIAGE

June 2, 2017

Contents

1. Principles underlying algorithm **3**

2. Analysis of existing algorithm (RBAV01) **3**
2.1. Structure of RBAV01 3
2.2. Orientations for improving RBA algorithm 5

3. Rewriting RBAs in full matrix form **6**

4. New formalism **9**
4.1. Final differences between RBAv01 and new formalism 9
4.2. Input data 9
4.3. From data to matrices 9

A. RBA XML **13**

B. Small bug in RBAv01 **14**

1. Principles underlying algorithm

Figure 1 shows how the fluxes of molecules are described by RBA and where the constraint are placed.

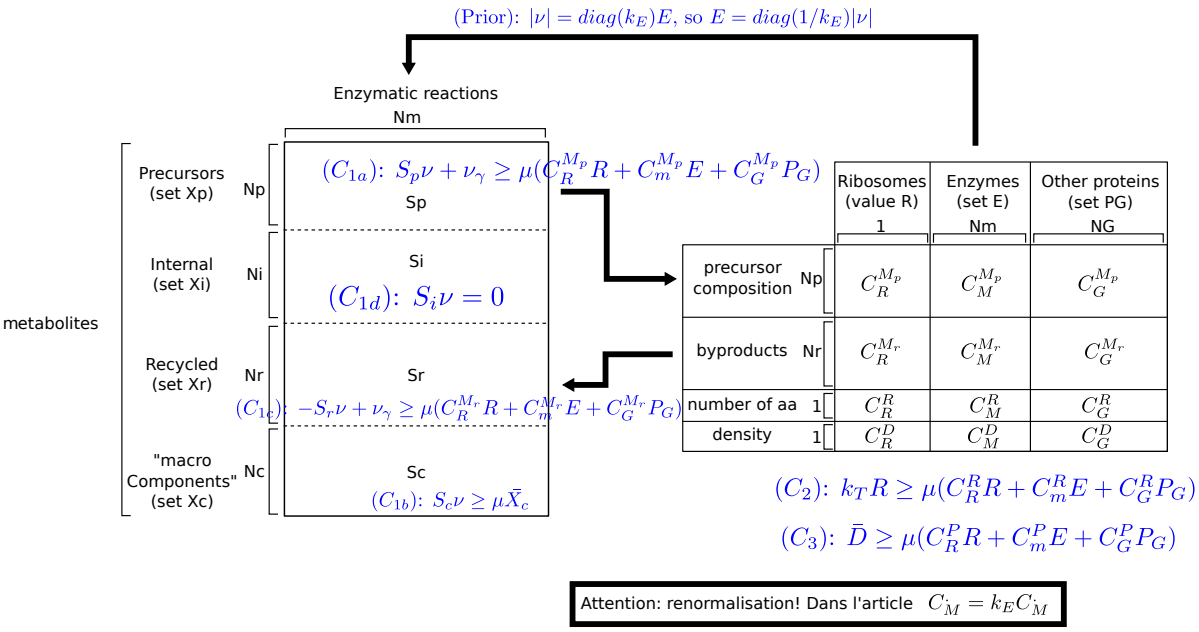


Figure 1: Description of cell by RBA.

2. Analysis of existing algorithm (RBAV01)

2.1. Structure of RBAV01

The original program is built around a multitude of functions and structures displayed on Figure 2. Figure 3 shows how matrices are built in the original algorithm.

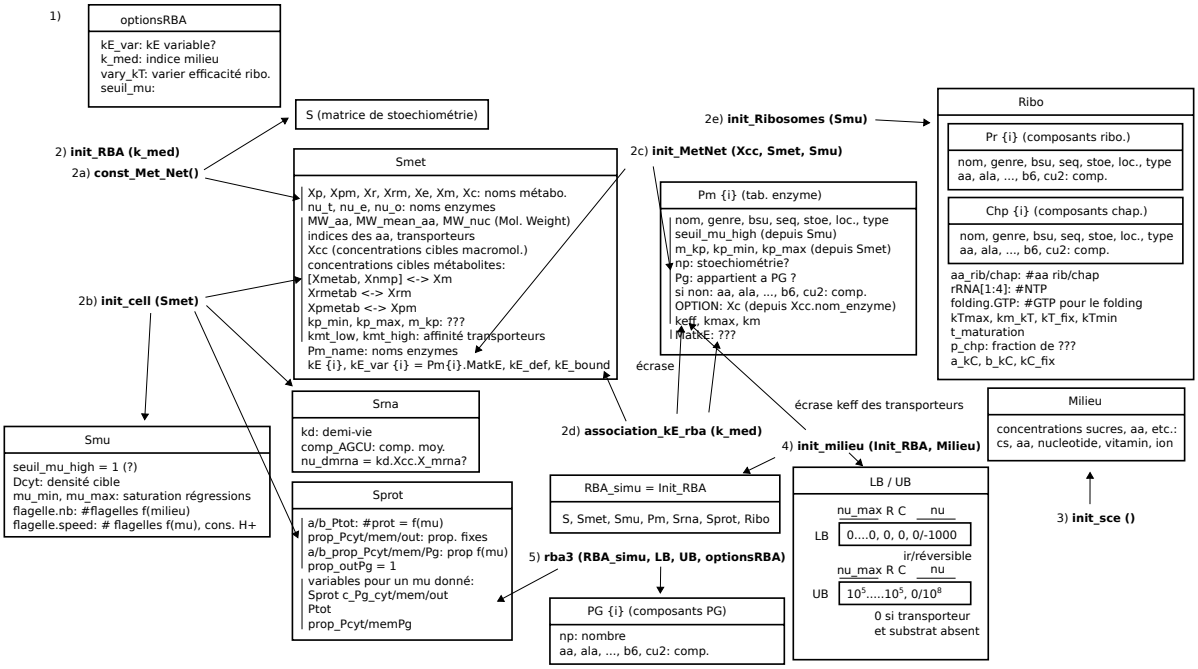


Figure 2: Algorithm used in RBAV01.

$$
\begin{array}{r|cccc}
 & \textbf{nu\_max} & \textbf{R} & \textbf{C} & \textbf{nu} \\
\textbf{f (fonction objective)} & 1 & 0 & 0 & 0 \\
\textbf{D (densité cytosol)} & \text{CD\_M} & \text{CD\_R} & \text{CD\_C} & \\
\textbf{M (densité membrane)} & \text{CD\_M} & & & \\
\textbf{taille Nm (flux)} & -\text{Pm.keff} & & & 1 \\
\textbf{taille Nm (flux)} & -\text{Pm.keff} & & & -1 \\
\textbf{ATP maintenance} & & & & \text{nu\_ATP}=-1
\end{array}
\quad \textbf{= A} \leq \textbf{b =} \quad
\begin{bmatrix}
\text{D - CD\_G.C\_PGcyt} \\
\text{Ptot.Pmem -CD\_G.C\_PGmem} \\
\\
\\
\\
\text{-atp\_maintenance}
\end{bmatrix}
$$

$$
\begin{array}{r|cccc}
\textbf{Ribosomes} & \text{muCR\_M} & \text{muCR\_R} & \text{muCR\_C} & \\
 & & -\text{kT.p\_act} & & \\
\textbf{Chaperones} & \text{muCC\_M} & \text{muCC\_R} & \text{muCC\_C} & \\
 & & & -\text{kC} & \\
\textbf{Xe} & & & & \text{Se} \\
\textbf{Xp} & \multicolumn{3}{c}{\multirow{2}{*}{muCMp\_M muCMp\_R muCMp\_C}} & \text{-Sp} \\
\textbf{Xpm} & & & & \text{-Spm} \\
\textbf{Xr} & \multicolumn{3}{c}{\multirow{2}{*}{muCMr\_M muCMr\_R muCMr\_C}} & \text{Sr} \\
\textbf{Xrm} & & & & \text{Srm} \\
\textbf{Xc} & & & & \text{-Sc} \\
\textbf{Xm} & & & & \text{-Sm} \\
\textbf{Coût flagelles} & & & & \text{nu\_H+=-1}
\end{array}
\quad \textbf{= Aeq = beq =} \quad
\begin{bmatrix}
\text{-CR\_G.mu.C\_PG} \\
\text{-CR\_G.pch.mu.C\_PG} \\
0 \\
\begin{bmatrix}\text{-muCMp\_G}\end{bmatrix} + \begin{bmatrix}0 \\ \text{-mu.Xp\_metab}\end{bmatrix} \\
\begin{bmatrix}\text{-muCMr\_G}\end{bmatrix} + \begin{bmatrix}0 \\ \text{-mu.Xr\_m}\end{bmatrix} \\
\text{-muXc} \\
\text{-muX\_metab} \\
\text{-muX\_nmp} \\
\text{-h\_flagelle}
\end{bmatrix}
+
\begin{bmatrix}
 \\
 \\
 \\
\text{-nu\_dmrna (X\_H2O)} \\
\text{nu\_dmrna (X\_H+)} \\
\text{-kd.Xc (X\_mrna)} \\
\text{comp\_AGCU.nu\_dmrna}
\end{bmatrix}
$$

mRNA

Figure 3: Matrices used in RBAV01.

2.2. Orientations for improving RBA algorithm

It would be more elegant and flexible to not name any metabolite or flux sets. These sets should be defined by data, not the program. The existence of $X_{e/c/r/rm/p/pm/m}$ is unnecessary. Note that there are even more subsets than defined in theory, we probably need to avoid proliferation of such subsets.

We need to find clear abstraction that can represent RBA entities as generically as possible. All macromolecules should be representable with a composition matrix that would help build the final matrices rapidly. We also need to express the second member as data, in terms of processes.

Final objectives are

- Completely separate data from code: Create data files that are meaningful to users.
- Make code more compact and easier to read (and potentially quicker): by using a full matrix formalism (avoiding loops). This can only be achieved by finding the right abstraction to represent RBA elements.

3. Rewriting RBAs in full matrix form

Composition vector for macromolecules In the following, each macromolocule is described by its **composition vector**. It is a column vector containing the metabolites necessary to build it, with a **minus** sign for metabolites consumed and a **plus** sign for byproducts generated.

Metabolism constraint The metabolism constraint (C_1) can be rewritten

$$\overbrace{\underbrace{S\nu}_{\substack{\text{metabolic flux}\\ \text{generated by}\\ \text{metabolism}}} + \underbrace{\mu[C_E, C_R, C_C][E; R; C]}_{\substack{\text{precursors}\\ \text{used/byproducts}\\ \text{generated by produc-}\\ \text{ing new molecules}}}}^{\text{'Variable' terms}} = \overbrace{\underbrace{-\mu[C_G, C_{X_c}][P_G; X_c]}_{\substack{\text{precursors}\\ \text{used/byproducts}\\ \text{generated by produc-}\\ \text{ing new molecules}}}}^{\text{Constant terms}}$$

$$-\text{diag}\,(k_E)\,E \leq \nu \leq \text{diag}\,(k_E)\,E$$

where C_i are composition matrices as defined above.

Written this way, it actually does not matter to which group a metabolite belongs. Its group is only dictated by the structures of the composition matrices. Based on this constraint only it does not make sense to create metabolic groups *in the program*.

In full matrix form, the above equations become

$$\begin{cases} [S, \mu C_E, \mu C_R, \mu C_C][\nu; E; R; C] & = & -[\mu C_G, \mu C_{X_c}][P_G; X_c] \\ \begin{bmatrix} I, & -\text{diag}\,(k_E)\,, & 0, & 0 \\ -I, & -\text{diag}\,(k_E)\,, & 0, & 0 \end{bmatrix} [\nu; E; R; C] & \leq & 0 \end{cases}$$

The most important here is the composition matrix. In the data, the user gives the list of metabolites used for synthesis and the list of byproducts. The program then has to figure out how to reorder terms to build the matrix.

Ribosome/chaperone constraints We can rewrite the ribosome constraint as follows:

$$k_T R = \mu[PC_E^R, PC_R^R, PC_C^R][E; R; C] + \mu PC_G^R P_G$$

where PC^R is the Processing Cost of a molecule and k_T the capacity of a ribosome (*e.g.* the capacity of a ribosome is the number of aas it can process and the processing cost is the number of amino acids of a protein). In full form:

$$[0, \mu PC_E^R, k_T + \mu PC_R^R, \mu PC_C^R][\nu; E; R; C] = [-\mu PC_G^R, 0][P_G; X_c]$$

The same applies for folding by chaperones.

These additionnal constraints have the form

$$a[\nu; E; R; C] = b_0 + b_1[P_G; X_c]$$

this means they can be appended to C_1 by **simple line concatenation**.

RNA degradation/replication In this type of constraints, we simply add new fluxes of metabolites. These fluxes are simply added up on the right hand side of (C_1)

$$\begin{aligned} ... &= -\mu[C_G, C_{X_c}][P_G; X_c] - \mu^d C_{process} X_{process} \\ &= -[\mu C_G, \mu C_{X_c}, \mu^d C_{process}][P_G; X_c; X_{process}] \end{aligned}$$

where $d = 0$ when the target flux is an abolute flux and $d = 1$ when it compensates dilution.

Density constraints The density constraint writes

$$[0, W_E, W_R, W_C][\nu; E; R; C] \leq \overline{D} - [W_G, W_c][P_G; X_c]$$

where W is the weight of each molecule. In general, these constraints can be expected to be of the form

$$a[\nu; E; R; C] \leq b_0 + b_1[P_G; X_c]$$

Maintenance ATP constraint (flux constraints) It is defined as

$$\mathbf{1}_{\nu_{constraint}}[\nu; E; R; C] = b$$

where $\mathbf{1}_{\nu_{constraint}}$ is an indicator matrix selecting the reaction associated with the production of maintenance ATP/flagella fuel. Generally this means a specific reaction has to be added in S. Another way to handle these constraints is to include them in the lower/upper bounds of the optimization problem.

Summary

$$\begin{array}{cc} [S, \mu C_E, \mu C_R, \mu C_C] & [\nu; E; R; C] = -\sum \mu^{d_i} C_i X_i \quad \text{(Base metabolism and process production)} \\ a & [\nu; E; R; C] = b_0 + b_1[\{X_i\}] \quad \text{(Process capacity)} \\ \mathbf{1}_{\nu_{constraint}} & [\nu; E; R; C] = b \quad \text{(Flux constraints)} \\ \begin{bmatrix} I, & -\mathrm{diag}\,(k_E)\,, & 0, & 0 \\ -I, & -\mathrm{diag}\,(k_E)\,, & 0, & 0 \end{bmatrix} & [\nu; E; R; C] \leq 0 \quad \text{(Enzymatic fluxes)} \\ a & [\nu; E; R; C] \leq b_0 + b_1[\{X_i\}] \quad \text{(Density constraints)} \end{array}$$

A more graphic representation of the matrix is provide in Figure 4.

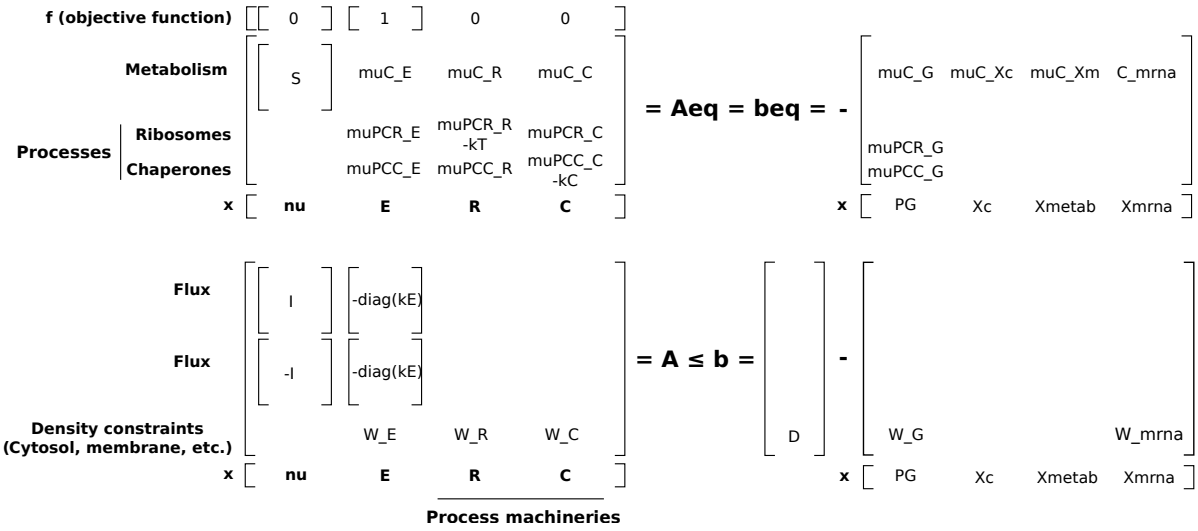


Figure 4: Structure of matrices. Note that the left hand side and the right hand side have a similar structure (except for the ν submatrix). Each column represents a set of molecules: on the right hand side, the production flux of the set has to be determined, on the left hand side it is already given *a priori*.

4. New formalism

The new formalism must allow simple interactions between user readable data and iffi-cient construction of the RBA problem. The data is going to be written is XML format, inspired by SBML standards.

4.1. Final differences between RBAv01 and new formalism

	RBAv01 mat12	RBAv01 mat15	RBAnew python
Input files	Custom files + partially hard coded		XML files (nothing hard coded)
Code length (commented)	$\simeq 3500$ ($\simeq 1000$)		$\simeq 2000$ ($\simeq 500$)
Parsing	15s	8s	< 1s
Solving (23 rounds)	15s	5s	< 2s
1 matrix update	580ms	100ms	3-10ms
1 CPLEX round	50ms	58ms	30ms

Table 1: Comparison between original algorithm (RBAv01) and new formalism (RBAnew). Because performance for the old algorithm depended on matlab version, we included performance for matlab R2012a (mat12) and R2015b (mat15).

Table 1 shows the main differences between RBAv01 and the new algorithm. The new algorithm uses generic XML files and is thus easier to modify for the end user. Using python, we could improve both parsing and solving time, which are significantly lower than the original algorithm.

4.2. Input data

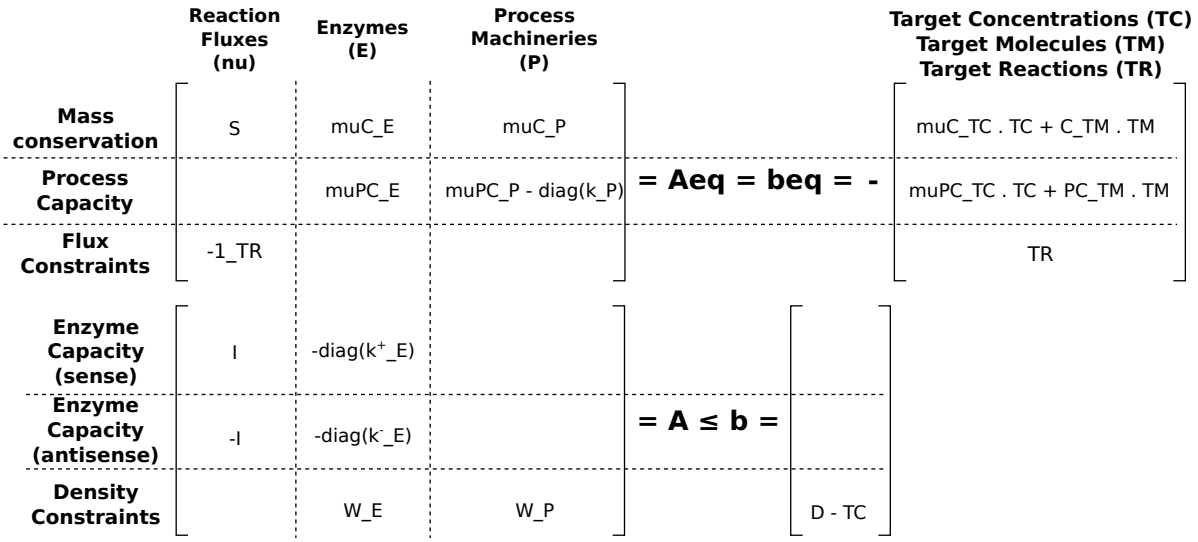
Data files are described in Appendix A.

4.3. From data to matrices

Algorithm overview Figure 5 shows the blocks that are used to build the final matrices.

Stoichiometry matrix The procedure is standard, we will not illustrate it here. However, note that external metabolites are removed from the metabolite pool.

Density limits These values are simply extracted from RBAParameters and assembled into a vector (one coefficient per compartment).



Blocks needed:
S: stoichiometry matrix.
C_E, PC_E, W_E, k_E: composition, processing cost, weight and capacity of enzymes.
C_P, PC_P, W_P, k_P: composition, processing cost, weight and capacity of process machineries.
D: density limits for every compartment.
C_TC: composition of molecules whose concentration needs to be kept at a certain value.
TC: vector of concentrations of molecules that need to be kept at certain value.
C_TM: composition of molecules which are generated at a given flux.
TM: vector of fluxes of molecules.
1_TR: indicator matrix of reactions whose flux is set to a fixed value.
TR: vector of set reaction fluxes.

Figure 5: Blocks that need to be assembled in the new algorithm.

Species matrices Figure 6 shows how macromolecules are breaken down into matrices describing their composition, processing cost and weight. In the end, they are merged into a single matrix describing composition, processing cost and weight of all metabolites and macromolecules.

Machinery matrices Figure 7 shows how the matrices stored in `rba.processes` are built.

Target matrices Targets are either metabolites or macromolecules. Their composition, processing cost and weight can be extracted as columns from the species matrices.

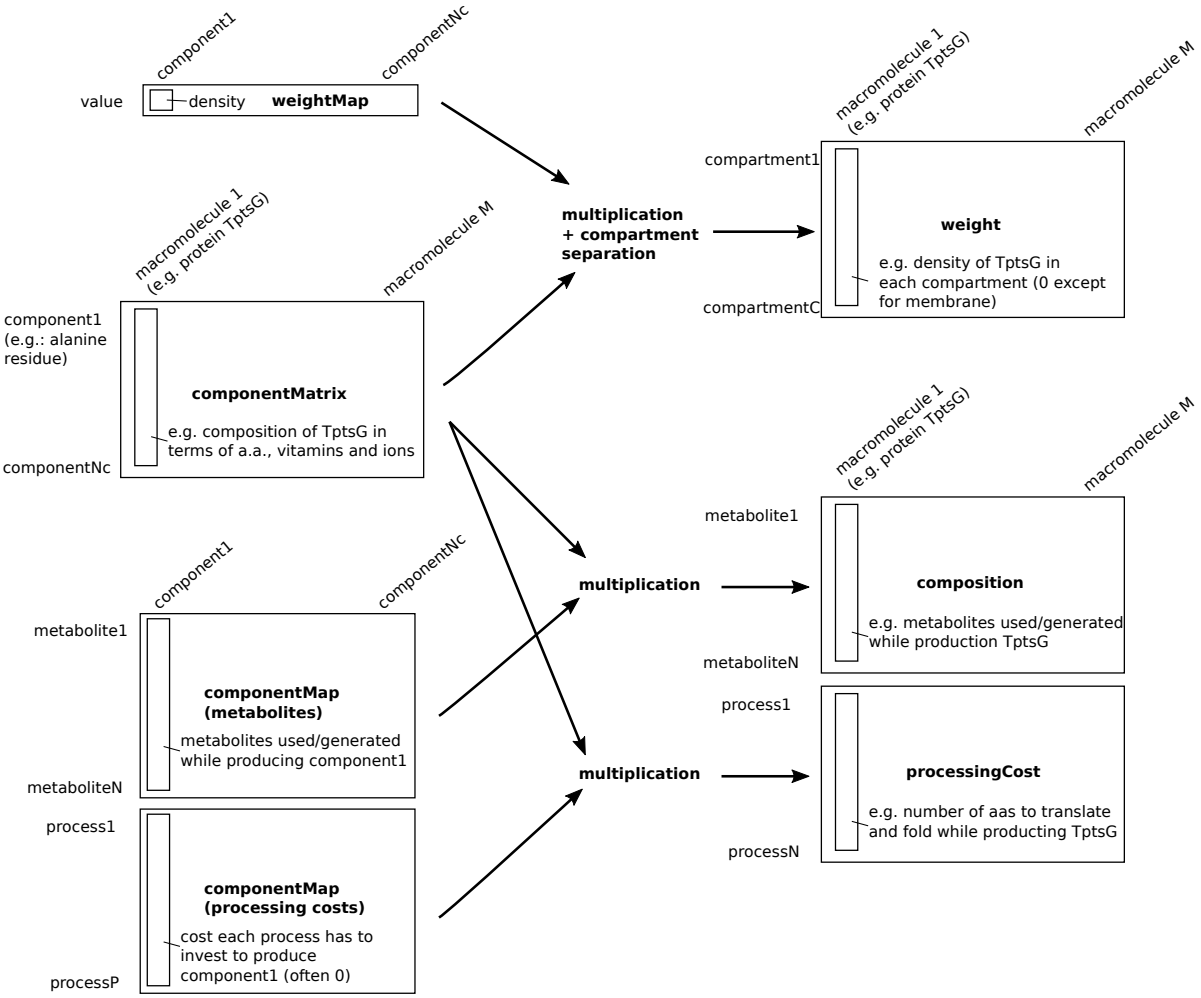


Figure 6: Matrices extracted from macromolecule information. An example is given with proteins but in the end, they contain all macromolecules and all internal metabolites.

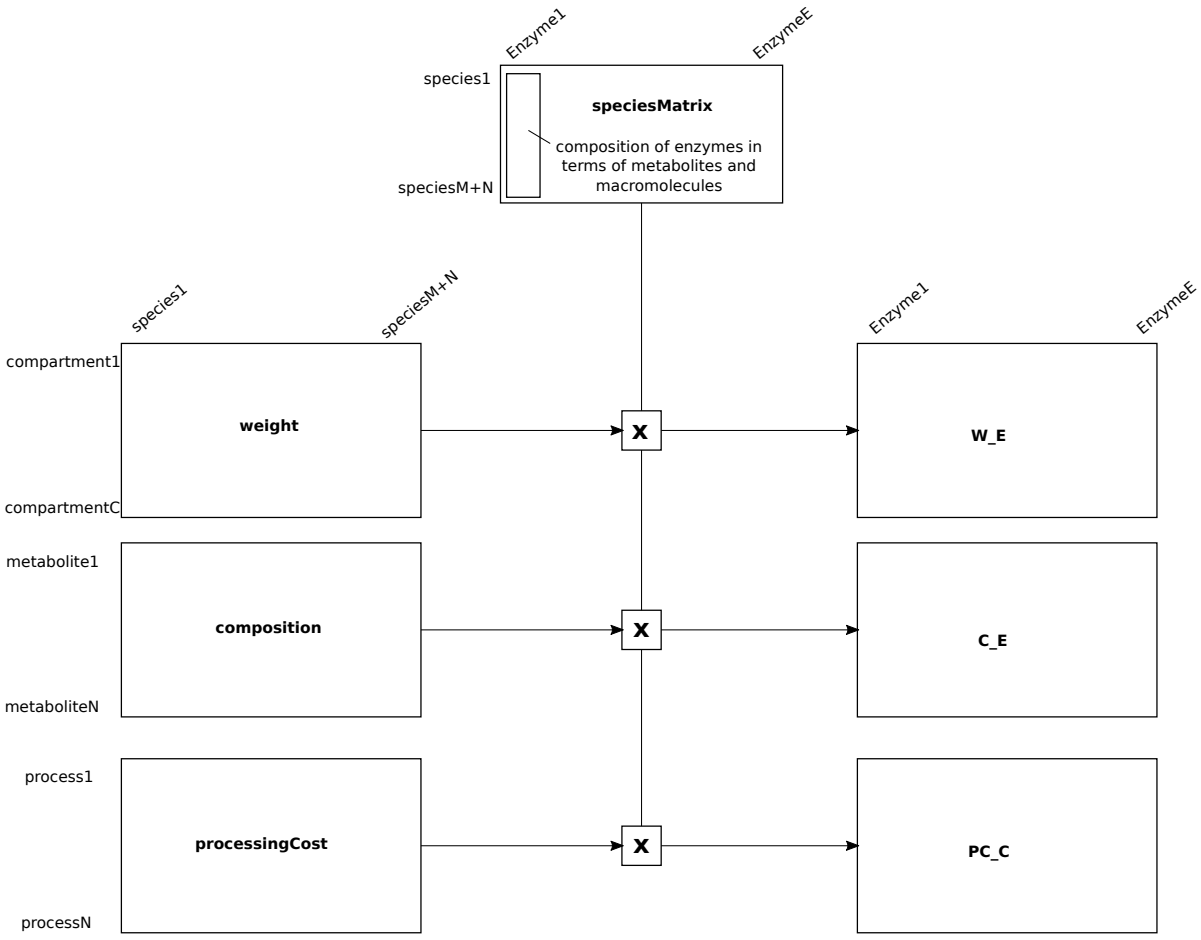


Figure 7: Every machinery can be described by a reaction matrix. Reactants are species (metabolites or macromolecules) needed to build the machinery and products are byproducts of the assembly process. Through matrix multiplication with the species matrices, we can deduce its composition, weight and processing cost.

A. RBA XML

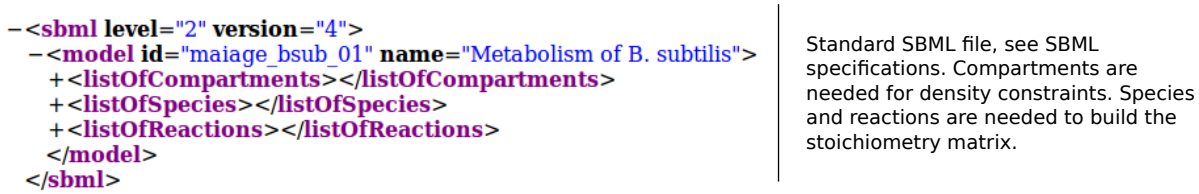


Figure 8: Metabolism file is a standard SBML file.

Metabolism file The metabolism file is a standard SBML file (Fig. 8). It contains information about cell compartments, metabolite species and metabolism reactions. Concentration of external metabolites are defined in the species section.

Parameter file The parameter file is an XML file composed of two subsections (Fig. 9). `listOfMaximalDensities` contains density constraints. `listOfFunctions` contains user-defined functions that can be used to set μ-dependent maximal densities or μ-dependent process targets (see below).

Macromolecule files Currently, RBA defines three sets of macromolecules: proteins, RNAs and DNA. One XML file is created for each set (Fig. 10). All macromolecules have a `listOfComponents` describing the building blocks used to produce them (*e.g.* amino acids, vitamins and ions for proteins). Then a `listOfSpecies` contains all molecules of the set, including their description in terms of components. Additionally, proteins have a `listOfEfficiencyFunctions` that lists efficiency models for enzymes. Enzymes contain an `enzymaticActivity` structure that defines the reaction they catalyze and the parameters for the efficiency models. Finally, transporters have a `transporterEfficiency` that modulates the enzymatic activity depending on substrate and cofactor availability.

Process file The process file is an XML file containing a `listOfProcesses` and a `listOfComponentMaps`. Each `process` can contain up to 3 subsections. The `capacityConstraint` defines a machinery used by the process that has a limited capacity. The `operatingCosts` defines which macromolecules the process produces/degrades/modifies and the cost associated with these operations. The `targets` are set fluxes that a process must maintain in order for the cell to work properly. Target fluxes can apply to metabolites (`targetValue`) and reactions (`targetReaction`). Target fluxes scale with μ if they contribute to `dilution_compensation` or if they are defined using a μ-dependent user-function. Finally `componentMaps` are used to compute the costs in the `operatingCosts` section.

Figure 9: Structure of the parameter file used by RBA.

B. Small bug in RBAv01

There was a slight bug in RBAv01. A typo in indexing while updating transporters had the following consequences:

- Two Fe transporter would not be initialized properly. They worked at default capacity. In the presence of Fe, this has only a low impact as they would be active anyway. In the absence of Fe, they would keep working.
- Instead, the catalytic efficiency of enzyme EPtsI was modified. In the presence of Fe, it would divide its catalytic activity by 2, leading to slightly lower growth rate than expected (something like 0.015). In the absence of Fe, the forward reaction catalyzed by EPtsI would be blocked.

Briefly, this bug has a small impact in the presence of Fe and a dramatic impact in the absence of Fe. To our knowledge, there is no dataset where Fe was removed from the medium, explaining why this bug has not been discovered earlier.

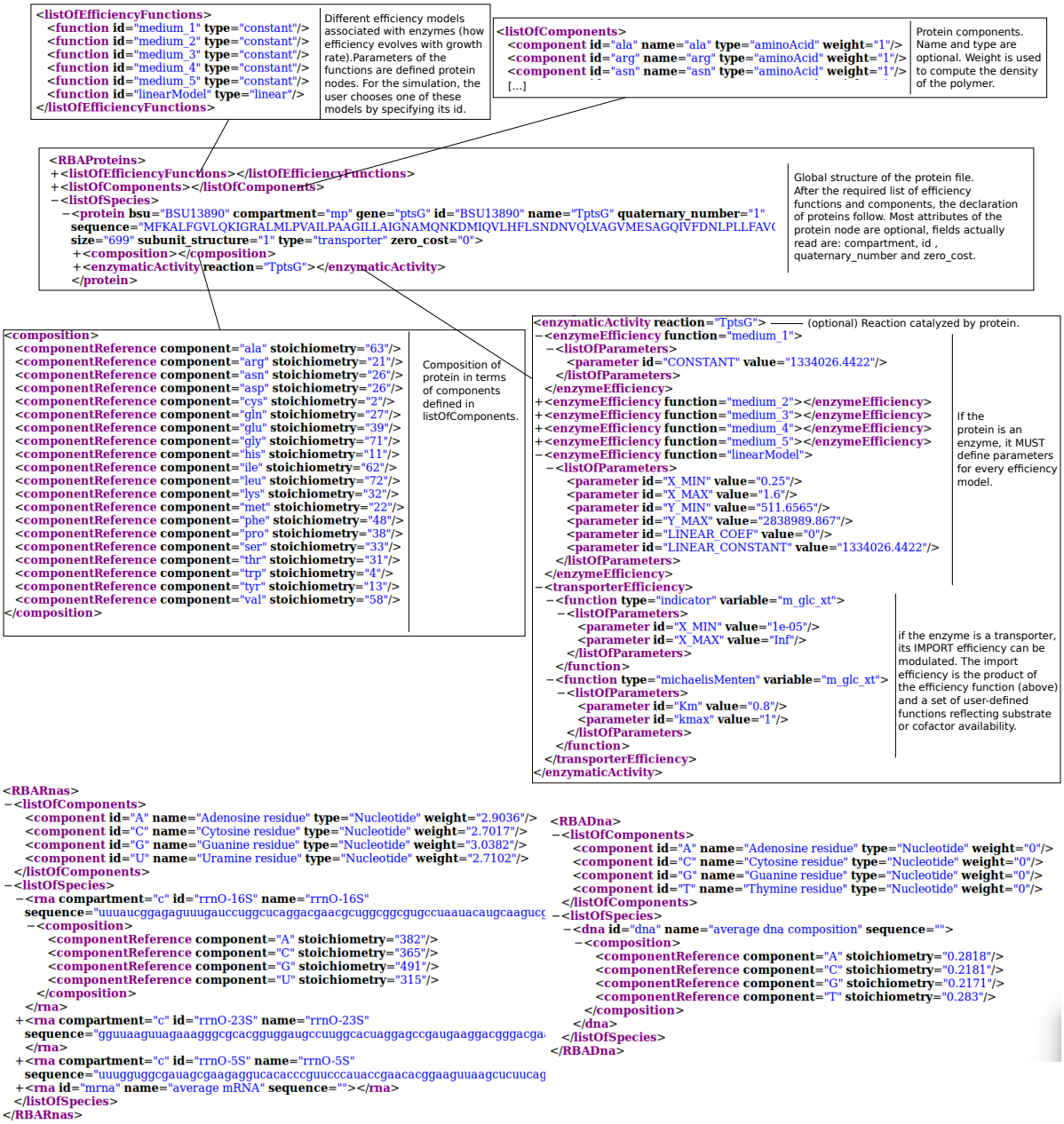


Figure 10: Structure of protein, RNA and DNA file.

Figure 11: Structure of process file.