RBA

S. Fischer - Biosys - MAIAGE

November 28, 2016

Contents

1 Principles underlying algorithm

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

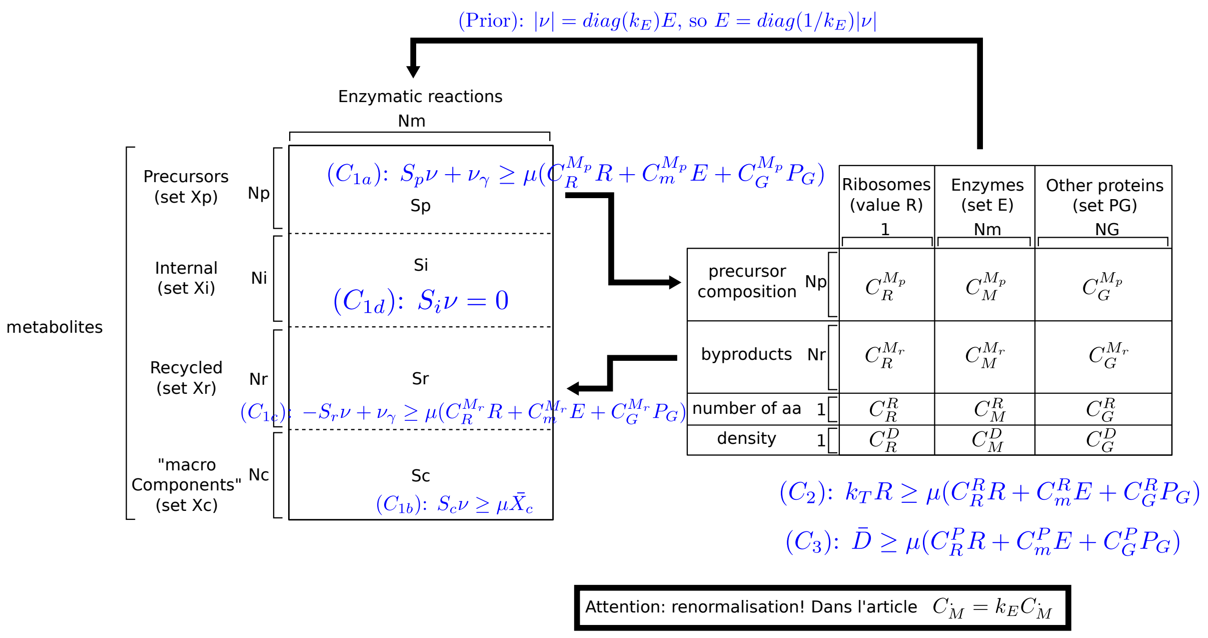

(Prior): $|\nu| = diag(k_E)E$, so $E = diag(1/k_E)|\nu|$
Enzymatic reactions
Nm
metabolites
Precursors (set Xp)
Np
Internal (set Xi)
Ni
Recycled (set Xr)
Nr
"macro Components" (set Xc)
Nc
(C_{1a}): $S_p\nu + \nu_\gamma \geq \mu(C_R^{M_p}R + C_m^{M_p}E + C_G^{M_p}P_G)$
Sp
Si
(C_{1d}): $S_i\nu = 0$
Sr
(C_{1c}): $-S_r\nu + \nu_\gamma \geq \mu(C_R^{M_r}R + C_m^{M_r}E + C_G^{M_r}P_G)$
Sc
(C_{1b}): $S_c\nu \geq \mu\bar{X}_c$
Ribosomes (value R)
1
Enzymes (set E)
Nm
Other proteins (set PG)
NG
precursor composition
Np
$C_R^{M_p}$
$C_M^{M_p}$
$C_G^{M_p}$
byproducts
Nr
$C_R^{M_r}$
$C_M^{M_r}$
$C_G^{M_r}$
number of aa
1
C_R^R
C_M^R
C_G^R
density
1
C_R^D
C_M^D
C_G^D
(C_2): $k_T R \geq \mu(C_R^R R + C_m^R E + C_G^R P_G)$
(C_3): $\bar{D} \geq \mu(C_R^P R + C_m^P E + C_G^P P_G)$
Attention: renormalisation! Dans l'article $C_M^{\cdot} = k_E C_M^{\cdot}$

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.

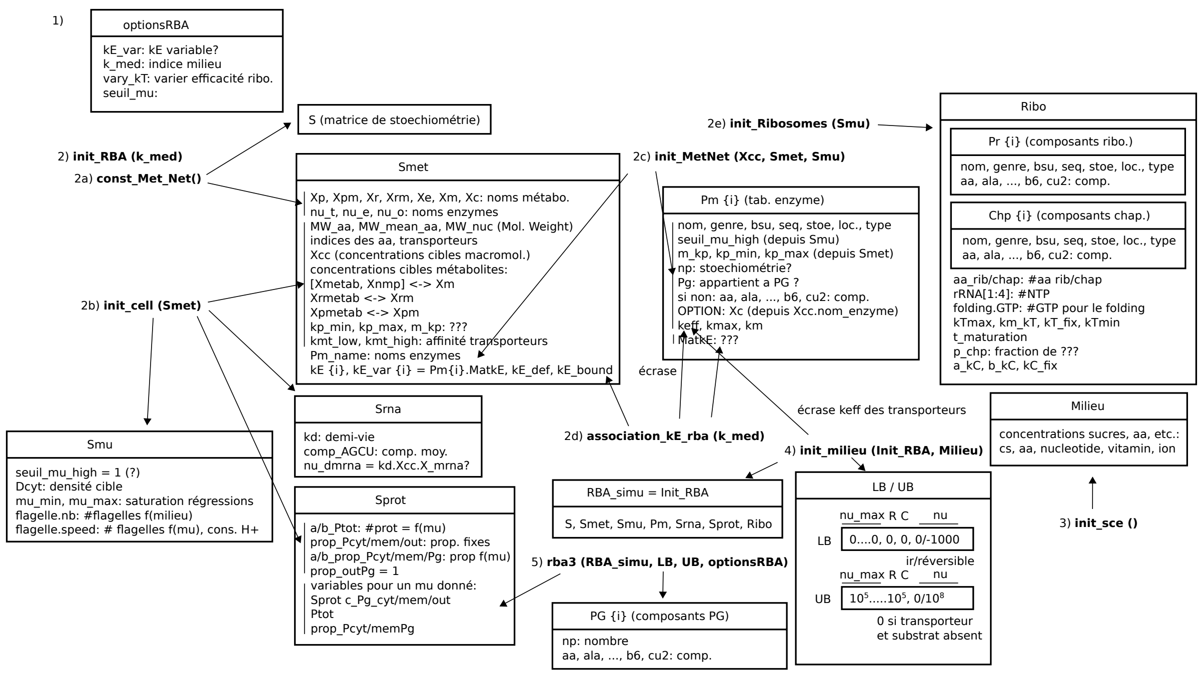

1)
optionsRBA
kE_var: kE variable?
k_med: indice milieu
vary_kT: varier efficacité ribo.
seuil_mu:
S (matrice de stoechiométrie)
2) init_RBA (k_med)
2a) const_Met_Net()
Smet
Xp, Xpm, Xr, Xrm, Xe, Xm, Xc: noms métabo.
nu_t, nu_e, nu_o: noms enzymes
MW_aa, MW_mean_aa, MW_nuc (Mol. Weight)
indices des aa, transporteurs
Xcc (concentrations cibles macromol.)
concentrations cibles métabolites:
[Xmetab, Xnmp] <-> Xm
Xrmetab <-> Xrm
Xpmetab <-> Xpm
kp_min, kp_max, m_kp: ???
kmt_low, kmt_high: affinité transporteurs
Pm_name: noms enzymes
kE {i}, kE_var {i} = Pm{i}.MatkE, kE_def, kE_bound
2b) init_cell (Smet)
Smu
seuil_mu_high = 1 (?)
Dcyt: densité cible
mu_min, mu_max: saturation régressions
flagelle.nb: #flagelles f(milieu)
flagelle.speed: # flagelles f(mu), cons. H+
Srna
kd: demi-vie
comp_AGCU: comp. moy.
nu_dmrna = kd.Xcc.X_mrna?
Sprot
a/b_Ptot: #prot = f(mu)
prop_Pcyt/mem/out: prop. fixes
a/b_prop_Pcyt/mem/Pg: prop f(mu)
prop_outPg = 1
variables pour un mu donné:
Sprot c_Pg_cyt/mem/out
Ptot
prop_Pcyt/memPg
2c) init_MetNet (Xcc, Smet, Smu)
Pm {i} (tab. enzyme)
nom, genre, bsu, seq, stoe, loc., type
seuil_mu_high (depuis Smu)
m_kp, kp_min, kp_max (depuis Smet)
np: stoechiométrie?
Pg: appartient a PG ?
si non: aa, ala, ..., b6, cu2: comp.
OPTION: Xc (depuis Xcc.nom_enzyme)
keff, kmax, km
MatkE: ???
écrase
2d) association_kE_rba (k_med)
2e) init_Ribosomes (Smu)
Ribo
Pr {i} (composants ribo.)
nom, genre, bsu, seq, stoe, loc., type
aa, ala, ..., b6, cu2: comp.
Chp {i} (composants chap.)
nom, genre, bsu, seq, stoe, loc., type
aa, ala, ..., b6, cu2: comp.
aa_rib/chap: #aa rib/chap
rRNA[1:4]: #NTP
folding.GTP: #GTP pour le folding
kTmax, km_kT, kT_fix, kTmin
t_maturation
p_chp: fraction de ???
a_kC, b_kC, kC_fix
écrase keff des transporteurs
Milieu
concentrations sucres, aa, etc.:
cs, aa, nucleotide, vitamin, ion
3) init_sce ()
4) init_milieu (Init_RBA, Milieu)
RBA_simu = Init_RBA
S, Smet, Smu, Pm, Srna, Sprot, Ribo
LB / UB
nu_max R C nu
LB 0....0, 0, 0, 0/-1000
ir/réversible
nu_max R C nu
UB 10^5.....10^5, 0/10^8
0 si transporteur
et substrat absent
5) rba3 (RBA_simu, LB, UB, optionsRBA)
PG {i} (composants PG)
np: nombre
aa, ala, ..., b6, cu2: comp.

Figure 2: Algorithm used in RBAV01.

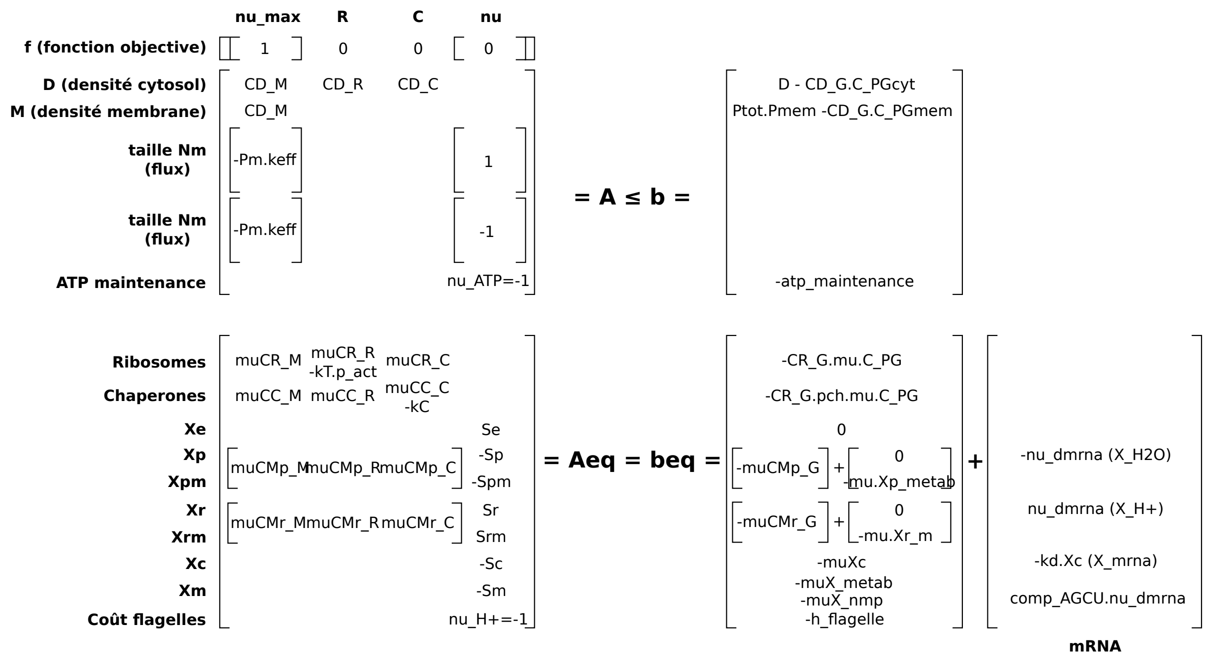


Figure 3: Matrices used in RBAV01.

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

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

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

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

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

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:

$$k_T R + \mu[C_E^R, C_R^R, C_C^R][E; R; C] = -\mu C_G^R P_G$$

In full form:

$$[0, \mu C_E^R, k_T + \mu C_R^R, \mu C_C^R][\nu; E; R; C] = [-\mu C_G^R, 0][P_G; X_c]$$

The number of aas needed to build a protein can be included in the description of molecule sets. 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)

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

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

Density constraints The density constraint writes

$$[0, C_E^D, C_R^D, C_C^D][\nu; E; R; C] \leq \overline{D} - [C_G^D, C_G^c][P_G; X_c]$$

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 (metabolite specific 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, as well as specific metabolites like ATP maintenance or protons dedicated to fuelling flagella.

Summary

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

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

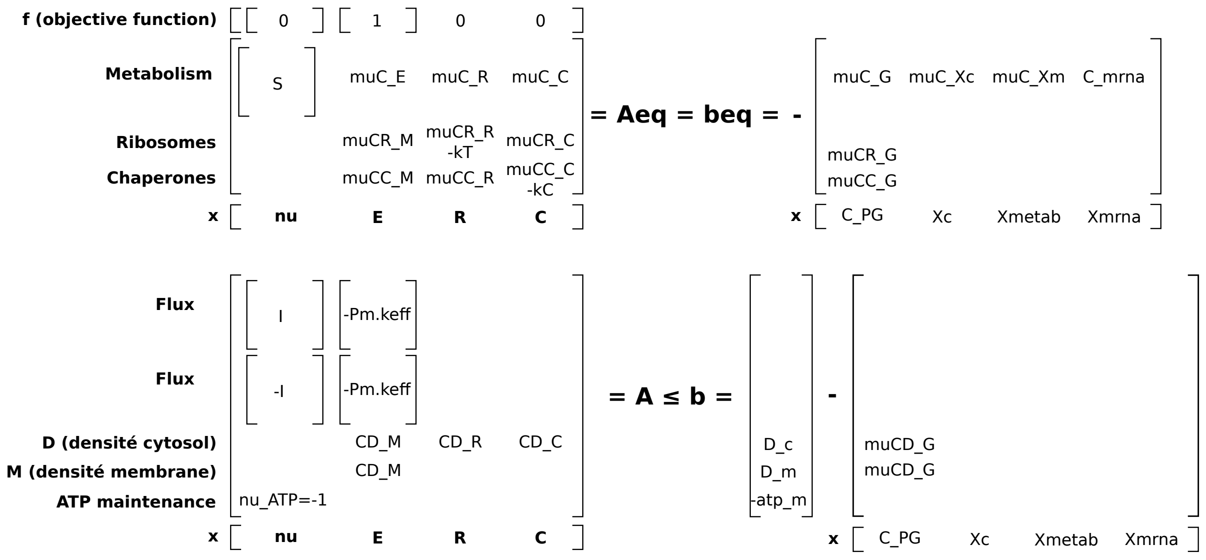


Figure 4: Structure of matrices. Note that the left hand side and the right hand side have a similar structure (except for the ν submatrix). Each column represents a set of molecules: on the right hand side, the production flux of the set has to be determined, on the left hand side it is already given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

4 New formalism

The new formalism must allow simple interactions between user readable data and efficient 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

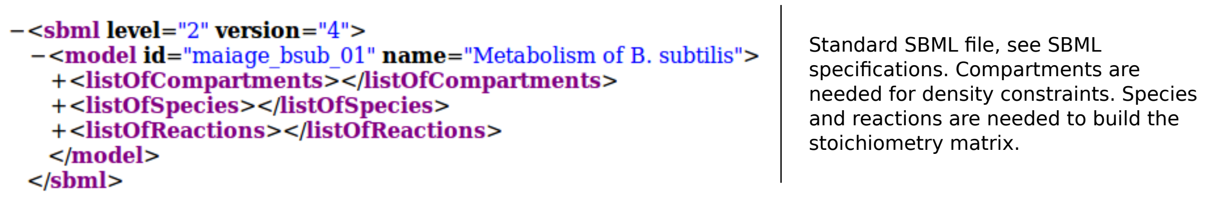
	RBAv01 mat12	RBAv01 mat15	RBAnew mat12	RBAnew mat15
Input files	Custom files + partially hard coded		XML files (nothing hard coded)	
Code length (commented)	$\simeq$ 3500 ($\simeq$ 1000)		$\simeq$ 1500 ($\simeq$ 500)	
Parsing	15s	8s	12s	12s
Solving (23 rounds)	15s	5s	1.5-2s	1.5-2s
1 matrix update	580ms	100ms	3-10ms	3-10ms
1 CPLEX round	50ms	58ms	50ms	58ms

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. Parsing files is less efficient than the original algoirthm but this is not very important as it is only done once. More important is the solving time, which is significantly lower than the original algorithm.

The new algorithm could be improved further by handling block-allocation more efficiently for sparse matrices, but current performance seems pretty good.

4.2 Input data

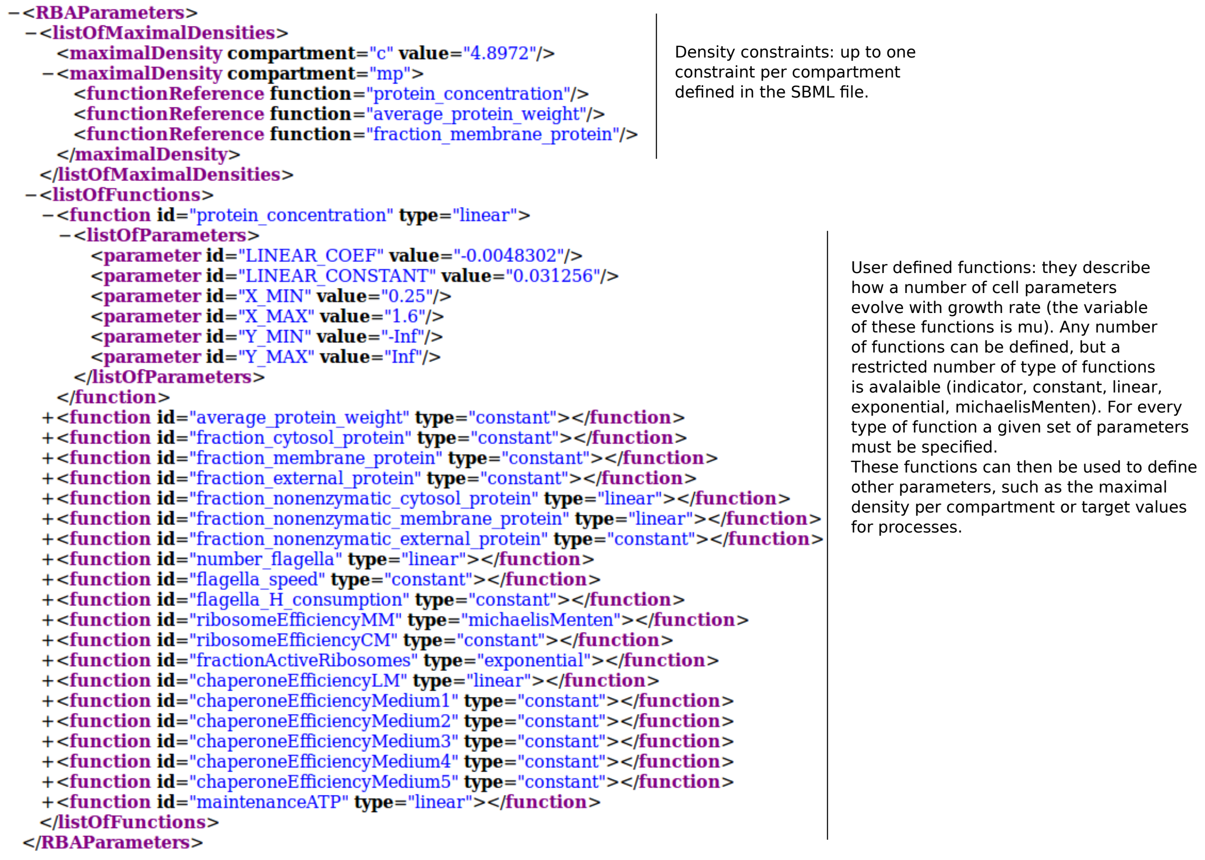


```
−<sbml level="2" version="4">
  −<model id="maiage_bsub_01" name="Metabolism of B. subtilis">
    +<listOfCompartments></listOfCompartments>
    +<listOfSpecies></listOfSpecies>
    +<listOfReactions></listOfReactions>
   </model>
 </sbml>
```

Standard SBML file, see SBML specifications. Compartments are needed for density constraints. Species and reactions are needed to build the stoichiometry matrix.

Figure 5: Metabolism file is a standard SBML file.

Metabolism file The metabolism file is a standard SBML file (Fig. 5). It contains information about cell compartments, metabolite species and metabolism reactions. Concentration of external metabolites are defined in the species section.



```
-<RBAParameters>
  -<listOfMaximalDensities>
     <maximalDensity compartment="c" value="4.8972"/>
    -<maximalDensity compartment="mp">
       <functionReference function="protein_concentration"/>
       <functionReference function="average_protein_weight"/>
       <functionReference function="fraction_membrane_protein"/>
     </maximalDensity>
   </listOfMaximalDensities>
  -<listOfFunctions>
    -<function id="protein_concentration" type="linear">
      -<listOfParameters>
         <parameter id="LINEAR_COEF" value="-0.0048302"/>
         <parameter id="LINEAR_CONSTANT" value="0.031256"/>
         <parameter id="X_MIN" value="0.25"/>
         <parameter id="X_MAX" value="1.6"/>
         <parameter id="Y_MIN" value="-Inf"/>
         <parameter id="Y_MAX" value="Inf"/>
       </listOfParameters>
     </function>
    +<function id="average_protein_weight" type="constant"></function>
    +<function id="fraction_cytosol_protein" type="constant"></function>
    +<function id="fraction_membrane_protein" type="constant"></function>
    +<function id="fraction_external_protein" type="constant"></function>
    +<function id="fraction_nonenzymatic_cytosol_protein" type="linear"></function>
    +<function id="fraction_nonenzymatic_membrane_protein" type="linear"></function>
    +<function id="fraction_nonenzymatic_external_protein" type="constant"></function>
    +<function id="number_flagella" type="linear"></function>
    +<function id="flagella_speed" type="constant"></function>
    +<function id="flagella_H_consumption" type="constant"></function>
    +<function id="ribosomeEfficiencyMM" type="michaelisMenten"></function>
    +<function id="ribosomeEfficiencyCM" type="constant"></function>
    +<function id="fractionActiveRibosomes" type="exponential"></function>
    +<function id="chaperoneEfficiencyLM" type="linear"></function>
    +<function id="chaperoneEfficiencyMedium1" type="constant"></function>
    +<function id="chaperoneEfficiencyMedium2" type="constant"></function>
    +<function id="chaperoneEfficiencyMedium3" type="constant"></function>
    +<function id="chaperoneEfficiencyMedium4" type="constant"></function>
    +<function id="chaperoneEfficiencyMedium5" type="constant"></function>
    +<function id="maintenanceATP" type="linear"></function>
   </listOfFunctions>
 </RBAParameters>
```

Density constraints: up to one constraint per compartment defined in the SBML file.

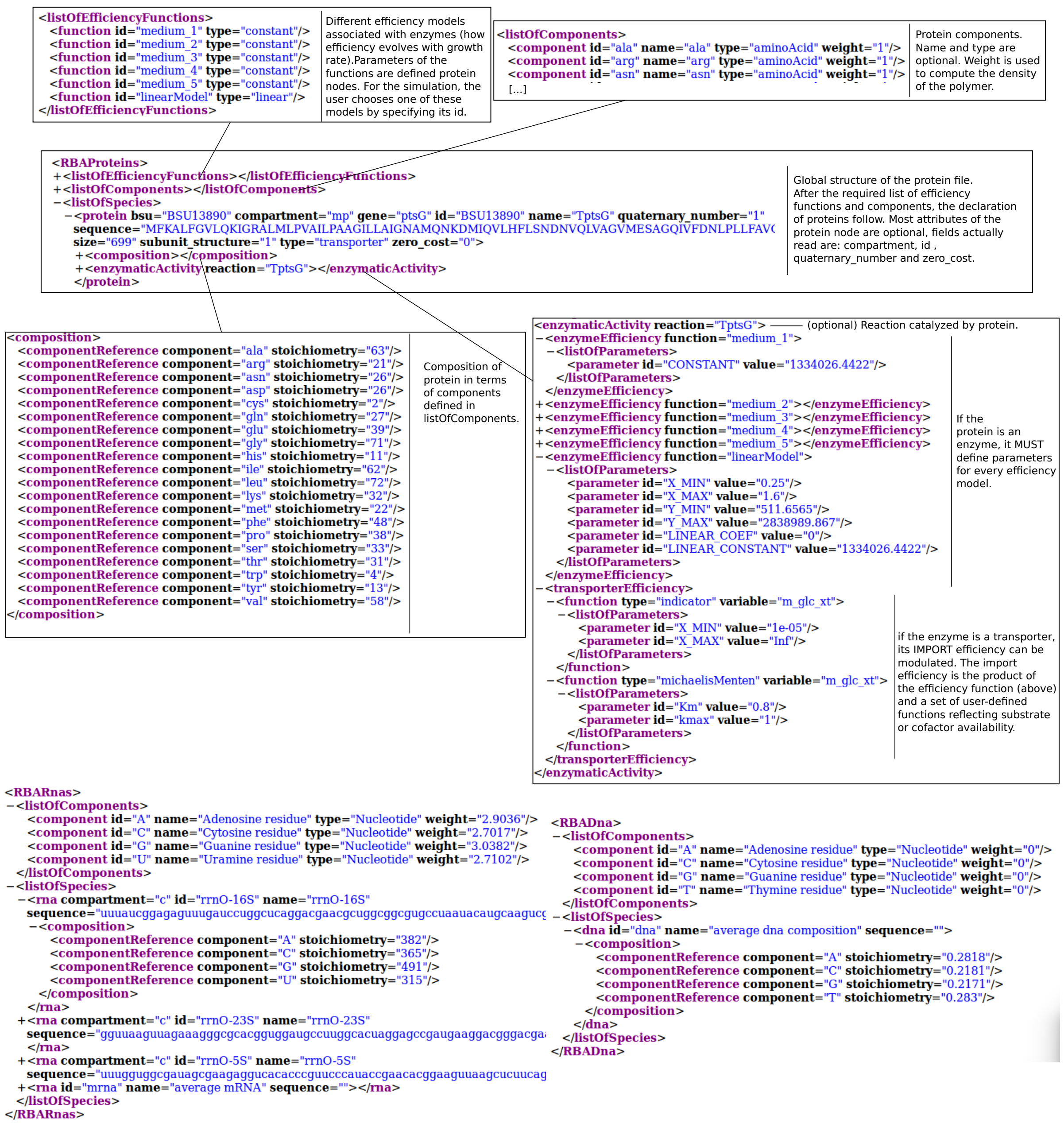
User defined functions: they describe how a number of cell parameters evolve with growth rate (the variable of these functions is mu). Any number of functions can be defined, but a restricted number of type of functions is avalaible (indicator, constant, linear, exponential, michaelisMenten). For every type of function a given set of parameters must be specified.
These functions can then be used to define other parameters, such as the maximal density per compartment or target values for processes.

Figure 6: Structure of the parameter file used by RBA.

Parameter file The parameter file is an XML file composed of two subsections (Fig. 6). `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. 7). 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.



```
<listOfEfficiencyFunctions>
   <function id="medium_1" type="constant"/>
   <function id="medium_2" type="constant"/>
   <function id="medium_3" type="constant"/>
   <function id="medium_4" type="constant"/>
   <function id="medium_5" type="constant"/>
   <function id="linearModel" type="linear"/>
</listOfEfficiencyFunctions>
```

Different efficiency models associated with enzymes (how efficiency evolves with growth rate).Parameters of the functions are defined protein nodes. For the simulation, the user chooses one of these models by specifying its id.

```
<listOfComponents>
   <component id="ala" name="ala" type="aminoAcid" weight="1"/>
   <component id="arg" name="arg" type="aminoAcid" weight="1"/>
   <component id="asn" name="asn" type="aminoAcid" weight="1"/>
   [...]
```

Protein components. Name and type are optional. Weight is used to compute the density of the polymer.

```
<RBAProteins>
+<listOfEfficiencyFunctions></listOfEfficiencyFunctions>
+<listOfComponents></listOfComponents>
-<listOfSpecies>
  -<protein bsu="BSU13890" compartment="mp" gene="ptsG" id="BSU13890" name="TptsG" quaternary_number="1"
   sequence="MFKALFGVLQKIGRALMLPVAILPAAGILLAIGNAMQNKDMIQVLHFLSNDNVQLVAGVMESAGQIVFDNLPLLFAV
   size="699" subunit_structure="1" type="transporter" zero_cost="0">
   +<composition></composition>
   +<enzymaticActivity reaction="TptsG"></enzymaticActivity>
   </protein>
```

Global structure of the protein file. After the required list of efficiency functions and components, the declaration of proteins follow. Most attributes of the protein node are optional, fields actually read are: compartment, id , quaternary_number and zero_cost.

```
<composition>
  <componentReference component="ala" stoichiometry="63"/>
  <componentReference component="arg" stoichiometry="21"/>
  <componentReference component="asn" stoichiometry="26"/>
  <componentReference component="asp" stoichiometry="26"/>
  <componentReference component="cys" stoichiometry="2"/>
  <componentReference component="gln" stoichiometry="27"/>
  <componentReference component="glu" stoichiometry="39"/>
  <componentReference component="gly" stoichiometry="71"/>
  <componentReference component="his" stoichiometry="11"/>
  <componentReference component="ile" stoichiometry="62"/>
  <componentReference component="leu" stoichiometry="72"/>
  <componentReference component="lys" stoichiometry="32"/>
  <componentReference component="met" stoichiometry="22"/>
  <componentReference component="phe" stoichiometry="48"/>
  <componentReference component="pro" stoichiometry="38"/>
  <componentReference component="ser" stoichiometry="33"/>
  <componentReference component="thr" stoichiometry="31"/>
  <componentReference component="trp" stoichiometry="4"/>
  <componentReference component="tyr" stoichiometry="13"/>
  <componentReference component="val" stoichiometry="58"/>
</composition>
```

Composition of protein in terms of components defined in listOfComponents.

```
<enzymaticActivity reaction="TptsG"> ——— (optional) Reaction catalyzed by protein.
-<enzymeEfficiency function="medium_1">
  -<listOfParameters>
     <parameter id="CONSTANT" value="1334026.4422"/>
   </listOfParameters>
 </enzymeEfficiency>
+<enzymeEfficiency function="medium_2"></enzymeEfficiency>
+<enzymeEfficiency function="medium_3"></enzymeEfficiency>
+<enzymeEfficiency function="medium_4"></enzymeEfficiency>
+<enzymeEfficiency function="medium_5"></enzymeEfficiency>
-<enzymeEfficiency function="linearModel">
  -<listOfParameters>
     <parameter id="X_MIN" value="0.25"/>
     <parameter id="X_MAX" value="1.6"/>
     <parameter id="Y_MIN" value="511.6565"/>
     <parameter id="Y_MAX" value="2838989.867"/>
     <parameter id="LINEAR_COEF" value="0"/>
     <parameter id="LINEAR_CONSTANT" value="1334026.4422"/>
   </listOfParameters>
 </enzymeEfficiency>
-<transporterEfficiency>
  -<function type="indicator" variable="m_glc_xt">
    -<listOfParameters>
       <parameter id="X_MIN" value="1e-05"/>
       <parameter id="X_MAX" value="Inf"/>
     </listOfParameters>
   </function>
  -<function type="michaelisMenten" variable="m_glc_xt">
    -<listOfParameters>
       <parameter id="Km" value="0.8"/>
       <parameter id="kmax" value="1"/>
     </listOfParameters>
   </function>
 </transporterEfficiency>
</enzymaticActivity>
```

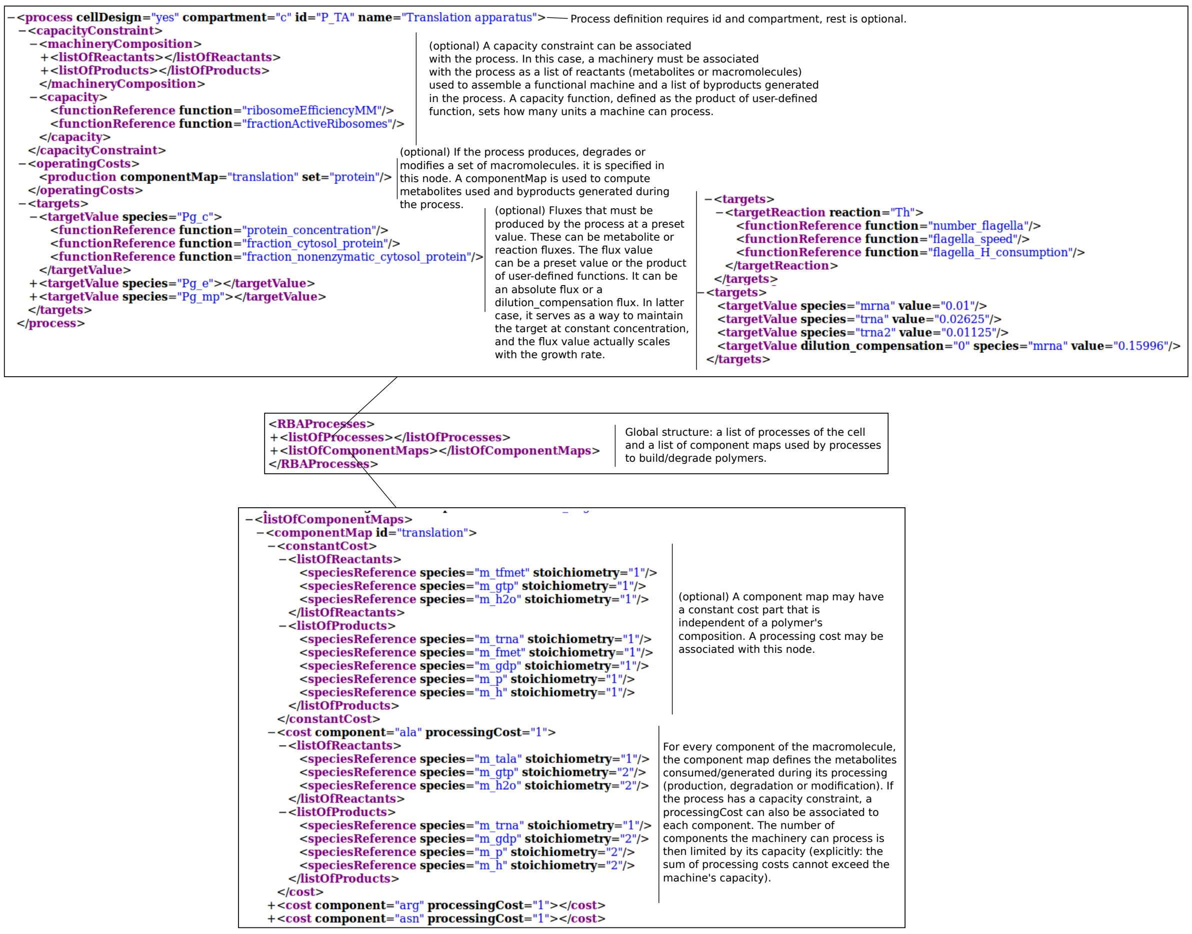
If the protein is an enzyme, it MUST define parameters for every efficiency model.

if the enzyme is a transporter, its IMPORT efficiency can be modulated. The import efficiency is the product of the efficiency function (above) and a set of user-defined functions reflecting substrate or cofactor availability.

```
<RBARnas>
-<listOfComponents>
    <component id="A" name="Adenosine residue" type="Nucleotide" weight="2.9036"/>
    <component id="C" name="Cytosine residue" type="Nucleotide" weight="2.7017"/>
    <component id="G" name="Guanine residue" type="Nucleotide" weight="3.0382"/>
    <component id="U" name="Uramine residue" type="Nucleotide" weight="2.7102"/>
  </listOfComponents>
-<listOfSpecies>
  -<rna compartment="c" id="rrnO-16S" name="rrnO-16S"
    sequence="uuuaucggagaguuugauccuggcucaggacgaacgcuggcggcgugccuaauacaugcaaguc
    -<composition>
        <componentReference component="A" stoichiometry="382"/>
        <componentReference component="C" stoichiometry="365"/>
        <componentReference component="G" stoichiometry="491"/>
        <componentReference component="U" stoichiometry="315"/>
     </composition>
    </rna>
  +<rna compartment="c" id="rrnO-23S" name="rrnO-23S"
    sequence="gguuaaguuagaaagggcgcacgguggaugccuuggcacuaggagccgaugaaggacgggacga
    </rna>
  +<rna compartment="c" id="rrnO-5S" name="rrnO-5S"
    sequence="uuugguggcgauagcgaagaggucacacccguucccauaccgaacacggaaguuaagcucuucag
  +<rna id="mrna" name="average mRNA" sequence=""></rna>
  </listOfSpecies>
</RBARnas>
```

```
<RBADna>
-<listOfComponents>
    <component id="A" name="Adenosine residue" type="Nucleotide" weight="0"/>
    <component id="C" name="Cytosine residue" type="Nucleotide" weight="0"/>
    <component id="G" name="Guanine residue" type="Nucleotide" weight="0"/>
    <component id="T" name="Thymine residue" type="Nucleotide" weight="0"/>
  </listOfComponents>
-<listOfSpecies>
  -<dna id="dna" name="average dna composition" sequence="">
    -<composition>
        <componentReference component="A" stoichiometry="0.2818"/>
        <componentReference component="C" stoichiometry="0.2181"/>
        <componentReference component="G" stoichiometry="0.2171"/>
        <componentReference component="T" stoichiometry="0.283"/>
     </composition>
    </dna>
  </listOfSpecies>
</RBADna>
```

Figure 7: Structure of protein, RNA and DNA file.



```
−<process cellDesign="yes" compartment="c" id="P_TA" name="Translation apparatus">
  −<capacityConstraint>
    −<machineryComposition>
      +<listOfReactants></listOfReactants>
      +<listOfProducts></listOfProducts>
     </machineryComposition>
    −<capacity>
       <functionReference function="ribosomeEfficiencyMM"/>
       <functionReference function="fractionActiveRibosomes"/>
     </capacity>
   </capacityConstraint>
  −<operatingCosts>
     <production componentMap="translation" set="protein"/>
   </operatingCosts>
  −<targets>
    −<targetValue species="Pg_c">
       <functionReference function="protein_concentration"/>
       <functionReference function="fraction_cytosol_protein"/>
       <functionReference function="fraction_nonenzymatic_cytosol_protein"/>
     </targetValue>
    +<targetValue species="Pg_e"></targetValue>
    +<targetValue species="Pg_mp"></targetValue>
   </targets>
 </process>
```

Process definition requires id and compartment, rest is optional.

(optional) A capacity constraint can be associated with the process. In this case, a machinery must be associated with the process as a list of reactants (metabolites or macromolecules) used to assemble a functional machine and a list of byproducts generated in the process. A capacity function, defined as the product of user-defined function, sets how many units a machine can process.

(optional) If the process produces, degrades or modifies a set of macromolecules. it is specified in this node. A componentMap is used to compute metabolites used and byproducts generated during the process.

(optional) Fluxes that must be produced by the process at a preset value. These can be metabolite or reaction fluxes. The flux value can be a preset value or the product of user-defined functions. It can be an absolute flux or a dilution_compensation flux. In latter case, it serves as a way to maintain the target at constant concentration, and the flux value actually scales with the growth rate.

```
−<targets>
  −<targetReaction reaction="Th">
     <functionReference function="number_flagella"/>
     <functionReference function="flagella_speed"/>
     <functionReference function="flagella_H_consumption"/>
   </targetReaction>
 </targets>
−<targets>
   <targetValue species="mrna" value="0.01"/>
   <targetValue species="trna" value="0.02625"/>
   <targetValue species="trna2" value="0.01125"/>
   <targetValue dilution_compensation="0" species="mrna" value="0.15996"/>
 </targets>
```

```
<RBAProcesses>
+<listOfProcesses></listOfProcesses>
+<listOfComponentMaps></listOfComponentMaps>
</RBAProcesses>
```

Global structure: a list of processes of the cell and a list of component maps used by processes to build/degrade polymers.

```
−<listOfComponentMaps>
  −<componentMap id="translation">
    −<constantCost>
      −<listOfReactants>
         <speciesReference species="m_tfmet" stoichiometry="1"/>
         <speciesReference species="m_gtp" stoichiometry="1"/>
         <speciesReference species="m_h2o" stoichiometry="1"/>
       </listOfReactants>
      −<listOfProducts>
         <speciesReference species="m_trna" stoichiometry="1"/>
         <speciesReference species="m_fmet" stoichiometry="1"/>
         <speciesReference species="m_gdp" stoichiometry="1"/>
         <speciesReference species="m_p" stoichiometry="1"/>
         <speciesReference species="m_h" stoichiometry="1"/>
       </listOfProducts>
     </constantCost>
    −<cost component="ala" processingCost="1">
      −<listOfReactants>
         <speciesReference species="m_tala" stoichiometry="1"/>
         <speciesReference species="m_gtp" stoichiometry="2"/>
         <speciesReference species="m_h2o" stoichiometry="2"/>
       </listOfReactants>
      −<listOfProducts>
         <speciesReference species="m_trna" stoichiometry="1"/>
         <speciesReference species="m_gdp" stoichiometry="2"/>
         <speciesReference species="m_p" stoichiometry="2"/>
         <speciesReference species="m_h" stoichiometry="2"/>
       </listOfProducts>
     </cost>
    +<cost component="arg" processingCost="1"></cost>
    +<cost component="asn" processingCost="1"></cost>
```

(optional) A component map may have a constant cost part that is independent of a polymer's composition. A processing cost may be associated with this node.

For every component of the macromolecule, the component map defines the metabolites consumed/generated during its processing (production, degradation or modification). If the process has a capacity constraint, a processingCost can also be associated to each component. The number of components the machinery can process is then limited by its capacity (explicitly: the sum of processing costs cannot exceed the machine's capacity).

Figure 8: Structure of process file.

4.3 From data to structures

Algorithm overview Figure 9 shows how the XML files are transformed into matlab structures. Note that a lot of elements are already in matrix form.

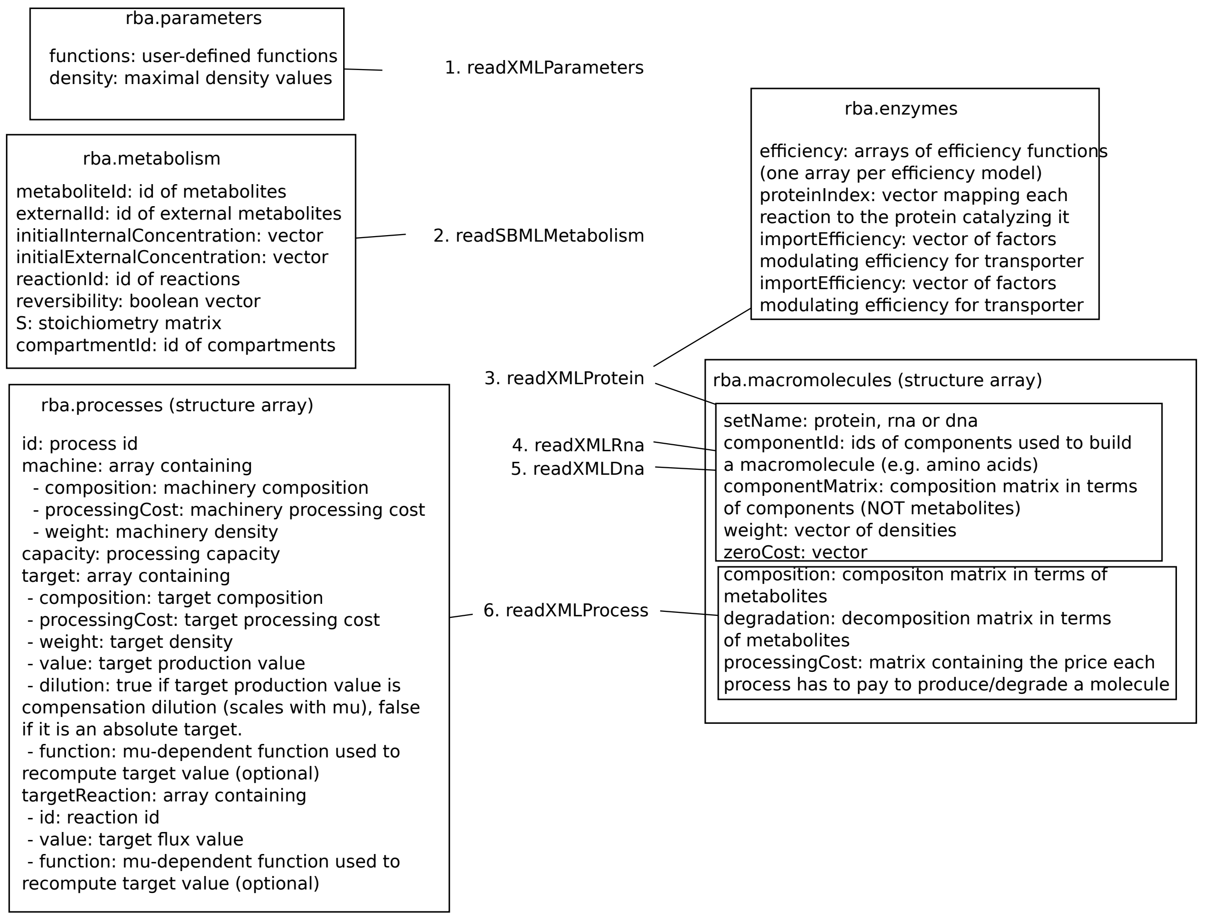

rba.parameters
functions: user-defined functions
density: maximal density values
1. readXMLParameters
rba.metabolism
metaboliteId: id of metabolites
externalId: id of external metabolites
initialInternalConcentration: vector
initialExternalConcentration: vector
reactionId: id of reactions
reversibility: boolean vector
S: stoichiometry matrix
compartmentId: id of compartments
2. readSBMLMetabolism
rba.enzymes
efficiency: arrays of efficiency functions
(one array per efficiency model)
proteinIndex: vector mapping each
reaction to the protein catalyzing it
importEfficiency: vector of factors
modulating efficiency for transporter
importEfficiency: vector of factors
modulating efficiency for transporter
3. readXMLProtein
rba.macromolecules (structure array)
setName: protein, rna or dna
componentId: ids of components used to build
a macromolecule (e.g. amino acids)
componentMatrix: composition matrix in terms
of components (NOT metabolites)
weight: vector of densities
zeroCost: vector
composition: compositon matrix in terms of
metabolites
degradation: decomposition matrix in terms
of metabolites
processingCost: matrix containing the price each
process has to pay to produce/degrade a molecule
4. readXMLRna
5. readXMLDna
6. readXMLProcess
rba.processes (structure array)
id: process id
machine: array containing
- composition: machinery composition
- processingCost: machinery processing cost
- weight: machinery density
capacity: processing capacity
target: array containing
- composition: target composition
- processingCost: target processing cost
- weight: target density
- value: target production value
- dilution: true if target production value is
compensation dilution (scales with mu), false
if it is an absolute target.
- function: mu-dependent function used to
recompute target value (optional)
targetReaction: array containing
- id: reaction id
- value: target flux value
- function: mu-dependent function used to
recompute target value (optional)

Figure 9: Algorithm used to read data in the new formalism.

Composition matrices Figure 10 shows how the matrices stored in `rba.macromolecules` are built.

Process-related matrices Figure 11 shows how the matrices stored in `rba.processes` are built.

4.4 Building final matrices

Figure 12 shows how the optimization matrices stored are built.

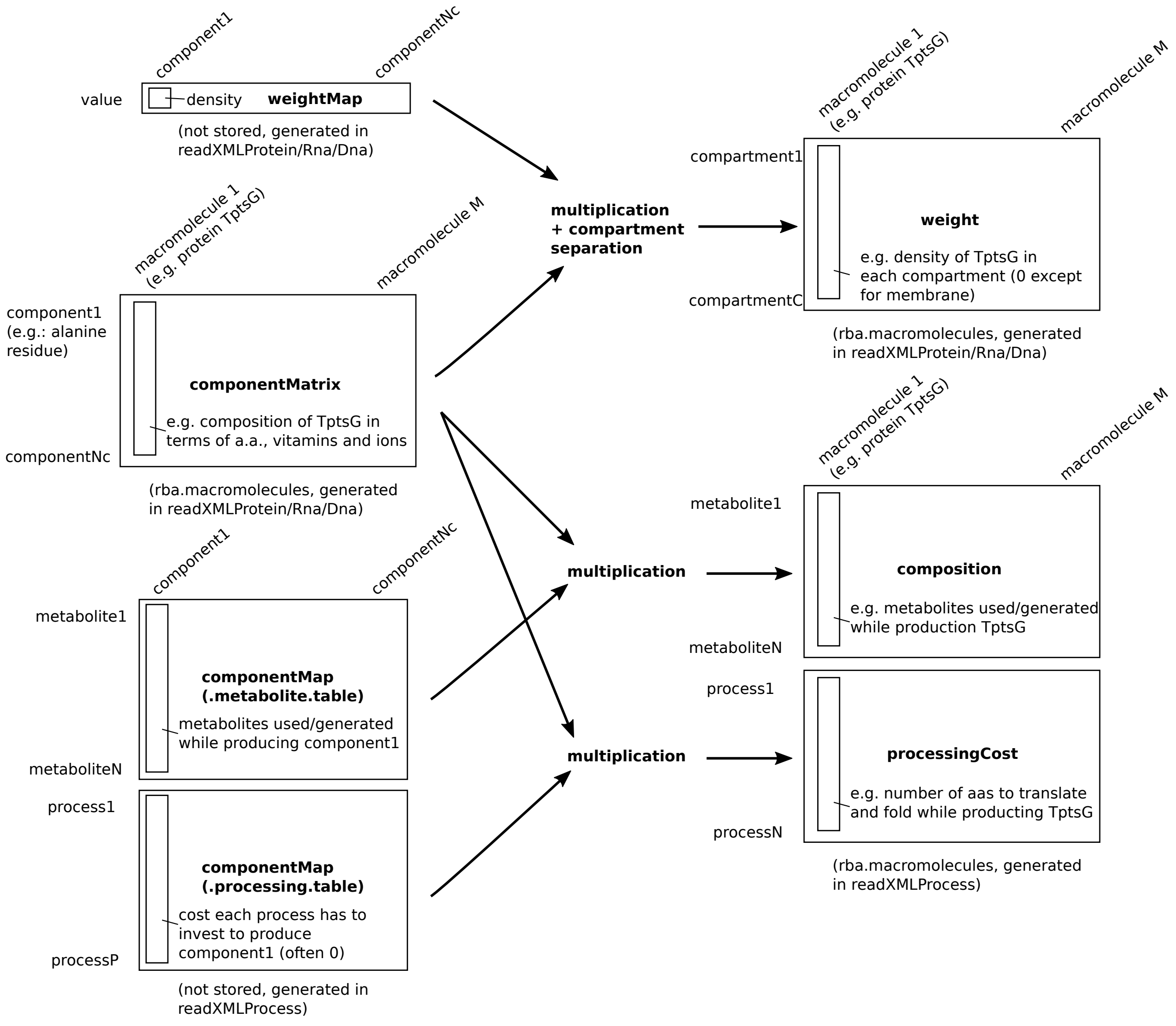

component1
componentNc
value
density
weightMap
(not stored, generated in readXMLProtein/Rna/Dna)
macromolecule 1 (e.g. protein TptsG)
macromolecule M
component1 (e.g.: alanine residue)
componentNc
componentMatrix
e.g. composition of TptsG in terms of a.a., vitamins and ions
(rba.macromolecules, generated in readXMLProtein/Rna/Dna)
component1
componentNc
metabolite1
metaboliteN
componentMap (.metabolite.table)
metabolites used/generated while producing component1
process1
processP
componentMap (.processing.table)
cost each process has to invest to produce component1 (often 0)
(not stored, generated in readXMLProcess)
multiplication + compartment separation
multiplication
multiplication
macromolecule 1 (e.g. protein TptsG)
macromolecule M
compartment1
compartmentC
weight
e.g. density of TptsG in each compartment (0 except for membrane)
(rba.macromolecules, generated in readXMLProtein/Rna/Dna)
macromolecule 1 (e.g. protein TptsG)
macromolecule M
metabolite1
metaboliteN
composition
e.g. metabolites used/generated while production TptsG
process1
processN
processingCost
e.g. number of aas to translate and fold while producting TptsG
(rba.macromolecules, generated in readXMLProcess)

Figure 10: Macromolecule matrices.

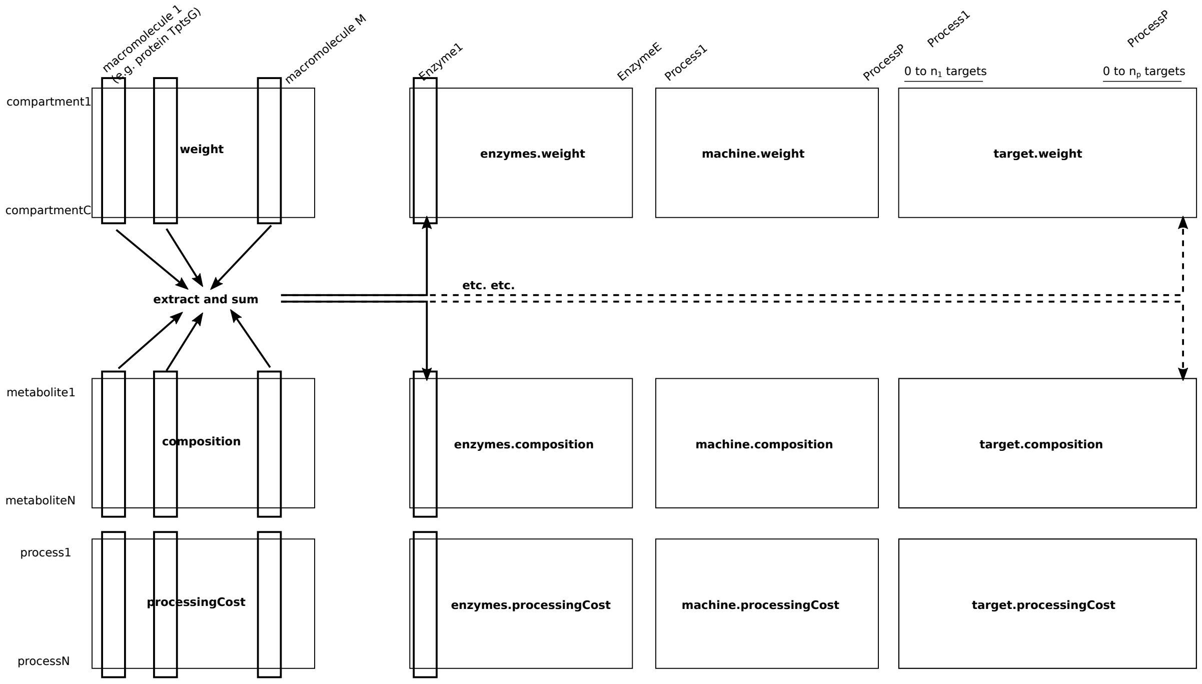


Figure 11: Process-related matrices.

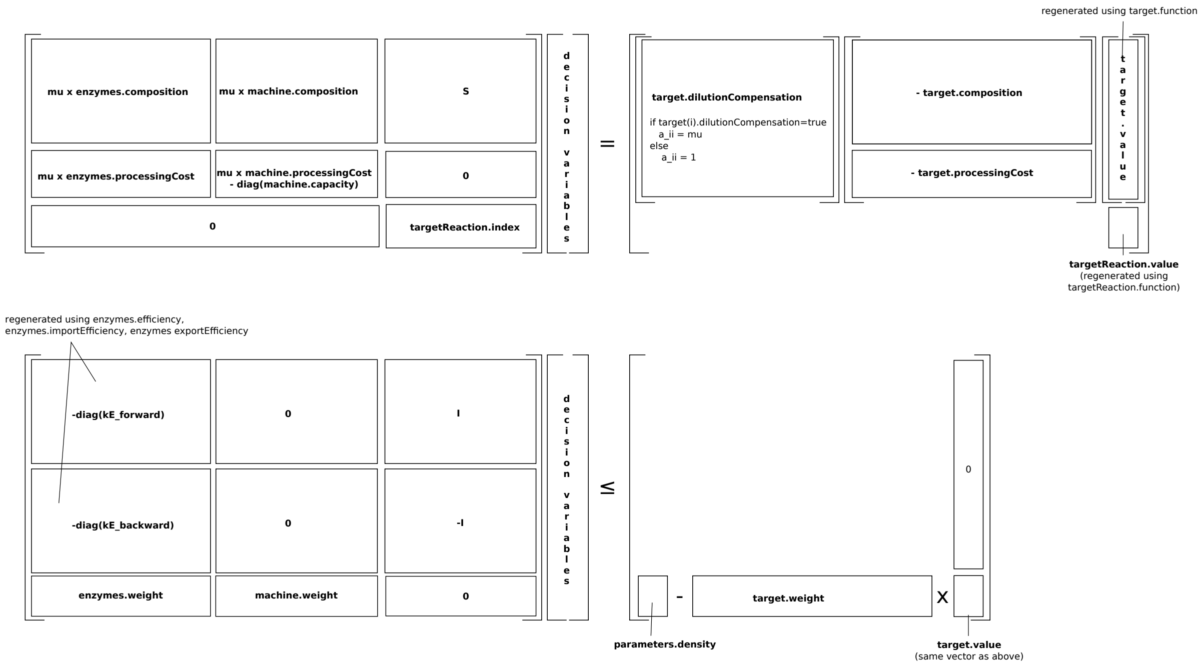


Figure 12: Matrices used to solve the optimization problem.